Joint Task Allocation and Path Planning for Space Robot

Yifei Sun
Guangdong University of Technology
Zikai Zhang
Beijing Jiaotong University
YiDong Li
Beijing Jiaotong University
Jigang Wu (asjgwucn@outlook.com)
Guangdong University of Technology

Research Article

Keywords: Space robot, Path planning, Combination algorithm, Completion latency

Posted Date: July 19th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1844809/v1

Joint Task Allocation and Path Planning for Space Robot

Yifei Sun[a,b], Zikai Zhang[b,c], YiDong Li[b], Jigang Wu[a,*]

[a]*School of Computer Science and Technology, Guangdong University of Technology*
[b]*School of Computer and Information Technology, Beijing Jiaotong University*
[c]*State Key Lab of Rail Traffic Control and Safety, Beijing Jiaotong University*

Abstract

Space robots have a wide application prospect in the aerospace industry. Due to the limited fuel for the space robots, which cannot support the space robots to run for a long time. In addition, the special working environment makes it impossible to replenish the fuel for the robot at any time. Therefore, a proper path is crucial for the operation of space robots. In order to ensure the operation of space robots, we optimize the allocation of exploration tasks and the selection of space robot paths, jointly. We propose two combination algorithms named Parallel Search and Task Allocation (PS-TA) and Subbranch Insertion and Task Allocation (SI-TA) to optimize the path and the task allocation, intend to obtain the minimum completion latency. We also construct Random Path Planning and Task Allocation (RTA) as the baseline. At last, we provide extensive experiments to demonstrate that proposed algorithms can obtain lower completion latency compared with RTA. Furthermore, SI-TA is more energy-efficient than PS-TA.

Keywords: Space robot, Path planning, Combination algorithm, Completion latency

1. Introduction

Space robot technology obtain continuous progress with the sustainably developed space industry. The space robot can perform various space tasks such as area exploration, and sample collection without astronauts [1, 2]. Generally, space robots can be categorized into on-orbit and exploration robots. The on-orbit robots mainly provide diverse on-orbit services. While the main services provided by exploration robots mainly include extravehicular exploration and base construction, etc.

Advances in the space field have greatly promoted the research of space robots, such as path planning of space robots and system design [3, 4]. As early

*Corresponding author
Email address: asjgwucn@outlook.com (Jigang Wu)

as 1994, some scholars have already systematically studied the path planning problem [5]. The path planning can be classified into local path planning and global path planning, depending on the knowability of global information [6, 7]. Global path planning is implemented under the condition that global information is known. The three most established global path planning methods are intelligent bionic algorithm, graph search algorithm, and random sampling algorithm [8]. In the static environment, global path planning can provide the optimal path for each space robot [9]. While the local path planning (online path planning) performs better in the dynamic environment [10]. Local path planning can adaptively generate the current optimal trajectory when obstacles occur between the source and destination [11]. To obtain optimal path planning, many algorithms have been adopted such as the simulated annealing algorithm [12], and fuzzy logic algorithm [13].

Since the practical robots have limited resources for space work, path planning is a crucial issue for studying to improve the efficiency of planet exploration. Path planning is a typical complex nonlinear optimization problem. Traditional optimization strategies are ineffective in resolving the issue. The bionic has been widely adopted to solve complex optimization since its appearance. Researchers have proposed different biological-based solutions to tackle path planning for space robots [14, 15, 16, 17, 18, 19]. The above works mainly study the single space robot path planning. Few works focus on multi-space robot path planning.

In this paper, we mainly investigate the joint task allocation and path planning for multiple space robots. Each exploration region must be explored and only require exploration once to decrease overall completion delay. Multiple space robots set out simultaneously from different starting areas to perform exploration tasks and assemble in a single destination after all areas have been explored. The most similar problem to this paper is the Multi-Depots Vehicle Routing Problem (MD-VRP) [20]. In MD-VRP, multiple vehicles start from different depots, serve a set of customers, and terminate their tours at the depots. The time spent by a vehicle moving between two customers is much longer than the time consumption spent serving customers. However, in this paper, the moving time of a space robot is much less than the exploration time. Therefore, the algorithms used for MD-VRP can not be applied to our problem directly. Although an exploration area may appear in multiple paths, only one space robot need to explore this area. This makes us aware of the importance of a proper task allocation strategy. It is difficult to predict the time of a space robot moving in two areas and exploring one area in advance. In this case, space robots can only generate one-step trajectory after each actions, including moving and exploring. It is vital to considerably allocate the exploration tasks while optimizing path planning to lower the exploration latency [21].

To our knowledge, this is the first research to look into multi-space robot task allocation and path planning, jointly. The main contributions are as follows.

