Robotica (2026), **44**, pp. 1843–1861
doi:10.1017/S0263574726103592

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Divide Areas based on Area Growing via Wavefront Propagation: a near-optimal multi-robot coverage path planning algorithm for large-scale maps with low computational cost

Renjie Guo†, Yiyi Zhang† and Peng Guo

School of Electronic Information and Communications, Huazhong University of Science and Technology, China
Corresponding author: Peng Guo; Email: guopeng@hust.edu.cn

Received: 4 September 2025 UTC; **Revised:** 2 May 2026 UTC; **Accepted:** 15 May 2026 UTC;
First published online: 28 July 2026 UTC

Keywords: multi-robot coverage path planning; coverage time; computational cost

Abstract

Multi-robot coverage path planning (MCPP) is a typical problem in multi-robot applications. Its goal is to decompose and assign coverage tasks, enabling mobile robots to collaboratively perform complete coverage of the map within a short time (referred to as coverage time). The coverage time of MCPP is generally determined by the path lengths of the robots and the number of turns on the paths. Most existing studies focus on only one of the two metrics, and few address the simultaneous optimization of both aspects. In this letter, we propose the Divide Areas based on area Growing via Wavefront Propagation (DAWG) algorithm to solve this problem through a staged co-optimization approach. Specifically, DAWG combines area growing and wavefront propagation to divide the map into equally sized subareas, ensuring equal path lengths. Meanwhile, DAWG incorporates a greedy strategy that reduces computation time by 81% while maintaining turn counts comparable to TMSTC*, a state-of-the-art (SOTA) method that minimizes path turns. Experiments on dozens of large-scale map instances demonstrate that DAWG achieves superior performance in both computation time and path quality compared to existing methods.

1. Introduction

Multi-robot path planning is a fundamental problem in multi-mobile robot systems. It aims to enable multiple robots to perform cooperative navigation and task execution in complex environments. In recent years, with the rapid development of unmanned systems, extensive research has been conducted in this area. Significant progress has been achieved, particularly in point-to-point path planning under dynamic and uncertain environments.

For example, in perception-limited environments, adaptive collision avoidance strategies have been developed for unmanned surface vehicles (USVs) by incorporating dynamic priority mechanisms and multi-agent reinforcement learning [1]. In addition, some studies adopt distributed reinforcement learning frameworks to jointly optimize target assignment and path planning in multi-USV systems, thereby improving task completion efficiency and system safety [2]. Furthermore, for collaborative tasks in complex environments, recent work has considered energy constraints and heterogeneous system characteristics, leading to multi-objective optimization approaches for cooperative path planning [3].

However, most of these studies focus on point-to-point path planning. Their goal is to guide robots to specific target locations rather than to systematically traverse the entire environment. In this context, multi-robot coverage path planning (MCPP) has attracted increasing attention.

† Equal contributions.

© The Author(s), 2026. Published by Cambridge University Press

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

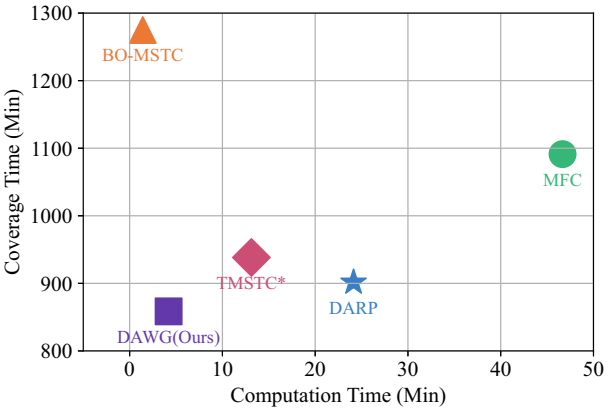


Figure 1. *The performance comparison of our proposed DAWG algorithm with four existing algorithms: TMSTC* [17], DARP (Divide Areas based on Robots Initial Positions) [10], BO-MSTC (Backtracking-Multi-robot Spanning Tree Coverage) [14], and MFC (Multi-robot Forest Coverage) [15], where all methods are evaluated on the same* 512×512 *map.*

MCPP is a key application in multi-mobile robot systems, aiming to enable multiple robots to collaborate in avoiding obstacles while minimizing the time required to cover the free area in a map [4]. MCPP is widely applied in various indoor and outdoor scenarios, such as robotic vacuum cleaners [5], drone search and rescue operations [6], and agricultural harvesters [7]. The spanning tree coverage (STC) [8] algorithm, as a classical single-robot coverage path planning method, has been widely extended to MCPP research.

The core performance metric of MCPP is coverage time, defined as the time required by the last robot to complete the coverage task. Coverage time depends on both the path length and the number of turns in the robot paths [9]. While most studies target either metric independently, few address the joint optimization of both.

MCPP algorithms that aim to reduce path length can be classified into two categories based on whether size division is performed: size division-based methods and non-size division-based methods. The first category [10, 11, 12, 13] divides the free area into several equal-sized subareas to ensure the minimal path length for each robot. The second category constructs a spanning tree to obtain a closed path, which is then assigned to multiple robots [14, 15, 16]. Due to overlapping coverage in the paths, the path length of the second category is generally longer than that of the first category. However, neither of these two categories considers the impact of robot turns on coverage time.

TMSTC* (Turn-minimizing Multi-robot Spanning Tree Coverage*) [17] exhibits state-of-the-art (SOTA) performance compared to other algorithms by minimizing path turns. However, since the TMSTC* algorithm does not perform size division, its path length is not minimized.

In addition to coverage time, another key metric for MCPP is computation time, especially in large-scale maps. However, the computation time required by the aforementioned methods on large-scale maps is prohibitively high for practical applications.

To address above issues, we propose the DAWG (Divide Areas based on area Growing [18] via wavefront Propagation [19]) algorithm, which can equally divides free area and minimizes path turn counts. Compared to existing methods, our approach not only achieves shorter coverage times but also reduces computation time, as shown in Figure 1. Our contributions are summarized as follows:

- We draw on the ideas of area growing and wavefront propagation to achieve an equal division of free area while ensuring the connectivity of each subarea. As a result, the path length is minimized, and computation time is kept low. We also propose a greedy method to minimize the number of bricks, achieving only a 3% increase in turn counts compared to TMSTC*, while reducing computation time by 81%. Additionally, by adjusting the distance metric used in wavefront propagation, we optimized the shapes of the subareas, further reducing turn counts.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

- The proposed DAWG algorithm achieves the minimum path length while ensuring near-optimal turn counts. The overall coverage time of this algorithm outperforms SOTA algorithms, while its computation time remains at a low level, as shown in Figure 1.
- We observe that existing MCPP algorithms exhibit unstable computation times (e.g., computation times exceeding a specified limit, defined in this letter as no solution) when faced with large-scale map diversity. In contrast, the proposed DAWG algorithm always finds a feasible solution for all 24 different instances.

2. Related work

The core idea of MCPP is to divide and allocate the coverage tasks of the entire map among multiple robots, enabling them to collaboratively complete the coverage within the shortest possible time [20]. The coverage time of MCPP is determined by the last robot to finish and depends on both the path length and the number of turns.

The STC [8] is a classical single-robot CPP algorithm, and many existing MCPP methods have been extended based on it. These methods either focus solely on optimizing path length or exclusively on minimizing the number of turns, resulting in a notable lack of research that comprehensively addresses both aspects.

To reduce path length, MCPP algorithms can be categorized into two types based on whether size balancing is considered: size division-based methods and non-size division-based methods. Among the first category, DARP (Divide Areas based on Robots Initial Positions) [10] (which partitions the area based on the initial positions of the robots) assigns each STC node to the nearest robot and iteratively adjusts the assignment to achieve equal-sized subareas. However, subsequent studies based on DARP [11, 12] have shown that in maps with complex obstacle distributions, when the initial positions of robots are concentrated, the DARP may fail to ensure the connectivity of each robot's subarea. Forcing subarea connectivity in such cases can result in redundant coverage, thereby increasing the path length. Among the second category, BO-MSTC (Backtracking-Multi-robot Spanning Tree Coverage) [14] algorithm treats the single-robot coverage path planned by the STC algorithm as a ring and divides the ring based on the initial positions of the robots to obtain coverage paths for multiple robots. Zheng et al. introduced MFC (Multi-robot Forest Coverage) [15], whose core idea is to construct subtrees with similar weights and assign them to multiple robots. Tang et al. proposed MSTC* [16], which uses a greedy strategy to address the imbalance in the ring division in BO-MSTC. However, both categories of studies have overlooked the minimization of the number of turns in the path.

