Çankaya University Journal of Science and Engineering
Volume 11 (2014), No 1, 29-63

A Failsafe Fine Resolution Online Grid Coverage Algorithm for Autonomous Agents

Hakan Aydemir[1], Sinan Kalkan[2], Veysi İşler[3]

[1]Defence Science Institute, Army Academy, Ankara, Turkey
[2]Computer Engineering Department, Middle East Technical University, Ankara, Turkey
[3]Modeling and Simulation R&D Center, Middle East Technical University, Ankara, Turkey
e-mail: haydemir74@yahoo.com, sinankalkan@gmail.com, isler@ceng.metu.edu.tr

Abstract: Planning paths for vehicles which take aerial photographs/images or which collect topographic data for mapping from the air, on the ground or inside water is a challenging problem. Moreover, if such vehicles are expected to move autonomously and if the terrain consists of obstacles, the complexity increases. In this article, we propose an area coverage algorithm that is suitable for planning paths for autonomous agents assigned to cover an area. Our algorithm is a polynomial time heuristic algorithm which guarantees complete coverage of an area consisting of obstacles. At each step of the algorithm, an agent observes the neighboring cells and moves to a cell that is surrounded by more obstacles or by already visited cells. With this simple behavior, we show in a simulated environment with a different number of agents that our method performs comparably to, and in certain configurations, better than the existing methods. An important advantage of our method is that it is failsafe. In other words, if a subset of the agents fails to complete their duties, the remaining agents suffice to cover any unvisited parts of an assigned area. This is due to the fact that (i) our algorithm can run online, (ii) the agents are able to perceive only the grids in their neighborhood and (iii) there is no cooperation among them.

Keywords: Area coverage, Multi agent, Autonomous agent, Shortest path.

1. Introduction

In order to explore space and air, various types of vehicles are being used for different purposes both military and civilian. Since it is easier to investigate the Earth from the air, studies have concentrated mostly on utilization of air and space vehicles and a great deal of effort has been spent on developing unmanned systems or autonomous robots to perform this investigation of the Earth. An important problem in such autonomous mobile systems (which will be designated agents for the remainder of the article) is planning a route, or a path, for the task at hand, satisfying the given constraints. The literature focuses to a great extent on path planning for single [1-4] and multiple agents [5-10].

In this article, we study a special case of path planning, called area coverage. In the area coverage problem, there is a set of agents that are to cover a pre-determined area of the environment by visiting each part. An example is seen in Fig. 1, where two agents cover green-marked regions in the environment. The main objective is to cover the whole terrain with one or more agents in minimum time with minimum re-coverage of previously covered locations. The possible application areas of the problem are observation of the Earth by satellites or unmanned air vehicles [11], lawn mowing [12], wall painting/inspection [13], floor cleaning [14], harvesting [15], mine cleaning [16], intrusion detection [17] and search-rescue missions [18]. The single agent case of the coverage problem is simply the traveling salesman problem in which nodes are the cells and the agent's movement is limited to the neighbor cells. The multi agent case is naturally more difficult than the single agent case and finding the optimal solution is proven to be NP-Complete [19].

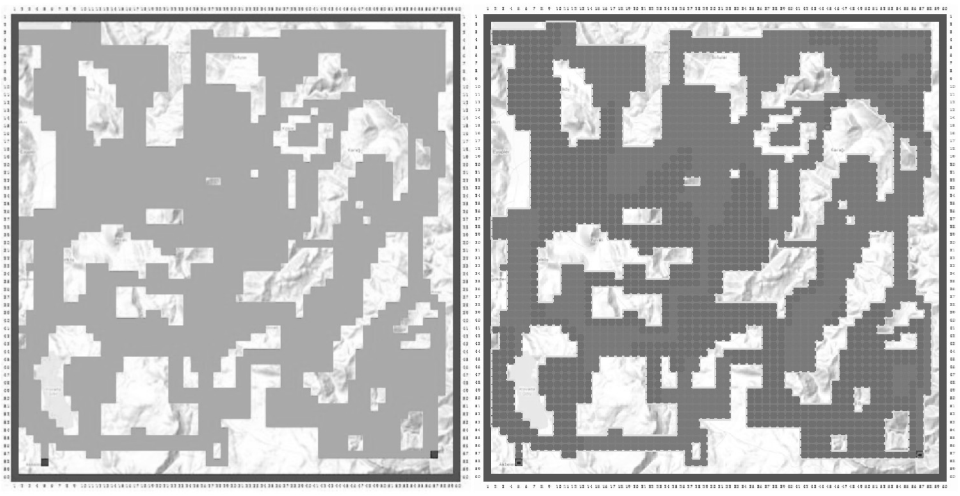

(a) Initial state (b) After coverage state

FIGURE 1. In the initial state (a), the agents (in blue) are located on the bottom left and bottom right corners. The green area is to be covered. The covered area (b) by the first agent is colored pink, and brown for the second.[1]