- We investigate the joint exploration task allocation and path planning for space robots. Multiple space robots locate their respective starting areas

and set off at the same time to execute exploration tasks. Finally, all space robots assemble at the same destination after completing tasks. We model the problem to minimize exploration delay.

- We propose two combination algorithms named SI-TA and PS-TA. The sequential search subbranch insertion in SI-TA allocates the remaining areas according to the graph construction and initial path of each space robot. The forward parallel search algorithm in PS-TA finds the next area in each round depending on its distance to the destination. Based on the paths generated by SI-TA and PS-TA, a Greedy-based Task Allocation (GRTA) algorithm based on greedy policy is proposed to minimize the exploration latency.

- We create the RTA algorithm as the baseline. Then, we implement extensive experiments to demonstrate the effectiveness of SI-TA and PS-TA. PS-TA achieves lower compared with SI-TA, while SI-TA consumes less fuel than PS-TA, according to the results of the experiments. Moreover, in terms of completion delay and fuel usage, PS-TA and SI-TA all outperform RTA.

The remainder is organized as follows. We summarize the related work in Section 2. We then describe the studied problem and model the problem in Section 3. In Section 4, we propose two algorithms named PS-TA and ST-TA, respectively. Extensive experiments and analyses are presented in Section 5 Finally, the conclusion of this paper is presented in 6.

2. Related Works

Path planning in dynamic situations is one of the most essential study issue of robots industry, especially when the extreme conditions are subject to robots in space environment.

Since the path planning problem is NP hardness [22], it is hard for a exponentially complicated traverse algorithm to satisfy the time limited practical applications Path planning is also a fundamental question in mobile robots to generate an optimal or sub-optimal path to enable the efficient operation of the robot [23, 24, 25]. In general, path planning approaches may be divided into three categories: biologically inspired, combinatorial, and sampling-based [26]. The biological-based path planning is a type of intelligent algorithm that simulates the evolutionary behaviors of biology. The combinatorial path planning usually integrates the graph search algorithm and workspace representation methods to optimize the path planning problem [27]. The sampling-based algorithms are usually utilized to find a path quickly [28]. However, the path found by sampling-based algorithms may not be the optimal path.

In recent years, biologically-based techniques have been increasingly popular in space robot path planning. The authors in [14] presented a genetic algorithm to find the path to ensure that the on-orbit space robots get close to the target satellites, safely. The authors in [15] developed a self-adaptive ant colony

algorithm, which has a faster convergence rate than the traditional ant colony algorithm. To avoid the obstacle, the authors in [16] proposed an artificial bee colony algorithm. In addition, some academics have developed a series of combination techniques to boost path planning performance even more. The authors in [17] integrated a genetic algorithm and simulated an annealing algorithm to generate a path with the capability of obstacle avoidance. In a 3D environment, a hybrid genetic-cuckoo search algorithm was designed in [18] to generate an optimal path and can alleviate the conflict between optimality and delay.

Except for the biology-based algorithms, the authors of [19] suggested a path planning algorithm on the basis of enhanced fuzzy control to avoid the obstacles that appeared continuously. The authors of [29] designed a deep deterministic policy gradient method with multi-constrained reward to find the best path while keeping path length, coupling disturbance, and safety in mind.

The existing works mentioned above can solve the path planning problem for single space robots with different methods. However, the cooperation of multiple space robots in a space environment is less considered to the most of our knowledge.

3. Scenario description and problem definition

In our studied scenario, multiple space robots are required to execute multiple areas exploring tasks, collaboratively. To avoid additional fuel consumption, we assume that each area needs to be explored only once. All space robots undertake exploration activities at the same time in different starting locations and finally stay in the same destination area when all areas have been explored. In our work, we conjointly study the execution decisions of exploration tasks and the multi-space robot path planning to minimize total exploration task completion delay. Besides, we consider homogeneous space robots for simplicity. A space robot does not take time to explore the area which has been explored by other space robots. Note that, a space robot is required to explore the area which is not explored as it arrive. If some space robots arrive in a previously unexplored area at the same time, the lower serial number space robot explore this area. We will cover the serial number later.