To reduce the number of turns, Vandermeulen et al. proposed a checkerboard-style decomposition of the free space into multiple rectangular cells and applied an iterative heuristic to adjust their orientations [21]. However, this method is limited in applicability as it cannot handle non-rectilinear environments effectively. Ramesh et al. introduced the OARP (Optimal Axis-Parallel Rank Partitioning) method, which minimizes the number of axis-parallel coverage lines by formulating the problem as a linear program (LP) [22]. However, the LP-based method is computationally expensive for large-scale environments, which limits its practicality. TMSTC* [17] transforms the problem of minimizing turns into dividing the free area into the fewest number of bricks and then solving the maximum independent set of a bipartite graph to determine the minimum turns. Although TMSTC* achieves SOTA performance, its path length remains suboptimal.

In recent years, research on large-scale MCPP has begun to move beyond traditional small-scale experimental settings. For example, refs. [23] and [24] investigate large-scale grid-based coverage strategies from the perspectives of path conflict resolution and local search optimization, respectively. However, most existing MCPP methods are still primarily evaluated in small-scale scenarios such as household cleaning or indoor service environments. This creates a clear gap with real-world applications, including large public-space cleaning, warehouse logistics, and city-scale UAV inspection. In this context, it is necessary to systematically evaluate the computational efficiency and scalability of different algorithms on large-scale maps to better reflect their practical performance.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

Traditional grid-based methods usually assume that the grid size matches the robot size. A grid cell is considered covered when the robot center passes through its center. However, in practical applications, grid cells may be smaller than the robot body. This makes it difficult for robots to navigate narrow areas, which reduces overall coverage performance. A straightforward solution is to reduce the robot size. However, this often leads to lower coverage efficiency.

To balance coverage completeness and efficiency, Samarakoon et al. [25] proposed a size-adaptive robot model. The robot can dynamically adjust its size according to environmental constraints. It maintains a larger size in open areas to improve efficiency, while shrinking in narrow regions to ensure accessibility. In addition, Samarakoon et al. [26] proposed an online path planning method based on a Glasius neural network for such adaptive robots. This method enables real-time motion adaptation in dynamic environments. However, these methods are still mostly at the prototype stage. Their applicability in real large-scale systems has not yet been fully validated.

Recently, energy-aware and terrain-aware coverage path planning has received increasing attention. In these methods, robot motion is not only evaluated by geometric metrics such as path length but also by energy-related factors and environmental characteristics. EA-OSPGB (Energy Activity-Obstacle and Start Point Guidance and Backtracking mechanisms) [27] introduces an energy activity (EA) function to guide multi-robot coverage in unknown environments. The algorithm constructs a local raster activity value function to evaluate coverage decisions and incorporates a terrain-based backtracking mechanism to improve exploration efficiency. This mechanism enables robots to escape so-called "dead zones" during online exploration and reduces redundant backtracking paths caused by environmental uncertainty. Experimental results demonstrate that considering energy activity and terrain factors can effectively reduce both path length and turning cost in dynamic environments.

However, EA-OSPGB is fundamentally an online reactive coverage framework. Its decision-making process relies on real-time environmental feedback and energy activity evaluation, which leads to dynamic path adaptation during execution. While this improves adaptability in unknown environments, it may also introduce additional computational overhead and complex coordination behavior in large-scale scenarios.

In contrast, the method proposed in this paper, DAWG, does not rely on energy-driven reactive decision-making or terrain-dependent backtracking mechanisms. Instead, it adopts an offline deterministic decomposition strategy, where the coverage problem is first decomposed into balanced and connected subareas, followed by STC-based path construction within each subarea. This structural design allows DAWG to decouple partitioning and path optimization, enabling efficient global planning with significantly reduced computational cost.

Therefore, this letter adopts equal-size division to minimize path length and addresses the long computation time of TMSTC* on large-scale maps while maintaining a similar number of turns. Next, the algorithm design is presented in Section 4.

3. Preliminaries & motivation

3.1. MCPP problem definition

As illustrated in Figure 2(a), in the MCPP algorithm based on STC, assume that the robot's coverage tool has a width of d, and the map is discretized into square grid cells with a width of $2d$. Each grid cell, composed of four small square grid cells with a width of d, is either completely free or entirely occupied by obstacles. This dual-scale grid division enables the construction of two graphs: the coverage graph G and the spanning tree graph H. Specifically, the center of each small grid cell is taken as a coverage node, and edges are established between adjacent coverage nodes to form the coverage graph G. Similarly, the center of each grid cell is taken as a spanning tree node, and edges between adjacent spanning tree nodes form the spanning tree graph H.

As shown in Figure 2(c), by executing the STC algorithm on the spanning tree graph H, the coverage paths Π_i for each robot can be obtained, where each path is generated by traversing around the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

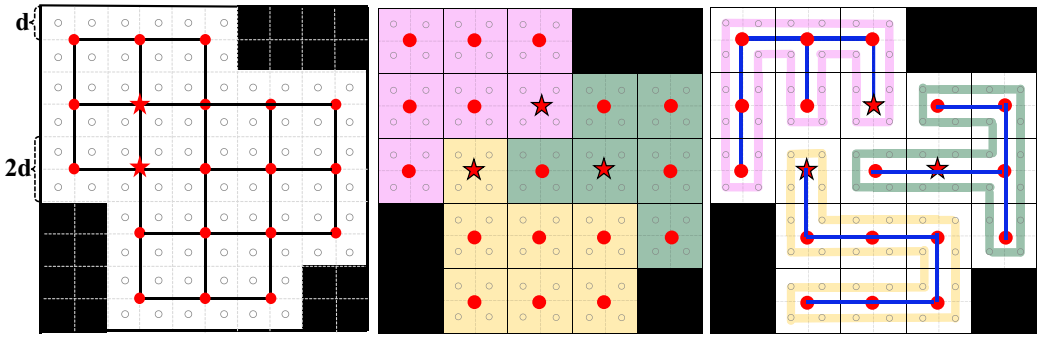


Figure 2. *Multi-robot coverage path planning using our method. (a) Coverage graph G and spanning graph H. (b) Division of the free area among robots, with initial positions marked by stars. (c) Coverage paths of the robots shown in different colors.*

corresponding spanning tree (hereafter referred to as the STC path). Therefore, the optimization goal of MCPP is to find a set of coverage paths such that the maximum coverage time is minimized:

$$\arg\min_{\{\Pi_i\}} \left(\max_{1 \le i \le K} \left(T_{\text{Cover}_i} \right) \right) \tag{1}$$

where T_{Cover_i} is the coverage time of Π_i.

3.2. MCPP problem analysis

In MCPP, the coverage time is influenced by both the path length and the number of turns. To reduce the coverage time, this work simultaneously optimizes both factors. On the one hand, the STC path Π_i of robot i has a length of $S_i \times 4d$, where S_i denotes the number of spanning tree nodes in the subarea assigned to robot i, which can also be regarded as the number of grid cells in that subarea. According to Eq. (1), dividing the free space into subareas with equal size helps minimize the maximum path length, thereby reducing the overall coverage time. On the other hand, further reducing the number of turns in each subarea can also effectively shorten the time required for individual robots to complete their tasks. Based on this analysis, the following two subsections present the strategy for equal-size subarea division and the construction of minimal-turn spanning trees, respectively.

