Complete coverage problem of multiple robots with different velocities

Lin Li¹, Dianxi Shi²,³, Songchang Jin²,³, Ying Kang⁴, Chao Xue³, Xing Zhou⁵, Hengzhu Liu¹ and XiaoXiao Yu¹

Abstract
Complete coverage, which is integral to many robotic applications, aims to cover an area as quickly as possible. In such tasks, employing multiple robots can reduce the overall coverage time by appropriate task allocation. Several multi-robot coverage approaches divide the environment into balanced subareas and minimize the maximum subarea of all robots. However, balanced coverage in many situations, such as in the cases of robots with different velocities and heterogeneous multi-robot systems, may have inefficient results. This study addresses the unbalanced complete coverage problem of multiple robots with different velocities for a known environment. First, we propose a novel credit model to transform the unbalanced coverage problem into a set of single-objective optimization problems, which can find a combinational optimal solution by optimizing each separate objective function of the single-objective optimization problem to alleviate the computational complexity. Then, we propose a credit-based algorithm composed of a cyclic region growth algorithm and a region fine-tuning algorithm. The cyclic region growth algorithm finds an initial solution to the single-objective optimization problems set by a regional growth strategy with multiple restricts, whereas the region fine-tuning algorithm reallocates the tasks of the partitions with too many tasks to the partitions with too few tasks by constructing a search tree, thereby converging the initial solution to the optimal solution. Simulation results indicate that compared with conventional multi-robot complete coverage problem algorithms, the credit-based algorithm can obtain the optimal solution with the increased number of robots and enlarged size of the mission environment.

Keywords
Complete coverage problem, multi-robot task allocation, balanced connected task allocation

Date received: 16 November 2021; accepted: 14 March 2022

Introduction
Complete coverage problem (CCP) is the task of visiting all the points over the available space while avoiding obstacles and minimizing the coverage time.¹ This task is integral to many robotic applications, such as area exploration,²³⁴ environment monitoring⁶⁷⁸⁹ oil pill cleaning,¹⁰ searches and rescue,¹¹ and automation in agriculture¹²¹³ Utilizing multiple robots may reduce the overall coverage time by appropriate task allocation and enhance the robustness to the potential failures.³ However, the collaboration within

¹ College of Computer, National University of Defense Technology, Changsha, Hunan, China
² Artificial Intelligence Research Center (AiRC), Defense Innovation Institute, Beijing, China
³ Tianjin Artificial Intelligence Innovation Center (TAIIC), Tianjin, China
⁴ People's Liberation Army of China, Changchun, China
⁵ College of Intelligence Science and Technology, National University of Defense Technology, Changsha, Hunan, China

Corresponding authors:
Dianxi Shi and Songchang Jin, Artificial Intelligence Research Center (AiRC), Defense Innovation Institute, Beijing, China; Tianjin Artificial Intelligence Innovation Center (TAIIC), Tianjin, China.

Emails: dxshi@nudt.edu.cn; jsco4@tsinghua.org.cn
The number of robots increases the algorithmic complexity and logistic management. This paper addresses the problem of covering a known environment with a group of mobile robots. The mission environment has been decomposed into regions, and each region has a known weight representing its area to cover. After decomposition, the mission environment can be abstracted as a node-weighted graph, where nodes represent regions and edges represent the common boundary between two regions. Figure 1(a) illustrates a known environment decomposed into cells, whereas Figure 1(b) represents the abstract graph of the environment after cell decomposition. Each vertex in the abstract graph represents a task. Complete coverage requires that each task has been assigned to at least one robot. If there are no tasks been repeatedly assigned, it is called disjoint task allocation. In addition, the tasks assigned to the same robot should be connected, thus avoiding the disasters of robot collisions. After task allocation, a single-robot coverage path planning approach is utilized to plan coverage paths individually for each robot. The implementation of single-robot coverage path planning is beyond the scope of this article, and reviews of such approaches are presented in the literature.

Several research approaches have been proposed in the past to perform multi-robot CCP. Most multi-robot CCP approaches divide the environment into balanced subareas and minimize the maximum subarea of overall robots. This min–max problem is usually modeled as the balanced, connected $q$ partition (BCP$q$) problem. While for some applications, utilizing BCP$q$ for area coverage achieves higher efficiency in many situations, such as robots with different velocities and heterogeneous multi-robot systems, balanced task allocation may have inefficient results. With this consideration, multi-robot CCP algorithms can be categorized as either balanced or unbalanced. Balanced coverage algorithms ensure that the coverage task is divided into equal subtasks, whereas for an unbalanced approach, there is necessary to assign tasks to robots considering their relative capabilities. Figure 1(c) illustrates an example of unbalanced task allocation, in which two robots with different velocities are used for coverage. The fast robot deserves more tasks to improve the coverage efficiency, while the slow robot deserves few tasks. For such an unbalanced coverage approach, the challenge is developing a suitable optimization methodology to find the optimal solution with higher accuracy and a broader range.

This article proposes a credit-based algorithm based on a novel credit model to address the unbalanced task allocation problem for multiple robots. Unlike previous works, this article assumes that each robot’s velocity in a multirobot system can be different. The credit-based algorithm divides the coverage tasks into several disjoint and interconnected subtasks according to robots’ velocities. Simulation results indicate that the proposed algorithm performs well in terms of precision and scalability. In summary, the contributions are as follows:

- We model the unbalance coverage problem into a multi-objective optimization problem (MOP) and propose a novel credit model to decouple the MOP into a set of single-objective optimization problems (SOP) to alleviate the computational complexity.
- We propose a credit-based algorithm composed of a cyclic region growth algorithm and a region fine-tuning algorithm. The cyclic region growth algorithm obtains an initial combinatorial solution for the set of SOPs by a multi-constrained task assignment strategy, whereas the region fine-tuning algorithm converges the initial solution to the optimal solution by reallocating the tasks of the partitions.