1. Note that the colored figures appear in the online version (www.cujse.cankaya.edu.tr).

When analyzing existing multi agent coverage systems, we can classify the approaches based on the following properties:

- Representation of the terrain: H. Choset [20] classified terrain representation approaches into three: (i) *approximate cellular decomposition,* where the specified area is covered by a grid of equally-shaped cells [1-10]. (ii) semi-approximate decomposition, where the free space is divided into lines with equal width but irregular shape at the top and bottom [21, 22] and (iii) *exact decomposition,* where the free space is decomposed to an unequal set of regions whose union fills the entire area exactly [23, 24]. In this article, approximate cellular decomposition is preferred and the size of each grid is fixed to the coverage size of an agent at each time step.
- The number of agents: The completion time, which is the main performance criteria in coverage algorithms, is supposed to decrease with the multi agent solutions [8]. On the other hand, the computational complexity and memory consumption in crease as the number of agents increases.
- Offline vs. online: An algorithm is considered to be *offline* if the agents plan their paths before the simulation or the actual test begins (see, e.g., [2, 7, 25] for offline algorithms). However, if the agents acquire the information during runtime from their sensors and decide on the next movement by evaluating the information *online*, an algorithm is considered *online* [20] - see, e.g., [2, 4, 14, 26] for online algorithms.
- Terrain and cell properties: The terrain may include obstacles whose location, size or number might change in time. The agents are supposed to avoid the obstacle cells. Moreover, the cells on the terrain can have different weights (*weighted* or *un weighted terrain*) [19]. The existence of obstacles and cell weights on a terrain increases the complexity of the problem.
- Cooperation: Cooperation between agents implies information exchange between agents to achieve the main shared goal and incorporating a conflict resolution mechanism [6]. The cooperation can be coordinated from the center, by one of the agents or via one-to-one local interactions between the agents. It is also possible to form coalitions during an operation [27]. On the other hand, there may not be any cooperation where the agents are completely independent for their future decisions.

- Communication: This is the ability to exchange information between the agents. Depending on the structure of the solution approach, the agents may be capable of sending information about their current locations, positions of the obstacles they encountered or the cells they covered. Communication capability is the basic necessity for cooperation [6].
- Agent movement: On grid-structured terrains, agents are generally allowed to move just one step at a time towards an edge of the cell (*left, right, up* and *down* for squared grids) if no obstacle exists in that direction (see Fig. 2). However, it is also possible to allow the agents to move diagonally.
- Backtracking: *Backtracking* means re-covering or re-stepping on any previously covered cells. Some of the algorithms allow the agent to backtrack while the other works in a non-backtracking manner. Although backtracking seems to decrease the time and re-coverage efficiency at first glance, non-backtracking algorithms may be inefficient and efficient algorithms may be backtracking surprisingly [7].
- Initial and final positions of the agents: The algorithm may require the robots to start near a wall or border, together at neighboring cells adjacent to each other or from distant locations. It might also be required that the final positions of the agents should be the same as their initial positions.
- Sensor capacity: In the grid terrains, it is generally accepted that the sensor mounted on the robots can detect only the next cell along the possible directions of movement. The size of the grid is also fixed to the coverage width of the sensor at a time step.