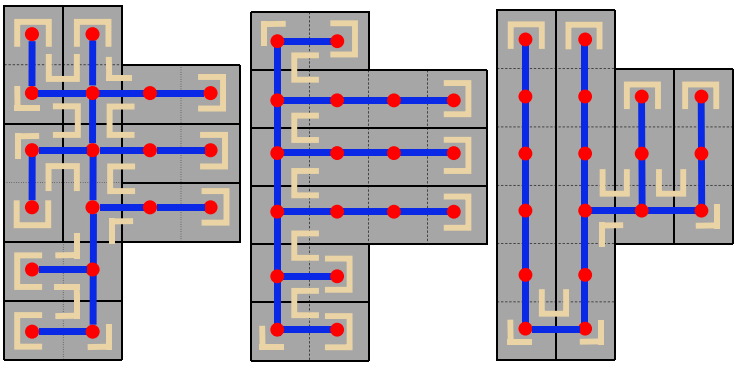
3.2.1. Minimization of STC path length

Given the fixed number of spanning tree nodes in the free space, it is equally divided into K subareas, where K denotes the number of robots. Each subarea has an approximate size S_i. The STC algorithm is then applied within each subarea, which helps balance the path lengths of all robots. Additionally, these K subareas should remain connected, and each robot's initial position X_i is located within its corresponding subarea A_i. To avoid time waste caused by overlapping coverage, these subareas should ideally be non-overlapping. Therefore, the optimization objectives and constraints for size division are as follows:

$$\begin{aligned} &\min \ \max\left(S_1, S_2, \ldots, S_K\right) \\ &\text{s.t.} \quad A_i \cap A_j = \emptyset, \quad \forall\, i \neq j, \\ &\qquad \text{Connectivity}(A_i) = 1, \quad \forall\, i \in \{1, 2, \ldots, K\}, \\ &\qquad \mathbf{X}_i \in A_i, \quad \forall\, i \in \{1, 2, \ldots, K\}. \end{aligned} \tag{2}$$

Eq. (2), specifically the second constraint, ensures that each subarea A_i assigned to a robot satisfies the following connectivity requirement. Since the robot is only allowed to move in four directions (up, down, left, and right), the 4-neighborhood is adopted as the basis for defining connectivity. Specifically, the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592



(a) 42 turns (b) 24 turns (c) 18 turns

Figure 3. *Comparison of turn counts (highlighted in beige, each representing a 90-degree turn) across different spanning tree constructions. Dashed lines divide the free space into grid cells, while solid lines divide it into bricks. Three spanning tree construction strategies: random, greedy, and TMSTC*, correspond to 9 bricks in (a), 6 bricks in (b), and 4 bricks in (c).*

grid cells in A_i are regarded as nodes in a graph. If two cells satisfy the 4-neighborhood relationship, an edge is defined between them. Based on this definition, A_i can be decomposed into several connected subsets under 4-neighborhood connectivity, that is, 4-connected components. The subarea A_i is considered connected if and only if the number of 4-connected components is equal to one. Otherwise, it is regarded as disconnected. Therefore, the connectivity constraint can be formulated as:

$$\text{Connectivity}(A_i) = \begin{cases} 1, & \text{if } N_{\text{comp}}(A_i) = 1 \\ 0, & \text{otherwise} \end{cases}$$

where $N_{\text{comp}}(A_i)$ denotes the number of 4-connected components in subarea A_i.

As a representative study on size division, the DARP algorithm transforms the problem of equally dividing the area into a clustering problem. However, the DARP algorithm exhibits three main limitations: it disregards the number of turns in the path while optimizing coverage time; it incurs significant computation time in large-scale maps; and it is sensitive to the initial distribution of robot positions, which can cause the algorithm to enter an infinite loop.

3.2.2. Minimization of STC path turns

The concept of brick was proposed in ref. [17], where it is defined as a regular rectangular strip formed by merging multiple grid cells along either the horizontal or vertical direction, with a fixed width or height of $2D$. A brick can alternatively be regarded as a segment of the spanning tree T, consisting of multiple nodes aligned in the same direction. Since the STC path only turns at the two endpoints of each brick and at the junctions between different bricks, a smaller number of bricks within the spanning tree T generally leads to fewer turns in the corresponding STC path. As shown in Figure 3, for a given spanning tree graph H, different spanning tree construction strategies have a significant impact on the number of turns in the robot's coverage path. According to ref. [17], the problem of minimizing the number of turns can be equivalently transformed into minimizing the number of bricks B_i contained in the spanning tree T:

$$\begin{aligned} &\min \|B_1, B_2, \ldots, B_m\| \\ &\text{s.t.} \quad B_1 \cup B_2 \cup \cdots \cup B_m = H, \\ &\qquad B_i \cap B_j = \emptyset, \quad \forall\, i, j \in \{1, 2, \ldots, m\},\ i \neq j. \end{aligned} \tag{3}$$

TMSTC* algorithm further transforms the problem of minimizing the number of bricks into solving the maximum independent set problem in a bipartite graph. Although TMSTC* achieves the best results

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

so far in minimizing the number of turns, it does not divide the area, leaving room for improvement in reducing path length. Furthermore, the computation time of TMSTC* in large-scale maps also limits its practicality.

3.3. Motivation

In practical applications, robots cannot maintain a constant speed throughout the entire process. This is because turning introduces additional time cost. To simplify the problem, the following assumptions are made for the robot's coverage process. First, the robot decelerates and accelerates with constant acceleration before and after turning. Second, when changing direction, the robot must reduce its linear velocity to zero and then perform a 90° in-place rotation with a constant angular velocity. Under these assumptions, as shown in Eq. (5), both the path length and the number of turns should be optimized to reduce the overall coverage time.

Since the simultaneous optimization of path length and the number of turns is an NP-hard problem, this letter simplifies the problem into two sequential steps: generating equal-sized subareas, followed by constructing a minimal-turn spanning tree within each subarea. It should be noted that this simplification yields an approximate solution rather than the theoretical global optimum. Due to the high computational cost of the TMSTC* algorithm in the spanning tree construction phase, we propose a greedy strategy that significantly reduces the computation time while maintaining a comparable number of turns.

To improve readability and ensure consistency of notation throughout the paper, we summarize the main symbols and variables used in this work in Table I. The table provides unified definitions for all key notations appearing in the main text, algorithms, and mathematical formulations. Symbols that are not explicitly defined in the main text are intentionally excluded to avoid ambiguity.

4. Our algorithm

In this section, we propose the DAWG algorithm, which aims to optimize the coverage time by simultaneously reducing the path length and the number of turns, while achieving a shorter computation time compared to existing SOTA algorithms.

4.1. Connectivity-assured equal-size division

The size division process of the DARP algorithm assigns each grid cell to the nearest robot based on distance. However, since the division of each grid cell is done independently, DARP may fail to ensure the connectivity of the subarea assigned to robot i (Figure 4(a)). Differing from DARP, we draw inspiration from the area growing method in image segmentation [18] and the wavefront propagation method [19] and gradually expand the boundaries of subareas through area growing to naturally maintain the connectivity of subareas during the division process (Figure 4(a)). Algorithm 1 summarizes the pseudocode of the proposed algorithm.

4.1.1. The process of grid cell transfer between adjacent subareas

Figure 5 illustrates the process of grid cell transfer between two adjacent subareas, aiming to equalize their sizes. Algorithm 1.

As shown in Figure 5(a), based on the Manhattan distance metric, the initial positions of the robots, denoted as $X_1, X_2, \ldots, X_K$, are used as seed points to perform a Voronoi partition over the full map. Each grid cell g is assigned to the subarea corresponding to the robot i for which the Manhattan distance $\mathrm{Dist}(X_i, g)$ is the smallest, where $\mathrm{Dist}(X_i, g)$ denotes the Manhattan distance between the robot's initial position X_i and the grid cell g. Initially, the sizes of subareas A_0 and A_1 are 18 and 75, respectively.