**Figure 1.** A task allocation solution for two robots that travel at 1.5 m/s and 0.5 m/s. The tasks of multiple robots are disjoint, whereas tasks of a single robot are connected. The single-robot coverage path planning algorithm is applied to each robot’s tasks, generating the coverage path for a single robot. (a) The mission environment has been decomposed into multiple cells, in which blue cells and white cells represent obstacles and the area to be covered, respectively. (b) Environment abstraction. Vertices represent coverage cells, whereas edges between vertices refer to common borders. (c) Task allocation. The task set has been divided into two partitions. The large partition is assigned to the fast robot, whereas the small partition corresponds to the slow robot.
with too many tasks to the partitions with too few tasks.

- Experiments demonstrate that compared with conventional balanced multi-robot task allocation (MRTA) algorithms, the credit-based algorithm can obtain the optimal solution with the increased number of robots and enlarged size of the mission environment. Besides, the credit-based algorithm can obtain the optimal unbalanced solution for 300 robots in the environment with a size of 300*300 cells.

The rest of the article is organized as follows. The second section presents the related work. In the third and fourth sections, we present the problem statement and outline the proposed algorithm. Then, experimental results are presented in the fifth section. Finally, the conclusion is presented in the sixth section.

**Related work**

Task allocation is one of the fundamental problems of multi-robot systems, which involves matching tasks and robots to optimize the overall system performance subject to specific constraints. According to the requirements of tasks and the characteristics of multi-robot systems, MRTA can be divided into diverse categories. Different from other approaches, this article studies the MRTA for the application of area coverage. We assume that the mission environment is already known and has been abstracted into a node-weighted graph. The robots have ideal execution capabilities, each robot can only perform one task simultaneously, and each task requires a robot to perform. The goal of the task allocation problem is to divide the graph into several partitions, which are connected with the optimal number of tasks.

The problem of balanced, connected, and disjoint partitions of known graphs are usually modeled as a min–max BCPq problem. Several approaches optimize the BCPq by tree partition, which generates a virtual tree based on the graph and then divides the tree in specific ways to obtain \( q \) partitions with the same weights. Hazon and Kaminka first generated a spanning tree coverage path based on the graph and then assigned \( k \) portions of the path to different robots. The coverage time of work was reduced by two times under the condition that no intersects tasks were generated. However, it cannot guarantee that task assignment is close to optimal as the number of robots grows. Zheng et al. proposed a polynomial-time algorithm to find \( k \) (\( k \) is the number of robots) trees that covered all vertices and assigned an even set of tasks to \( k \) robots. Although this method guarantees that the coverage time is eight times the optimal coverage time, it assumes that trees can overlap.

Different from tree partition, graph partition has been proved to be an NP-hard problem, even when \( q = 2 \) and the weights of vertices are all one. In 2019, the authors presented an exact mixed linear integer program based on the flow model, which partitions graphs with 170 vertices in \( q = 2 \). The approach proposed by Kapoutsis et al. updated an evaluation matrix to modify the task assignment results to obtain a balanced task assignment. In addition to these classic approximation methods, various meta-heuristic methods are also proposed, such as variable neighborhood search algorithms and genetic algorithms. However, the neighborhood search algorithms were designed for the 2-partition on the general graph, whereas genetic algorithms traded between accuracy and speed.

Balanced task allocation is a particular case of unbalanced task allocation. For unbalanced task allocation, the most widely known method is the market-based method, especially auction algorithms. The core idea of auction algorithms is that each agent bids on a task and the winner is assigned to this task. Literature found that simple auction-based allocation linearly increased the computational cost as the number of robots grew. The linear computational complexity effectively improves the algorithm’s scalability. However, its performance in a multi-agent environment is still an issue. ElGibreen and Youcef-Toumi used the auction method to assign appropriate tasks to heterogeneous robots. Although this approach performs well, it is a time-extended assignment but not an instantaneous assignment. Liu and Shell mixed centralized and distributed methods at different levels to improve the algorithm’s scalability. The allocation method is scalable, but it sacrifices the quality of the results.

The analysis of related works indicates that there is always a trade-off between scalability and accuracy. Exact methods take time to find the optimal solution, and thus they do not scale well. Although heuristic methods always find the solution quickly and effectively, it tends to be affected by a heuristic, which may lead to local optimal rather than global optimal. Therefore, there is room for contributions to improve the scalability and accuracy of task allocation approaches. According to this necessity, this work proposes a novel algorithm that considers both scalability and accuracy for the unbalanced MRTA problem.

**Problem statement and formulation**

In the following section, the MRTA of area coverage is formally presented together with relevant definitions. We assumed to have a multi-robot system \( R = \{r_1, \ldots, r_K\} \), where \( r_k, 1 \leq k \leq K \) represents the \( k \)-th robot in \( R \). Robots in \( R \) are equipped with a task sensor for performing the specific task with a fixed-size footprint \( L \) and can travel at different velocities \( v = \{v_1, \ldots, v_K\} \), where \( v_k \) represents the velocity of \( r_k \). The starting point of \( R \) is denoted by \( s = \{s_1, \ldots, s_K\} \), which is mutually exclusive, i.e. \( s_i \cap s_j = \emptyset, i \neq j, i,j = 1, \ldots, K \). The mission environment \( A \subset \mathbb{R}^2 \) is assumed to be a bounded planar field populated with a finite and known number of obstacles.
To achieve area coverage, \( A \) is assumed to be decomposed into a set of square-shaped cells of size \( L \). This cell set can be abstracted into a node-weight connected graph \( G = (V, E, W) \), where the vertex set \( V \) represents the set of cells to be covered with a known weight function \( W: V \rightarrow \mathbb{N}^+ \) and the edge set \( E \) represents common boundaries between cells. For any vertex set \( X \subseteq V \), let \( w(X) \) denote the sum of weights of vertices in \( X \). Given that a cell represents both a vertex in \( G \) and a coverage task, the terms “cell,” “vertex,” and “task” are used interchangeably in this article.