Next, we will introduce the studied scenario and problem in this paper and then formulate the problem. We define an undirected graph $G = (\mathcal{V}, E)$ to abstractly represent the entire exploration area. We denote $E = \{1, 2, \cdots, e\}$ to represent the paths between areas. Besides, let $V = \{1, 2, \cdots, v\}$ be the area set. In order to facilitate the description, we will use word "node" and "area" interchangeably in the following text, indicating the same meaning. For an exploration task, we can obtain the corresponding graph $G = (\mathcal{V}, E)$ in advance. Then, we number the nodes in a graph. We set the serial number of the starting node to start at 0, that is $0, 1, \cdots, s-1$. Space robots at different starting nodes have the same serial number as the starting nodes where they locate, which also means that there are s space robots. Note that the destination is the node with the largest serial number. Let $\lambda_b^{i,j}$ indicate whether the space

robot b moves between area i and j.

$$\lambda_b^{i,j} = \begin{cases} 1 & \text{The space robot } b \text{ moves between area } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

When $i = j$, $\lambda_b^{i,j} = 0$. We can use an order sequence $\langle \cdot \rangle$ to represent a space robot path. For example, if a space robot locate at area 1 in the begin, goes through area 2 and comes back to area 1, then the path of this space robot can be represent as $\langle 1, 2, 1 \rangle$. Next, a discriminant operation $\ominus$ was defined as follows. If $\lambda_b^{i,j} = 0$, $\lambda_b^{i,j} \ominus j = \{\emptyset\}$. When $\lambda_b^{i,j} = 1$, $\lambda_b^{i,j} \ominus j = \{j\}$. Then, a splicing operation $\otimes$ between sets is also defined. The rule for $\otimes$ is as follows.

$$\{f\} \otimes \{g\} = \langle f, g \rangle. \tag{2}$$

Finally, let $\bigsqcup$ be the operation of set accumulation.

$$\bigsqcup_a^v \{\phi_a\} = \{\phi_a\} \otimes \{\phi_{a+1}\} \otimes \cdots \otimes \{\phi_v\}. \tag{3}$$

Then, let P_b be the path of space robot b, which can be obtained by follow formula.

$$P_b = \{b\} \otimes (\bigsqcup_{i=b}^{v-1} \bigsqcup_{j=0}^{v-1} \lambda_b^{i,j} \ominus j), \tag{4}$$

Our main goal is to achieve minimum exploration latency. We define the time for space robot b to explore area i as τ_b^i. If the multiple paths contain the same area, then, which space robot takes the responsibility for the exploration of this area should be determined carefully. Let γ_b^i indicate whether the space robot b performs the exploration tasks of area i, where $b \in \{0, 1, \cdots, s-1\}$, $i \in \{0, 1, \cdots, v-1\}$.

$$\gamma_b^i = \begin{cases} 1 & \text{The space robot } b \text{ explore area } i \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Let T_b^{exe} denote the time spent by space robot b completing the exploration task.

$$T_b^{exe} = \sum_{i=0}^{v-1} \gamma_b^i \cdot \tau_b^i. \tag{6}$$

Let $\kappa_b^{i,j}$ indicate the delay for space robot b moving between area i and area j. Then, we denote θ to indicate the position of a node in P_b, where $1 \leq \theta < |P_b|$. As an illustration, the θ-th node on path P_b can be represented as $P_b[\theta]$. Let T_b^r be the overall movement delay of space robot b.

$$T_b^r = \sum_{\theta=1}^{|P_b|-1} \kappa_b^{P_b[\theta], P_b[\theta+1]}. \tag{7}$$

We denote T to represent as the overall task completion latency.

$$T = max\{T_b^{exe} + T_b^r\}. \tag{8}$$

We can formalize the latency minimization problem as follows.

$$min\, T \tag{9}$$

$$\text{s.t.} \sum_{b=0}^{s-1} \sum_{i=0}^{v-1} \gamma_b^i = v. \tag{10}$$

Formula (10) ensures that each area is explored.

Algorithm 1 Find Shortest Path

Input: s, v, G
Output: l

Initial l
Append s to l //Append start node to path l
while l does not end with v **do**
    Get the last node e of l
    Obtain the neighbor node set C of e according to G
    Calculate the distance d between first node in set C and v
    $d_{e,v} = Caldis(e, v)$ //Calculate the distance between two nodes
    for each c in C **do** $d_{c,v} = Caldis(c, v)$
        if $d_{c,v<d_{e,v}}$ and $d_{c,v} < d$ **then**
            $d \leftarrow d_{c,v}$
    Obtain the node c^* corresponding to d
    Append c^* to l
return l

4. Solutions

The joint optimization problem is decomposed into two subproblems, path planning, and task allocation. Before solving the task allocation problem, we need to find the working path for all space robots. The path of each space records the movement sequence of the robot from a starting area to the destination. The generated paths must cover all nodes in the given graph. Then, we solve the task allocation problem to obtain the execution decision of exploration tasks to minimize the overall latency. In particular, we propose two path planning algorithms named Sequential Search Subbranch Insertion (SSSI) and Forward Parallel Search (FPS), respectively. After that, a task allocation algorithm named Greedy-based Task Assignment (GRTA) is proposed to generate the execution decision of each exploration area. Finally, we combine SSSI with GRTA to obtain SI-TA and combine FPS with GRTA to obtain PS-TA.