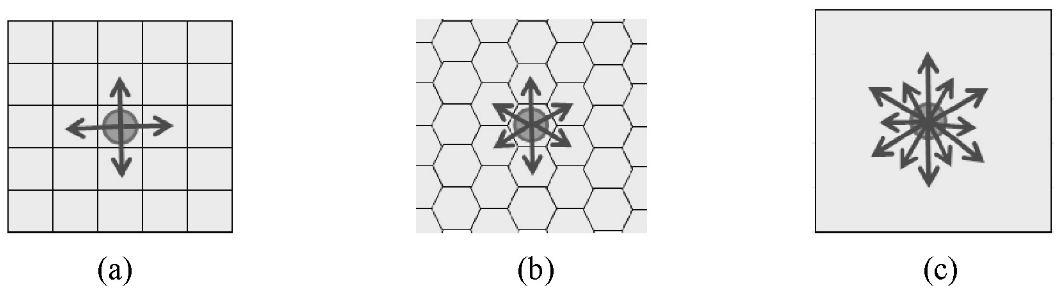


FIGURE 2. Possible directions of movement for three terrain types. (a) 4 directions for square grids, (b) 6 directions for hexagonal grids and (c) no limits on directions for non-grid terrains.

- Memory and Cell Marking: Agents might (i) permanently mark the cells they have covered [29, 30], (ii) leave temporary marks (like ants) [9, 31, 32], or (iii) leave no mark on the terrain at all [5, 33]. The information about the cell marking can be saved externally on the center or internally on the agent.
- Dynamic terrain: The properties, types or coverage status of the cells in the terrain might be changeable by time (e.g., some of the covered cells might become uncovered in time, some of the obstacles change their location or turn into target cells or the size of the terrain might increase, etc.
- Failure Behavior: This is the ability of the algorithm to react in case of agent or communication failure [6]. In order to perform a complete coverage, the algorithm must satisfy failure occurrences.
- Performance criteria: Multi agent coverage algorithms are compared and evaluated based on the following: (i) the time of coverage completion, (ii) the rate of the re-covered cells over the number of covered cells (in total and separately for each agent), (iii) the difference of coverage number between the maximum covering robot and minimum covering robot, and (iv) the deviation from the optimal for each robot [8, 34].

In this article, we propose an algorithm for area coverage which is a fine resolution, online heuristic algorithm (called Multi Agent Grid Environment Coverage Algorithm - MGECA) that guarantees to cover the entire grid environment with multiple agents. The algorithm uses approximate cellular decomposition for un weighted and unknown terrains. Since the agents have no information about the terrain (location of borders, obstacles, other agents and covered cells) initially, it works online both for single and multiple agents. The agents are able to move along four directions. They can mark the cells permanently and send the information of the cells they passed to the other agents. This provides the other agents with awareness of the map of the current covered region so the agents can find a way to the nearest uncovered cell when they are completely surrounded by covered cells. There is no necessity to return to the initial position. The agents are allowed to step on any covered cells in the neighborhood. The scale of the cells is as wide as the sensor capacity. The cells on the terrain are not classified as "*large cell*" or "*small cell*" as carried out in existing studies explained below.

2. Existing Studies on Area Coverage

Area coverage is usually studied in Robotics, which presents a wide range of useful applications. Based on the objectives of specific applications, many approaches have been proposed for both terrain representation and its coverage (see, e.g., Choset [20] for a review).

In this study, we concentrate on approximate cellular decomposition for terrain representation. It is possible to classify the area coverage algorithms with approximate cellular decomposition into two groups as (i) the $D \times D$ algorithms and (ii) the $2D \times 2D$ algorithms. In the first group, the territory is decomposed with a fine resolution into $D \times D$ grids and in the second, the area is decomposed into $2D \times 2D$ grids where the coverage size of the agent is accepted as D.

2.1. $\boldsymbol{D \times D}$ Algorithms

The coverage with D × D decomposition is generally handled with a multi agent approach with different search strategies being studied [3]. The simplest of these filling strategies is the Brownian Motion, where the agents move randomly on the terrain [35, 36]. As it turns out, this strategy results in low performance in terms of completion time and re-coverage since it permits the agent to cover a cell several times. On the other hand, it consumes fewer computational resources compared to other systems since the agents are very simple [36]. Spread Seeder Trajectory is another strategy where the agents move along the covered line until it meets the border and follows a zigzag path to fulfill the coverage of the terrain [22, 37]. The efficiency of this strategy is highly dependent on the first location and orientation of the agent [3]. Spiral Trajectory is the last of the strategies in which the agent starts near a wall, follows a spiral path and finishes at the center of the spiral [14].