To balance the sizes among subareas, we first identify the subarea with the largest size, denoted as $A_{\max}$, and the one with the smallest size, denoted as $A_{\min}$. Before introducing the transfer process, we define the adjacency relationship between subareas. Two subareas A_i and A_j $(i \neq j)$ are defined as adjacent if there exists at least one pair of grid cells $g \in A_i$ and $g' \in A_j$ that satisfy the 4-neighborhood relationship.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

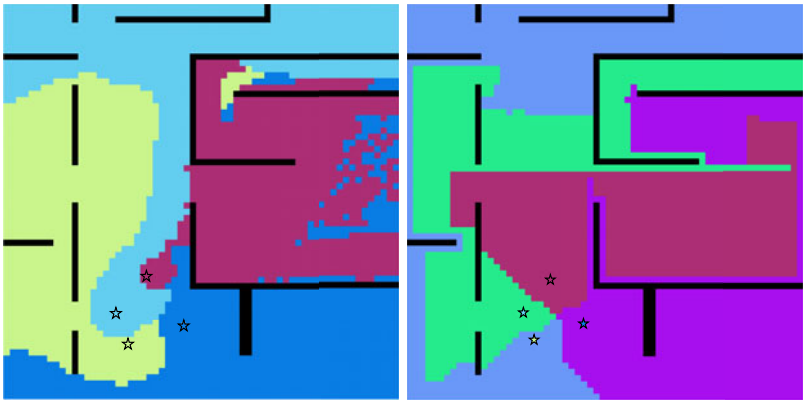
Table I. *Summary of symbols and variable definitions used in this paper.*

Symbol	Definition
A_i	Subarea assigned to robot i
S_i	Number of grid cells in subarea A_i
S	Total number of free grid cells
K	Number of robots
X_i	Initial position of robot i
g	Grid cell
Dist(X_i, g)	Manhattan distance between X_i and grid cell g
$N_{\text{comp}}(A_i)$	Number of 4-connected components in A_i
Connectivity(A_i)	Connectivity indicator of subarea A_i
G	Coverage graph
H	Spanning tree graph
Π_i	Coverage path of robot i
T_{Cover_i}	Coverage time of robot i
B	Brick (rectangular strip in spanning tree)
B_i	i-th brick
m	Number of bricks/iteration index
T	Spanning tree
F	Node set of spanning graph H
V_B	Node and edge set of brick B
$A_{\max}$	Largest subarea
$A_{\min}$	Smallest subarea
P	Grid cell transfer path between subareas
$S_{\min}$	Minimum subarea size among all S_i
T_h	Threshold for size balancing
$w(n_1, n_2)$	Cost change of turns after adding edge (n_1, n_2)
$f(n)$	Number of turns associated with node n
$f^+(n)$	Number of turns after adding an edge
n	Node in spanning tree (corresponding to grid cell g)
γ_j	Turning point in path
t_j	Motion time between two turning points
$d_{\gamma_j,\gamma_{j+1}}$	Euclidean distance between two turning points
$v_{\max}$	Maximum robot velocity
a	Robot acceleration
ω	Angular velocity during turning

In other words, the two subareas share at least one pair of 4-neighboring boundary cells. Based on this definition, a valid grid cell transfer path is constructed on the adjacency graph of subareas using the function $P_{A_{\max}\to A_{\min}} = \texttt{get_transfer_path}(A_{\max,A_{\min}})$. For notational simplicity, we omit the dependency on $(A_{\max}, A_{\min})$ and use P to denote the corresponding transfer path in the remainder of this paper. Since grid cells can only be transferred directly between adjacent subareas, if $A_{\max}$ and $A_{\min}$ are not adjacent, the function returns a sequence of intermediate subareas that forms a connected path between them.

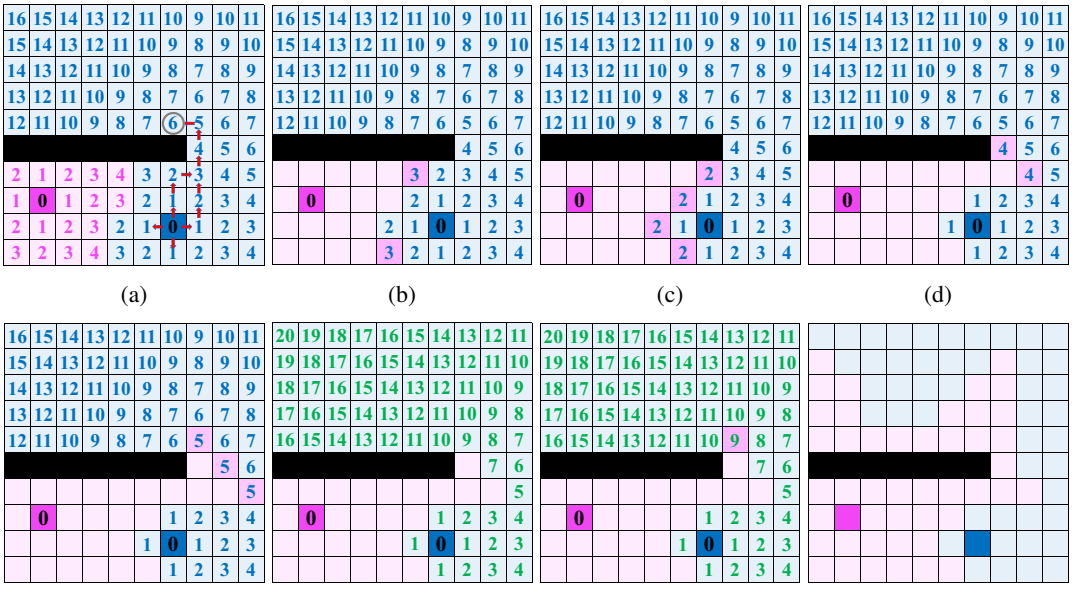
For example, let $P = [1, 2, 3, 4]$, where subareas 1 and 4 correspond to $A_{\max}$ and $A_{\min}$, respectively. In this case, the transfer from subarea 1 to subarea 4 must pass through subareas 2 and 3, and any two consecutive subareas in P are adjacent according to the above definition. Therefore, the grid cell transfer path is A_1 (sender) $\to$ A_0 (receiver), and the number of transferred grid cells is $\left\lfloor \frac{S}{K} \right\rfloor - S_0 = 28$. Here, S denotes the total number of free grid cells (i.e., excluding obstacle cells), where $S = \sum_{i=1}^{K} S_i$, and $i \in \{1, \ldots, K\}$. Therefore, $\lfloor S/K \rfloor$ represents the discretized average number of free grid cells assigned to each subarea, corresponding to the desired balanced coverage load.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592



(a) (b)

Figure 4. *Comparison of subarea connectivity between the DARP algorithm and the proposed DAWG algorithm. (a) Subareas are not connected. (b) Subareas are connected.*



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5. *The process of grid cell transfer between adjacent subareas of two robots until the free area is equally divided.*

As shown in Figure 5(b), the part of the sender area adjacent to the receiver contains four grid cells. If these grid cells are all assigned to the receiver at once, it may lead to two issues: the sender area becoming disconnected, or the sender's initial position being assigned to the receiver. To avoid these problems, the appropriate approach is to gradually assign the grid cells farthest from the sender's initial position to the receiver. This approach is based on the observation that, when the sender area expands outward, the farthest grid cells are the last ones added; conversely, when the sender area shrinks, the farthest grid cells should be the first to be removed.

As shown in Figure 5(c), during the first iteration, two grid cells were successfully transferred from the sender to the receiver.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

Algorithm 1. Equal-size division

Input: Number of robots K, subareas A_1 to A_K, total number of grid cells S, threshold T_h
Output: Balanced subareas A_1 to A_K
while $\exists i \in \{1, \ldots, K\}$ *such that* $\left|\frac{S_i - \lfloor S/K \rfloor}{\lfloor S/K \rfloor}\right| > T_h$ **do**
    $A_{\max} = \arg\max_{i \in \{1,\ldots,K\}} S_i, \quad A_{\min} = \arg\min_{i \in \{1,\ldots,K\}} S_i$
    $S_{\min} = \min(S_1, \ldots, S_K)$
    $P = \text{get_transfer_path}(A_{\max}, A_{\min})$
    for $j \leftarrow 0$ **to** $|P| - 2$ **do**
        $temp_S, temp_R = P[j], P[j+1]$
        $T_n = 0$, $flag$ = False
        while $T_n < \lfloor S/K \rfloor - S_{min}$ **do**
            $C_0 = \{g \in temp_S \mid g \text{ is adjacent to } temp_R\}$
            $C = \{g \in C_0 \mid Dist(X, g) \text{ is maximum}\}$
            if *temp_S is connected* **then**
                $T_n = T_n + \text{len}(C)$
                Transfer C from $temp_S$ to $temp_R$
            else
                Fallback the transfer of C
                if *not* $flag$ **then**
                    $flag$ = True
                    Update $Dist$ for $temp_S$
                else
                    Fix narrow channel in $temp_S$
                    $flag$ = False
return A_1 to A_K

It is important to note that, in theory, the wavefront distance within the sender area should be updated in each iteration (i.e., grid cells outside the sender area are treated as obstacles); otherwise, the situation shown in Figure 5(e) may occur, where the sender becomes disconnected from the receiver. However, repeatedly updating the wavefront distance incurs a significant computational cost, especially in large-scale maps. To reduce the computational time, as shown in Figure 5(b)–(e), the wavefront distance is not updated after each iteration.