4.1. Solutions for Path Planning

We next introduce the SSSI and FPS, respectively. We first introduce SSSI. The Algorithm 1 is used to obtain the shortest path for each space robot, which is usually used to generate the shortest path in many works. However, the paths obtained by using Algorithm 1 can not cover all nodes. Next, the SSSI algorithm is used to insert nodes that are not covered into the shortest paths obtained by Algorithm 1. The details of SSSI are described as follows.

Algorithm 2 Sequential Search Subbranch Insertion

Input: $G, l_b, b \in \{1, 2, \cdots, s\}$ //l_b is initial path of space robots b
Output: l_b

$w = 0$
Initialize F_b
Denote z to indicate state of nodes //explored or unexplored
while There exists paths b where $l_b[w]$ is not destination **do**
    $\Phi = \{\emptyset\}$
    Record serial number of all space robots that meet the conditions in Φ.
    for each φ in Φ **do**
        Construct h_φ //neighbor nodes set of $l_\varphi[w]$
        if There exits nodes that not explored **then**
            Insert the node $l_\varphi[w]$ into the last position of F_φ
            Choose an unexplored node h' in h_φ
            Insert h' into position $l_\varphi[w+1]$
            Update z
        if The successor node g of $l_\varphi[w]$ are contained by h_φ **then**
            if The last node of F_φ is in g **then**
                Insert g into position $l_\varphi[w+1]$
                Remove g from F_φ
    $w = w + 1$;
return l_b

First, stores all paths that do not reach the destination according to w, where w is the search position on paths. Define set Φ to record the space robot serial number corresponding to the paths. Then, we can construct the set h_φ to record serial numbers of neighbor nodes of node $l_\varphi[w]$ according to the structure of the given graph G, where l_φ represents the path of robot φ.

Next, we judge the type of nodes in Φ according to z, where z is used to indicate whether a node is explored. If the node $l_\varphi[w]$ is an unexplored node, we insert $l_\varphi[w]$ into the last position of F_φ, where F_φ is the set of bifurcation nodes. We denote a node as a bifurcation node if the neighbor nodes of this node are not all explored nodes. Then, we select an unexplored node h' in h_φ and insert h' into the position $l_\varphi[w+1]$. We next update the z. If all nodes in h_φ are explored. We insert node g into position $l_\varphi[w+1]$, where g is contained by h_φ and is the successor node of $l_\varphi[w]$ and the last node of F_φ. Next, we remove g from F_φ. Finally, update the search position w. The pseudocode of SSSI is described in Algorithm 2.

Algorithm 3 Distance Calculation

Input: $G = \{\mathcal{N}, E\}$
Output: $\mathcal{D}$ //distance information
1: $\mu = 1$
2: Initial $\mathcal{E}$ //indicator vector
3: **while** $Sum(\mathcal{E}) < |\mathcal{N}|$ **do** //Sum($\mathcal{E}$) represent the sum of elements in $\mathcal{E}$
4: Initialize ς //set of nodes that are not operated
5: Initialize η
6: **for** $\alpha = 0$ to $|op| - 1$ **do**
7: **for** $\beta = 0$to $|\mathcal{N}| - 1$ **do**
8: **if** $op[\alpha]$ is a neighbor of β **then**
9: **if** $ind[\beta] == 0$ **then**
10: $D[\beta] = \mu$ //record distance
11: $\mathcal{E}[\beta] = 1$
12: append the node β to ς
13: $\mu = \mu + 1$
14: $\eta = \varsigma$
15: **return** $\mathcal{D}$

Next, we introduce FPS. The main idea of FPS is to synchronously search the next node to go for each space robot that has not reached the destination. The successful execution of FPS requires preliminary knowledge of the distance from all nodes to the destination. We adopt Algorithm 3 to obtain the distance information. In this process, we assume that the distance between two adjacent nodes is 1. We first set the destination node to be the initial node. Then, the distance of nodes that are directly adjacent to the destination and have not been accessed is set as 1. Continue to find the nodes which are directly adjacent to the nodes with 1 distance to the destination and have not been accessed, and the distance of these nodes to the destination is set as 2. Repeat the above steps until all nodes have been accessed. We denoted D to record the distance information. After that, we obtain the path of each space robot by using Algorithm 4. The main process is summarized as follows.

Step1: Find all space robots which have not reached the destination and construct set S to store the found space robots serial number.

Step2: Construct set C_r of neighbor nodes of the last node for each p_r according to the given G. Each node in C_r may be an explored node or an unexplored node. We denoted W_r to represent the set of unexplored nodes.

Step3: Calculate the number η_r of nodes in C_r.