Since the strategies explained above did not ensure the complete coverage of non-convex and obstacle-consisting regions, they are equipped with additional properties. The *Backtracking Spiral Algorithm (BSA)* [3] was proposed by Gonzales for the single agent case in 2005 in which agents link the spiral paths with the shortest path algorithms. An example of BSA is shown in Fig. 3. Later in 2011, *BSA* was improved by Gerlein and Gonzales with a multi agent approach. They proposed the Backtracking

Spiral Algorithm – Cooperative Multirobot (*BSA-CM)* for complete coverage of the terrain by multiple agents [6]. In their algorithm, agents can cooperate with other agents, follow a spiral path until they complete their spiral and are able to negotiate with backtracking points to find the most suitable one.

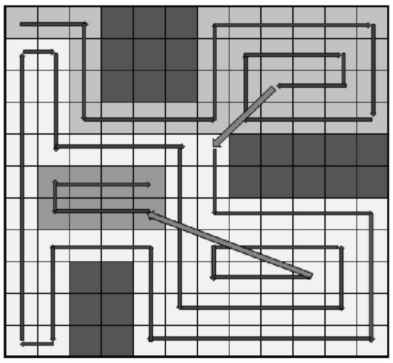


FIGURE 3. Example of the BSA Algorithm. 3 spiral zones are shown with different colors. Blue lines are backtracking directions (adapted from [4, 6].

The *Spiral Trajectory* was also used by Choi for the online single agent solution in 2009. They proposed the *Linked Spiral Paths Using Constrained Inverse Distance Transform* (LSP-CIDT) method, which uses distance transforms to link the separated spiral paths [10].

Additionally, Zelinsky studied the single coverage using the distance transform path planning method, which uses the shortest path from a starting cell to a goal cell using the Wavefront Algorithm [1]. Altshuler and Bruckstein [38] also studied the strengths and limitations of collaborative teams of simple agents and presented the complexity of the coverage problem with the same size and type of cells.

2.2. $2D \times 2D$ Algorithms

The algorithms that use $2D \times 2D$ terrain representation generally use a spanning tree approach and the environment is represented as a $2D$ grid of large square cells which are composed of four small equal cells. All of the agents on the terrain are of the size of a small cell.

A popular approach for $2D \times 2D$ terrain representation is the *Spanning Tree Coverage (STC)* by Gabriely and Rimon [2]. Their algorithm solves the single-robot coverage problem. Hazon and Kaminka later extended STC to *Multi-Robot Spanning Tree Coverage (MSTC)* [7]. Additionally, Senthilkumar and Bharadwaj proposed the *Simultaneous Multiple STC (S-MSTC)* and *Extended S-MSTC (ES-MSTC)* [39]. While MSTC improves the cover times compared to STC, the results were not satisfactory. Recently, STC is generalized by Zheng et al. to the *Multi-Robot Forest Coverage (MFC) Algorithm,* which solves the problem at polynomial time [8]. They also improved the MFC to solve the coverage problem for weighted terrain [5]. STC, MSTC and MFC are explained briefly below:

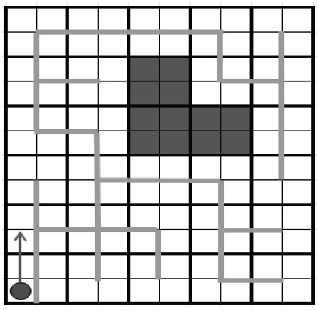


FIGURE 4. An example of STC. The robot moves in the direction of the red arrow and follows the blue line.

Spanning Tree Coverage solves the single-robot coverage problem in polynomial time. It first constructs a spanning tree of the graph, whose vertices are the large cells, and whose edges connect adjacent unblocked large cells. The robot then circumnavigates the spanning tree. STC never visits any small cell twice and thus minimizes the cover time. In addition, the robot essentially returns to its initial small cell, facilitating its collection and storage [5]. An example of STC is shown in Fig. 4.

STC was generalized to Multi-Robot Spanning Tree Coverage, a polynomial-time multi-robot coverage heuristic by Hazon and Kaminka [7, 40]. MSTC first constructs the same spanning tree as STC, and considers the tour that circumnavigates the spanning tree. Each robot follows the tour segment clockwise ahead of it. To improve the cover time, the longest segment can be divided evenly between the two adjacent robots. Each small cell is visited by only one robot, so there are never any collisions or blockages. Fig. 5 illustrates an example of MSTC in operation. As shown in these figures, efficiency of MSTC is highly dependent on the initial positions of the robots and on the structure of the spanning tree constructed [25, 41].