When the sender area becomes disconnected, as shown in Figure 5(e)–(g), the division process rolls back to the previous round and updates the wavefront distance. As shown in Figure 5(h), the division process continues while the imbalance among subareas remains significant. Specifically, the algorithm continues if the relative deviation of any subarea size from the average value exceeds a predefined threshold T_h, as described in Step 1 of Algorithm 1. The process terminates only when the sizes of all subareas are sufficiently close to the average value.

Here, T_h represents a tolerance threshold that controls the maximum allowable relative imbalance between each subarea and the ideal average size. Formally, the continuation condition is defined as:

$$\left|\frac{S_i - \lfloor S/K \rfloor}{\lfloor S/K \rfloor}\right| > T_h,$$

where S_i denotes the number of grid cells in subarea A_i, and $\lfloor S/K \rfloor$ is the discretized average number of grid cells per subarea.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

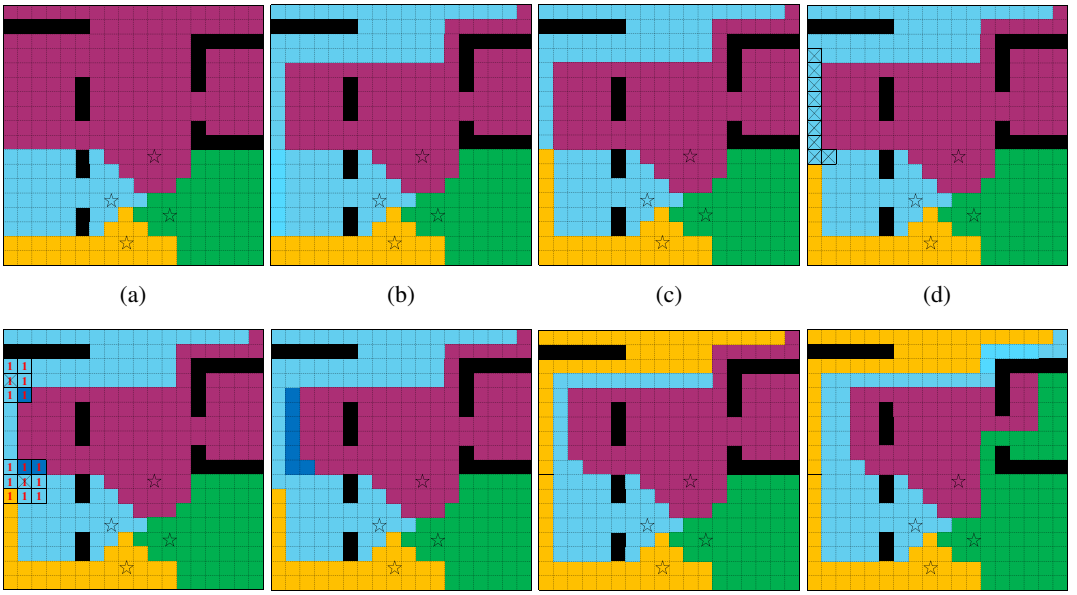


Figure 6. *The process of grid cell transfer between the subareas of four robots until the free area is equally divided. Subareas assigned to Robots 1 through 4 are distinguished by different colors: yellow, green, blue, and purple, respectively.*

4.1.2. The process of grid cell transfer between non-adjacent subareas

In Figure 5, we introduced a simple scenario with only two robots, where $A_{\max}$ and $A_{\min}$ are directly adjacent. However, as the number of robots increases, $A_{\max}$ and $A_{\min}$ may not be directly adjacent. The following two paragraphs discuss a new issue arising in this scenario, which is the narrow channel fixing issue.

As shown in Figure 6(a), in the initial size division, $A_{\max}$ (Robot 4) and $A_{\min}$ (Robot 1) are not directly adjacent, so the grid cell transfer path is $4 \rightarrow 3 \rightarrow 1$. Figures 6(b) show that Robot 3 first receives a number of grid cells of size $\lfloor \frac{S}{K} \rfloor - s_{\min}$. In Figure 6(c), when the grid cell transfer reaches the m-th round, the sender area is cut off by the receiver area, and the fallback + updated wavefront distance strategy is no longer applicable. Figure 6(d) shows that when falling back to the m-1-th round, deleting the grid cells marked as "x" in the sender area will disconnect the area. These grid cells collectively form a set called the "narrow channel."

As shown in Figure 6(e), to fix the "narrow channel" in the sender area, we perform omnidirectional expansion centered at each grid cell. The expanded sender area will occupy the grid cells originally belonging to robot p (p≠receiver) (depicted in dark blue in the figure). Figure 6(f) shows that when robot p's area remains connected, the "narrow channel" in the sender area is successfully fixed. If fixing the sender's "narrow channel" causes robot p's area to become disconnected, it indicates that robot p, like the sender, also has a "narrow channel" that needs to be fixed. In this case, robot p performs the same fix operation, occupying robot q's grid cells, and then checks if robot q's area remains connected. This fix process is recursive, with a maximum recursion depth equal to the number of robots K. As shown in Figure 6(g), after Robot 3 completes the fix, it continues transferring the grid cells to Robot 1 until the number of grid cells reaches $\lfloor \frac{S}{K} \rfloor$. Figure 6(h) shows that when Robot 4 transfers the grid cells to Robots 2 and 3, the number of grid cells of the four robots becomes nearly equal, thus completing the size division.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

Algorithm 2. Minimize the number of turns

Input: F: set of nodes in the spanning graph H
Output: T: spanning tree with minimum turns

1 $T \leftarrow \emptyset$, $m \leftarrow 0$;
2 **while** $F \neq \emptyset$ **do**
3 $\quad B \leftarrow$ longest brick determined from F;
4 $\quad m \leftarrow m + 1$;
5 $\quad V_B \leftarrow$ edge and node sets of B;
6 $\quad T \leftarrow T \cup B$;
7 $\quad F \leftarrow F \setminus V_B$;
8 **while** $m > 1$ **do**
9 $\quad e_{n_i,n_j} \leftarrow$ edge with smallest cost $w(\cdot)$ connecting adjacent nodes n_i and n_j where n_i and n_j are in different components of T;
10 $\quad T \leftarrow T \cup e_{n_i,n_j}$;
11 $\quad m \leftarrow m - 1$;
12 **return** T;

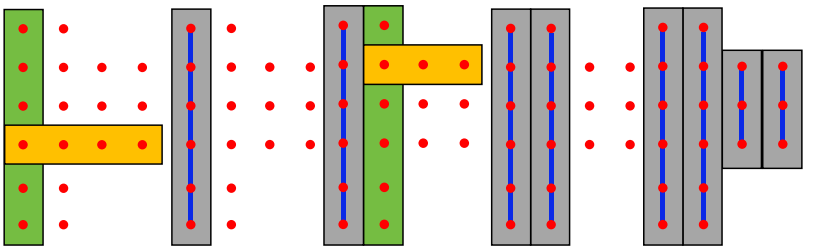


Figure 7. *Illustration of the greedy process for selecting the minimal number of bricks. Green and orange represent the longest vertical and horizontal candidate bricks, respectively, while gray indicates the brick B selected by the greedy algorithm; blue highlights the spanning tree trunk within B.*

4.2. Computation-efficient minimization of STC path turns

The problem of minimizing the number of bricks is addressed in the TMSTC* algorithm by solving the minimum vertex cover problem on a bipartite graph. In contrast, our method adopts a greedy-based strategy: select the longest brick from the spanning tree node set, remove the nodes contained in the brick from the set, and repeat this process until the set is empty. The greedy process is illustrated in Figure 7, and the corresponding pseudocode can be found in lines 1–7 of Algorithm 2.