Step4: Append the only node in W_r to l_r when $\eta_r = 1$, where l_r is the path of space robot r.

Step5: Find out all nodes which are farthest to the destination according to D and W_r. Construct D^r_{max} to record the corresponding space robots serial numbers.

Algorithm 4 Forward Parallel Search

Input: $G = \{\mathcal{N}, \mathcal{E}\}, D$
Output: l_i

```
1: Initialize ρ
2: Initialize l_i to record path for i
3: while There are robot r ∈ S that has not reached destination node do
4:     for each r in S do
5:         Get neighbor nodes set C_r of last node of path according to the G
6:         Get unexplored nodes set W_r of i according to C_r and ρ
7:         Calculate the number η_r of nodes in W_r
8:         if η_r == 1 then
9:             Append this node to path l_r
10:            Update ρ
11:        if η_r > 1 then
12:            Construct set D^r_max of nodes that are farthest to destination
               node according to D, W_r
13:            if D^r_max only contains one node then
14:                Append this node to path l_r
15:            else
16:                Select a node from D^r_max randomly to append to l_r
17:                Append the penultimate node on l_r to l_r again
18:            Update ρ
19:        if η_r ≤ 0 then
20:            while true do
21:                Find node with minimal distance to destination node in C_r
22:                Append this node to l_r
23:                if r has reached the destination node then
24:                    break                //terminates the path search for r
25:                Repeat line 5 to 18
26:                break;
27: return l_i
```

Step6: Choose a node from D^r_{max} to add to the last position of l_r. After that, we append the penultimate node on l_r to l_r again.

Step7: Find out the node with minimal distance from starting node to destination in C_r and append this node to l_r. If r reaches the destination, we terminate the algorithm. Otherwise, repeat Step4 - Step6 once. Then, we terminate the algorithm.

For the pseudocode of FPS, see Algorithm 4

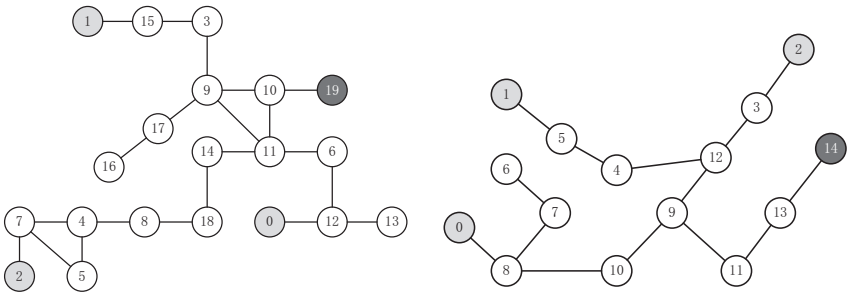
4.2. Greedy-based Task Assignment

In this subsection, we propose GRTA based on greedy policy to generate the execution decisions of exploration tasks. The details of GRTA are described in Algorithm 5.

Algorithm 5 Greedy-based Task Allocation

Input: p_b
Output: T
$w = 1$, $\gamma = p_b[0]$
$T = \tau_b^{\gamma}$
Let vector z indicate whether nodes are explored
Let m_b represent the length of p_b
while There exists $w \leq m_b$ **do**
    $U = \{\emptyset\}$
    for $b = 0$ to $s - 1$ **do**
        if $w \leq m_b$ **then**
            $U = U \cup b$
    for each u in U **do**
        if Node $p_u[w]$ is an explored node **then**
            $T_u = T_u + \kappa_u^{p_u[w-1],p_u[w]}$
        else
            Construct set Ω to record paths whose last node is $p_u[w]$
            Select space robot a whose T_a is minimum.
            $T_a = T_a + \tau_a^{p_a[w]} + \kappa_a^{p_a[w-1],p_a[w]}$
            for Robot α in $\Omega \setminus a$ **do**
                $T_\alpha = T_\alpha + \kappa_\alpha^{p_\alpha[w-1],p_\alpha[w]}$
        Update z
    $w = w + 1$
Compute T with formula (8)
return T

First, let m_b be the length of path p_b and w be the index to indicate the current search position of all paths. We define U to record the space robot serial number. If $w \leq m_b$, we append b to U. Then, we judge whether the node $p_b[w]$ has been explored for each space robot b. If node $p_b[w]$ has been explored, we just compute the moving delay for b arriving at this node. If this node is an



(a) Graph G_1 (b) Graph G_2

Figure 1: Graphs for experiment

unexplored node, we record all paths whose node in position w is this node. Then, we construct Ω to store the robot serial number corresponding to these paths. We next choose a node a from Ω with a minimum T_a and compute the exploring time and moving time of a. For node α in $\Omega \setminus a$, we only compute the movement time of the robot arriving at this node. Finally, we update the state of nodes in ρ. We calculate the T with Equation (8) when $h > max(S_i)$.