This article discusses the complete and optimal coverage problem for a multi-robot system. The complete coverage denotes that each vertex in the graph \( G \) has to be assigned to at least one robot, and the optimal coverage refers to dividing \( V \) into \( K \) disjoint partitions \( \{V_1, \ldots, V_K\} \), in which \( V_k, k = 1, \ldots, K \) is connected and has the optimal number of tasks \( p_k \) (see Eq. (2)). We model the complete and optimal coverage problem as the following MOP with multiple constraints.

Objective

\[
\min |w(V_k) - p_k|, k = 1, \ldots, K
\]

s.t.

\[
p_k = \frac{v_k}{\sum_{k=1}^{K} v_k}
\]

\[
V_1 \cup V_2 \cup \ldots \cup V_K = V
\]

\[
V_i \cap V_j = \emptyset, i, j = 1, \ldots, K \text{ and } i \neq j
\]

\[
G(V_k) \text{ is connected}
\]

\[
s_k \in V_k
\]

where \( |w(V_k)| \) represents the sum weights of \( V_k \) and \( p_k \) represents the optimal number of tasks of \( p_k \). Equation (3) indicates that each task in \( V \) should be assigned to at least one robot. Equation (4) represents that tasks would not be assigned to multiple robots. Equation (5) means that tasks in \( V_k \) should be connected to prevent the collision disaster between robots. Equation (6) indicates that each robot’s starting point should belong to its corresponding task set.

Remark 1. A partition \( \{V_1, \ldots, V_K\} \) composes an optimal solution for the MOP, if [1.]

1. \( |w(V_k)| = p_k \).
2. \( V_1 \cup V_2 \cup \ldots \cup V_K = V \).
3. \( V_i \cap V_j = \emptyset, i, j = 1, \ldots, K \text{ and } i \neq j \).
4. \( G(V_k) \) is connected.
5. \( s_k \in V_k \).

Because of the start points of robots and the distribution of obstacles, the optimal solution of the MOP may not exist. For example, if other robots completely surround a robot, it is impossible to find a solution that satisfies both condition 1 and condition 3 in Remark 1. Therefore, we focus on the case that there is at least one optimal solution in this article.

Proposed solution

To address the unbalanced task allocation problem of multi-robot systems, this article presents a credit model and a credit-based algorithm.

Credit model

Several models have been proposed to address the MRTA problem of area coverage. For example, a flow model\(^3\) was proposed to address the balanced and connected task allocations for multi-robot systems. However, the flow model was designed for graphs of less than 200 nodes of two robots, which may not suit the large and complicated coverage applications. This article proposed a novel credit model to solve the unbalanced MRTA in a broader range and with higher accuracy. The credit model takes inspiration from the conventional market-based methods\(^3\) and uses the concepts of traders and commodities. Robots in the credit model are set up as traders in a virtual economy, whereas coverage tasks are tradable commodities with measurable values. Unlike market-based methods, we introduce a virtual bank that opens a credit account for each robot and manages traders’ accounts and assets. Each account’s capital is initialized to the robot’s optimal number of tasks and can be changed with the transactions between traders. Two types of transactions may occur: (i) if a robot buys a task, its account balance decreases by one and (ii) if a robot sells a task, its account balance increases by one. If a trader wants to purchase a task but does not have any money in his account, he can get a loan from the virtual bank without interest and his account balance will be negative. Thus, each account’s balance is used to quantify the difference between actual assignment and optimal assignment. All accounts have a balance of zeros, indicating that \( K \) robots all have the optimal number of tasks, which means the first condition in Remark 1 is satisfied. In addition, if all partitions simultaneously meet the other conditions in Remark 1, the partitions are the optimal combinatorial solution of the MOP. Figure 2 illustrates an example of the credit model, in which four robots are assigned to cover the mission environment.

Based on the credit model, we set a specific objective function for each partition

\[
f(t + 1, k) = \begin{cases} 
  f(t, k) + 1 & \text{if } V_k \text{ sell a task} \\
  f(t, k) - 1 & \text{if } V_k \text{ buy a task} \\
  f(t, k) & \text{otherwise}
\end{cases}
\]

where \( f(t, k) \) represents the capital of each robot at time \( t \), s.t., \( f(0, k) = p_k \). \( f(t, k) < 0 \) represents \( V_k \) has fewer cells than the optimal \( p_k \) by \( f(t, k) \) cells, while \( f(t, k) \geq 0 \)
indicates $V_k$ has more cells than the optimal $p_k$ by $f(t, k)$ cells. If $K$ partitions that satisfy $f(T_e, k) = 0, k = 1, \ldots, K, T_e > 0$ and meet the second to the fifth conditions in Remark 1 are found, then an optimal solution of the MOP is obtained.

The credit-based algorithm

We apply the credit model to the CCP application and propose a novel credit-based task allocation algorithm to find the combinational optimal solution of $K$ partitions. The credit-based algorithm comprises the cyclic region growth algorithm and the region fine-tuning algorithm. The cyclic region growth algorithm obtains $K$ initial partitions by a multi-constrained task assignment strategy, whereas the local task reassignment algorithm optimizes the $K$ initial partitions based on a search tree.

Cyclic region growth algorithm. The certain environment has been decomposed by the grid-based decomposition methods and been modeled as a binary map $M$. In $M$, the cell with a value of 0 represents the obstacle cell, whereas the cell with a value of 1 refers to the free space cell to be covered. Once a cell is allocated to one robot, the map updates this information in real time. Figure 3 demonstrates an example of map initializing and updating, in which cells allocated to $r_1$ are transited from “1” to “1.1,” whereas cells assigned to $r_2$ are marked from “1” to “1.2.”

The cyclic region growth algorithm works as follows: each partition alternately purchases tasks, and all partitions incrementally get larger and larger as the numbers of purchased cells increase. Partitions will stop growing once all tasks have been purchased and $V$ has been divided into $K$ partitions. Specifically, five restrictions are proposed to make all partitions meet the second to the fifth conditions in Remark 1.