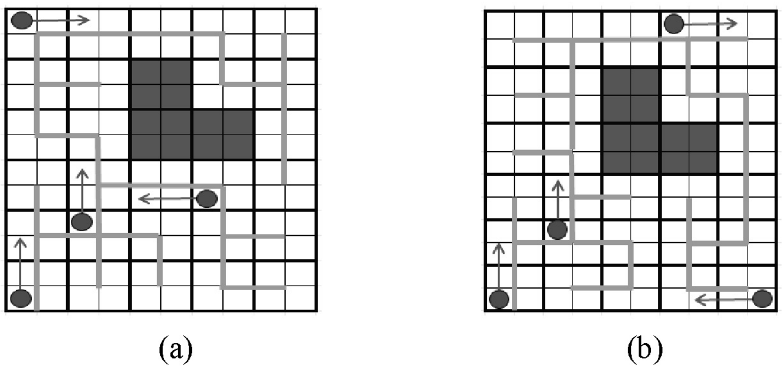


FIGURE 5. Two examples for MSTC. In (a), the agents move 8, 9, 66 and 21 steps respectively while they move 8, 23, 21, 52 steps in (b).

MFC operates on the graph whose vertices are the large cells, and whose edges connect adjacent unblocked large cells. If r robots start in a large cell, then MFC makes r identical copies of that vertex. MFC first finds a rooted tree cover for this graph in polynomial time, where the roots are the vertices that contain robots. The graph is allowed to be disconnected, so long as each of its components contains at least one robot. Each robot then circumnavigates its own tree [8]. MFC is illustratedin Fig. 6.

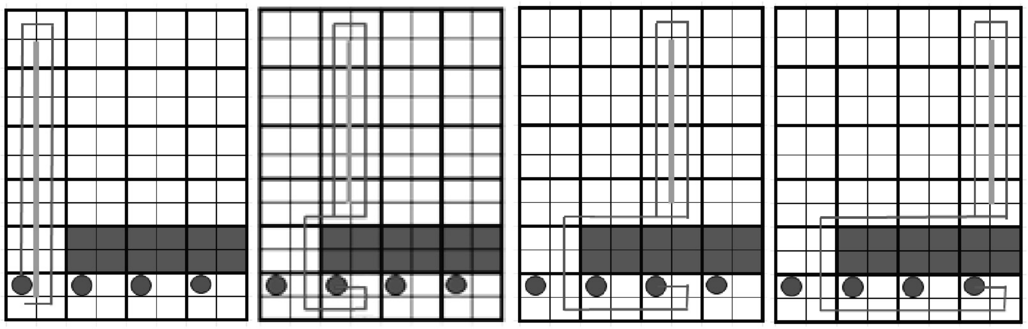


FIGURE 6. An example of MFC. The red lines are the routes of each robot (adapted from [8]).

The algorithms both with $D \times D$ and $2D \times 2D$ terrain represantations are summarized in Table 1 with their basic properties.

TABLE 1. Basic properties of the grid coverage algorithms

PROPERTY-Algorithm	***BSA-CM [6]***	***LSP-CIDT [10]***	***STC [2]***	***MSTC [7]***	***ES-MSTC [39]***	***MFC [8]***	***MGECA (Proposed Algorithm)***
Representation of the terrain	$D \times D$	$D \times D$	$2D \times 2D$	$2D \times 2D$	$2D \times 2D$	$2D \times 2D$	$D \times D$
The number of agents	Multi	Single	Single	Multi	Multi	Multi	Multi
Offline vs. online	Online	Online	Online	Offline	Online	Offline	Online
Terrain and cell Properties	Obstacle, Unweighted	Obstacle, Unweighted	Obstacle, Unweighted	Obstacle, Unweighted	Obstacle, Unweighted	Obstacle, Unweighted, Weighted	Obstacle, Unweighted
Cooperation	Yes	Yes	No	No	No	No	Yes
Communication	Yes	Yes	No	No	No	No	Yes
Agent movement	Vertical, Horizontal	Vertical, Horizontal, Diagonal	Vertical, Horizontal	Vertical, Horizontal	Vertical, Horizontal	Vertical, Horizontal	Vertical, Horizontal
Backtracking	Yes	Yes	No	No, Yes	Yes	Yes	Yes
Initial position	Limited	Limited	Non-limited	Non-limited	Non-limited	Non-limited	Non-limited
Return to initial position	No	No	Yes	No	Yes	Both	No
Cell marking	Permanent	Permanent	Permanent	Permanent	Permanent	Permanent	Permanent
Dynamic terrain	No	No	No	No	No	No	Possible