GAT can be formally described by Algorithm 5.

4.3. Combination algorithm

We combine Algorithm 2 and Algorithm 5 to obtain SI-TA algorithm. Besides, we also obtain a combination algorithm named PS-TA by combining Algorithm 4 and Algorithm 5. SI-TA and PS-TA solve the path planning problem through Algorithm 2 and Algorithm 4, respectively. After that, SI-TA and PS-TA obtain the execution decision of exploration task by using GRTA algorithm.

5. Simulation Results and Analysis

A series of experiments are implemented in this section to demonstrate the performance of SI-TA and PS-TA. Furthermore, RTA algorithm is created for comparison. The RTA selects nodes randomly for each space robot and uses GRTA to make decisions for exploration tasks.

5.1. Simulation Setup

In the experiments, we construct a specific scenario where three space robots execute an exploration task. As shown in Figure 1(a)[30] and Figure 1(b) which are graphs abstracted from two areas. The first exploration task consists of 20 exploration areas and the second exploration task consists of 15 exploration areas. In the beginning, all space robots stay at their respective starting areas (0, 1, and 2). We set the largest serial number node as the destination. We set the moving delay between two areas is uniformly distributed in [10, 20]s. The time consumed by exploring an area is distributed in [200 500]s.

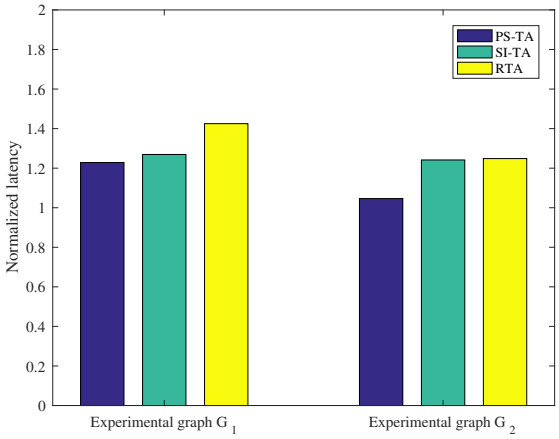


Figure 2: Completion latency.

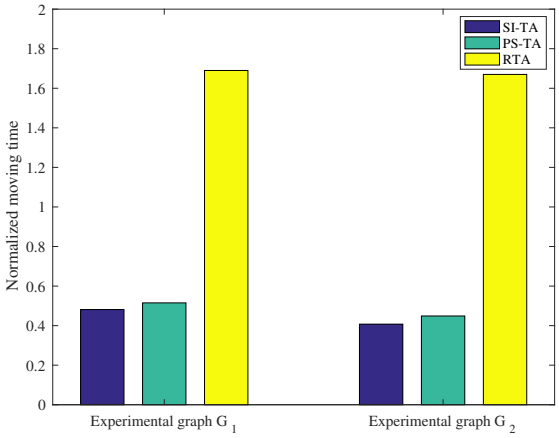
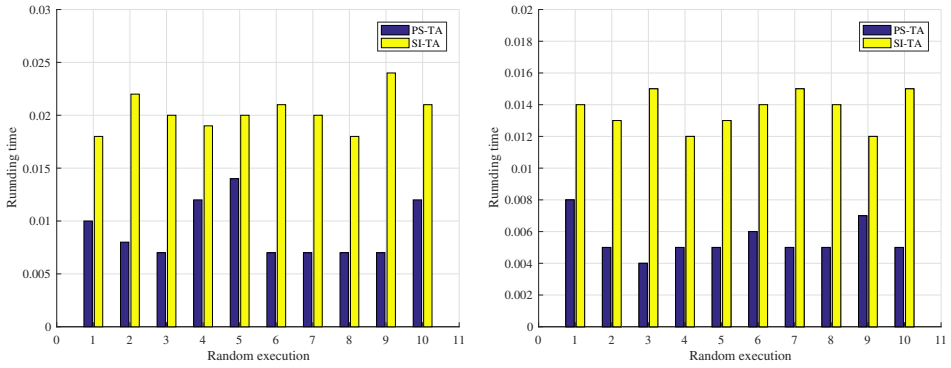


Figure 3: The movement delay obtained by algorithms.



(a) On graph G_1 (b) On graph G_2

Figure 4: Algorithms running time on two graphs

5.2. Experimental Results and Performance Analysis

First, we focus on the performance of algorithms on task completion latency. We normalize the latency by dividing the same constant. The Figure 2 shows the final results. It is observed that the latency of PS-TA is minimum. Because PS-TA can make the load between space robots more balanced. We can also see that the delay of RTA is the largest, mainly due to the high randomness of the path planning algorithm used by RTA.