- Complete coverage restriction. To make $K$ partitions meet condition 2 of Remark 1, we assume that the cyclic region growth algorithm ends if all coverage tasks are purchased, i.e. each task is allocated to at least one robot.
- Disjoint restriction. To meet condition 3 of Remark 1 for $K$ partitions, it is assumed that each partition can only buy the tasks which other partitions have not bought. This restriction enables that a coverage task would not be assigned to multiple robots.
- Connective restriction. The connective restriction makes each partition meet the fourth condition in Remark 1. We call the task set adjacent to $V_k$ and have not yet been purchased by other partitions as the adjacent task set of $V_k$, denoted as $NV_k$. Figure 4(a) and (b) demonstrates examples of the adjacent task that are marked in grey. To maintain the connectivity of $V_k$, it is assumed that $V_k$ can only buy tasks from $NV_k$.
- Starting point restriction. We assume that $K$ partitions grow around their starting point to meet condition 5 of Remark 1.
- Growth rate restriction. To make the number of tasks of each partition close to the optimal, we set that partitions alternately purchase tasks at specific rates. The partition with a large number of optimal

![Figure 2](image_url)

**Figure 2.** An example of the credit model for four robots. Given that the velocities of robots are 1m/s, 1m/s, 2m/s, and 2m/s, their optimal numbers of tasks are set as 16, 16, 32, and 32 cells, respectively. (a) The mission environment after being decomposed. The white, black, green, blue, red, and yellow cells represent the coverage task, obstacle, and starting points of the four robots, respectively. (b) Task allocation. The task set has been divided into four partitions with 16, 16, 32, and 32 cells. Partitions’ tasks are disjoint, and the tasks of each partition are interconnected.

![Figure 3](image_url)

**Figure 3.** The dynamic updating process of the map for two robots. 0 and 1 represent the obstacle cell and free space cell, respectively, whereas 1.1 and 1.2 refer to the cell that has been allocated to $r_1$ and $r_2$, respectively. (a) Obstacle layout and the initial map encoding. (b) Task allocation screenshots and map encoding updates.
tasks can purchase more tasks, whereas the partition with a small number of potimal tasks can buy only a few tasks. For example, in Figure 3, the optimal number of tasks for five partitions is 68, 136, 202, 67, and 67, indicating that the number of purchasing tasks at one time and the total number of tasks for five partitions have the ratio of 1:2:3:1:1.

Remark 2. Let \( v_1 \in V \) and \( v_2 \in V \) be the two vertices of the graph \( G = (V, E, W) \). Then, \( v_1 \) and \( v_2 \) are said to be adjacent, if \( \exists e \in E \), s.t. \( e \) connects \( v_1 \) and \( v_2 \). Let \( X_1 \subseteq G \) and \( X_2 \subseteq G \) be the vertex sets. Then, \( X_1 \) and \( X_2 \) are said to be adjacent, if \( \exists v_1 \in X_1, v_2 \in X_2 \), s.t. \( v_1 \) and \( v_2 \) are adjacent.

The pseudo-code of the cyclic region growth algorithm is present in Algorithm 1. The given inputs are a binary matrix \( M \), robots’ starting points \( \{s_1, \ldots, s_K\} \), and the optimal number of tasks \( \{p_1, \ldots, p_K\} \). First, the total number of coverage tasks \( N \), partitions’ objective functions \( \text{func} \), and growth rates \( \text{rate} \) are calculated (lines 1–3). If tasks have not been wholly allocated (line 4), robots alternately purchase tasks (line 5) at a specific growth rate (line 6). In every purchase operation, the cyclic region growth algorithm first updates the adjacent cell set \( NV_k \) by querying neighbors of the latest assigned task (line 7). Then, a cell \( M(x, y) \) is selected from \( NV_k \) (line 9) and marked as \( 1 + 0.1 \cdot k \) (line 10), indicating that the partition \( V_k \) has purchased the task. Then, the objective function of partition \( V_k \) is reduced by one (line 11). Conversely, if all coverage tasks have been assigned, then the cyclic region growth algorithm ends.

Algorithm 1: Cyclic Region Growth Algorithm

\begin{itemize}
\item \textbf{Input:} \( M \): the binary image with size \([\text{ROW}, \text{COL}]\) of the target area, \( s = \{s_1, \ldots, s_K\} \): the start points of \( R \), \( p = \{p_1, \ldots, p_K\} \): the optimal number of tasks of \( K \) partitions.
\item \textbf{Output:} \( M', \text{func}' \).
\end{itemize}

1. \( N \leftarrow \sum_{x=1}^{\text{ROW}} \sum_{y=1}^{\text{COL}} M(x, y) \);
2. \( \text{func} \leftarrow \{p_1, \ldots, p_K\} \); //Eq.2
3. \( \text{rate}(k) \leftarrow \frac{\text{GCD}(k)}{\text{NB}} \); \forall k = 1, \ldots, K. \text{GCD} \) is the greatest common divisor of \( p_i \).
4. \textbf{While} \( \{N > 0\} \{ \)
5. \hspace{1em} \textbf{For} \( \{k = 1 \rightarrow K\} \{ \)
6. \hspace{2em} \textbf{For} \( \{j = 1 \rightarrow \text{rate}(k)\} \{ \)
7. \hspace{3em} \textbf{If} \( \{\text{NV}_k \neq \emptyset\} \{ \)
8. \hspace{4em} \( M(x, y) \leftarrow \text{select a cell from NV}_k \).
9. \hspace{4em} \text{func}(k) = \text{func}(k) - 1 \); //Eq.7
10. \hspace{3em} \textbf{Else} \{ \)
11. \hspace{4em} \( M(x, y) = 1 + 0.1 \cdot k \);
12. \hspace{4em} \text{NV} = \text{NV} - 1 \;
13. \hspace{3em} \} \)
14. \hspace{2em} \} \)
15. \hspace{1em} \} \)
16. \} \)
17. \( M' = M; \text{func}' = \text{func}; \)
18. \( \text{Return} \ M', \text{func}' \).