2.3. Performance Criteria

There are two performance criteria commonly used to evaluate area coverage algorithms. The first is *time rate* (*TR*), defined as follows:

$$TR = \frac{CT}{IT}, \tag{1}$$

where CT is the *completion time* ($CT = \max\{t_a\}_{a=1}^{|A|}$ where a denotes an agent, A is the set of agents, and t_a is the time spent by the agent) and IT is the *ideal number of cells per agent* (for an $M \times N$ terrain):

$$IT = \frac{M \times N}{|A|} \tag{2}$$

The second performance criterion is the *coverage rate* (*CR)*, which is the ratio of *total movement* ($TM = \sum_{a \in A} t_a$) over *total target cell:*

$$CR = \frac{TM}{|G|}, \tag{3}$$

where G is the set of target cells. The time rate measures time efficiency of the algorithm in terms of the number of cells visited, whereas the coverage rate tells us about the re-visiting performance of the algorithm.

There are also some secondary level criteria such as total cover (TC_a) and re-cover (TRC_a) numbers or rates (TR_a, CR_a) for each agent$a$, the values of minimum (minCR_a) and maximum (maxCR_a) covering agents or the differences (maxCR_a- minCR_a) between them:

$$TR_a = \frac{t_a}{IT}, \tag{4}$$

$$CR_a = \frac{(TC_a + TRC_a)}{TC_a}. \tag{5}$$

3. Our Proposal: Fine Resolution Multi Agent Grid Environment Coverage Algorithm (MGECA)

In MGECA, like other coverage algorithms, the terrain T is taken to be a 2-dimensional array, which can be considered as a snapshot of the environment from the top. The terrain T is decomposed into $M \times N$ many cells: $T = \{c\}_{i,j=1,1}^{M,N}$, where each cell c has size $D \times D$. The $D \times D$ resolution is better than most of the existing algorithms (see section 2.2) which require that each cell is composed of $2D \times 2D$ units. The terrain

T also contains set of obstacles (O) which the agents avoid entering. The agents initially haveno *a priori* information about the terrain and cells. However, when they are all surrounded by obstacles or already-covered cells, they make use of the current coverage status of the terrain in order to find the shortest path to the nearest uncovered cell. The current status of the covered region is accumulated as the agents explore the terrain. The agents are able to move only to the neighboring cells and acquire information from the adjacent cells. Basically, the algorithm is based on selecting the most valuable neighbor for the next time step and finding a way to an uncovered cell by using a polynomial time shortest path algorithm.

For both multi and single agent cases, the territory and the cells are preprocessed. All the cells in the area are labeled and classified as either *uncovered, covered* and *obstacle* cells; i.e., at any time, $T = U \cup C \cup O$, where U is the set of uncovered cells, C the set of covered cells, O the obstacles. An example of terrain is shown in Fig. 7. For the initial position, the number of covered cells is zero. For each time step, an agent moves to one of its neighbors by executing a decision procedure. As the agents move and cover the *uncovered cells* (U), the number of *covered cells* (C) increases. ($G = U$ at the beginning and $G = U \cup C$ during process).

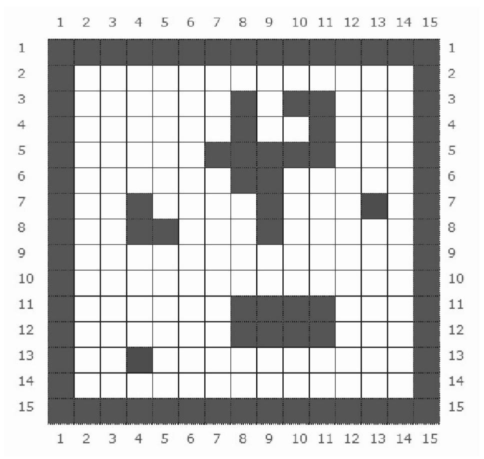


FIGURE 7. The red cells are obstacles while the blue cells are agents and the white cells are targets.