Because space robot fuel is limited, and refueling is difficult. Therefore, it is significant to consider the fuel consumption of space robots. Due to the homogeneity of space robots, the exploration time for all space robots to explore an area is same under different algorithms. The main difference in completion latency of different algorithms main depends on the moving delay. Therefore, we can compare the energy efficiency of algorithms according to the space robots moving delay. For the sake of simplicity, we normalized the movement time. The Figure 3 show the experimental results. From the figure, we can see that the SI-TA can save more fuel than PS-TA. The fuel consumption of RTA is the highest. This is caused by the randomness of path planning in RTA.

Next, the running time of the SSSI and the FPS are counted. To avoid contingency, we count the results of 10 experiments, respectively. The Figure 4(a) and Figure 4(b) show the results on G_1a nd G_2, respectively. From Figure 4, the running time of SSSI is higher. The main reason is that SSSI spent more time executing the insert operation.

6. Conclusion

In this paper, we have investigated the joint path planning and task allocation problem for space robots and formalized the problem to minimize the exploration task completion latency. In this problem, multiple space robots set out simultaneously from different starting areas to perform exploration tasks

and assemble at a single destination after all areas have been explored. We have proposed two algorithms named SI-TA and PS-TA by integrating a task allocation algorithm and a path planning algorithm to solve the completion latency minimization problem. In particular, we have designed two path planning algorithms named SSSI and FPS, which can generate paths for all space robots moving from their respective starting node to the destination. In addition, each node of the given graph must be contained by at least one path. Furthermore, we have proposed an algorithm named GRTA as the task allocation algorithm to allocate the exploration tasks based on the generated paths to space robots with the greedy idea to obtain minimum completion latency. To further prove the performance of the proposed algorithms, the RTA algorithm has been constructed as a baseline, which generates paths for space robots by randomly selecting nodes and allocating tasks with GRTA. To demonstrate the performance of proposed algorithms, extensive experiments have been implemented. Experimental results show the effectiveness of the proposed algorithm. In terms of fuel consumption and completion latency, the proposed algorithms perform better compared with RTA. The proposed algorithms are all better than RTA on delay and fuel consumption. Specifically, the PS-TA can save more time to complete the exploration tasks than SI-TA. Besides, SI-TA consumes less fuel than PS-TA.

7. Acknowledgement

Funding:This work is supported by the National Natural Science Foundation of China [No. 62106052 and 62072118], Guangdong Key R&D Project of China [No. 2019B010121001].

8. Data availability

The data underlying this article will be shared on reasonable request to the corresponding author.

9. Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

[1] Z. XING, Y. ZHAO, S. Zhu, Path planning method design and dynamic model simplification of free-flying space robot, in: 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2020, pp. 1501–1505. doi:10.1109/ICIEA48937.2020.9248286.

[2] Y. Lin, D. Li, Y. Wang, et al., Current status and analysis of space robot, Spacecr. Eng 24 (5) (2015).

[3] L. E. Xiong Youlun, Ding Han, Robotics, Mechanical Industry Press, Beijing, 1993.

[4] A. N. Atiyah, N. Adzhar, N. I. Jaini, An overview: on path planning optimization criteria and mobile robot navigation, Journal of Physics: Conference Series 1988 (1) (2021) 012036. doi:10.1088/1742-6596/1988/1/012036.

[5] L. Chen, Y. Huang, H. Zheng, H. Hopman, R. Negenborn, Cooperative multi-vessel systems in urban waterway networks, IEEE Transactions on Intelligent Transportation Systems 21 (8) (2020) 3294–3307. doi:10.1109/TITS.2019.2925536.

[6] P. B. Kumar, C. Sahu, D. Parhi, K. Pandey, A. Chhotray, Static and dynamic path planning of humanoids using an advanced regression controller, Scientia Iranica 26 (1) (2019) 375–393.

[7] Z. Tang, H. Ma, An overview of path planning algorithms, IOP Conference Series: Earth and Environmental Science 804 (2) (2021) 022024. doi:10.1088/1755-1315/804/2/022024.

[8] K. Cai, C. Wang, J. Cheng, C. W. De Silva, M. Q.-H. Meng, Mobile robot path planning in dynamic environments: a survey, arXiv preprint arXiv:2006.14195 (2020). doi:10.15878/j.cnki.instrumentation.2019.02.010.

[9] G. Klancar, A. Zdesar, S. Blazic, I. Skrjanc, Wheeled mobile robotics: from fundamentals towards autonomous systems, Butterworth-Heinemann.