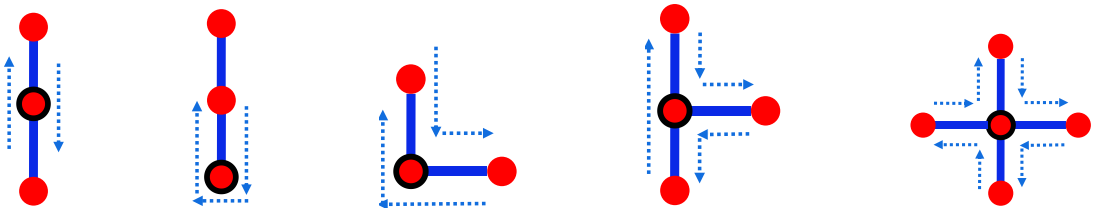
Figure 7(a) shows the initial state, where all nodes can be used to construct bricks, and the longest vertical and horizontal bricks are selected as candidates. Figure 7(b) illustrates that the longest brick is chosen from the candidate set, and the nodes within this brick are connected as part of the final spanning tree. Figure 7(c) shows that, after removing the connected nodes, the remaining nodes are used to construct new bricks, and the longest vertical and horizontal bricks are again selected as candidates. Figures 7(d)–(e) show that this process is repeated until no remaining nodes can be used to form bricks.

As shown in Table II and Figure 8, spanning tree nodes are classified into five categories, with each category corresponding to a specific number of turns, denoted as $f(n)$. Here, n represents a node in the spanning tree H, and each node n corresponds to a grid cell g in the map.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

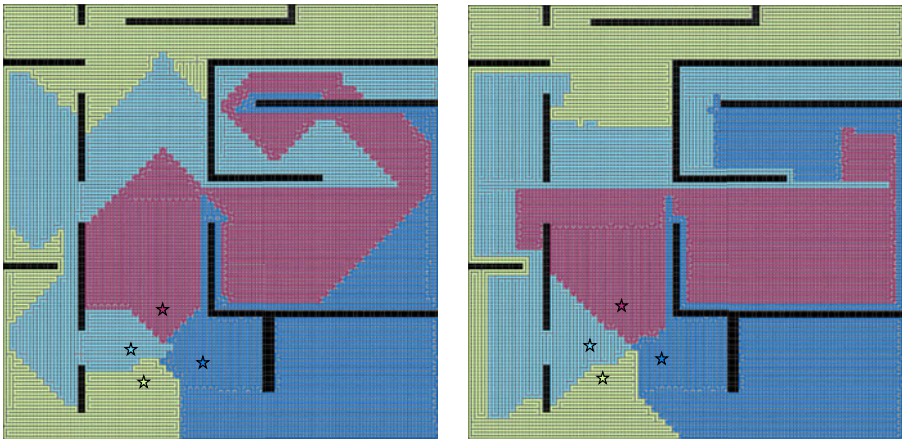
Table II. *Number of turns for different node categories.*

I	L	+ −	Leaf	T
0	2	4	2	2



(a) (b) (c) (d) (e)

Figure 8. *Five typical node types in the spanning tree T (*I*-type, leaf,* L*-type,* T*-type, and +-type) with corresponding turn numbers. The dotted blue arrow indicates the robot's trajectory.*



(a) Manhattan distance (b) Chebyshev distance

Figure 9. *Comparison of two distance metrics used in DAWG. (a) Manhattan distance, with an average of 316.5 turns. (b) Chebyshev distance, with an average of 213.5 turns.*

Therefore, the difference in the number of turns before and after adding a candidate edge to the spanning tree can be calculated:

$$w(n_1, n_2) = f^+(n_1) + f^+(n_2) - f(n_1) - f(n_2) \tag{4}$$

Here, $f^+(n_1)$ and $f^+(n_2)$ denote the number of turns at n_1 and n_2 after adding the edge e_{n_1,n_2}, respectively.

To minimize the number of turns, at each step we select an edge from the candidate edges that connect two adjacent bricks, where both bricks are not yet in the spanning tree T, and choose the edge with the minimum value of the function $W(n_1, n_2)$. After selecting the edge, it is added to the spanning tree T, connecting the two bricks. This process is repeated $m - 1$ times to construct the minimum-turn spanning tree. The corresponding steps are shown in lines 8–11 of Algorithm 2.

As shown in Figure 9, we found that during the size division phase, different distance metrics affect the average number of turns of each robot. When wavefront propagation is performed using Manhattan distance and Chebyshev distance, the average number of turns for each robot is 316.5 and 213.5,

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

Table III. *MCPP instance specifications.*

Map name	Grid size	Obstacle density (%)	K
Floor	128×128	6.08	4/6
Random	128×128	9.96	4/6
Room	128×128	9.59	4/6
Denver1	256×256	25.05	4/6
Berlin1	256×256	28.52	4/6
Maze	256×256	9.56	4/6
NewYork1	1024×1024	25.68	4/6/8/10
AR0701SR	1024×1024	66.33	4/6/8/10
Boston0	1024×1024	25.13	4/6/8/10

respectively. Theoretically, when wavefront propagation is carried out using these two different distance metrics, the isochrones of the wavefront are square and diamond-shaped, respectively, which leads to differences in the number of turns at the boundaries of the subareas. Therefore, in the area equal division phase, using Chebyshev distance for wavefront propagation can further reduce the number of turns of each robot.

5. Results & analysis

The experiments were conducted on a device with an Intel Core i5-11,400 processor (2.60 GHz) and 16 GB of memory. Our code is open-source and available on GitHub.

Parameter: In DARP, the maximum number of iterations was set to be nearly infinite. For the other three algorithms, the experimental parameters were kept consistent with the descriptions in the original papers.

Instances: As shown in Table III, we generated 24 test instances by placing different numbers of robots, denoted as K, on 9 maps with varying obstacle distributions. The values of K are selected with reference to ref. [16]. Each instance corresponds to a specific robot distribution on a given map. Among these, the first map is a typical floor map created by the authors, while the remaining eight maps are sourced from the widely used 2D pathfinding benchmarks [28]. The existing large-scale MCPP study [24] is conducted on a map of size 256×256. Therefore, in our experiments, we define maps of size 256×256 and above as large-scale maps, while other maps are referred to as small-scale maps.

Furthermore, to provide a more comprehensive evaluation of the proposed method, the test maps are designed by considering three key factors: map scale, obstacle density, and the distribution of initial robot positions. As shown in Figure 14(c), on the map *Room*, the obstacle density is only 9.59%, and the initial positions of the four robots (indicated by white pentagrams) are highly clustered. In contrast, as shown in Figure 14(d), the map *AR0701SR* has a high obstacle density of 66.33%, and the initial positions of the 10 robots are more spatially dispersed.

5.1. Solvability comparisons

In this letter, the solvability of an algorithm is defined as whether it can produce the result of coverage path planning within 3 hours. To evaluate the solvability of our algorithm compared to other existing algorithms, we conducted experiments on both small-scale and large-scale maps.

As shown in Figure 10, our algorithm achieves 100% solvability on both small-scale and large-scale maps, while DARP fails on some maps. Furthermore, although BO-MSTC, MFC, and TMSTC* are not MCPP algorithms based on size division, their solvability rate on large-scale maps is still below 100%. These results demonstrate that the proposed DAWG algorithm exhibits stronger solvability in large-scale maps.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

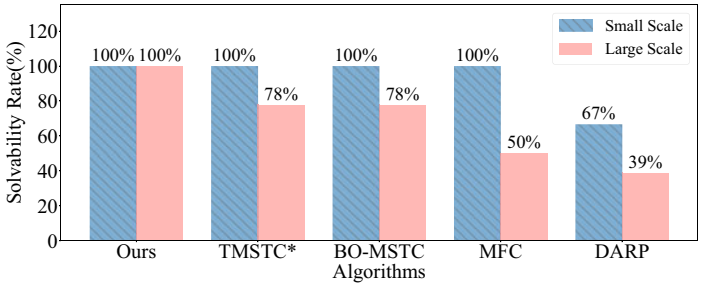


Figure 10. *Comparison of solvability rates on large-scale and small-scale maps using DAWG and baseline algorithms.*