Computational complexity. The cyclic region growth algorithm trades \( N \) tasks. For each task, it performs nine operations, while four operations are for updating \( NV_k \) and the remaining are for buying tasks. Therefore, the overall computational complexity of the cyclic region growth algorithm is \( O(N) \).

Region fine-tuning algorithm. Because of the random starting positions of robots and complex obstacles, the number of
tasks of \(K\) partitions obtained by the cyclic region growth algorithm may not be optimal. Therefore, a region fine-tuning algorithm is presented to eliminate the difference between the actual and optimal partitions. In the region fine-tuning algorithm, the objective function \(f(t,k)\) of the partition \(V_k\) indicates the difference between the partition \(V_k\) and its optimum. If \(f(t,k) > 0\), the number of tasks of the partition \(V_k\) should be increased by \(f(t,k)\) to converge to the optimum. Conversely, if \(f(t,k) < 0\), the number of tasks of the partition \(V_k\) should be reduced by \(|f(t,k)|\). Therefore, the partition with too many tasks has to assign tasks with too few tasks.

Assume that \(\text{start}\) and \(\text{dest}\) are the two partitions that need to reassign tasks. Two possible cases can occur based on the positions of \(\text{start}\) and \(\text{dest}\). In the first case, \(\text{start}\) and \(\text{dest}\) are adjacent, and they can directly trade tasks located on the boundaries of both partitions. The partition with a negative objective function sells cells to the partition with a positive objective function, making the task numbers of both partitions closer to the optimum. The second case occurs if \(\text{start}\) and \(\text{dest}\) are non-adjacent. Given that two partitions are non-adjacent, the trade of tasks between two partitions will cause the tasks within the partition to be disconnected. Therefore, a search tree is presented to find a task-reassignment path between two partitions. The search tree is built based on a graph in which vertices represent partitions, and the edge denotes that partitions at the end of the edge are adjacent. As shown in Figure 5, the depth-first algorithm is adopted to generate the search path that constituted a partition set \{\(\text{start}\), \(\text{E1}\), ..., \(\text{En}\), \(\text{dest}\)\}, where the two partitions next to each other in the path are adjacent. If the partition \(\text{start}\) meets \(f(t,\text{start}) \leq 0\), then the \(\text{E1}\) partition buys a task from the \(\text{start}\) partition, and the \(\text{E2}\) partition buys a task from the \(\text{E1}\) partition, etc. until the \(\text{dest}\) partition buys a task. Conversely, the partition \(\text{start}\) meets \(f(t,\text{start}) \geq 0\), then the \(\text{start}\) partition buys a task from the \(\text{E1}\) partition, and the \(\text{E1}\) partition buys a task from the \(\text{E2}\) partition, etc. until the \(\text{dest}\) partition sells one task. Repeated task reassignments eliminate the difference between the actual partitions and the optimal partitions. If all partitions have the optimal number of tasks, the solution converges to the global optimal. Figure 6 demonstrates an example of the region fine-tuning algorithm, which receives the output of the cyclic region growth algorithm in Figure 6(a), performs task reassignment in Figure 6(b), and obtains the optimal partition in Figure 6(c).

The pseudo-code of the region fine-tuning algorithm is present in Algorithm 2. The inputs are the output of Algorithm 1 (\(M'\) and \(\text{func}'\)) and the maximum number of iterations (\(\text{MaxInum}\)). First, the region fine-tuning algorithm calculates each partition’s set of tasks adjacent to other partitions. The sets of connected cells for \(K\) partitions form \(BV = \{BV_1, \ldots, BV_K\}\) (line 2), where \(BV_k\) is the set of cells of partition \(V_k\) connected to other partition. Then, the region fine-tuning algorithm identifies whether all partitions have the optimal task numbers (line 3). If not, the two partitions with the maximum and minimum \(\text{func}'\) are selected (lines 4–5). The depth-first algorithm is utilized to generate a task-reassignment path between the two

\begin{align*}
\text{start} & \rightarrow \text{E1} \\
\text{E1} & \rightarrow \text{E2} \\
\text{E2} & \rightarrow \text{E3} \\
\text{E3} & \rightarrow \text{E4} \\
\text{E4} & \rightarrow \text{E5} \\
\text{E5} & \rightarrow \text{E6} \\
\text{E6} & \rightarrow \text{dest}
\end{align*}

Figure 5. An example of the search path in the region fine-tuning algorithm. The vertices represent partitions, whereas edges indicate that the two partitions are adjacent. The dotted line represents the task exchanging path, and all partitions in the path need to exchange tasks.

\begin{align*}
\text{start} & \rightarrow \text{E1} \\
\text{E1} & \rightarrow \text{E2} \\
\text{E2} & \rightarrow \text{E3} \\
\text{E3} & \rightarrow \text{E4} \\
\text{E4} & \rightarrow \text{E5} \\
\text{E5} & \rightarrow \text{E6} \\
\text{E6} & \rightarrow \text{dest}
\end{align*}

Figure 6. An example of the region fine-tuning algorithm. Partitions with too many tasks reallocate tasks to the partitions with too few tasks. The reallocated cells between partitions are marked by white lines. (a) Initial task allocation obtained by the cyclic region growth algorithm. The task numbers of five partitions deviate from the optimum by \{-2, 3, -3, -10, 12\}. (b) Snapshot of the region fine-tuning algorithm. The partition in the lower right corner reassigns six tasks to the partition in the lower-left corner and four tasks to the partition in the upper right corner. (c) An optimal task allocation obtained by the region fine-tuning algorithm after multiple task reassignments.
Algorithm 2: Region Fine-tuning Algorithm