[10] M. M. Costa, M. F. Silva, A survey on path planning algorithms for mobile robots, in: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2019, pp. 1–7. doi:10.1109/ICARSC.2019.8733623.

[11] H. S. Dewang, P. K. Mohanty, S. Kundu, A robust path planning for mobile robot using smart particle swarm optimization, Procedia Computer Science 133 (2018) 290–297, international Conference on Robotics and Smart Manufacturing (RoSMa2018). doi:10.1016/j.procs.2018.07.036.

[12] B. Basbous, 2d uav path planning with radar threatening areas using simulated annealing algorithm for event detection, in: 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), 2018, pp. 1–7. doi:10.1109/IDAP.2018.8620881.

[13] A. Bakdi, A. Hentout, H. Boutami, A. Maoudj, O. Hachour, B. Bouzouia, Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control, Robotics and Autonomous Systems 89 (2017) 95–109. doi:10.1016/j.robot.2016.12.008.

[14] A. Seddaoui, C. M. Saaj, Collision-free optimal trajectory for a controlled floating space robot, in: Towards Autonomous Robotic Systems, Springer International Publishing, Cham, 2019, pp. 248–260.

[15] W. Ye, D. Ma, H. Fan, Path planning for space robot based on the self-adaptive ant colony algorithm, in: 2006 1st International Symposium on Systems and Control in Aerospace and Astronautics, 2006, pp. 4 pp.–33. doi:10.1109/ISSCAA.2006.1627696.

[16] F. Jin, G. Shu, Path planning of free-flying space robot based on artificial bee colony algorithm, in: Proceedings of 2012 2nd International Conference on Computer Science and Network Technology, 2012, pp. 505–508. doi:10.1109/ICCSNT.2012.6525987.

[17] F. Jin, Path planning of free-flying space robot using memetic algorithm, in: Ifost, Vol. 2, 2013, pp. 19–22. doi:10.1109/IFOST.2013.6616889.

[18] J. Wang, X. Shang, T. Guo, J. Zhou, S. Jia, C. Wang, Optimal path planning based on hybrid genetic-cuckoo search algorithm, in: 2019 6th International Conference on Systems and Informatics (ICSAI), 2019, pp. 165–169. doi:10.1109/ICSAI48974.2019.9010519.

[19] B. Shi, H. Wu, Space robot motion path planning based on fuzzy control algorithm, Journal of Intelligent and Fuzzy Systems (Preprint) 1–8.

[20] Y. E. M. Vieira, R. A. de Mello Bandeira, O. S. da Silva Júnior, Multi-depot vehicle routing problem for large scale disaster relief in drought scenarios: The case of the brazilian northeast region, International Journal of Disaster Risk Reduction 58 (2021) 102193.

[21] Y. Sun, Z. Zhang, Y. Li, J. Wu, Joint optimization of path planning and task assignment for space robot, in: 2021 12th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), IEEE, 2021, pp. 47–51.

[22] B. Chen, G. Quan, Np-hard problems of learning from examples, in: 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, Vol. 2, 2008, pp. 182–186. doi:10.1109/FSKD.2008.406.

[23] K. Karur, N. Sharma, C. Dharmatti, J. E. Siegel, A survey of path planning algorithms for mobile robots, Vehicles 3 (3) (2021) 448–468.

[24] Y. Zhuang, Y. Sun, W. Wang, Mobile robot hybrid path planning in an obstacle-cluttered environment based on steering control and improved distance propagating, Int. J. Innov. Comput. Inf. Control 8 (2012) 4095–4109.

[25] M. A. Contreras-Cruz, V. Ayala-Ramirez, U. H. Hernandez-Belmonte, Mobile robot path planning using artificial bee colony and evolutionary programming, Applied Soft Computing 30 (2015) 319–328. doi:10.1016/j.asoc.2015.01.067.

[26] S. K. Debnath, R. Omar, N. B. A. Latip, A review on energy efficient path planning algorithms for unmanned air vehicles, in: R. Alfred, Y. Lim, A. A. A. Ibrahim, P. Anthony (Eds.), Computational Science and Technology, Springer Singapore, Singapore, 2019, pp. 523–532.

[27] S. M. LaValle, Planning algorithms, Cambridge university press, 2006.

[28] F. Yan, E. Xia, Z. Li, Z. Zhou, Sampling-based path planning for high-quality aerial 3d reconstruction of urban scenes, Remote Sensing 13 (5) (2021) 989.

[29] X. Hu, X. Huang, T. Hu, Z. Shi, J. Hui, Mrddpg algorithms for path planning of free-floating space robot, in: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), 2018, pp. 1079–1082. doi:10.1109/ICSESS.2018.8663748.

[30] Mastermind – 2021 national algorithm design challenge, https://www.shenjims.com/.