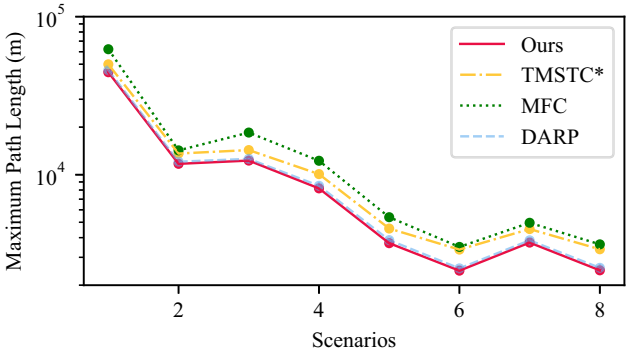


Figure 11. *Comparison of maximum coverage path length among four algorithms.*

5.2. Path length and number of turns comparison

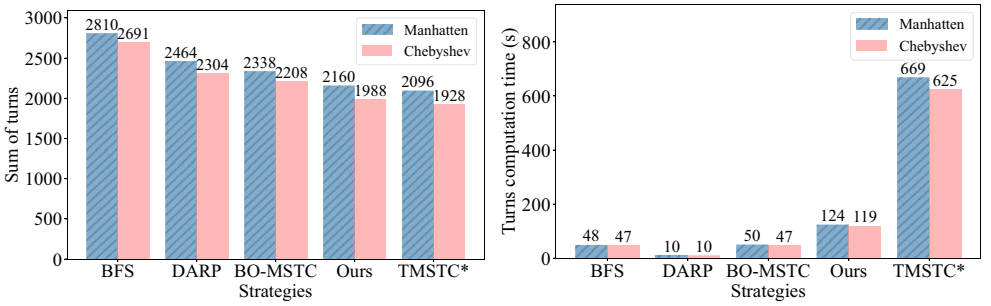
To compare the maximum coverage path lengths between our proposed algorithm and existing methods, we selected the intersection of solvable instances across four algorithms: DAWG, DARP, TMSTC*, and MFC. Figure 11 presents the maximum coverage path length among the K robots for each of the eight instances (the specific values of K are provided in Table III). Based on these results, we can observe that both our DAWG algorithm and the DARP algorithm, as size division-based MCPP approaches, achieve comparable and relatively shorter maximum coverage path lengths than the other two methods that do not employ size division.

To evaluate the performance of our Chebyshev distance-based size division and Greedy–Kruskal spanning tree construction strategy in reducing turns, we calculated the average total number of turns and the computation time for the spanning tree construction process. Four existing strategies were compared: DARP (using DFS), TMSTC*, BO-MSTC (using Kruskal), and BFS. Figure 12(a) shows the total average turns for size division using Manhattan and Chebyshev distances under different spanning tree construction strategies. The use of Chebyshev distance consistently resulted in fewer turns compared to Manhattan distance, indicating that it optimizes subarea shapes and reduces turns within each spanning tree. Notably, as shown in Figure 12, our method incurs only 3% more turns than TMSTC*, while the computation time is reduced by 81%.

5.3. Coverage time comparisons

To ensure a fair comparison, we adopt the same coverage time calculation method as described in TMSTC*. Specifically, the time t_j required for a robot to move between two consecutive turning points γ_j and γ_{j+1} in the path Π_i is computed as follows, where a turning point is defined as a point where the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592



(a) Number of turns. (b) Computation time.

Figure 12. *Comparison of the number of turns and computation time required to construct the spanning tree.*

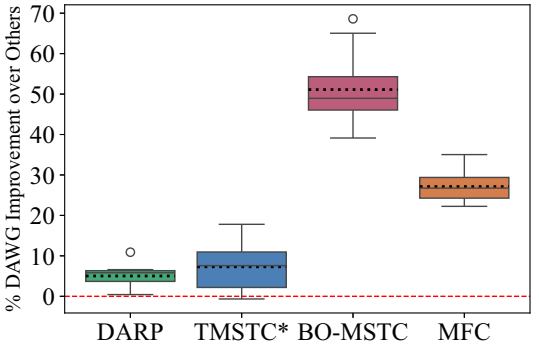


Figure 13. *Box plot of the performance improvement percentages of the proposed DAWG algorithm compared with four other algorithms, with dashed lines representing the mean. The mean performance improvement ratios relative to DARP, TMSTC*, BO-MSTC, and MFC are* 5%, 7.3%, 51.1%, *and* 27.1%, *respectively.*

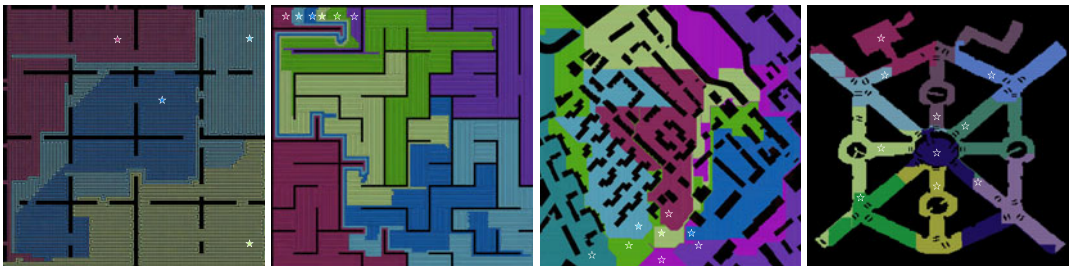
robot's direction of motion changes:

$$t_j = \begin{cases} \sqrt{\dfrac{d_{\gamma_j,\gamma_{j+1}}}{a}}, & \text{if } d_{\gamma_j,\gamma_{j+1}} \le \dfrac{v_{\max}^2}{a}, \\ \dfrac{d_{\gamma_j,\gamma_{j+1}}}{v_{\max}} + \dfrac{v_{\max}}{a}, & \text{otherwise.} \end{cases} \tag{5}$$

Here, $d_{\gamma_j,\gamma_{j+1}}$ denotes the Euclidean distance between two turning points, $v_{\max}$ is the maximum velocity, and a is the constant acceleration. The time spent on turning is also considered, where each turn costs $\frac{\pi}{2\omega}$, and ω represents the constant angular velocity. Other kinematic parameters and implementation details are consistent with those provided in ref. [17]. Additionally, in our experiments, we assume that robots can only move in four directions (up, down, left, right). Therefore, we made corrections to the coverage paths obtained from the TMSTC*, BO-MSTC, and MFC algorithms, where it is assumed that robots can move in eight directions. Specifically, when diagonal movement occurs in the path, we decompose it into two components along the horizontal and vertical directions using vector decomposition.

In the MCPP problem, coverage time is defined as the time taken for the last robot to complete its coverage task. As shown in Figure 13, to evaluate the effectiveness of the proposed algorithm in comparison with existing algorithms, we calculated the coverage time improvement percentage of DAWG relative to DARP, TMSTC*, BO-MSTC, and MFC. The results indicate that DAWG achieves an average coverage time improvement of 5%, 7.3%, 51.1%, and 27.1%, respectively, over the aforementioned four baseline algorithms across all test cases (assuming the algorithm is solvable for each test case). The experimental

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592



(a) (b) (c) (d)

Figure 14. *Four different environments and the corresponding coverage path planning results using the proposed DAWG algorithm: (a) Room,* 128 × 128*, with 4 robots; (b) Maze,* 256 × 256*, with 6 robots; (c) Boston0,* 1024 × 1024*, with 8 robots; and (d) AR0701 SR,* 1024 × 1024*, with 10 robots.*

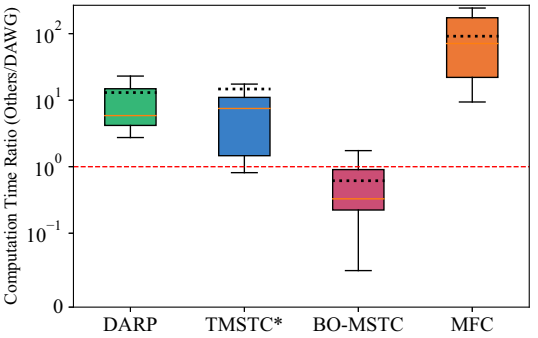


Figure 15. *Box plot showing the ratio of the computation time of DARP, TMSTC*, BO-MSTC, and MFC to that of DAWG, with the mean indicated by the dashed line. The mean ratios are 13, 14.7, 0.6, and 91.6, respectively.*

results clearly demonstrate that our algorithm outperforms all baseline algorithms in terms of coverage time, establishing it as an effective solution to the MCPP problem in large-scale maps.

Figure 14 illustrates the coverage path planning results of DAWG applied to four test cases of varying scales. In Figure 14(c) and (d), due to the larger map scale, the paths are more densely packed. In the maps, white pentagrams represent the initial positions of the robots, black regions indicate obstacles, and the coverage paths of different robots are distinguished using different colors.