Input: $M'$, func', Max[unum].
Output: $M''$. [1]
1. initialize: $M' = M$; max_func = max[func']; iter_number = 0;
2. $BV = \{BV_1, ..., BV_k\}$ ← generate the set of connected cells of $K$ partitions;
3. While (max_func $\neq 0$ and iter_number $\leq$ Max[unum]) {
4. start $\leftarrow$ the partition $\text{func}'(\text{start}) = \max(\text{func}')$;
5. dest $\leftarrow$ the partition $\text{func}'(\text{dest}) = \min(\text{func}')$;
6. path = {start, $E_1$, $E_2$, ..., $E_n$, dest} $\leftarrow$ generate the path of task exchange according to the depth first algorithm.
7. For{$i = 1 \rightarrow n + 2$} {
8. $M'(x, y)$ ← select a cell from $BV_{\text{path}(i+1)}$;
9. $M'(x, y) = \text{path}(i)$;
10. $BV_{\text{path}(i+1)} = BV_{\text{path}(i+1)} - M'(x, y)$;
11. $BV_{\text{path}(i)} = BV_{\text{path}(i)} \cup M'(x, y)$;
12. }
13. func'(start) = +;
14. func'(dest) = +;
15. max_func = max[func'];
16. iter_number = iter_number + 1;
17. }
18. Return $M''$;

Computational complexity. Suppose that $\sum_{k=1}^{K} f_{\text{unum}}(k) = M'$, the main operation loop of the region fine-tuning algorithm will perform for $M' / 2$ times. Then, for each loop, the local task reassignment algorithm has a complexity of $O(K^2)$. Therefore, the overall complexity of the region fine-tuning algorithm is $O(M \cdot K^2)$.

Experiment

The computational experiments are carried out on a PC with Intel(R) Core(TM) CPU i7-8565U, 1.8 GHz processor, 16 GB RAM, WIN10. We present three sets of simulations in MATLAB 2018a. The first set shows the validation of the proposed algorithm in different environments. In the second set, the credit-based algorithm was compared with the state-of-art balanced MRTA algorithms.2,23 The third set shows the scalability of the credit-based algorithm for multi-robot systems with unbalanced optimal tasks (the simulation results are available online https://github.com/lilin-ll/Complete-Coverage-Problem-of-Multiple-Robots-with-Different-Velocities.git).

Experimental validation in different environments

For validating the credit-based algorithm, different environments were used. In Figure 7, the top row presents the indoor environment, cave environment, and random environment, whereas the second row shows the respective task allocations obtained by the credit-based algorithm. Four robots with different velocities are assigned to perform the coverage. The second row of Figure 7 shows that the credit-based algorithm could assign coverage tasks according to robots’ velocities in different environments.

Comparison experiments with balanced MRTA algorithms2,23

This section presents a comparison study between the proposed credit-based algorithm and the state-of-art Mofint2 algorithm, and Darp23 algorithm. Mofint2 algorithm was proposed to partition a graph with $k$ tress, and the maximum weight of these trees should be minimized. Darp23 algorithm utilized a cyclic coordinate descent approach to assign balanced tasks to multiple robots.

Simulation setup. To generate comparable results, we adopt a similar simulation setup in Kapoutsis et al.23 A set of 20 mission environments is randomly generated and modeled into binary images according to the approach in Gabriely and Rimon.34 More precisely,

- The size of the mission environment includes two classes: Small ([ROW, COL] = 55 $\times$ 55 cells) and Medium ([ROW, COL] = 100 $\times$ 100 cells).
- The number of obstacles includes two levels: 10% and 20% of the total number of cells in the mission environment. A random function automatically generates the row and column coordinates of every obstacle.
- The number of robots varies from 10, 20, 30, 40, 50, 80 to 100 robots.
- Random robots’ starting positions.

To produce comparable results, we randomly generated 10 sets of robot starting points for each environment and applied the credit-based algorithm in environments with different starting points. As shown in Figure 8, for each combination of the coverage methods and scenario, the maximum number of tasks of all partitions for the credit-based algorithm, Mofint,2 and Darp23 are presented. Additionally, the optimal number of cells [Optimum], which represents the optimal solution, is shown in each scenario. The closer to the Optimum, the better the algorithm is. Mofint and Darp could not produce valid results in some scenarios, and we presented this case with a blank on the graph.

Results. The results for each combination of different scenarios are illustrated in Figure 8. As shown in Figure 8(a), on the premise of 10% obstacle density and a mission environment with a size of 55 $\times$ 55 cells, the output of Mofint increasingly deviates from Optimum with the increased number of robots. In contrast, the deviations of the optimum and the outputs of Darp and the credit-
Figure 7. Task allocations of the credit-based algorithm in different mission environments. Four robots assigned to the coverage task can travel at 0.5 m/s, 0.5 m/s, 1 m/s, and 1 m/s. (a) Indoor environment. (b) Random environment. (c) Cave environment. (d) Task allocation. Four partitions with 52, 52, 104, and 104 cells are obtained. (e) Task allocation. Four partitions with 57, 57, 112, and 112 cells are obtained. (f) Task allocation. Four partitions with 55, 55, 110, and 109 cells are obtained.

Figure 8. The maximum task allocation results compared to Optimum, Mofint, and Darp for a variable number of robots, a variable density of obstacles, and a variable size of the target area. (a) Small, 10%. (b) small, 20%. (c) medium, 10%, and (d) medium, 20%.
based algorithm are slight, which means the credit-based algorithm and Darp outperform Mofint. However, as shown in Figure 8(b), in the scenarios with 20% obstacle density and a target area with a size of $55 \times 55$ cells, Darp could not obtain effective outputs when the number of robots is greater than or equal to 30. In contrast, the output of the credit-based algorithm remains close to the optimum with the increased number of robots, which means the credit-based algorithm outperforms Darp and Mofint. Similar comparison results appear in Figure 8(c) and (d). The deviation of the output of Mofint with optimum becomes bigger and bigger with the increased number of robots, enlarged size of the target area, and the higher obstacle density. At the same time, Darp cannot obtain an effective solution in scenarios where the number of robots exceeds 50. However, no matter how the number of robots, the obstacle density, and the robots' initial positions vary, the results for the credit-based algorithm are all close to the optimum, which means the credit-based algorithm finds the optimal solutions in all scenarios. Figure 9 demonstrate task allocations of the credit-based algorithm in scenarios with small sizes and 10 robots.

**Comparison experiment with unbalanced optimal MRTA**

In this subsection, we study the applicability of the credit-based algorithm in multi-robot systems with unbalanced optimal tasks and discuss its performance for the size of the mission environment, the number of robots, and the density of the obstacles.

**Simulation setup.** A set of sixteen target areas are randomly generated and have been modeled into binary
images according to the grid decomposition approach. More precisely,

- The size of mission environment includes four classes: small ([ROW, COL] = 55 × 55 cells), medium ([ROW, COL] = 100 × 100 cells), large ([ROW, COL] = 200 × 200 cells), and very large ([ROW, COL] = 300 × 300 cells).
- The number of obstacles includes two levels: 10% and 20% of the total number of cells in the mission environment. A random function automatically generates the row and column coordinates of every obstacle.
- The number of robots varies from 10, 30, 40, 50, 100, 150, 200 to 300 robots.
- A random function automatically generates the row and column coordinates of robots' starting positions.
- We generate an integer array by a random function and normalize it to a float array. The optimal task numbers for a multi-robot system are obtained by multiplying the total number of tasks and the normalized array.

The optimal number of tasks for each robot is different. Therefore, to visually demonstrate the performance of the credit-based algorithm, the ratio of the actual number of cells to the optimal number of cells is present. Denote the robot, which has the maximum ratio or minimum ratio, as the specific robot. As shown in Figure 10, the maximum ratio represents the worst case, where the actual number of cells of the specific robot is higher than its optimal number. The minimum ratio represents the worst case, where the actual number of cells of the specific robot is less than its idealized number. Additionally, the idealized ratio 1 [Optimum], which represents the optimal solution, is presented in each scenario. The smaller the deviation from the idealism is, the closer the result is to the optimal solution. The Optimum is only an upper bound since the optimal number of cells may have the decimal, while the actual number of cells can only be an integer.

**Results.** The results for each combination of different scenarios are illustrated in Figure 10. As shown in Figure 10(a), on the premise of 10% obstacle density, mission environment with a size of 55 × 55, the maximum and minimum ratios of the credit-based algorithm are very close to 1 as the number of robots grows, indicating that the output of the credit-based algorithm is close to the optimal. As shown in Figure 10(a) to (d), the maximum and minimum ratios of the credit-based algorithm are close to the optimum as the size of the mission environment increases from 55 × 55 to 300 × 300, and the number of robots increases from 10 to 300, indicating that the credit-based algorithm can find the solution close to the optimum in all cases when the tasks are unbalanced.

It should be noted that there are slight differences between the Optimum and the minimum/maximum ratio. The quantization operation introduces these differences in the transformation of the normalized float array to the
optimal numbers of robots, and these differences are no more than one cell. For example, Figure 11(a) is the experiment scenario with a mission environment size of small and 10 robots, while Figure 11(b) is the corresponding task allocation. In Figure 11(b), the minimum ratio is 0.989732, whereas the maximum ratio is 1.002574. For the maximum ratio, the optimal number of tasks for the robot with the maximum ratio is 233.399222, while the actual number of tasks allocated is 234. For the minimum ratio, the optimal number of tasks for the robot with the smallest ratio is 77.798833, while the assigned tasks are 77. Therefore, the ratio of optimal to actual is 0.989732.

Conclusion
In this article, the unbalanced MRTA problem of area coverage is investigated. We formulate this problem into a MOP and propose a novel credit model to decompose the MOP into a series of SOPs. Then, the credit-based task allocation algorithm based on the credit model is presented to find the combined optimal solution of the SOPs. The credit-based algorithm includes the cyclic region growth algorithm and the region fine-tuning algorithm. The former generates multiple partitions of the task set by a multi-constrained region growing strategy, and the latter adopts a local task exchange strategy based on a search tree to eliminate the local optimum and optimum differences. Simulation experiments show that the credit-based algorithm can obtain the optimal solution with the increased number of robots and enlarged size of the mission environment in both balanced assignment and unbalanced assignment scenarios.

However, the credit-based algorithm still suffers from some limitations. First, the credit-based algorithm applies to the mission environment abstracted as a binary node-weight graph but not the general graph, limiting its scope of application. Second, the credit-based task allocation algorithm only addresses the task allocation problem and does not consider the path planning problem. In the future, we will further study the CCP method suitable for general graphs and practical coverage path planning.

Acknowledgments
The authors are grateful to the colleagues who are not on the author list for their valuable suggestions.

Authors’ contributions
Conceptualization: Lin Li, Dianxi Shi, Hengzhu Liu, and Songchang Jin; Methodology: Lin Li, and Dianxi Shi; Formal analysis and investigation: Lin Li and Xing Zhou; Writing—original draft preparation: Lin Li; and Writing — review and editing: Songchang Jin, Chao Xue, Ying Kang, Xiaoxiao Yu, and Dianxi Shi.

Authors Note
The author Chao Xue is now affiliated with Tianjin Artificial Intelligence Innovation Center (TAIIC), Tianjin, China.

Code availability
The data and code that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Natural Science Foundation of China (Grant No. 91948303).

ORCID iD
Lin Li https://orcid.org/0000-0001-8645-0392

References
1. Zhou X, Wang H, and Ding B. How many robots are enough: a multi-objective genetic algorithm for the single-objective time-limited complete coverage problem. In: 2018 IEEE international conference on robotics and automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018, pp. 2380–2387.
2. Kan X, Teng H, and Karydis K. Multi-robot field exploration in Hex-decomposed environments for Dubins vehicles. In: 2019 IEEE international conference on robotics and biomimetics (ROBIO), Dali, China, 6–8 December 2019, pp. 449–455. IEEE.
3. Kan X, Teng H, and Karydis K. Online exploration and coverage planning in unknown obstacle-cluttered environments. IEEE Robot Automat Lett 2020; 5(4): 9569–9576.
4. Wu C, Dai C, Gong X, et al. Energy-efficient coverage path planning for general terrain surfaces. IEEE Robot Automat Lett 2019; 4(3): 2584–2591.
5. Zhang P, Wang H, Ding B, et al. Cloud-based framework for scalable and real-time multi-robot slam. In: 2018 IEEE
Li et al.

international conference on web services (ICWS), San Francisco, CA, USA, 2–7 July 2018, pp. 147–154. IEEE.