5.4. Computation time comparisons

We computed the ratio of the computation time of DAWG relative to the four existing algorithms across all test cases (assuming the algorithm is solvable for each test case). As shown in Figure 15, the average ratios of the computation time of DAWG to DARP, TMSTC*, BO-MSTC, and MFC are 13, 14.7, 0.6, and 91.6, respectively. It can be seen that, compared to the other four algorithms, our proposed DAWG algorithm has a computation time only slightly longer than BO-MSTC. Compared to BO-MSTC, DAWG reduces coverage time by 51.1% and increases solvability by 33% on large-scale maps. More importantly, DAWG algorithm, compared to the existing SOTA algorithms, not only has shorter coverage time but also lower computational cost.

6. Conclusions

In this letter, the proposed DAWG algorithm achieves path length minimization by equally dividing free area. To address the issue of TMSTC*'s long computation time on large-scale maps, a greedy strategy is

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

introduced, which significantly reduces computation time compared to TMSTC* while ensuring near-optimal turn counts. The overall coverage time of this algorithm is shorter than that of SOTA algorithms, while its computation time remains at a low level. Additionally, experimental results demonstrate that DAWG exhibits higher solvability on large-scale maps.

The proposed method has potential applications in various real-world scenarios, such as cleaning and inspection tasks in large indoor environments. However, it still has certain limitations. Since the method is based on spanning tree construction, it imposes specific requirements on map modeling, that is, the grid size must match the robot size. In practice, the grid size may be smaller than the robot size. This makes it difficult for the robot to enter narrow areas and reduces the overall coverage rate. To balance coverage and efficiency, future work may consider size-adaptive robots as proposed in refs. [25, 26]. Such robots can maintain a larger size in open areas to improve coverage efficiency. They can also reduce their size in narrow areas to enhance accessibility. It should be noted that, when the speed remains constant, reducing the robot size decreases the area covered per unit time, which lowers coverage efficiency. By using size adaptation, a high coverage rate can be maintained while mitigating its negative impact on coverage efficiency.

Author contributions. **Yiyi Zhang** and **Renjie Guo** contributed equally to this work; **Yiyi Zhang** and **Renjie Guo** conceived and designed the study, and conducted the experiments; **Peng Guo** supervised the project and provided critical revisions to the manuscript. All authors read and approved the final manuscript.

Financial support. This research was supported by the National Key Research and Development Program of China under Grant 2024YFC3811102.

Competing interests. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] S. Yin and Z. Xiang, "Energy-constrained collaborative path planning for heterogeneous amphibious unmanned surface vehicles in obstacle-cluttered environments," *Ocean Eng.* **330**, 121241 (2025).
[2] S. Yin and Z. Xiang, "Real-time distributed decision-making for simultaneous target assignment and path planning in multiple unmanned surface vehicles," *Expert Syst. Appl.* **279**, 127457 (2025).
[3] S. Yin and Z. Xiang, "Adaptive collision avoidance strategy for USVs in perception-limited environments using dynamic priority guidance," *Adv. Eng. Inform.* **65**, 103355 (2025).
[4] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.* **61**(12), 1258–1276 (2013).
[5] R. Bormann, F. Jordan, J. Hampp and M. Hägele, "Indoor Coverage Path Planning: Survey, Implementation, Analysis," In: *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2018) pp. 1718–1725.
[6] L. Collins, P. Ghassemi, E. T. Esfahani, D. Doermann, K. Dantu and S. Chowdhury, "Scalable Coverage Path Planning of Multi-Robot Teams for Monitoring Non-Convex Areas," In: *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2021) pp. 7393–7399.
[7] D. P. Arab, M. Spisser and C. Essert, "Complete coverage path planning for wheeled agricultural robots," *J. Field Robot.* **40**(6), 1460–1503 (2023).
[8] Y. Gabriely and E. Rimon, "Spiral-STC: An On-Line Coverage Algorithm of Grid Environments by a Mobile Robot," In: *Proc. IEEE Int. Conf. Robot. Autom.*, vol. **1** (2002) pp. 954–960.
[9] R. Almadhoun, T. Taha, L. Seneviratne and Y. Zweiri, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *reconstruction* **69**, 79 (2019).
[10] A. C. Kapoutsis, S. A. Chatzichristofis and E. B. Kosmatopoulos, "DARP: Divide areas algorithm for optimal multi-robot coverage path planning," *J. Intell. Robot. Syst.* **86**(3–4), 663–680 (2017).
[11] O. Idir and A. Renzaglia, "Multi-Robot Weighted Coverage Path Planning: A Solution Based on the DARP Algorithm," In: *Proc. Int. Conf. Control, Autom., Robot. Vision (ICARCV)* (2022) pp. 98–104.
[12] N. Baras and M. Dasygenis, "Size division using affinity propagation for multi-robot coverage path planning," *Appl. Sci.* **13**(14), 8207 (2023).
[13] S. D. Apostolidis, P. C. Kapoutsis, A. C. Kapoutsis and E. B. Kosmatopoulos, "Cooperative multi-UAV coverage mission planning platform for remote sensing applications," *Auton. Robots* **46**(2), 373–400 (2022).

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592

[14] N. Agmon, N. Hazon and G. A. Kaminka, "The giving tree: Constructing trees for efficient offline and online multi-robot coverage," *Ann. Math. Artif. Intell.* **52**(2–4), 143–168 (2008).
[15] X. Zheng, S. Jain, S. Koenig and D. Kempe, "Multi-Robot Forest Coverage," In: *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)* (2005) pp. 3852–3857.
[16] J. Tang, C. Sun and X. Zhang, "MSTC: Multi-Robot Coverage Path Planning Under Physical Constraint," In: *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2021) pp. 2518–2524.
[17] J. Lu, B. Zeng, J. Tang, T. L. Lam and J. Wen, "TMSTC*: A path planning algorithm for minimizing turns in multi-robot coverage," *IEEE Robot. Autom. Lett.* **8**(8), 5275–5282 (2023).
[18] R. Adams and L. Bischof, "Seeded area growing," *IEEE Trans. Pattern Anal. Mach. Intell.* **16**(6), 641–647 (1994).
[19] A. Zelinsky, R. A. Jarvis, J. Byrne and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," *Proc. Int. Conf. Adv. Robot.* **13**. Citeseer (1993).
[20] C. S. Tan, R. Mohd-Mokhtar and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access* **9**, 119310–119342 (2021).
[21] I. Vandermeulen, R. Groß and A. Kolling, "Turn-Minimizing Multi-Robot Coverage," In: *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2019) pp. 1014–1020.
[22] M. Ramesh, F. Imeson, B. Fidan and S. L. Smith, "Optimal partitioning of non-convex environments for minimum turn coverage planning," *IEEE Robot. Autom. Lett.* **7**(4), 9731–9738 (2022).
[23] J. Tang, Z. Mao and H. Ma, "Large-scale multi-robot coverage path planning on grids with path deconfliction," *IEEE Trans. Robot.* (2025).
[24] J. Tang and H. Ma, "Large-scale multi-robot coverage path planning via local search," *Proc. AAAI Conf. Artif. Intell.* **38**(16), 17567–17574 (2024).
[25] S. B. P. Samarakoon, M. V. J. Muthugala, M. Kalimuthu, "Design of a Reconfigurable Robot with Size-Adaptive Path Planner," In: *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)* (2022) pp. 157–164.
[26] S. M. B. P. Samarakoon, M. A. V. J. Muthugala, M. Kalimuthu, S. K. Chandrasekaran and M. R. Elara, "Online Coverage Path Planning Scheme for a Size-Variable Robot," In: *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2023) pp. 5688–5694.
[27] Y. Xu, Z. Diao, J. Li, Z. Liu, X. Zhang, J. Ye and J. Nie, "Path optimization for surface damage detection on large structures," *J. Build. Eng.* 115094 (2025).
[28] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *IEEE Trans. Comput. Intell. AI Games* **4**(2), 144–148 (2012).

Cite this article: R. Guo, Y. Zhang and P. Guo (2026). "Divide Areas based on Area Growing via Wavefront Propagation: a near-optimal multi-robot coverage path planning algorithm for large-scale maps with low computational cost", Robotica **44**, 1843–1861. https://doi.org/10.1017/S0263574726103592

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 06:21:25, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0263574726103592