6. Palacios-Gasós JM, Montijano E, Sagués C, et al. Distributed coverage estimation and control for multirobot persistent tasks. IEEE Trans Robot 2016; 32(6): 1444–1460.

7. Karapetyan N, Braude A, Moulton J, et al. Riverine coverage with an autonomous surface vehicle over known environments. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS), Macau, China, 3–8 November 2019, pp. 3098–3104. IEEE.

8. Karapetyan N, Moulton J, and Rekleitis I. Dynamic autonomous surface vehicle control and applications in environmental monitoring. In: OCEANS 2019 MTS/IEEE SEATTLE, Seattle, WA, USA, 27–31 October 2019, pp. 1–6. IEEE.

9. Wu Y, Zhang B, Yi X, et al. Communication-motion planning for wireless relay-assisted multi-robot system. IEEE Wireless Commun Lett 2016; 5(6): 568–571.

10. Jin X and Ray A. Navigation of autonomous vehicles for spill cleaning in dynamic and uncertain environments. Int J Control 2014; 87(4): 787–801.

11. Karapetyan N, Moulton J, Lewis JS, et al. Multi-robot Dubins coverage with autonomous surface vehicles. In: 2018 IEEE international conference on robotics and automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018, pp. 2373–2379. IEEE.

12. Duckett T, Pearson S, Blackmore S, et al. Agricultural robotics: the future of robotic agriculture. arXiv preprint arXiv:1806.06762, 2018; arXiv:1806.06762. [Online]. Available: https://arxiv.org/abs/1806.06762.

13. Lottes P, Khanna R, Pfeifer J, et al. UAV-based crop and weed classification for smart farming. In: 2017 IEEE international conference on robotics and automation (ICRA), Singapore, 29 May–3 June 2017. IEEE.

14. Karapetyan N, Benson K, McKinney C, et al. Efficient multirobot coverage of a known environment. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), Vancouver, 24–27 September 2017. IEEE.

15. Galceran E and Carreras M. A survey on coverage path planning for robotics. Robot Auton Syst 2013; 61(12): 1258–1276.

16. Pananjady A, Bagaria VK, and Vaze R. The online disjoint set cover problem and its applications. In: 2015 IEEE conference on computer communications, Hong Kong, China, 26 April–1 May 2015. IEEE.

17. Tan CS, Mohd-Mokhtar R, and Arshad MR. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. IEEE Access 2021; 9: 119310–119342.

18. Cabreira TM, Brisolara LB, and Ferreira PR, Jr. Survey on coverage path planning with unmanned aerial vehicles. Drones 2019; 3(1): 4.

19. Aggarwal S and Kumar N. Path planning techniques for unmanned aerial vehicles: a review, solutions, and challenges. Computer Commun 2020; 149: 270–299.

20. Hazon N and Kaminka GA. Redundancy, efficiency and robustness in multi-robot coverage. In: Proceedings of the 2005 IEEE international conference on robotics and automation, Barcelona, Spain, 18–22 April 2006. IEEE.

21. Zheng X, Jain S, Koenig S, et al. Multi-robot forest coverage. Proc Iros 26(6): 3852–3857, 2005.

22. Zhou X, Wang H, Ding B, et al. Balanced connected task allocations for multi-robot systems: an exact flow-based integer program and an approximate tree-based genetic algorithm. Expert Syst Appl 2019; 116: 10–20.

23. Kapoutsis AC, Chatzichristofis SA, and Kosmatopoulos EB. Darp: divide areas algorithm for optimal multi-robot coverage path planning. J Intell Robot Syst 2017; 86(3–4): 663–680.

24. Han Y, Li D, Chen J, et al. A multi-robots task allocation algorithm based on relevance and ability with group collaboration. Int J Intell Eng Syst 2010; 3(2): 33–41.

25. Nunes E, Manner M, Mitiche H, et al. A taxonomy for task allocation problems with temporal and ordering constraints. Robot Auton Syst 2016; 90: 55–70.

26. Dyer ME and Frieze AM. On the complexity of partitioning graphs into connected subgraphs. Discrete Appl Math 1985; 10(2): 139–153.

27. Matić D. A mixed integer linear programming model and variable neighborhood search for maximally balanced connected partition problem. Appl Math Computat 2014; 237: 85–97.

28. Garcia P, Caamaño P, Duro RJ, et al. Scalable task assignment for heterogeneous multi-robot teams. Int J Adv Robot Syst 2013; 10: 105.

29. ElGibreen H and Youcef-Toumi K. Dynamic task allocation in an uncertain environment with heterogeneous multi-agents. Auton Robots 2019; 43(7): 1639–1664.

30. Liu L and Shell DA. Large-scale multi-robot task allocation via dynamic partitioning and distribution. Auton Robots 2012; 33(3): 291–307.

31. Yu X, Jin S, Shi D, et al. Balanced multi-region coverage path planning for unmanned aerial vehicles. In: 2020 IEEE international conference on systems, man, and cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020, pp. 3499–3506. IEEE.

32. Hussein A and Khamis A. Market-based approach to multi-robot task allocation. In: 2013 international conference on individual and collective behaviors in robotics (ICBR), Sousse, Tunisia, 15–17 December 2013, pp. 69–74. IEEE.

33. Choset H. Coverage of known spaces: the boussphedon cellular decomposition. Auton Robots 2000; 9: 247–253.

34. Gabriely Y and Rimon E. Spanning-free based coverage of continuous areas by a mobile robot. Ann Math Artif Intell 2001; 31(1): 77–98.