At the preprocessing step, a *degree* is calculated for each cell where the *degree d(c)* of a cell c is the number of uncovered cells in the neighborhood, and it is the main decision factor for the next step of the agent:

$$d(c_{i,j}) = u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}, \forall\, i \geq 1, i \leq M-1, j \geq 1, j \leq M-1 \qquad (6)$$

where $c_{i,j}$ is the cell at coordinate (i,j), and $u_{i,j}$ is defined as follows:

$$u_{i,j} = \begin{cases} 1, \text{if} c_{i,j} \in U \\ 0, \text{if} c_{i,j} \in C,\ c_{i,j} \in O \end{cases} \tag{7}$$

The agent selects the lowest degree neighboring cell and the degrees of the neighbors of the cells are updated at each step. We assume that the agents are not supposed to return to their initial positions. The algorithm stops when the final uncovered cell is covered.

3.1 Single Agent Coverage with MGECA

For the single agent case, the algorithm is simpler since there is no other agent that disturbs its plan for the next movement. The main steps of the algorithm are presented in Algorithm1.

ALGORITHM1 The Single Agent Grid Coverage Algorithm

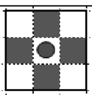
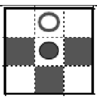
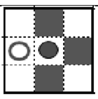
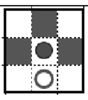
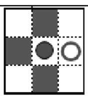
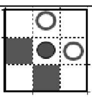
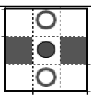
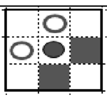
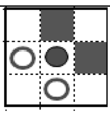
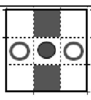
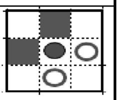
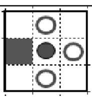
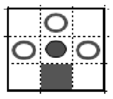
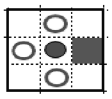
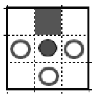
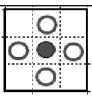
```
1.  for i=1 to M
2.    for j=1 to N
3.      if $c_{i,j} \in G$
4.        $d(c_{i,j}) \leftarrow (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})$   // Find the degree of each cell
5.        status$(c_{i,j}) \leftarrow UNCOVERED$
6.      else if $c_{i,j} \in A$// A, the set of agents
7.        $L(a) \leftarrow (i,j)$// Set current location (CL) of agent a to location (L) of $c_{i,j}$
8.  while completed = FALSE// Main loop
9.  $c_{i,j} \leftarrow L(a)$
10.   if $d(c_{i,j}) > 0$ // Cell has an uncovered neighbour
11.     $\Omega(c_{i,j}) = \{c_{i-1,j}, c_{i+1,j}, c_{i,j-1}, c_{i,j+1}\}$ // Neighbors of cell $c_{i,j}$
12.     $L(a) \leftarrow \arg\min_{c \in \Omega(c_{i,j})} d(c)$//The neighboring cell with the lowest degree. Choose
    in counterclockwise direction starting search from $c_{i,j-1}$ in case of multiple minimums
13.     status$(i,j) \leftarrow COVERED$
14.     Update Degree OfNeighbors $(i,j)$
15.   else// $d(c_{i,j}) = 0$
16.     If Find Shortest Path (Nearest Empty Cell)=TRUE
17.       $L(a) \leftarrow first\ cell\ through\ location\ of\ nearest\ empty\ cell$
18.     else if Find Shortest Path (Nearest Empty Cell)=FALSE
19.       completed =TRUE
```

3.1.1 Orientation and Decision at Each Step

The agent firstly orientates it self and checks the cells on four sides, compares their degrees and selects the lowest one. Giving priority to the lowest degree neighbor prevents the separation of the uncovered region and provides the covered cells attach together. If equality on the degrees of neighbor cells exists, the selection is made in a counter-clockwise order. The Wave front Algorithm which is an extension of Dijkstra's Shortest Path Algorithm is used to solve the surrounded case.

All possible combinations of the agent's location and neighbor cells are shown in Table 2. Red *squares* represent obstacles or covered cells. Empty blue circles are possible next cells and the filled cells are the agent's current location. The Shortest Path Algorithm is executed for the "$d(c) = 0$" case, the single alternative is selected for "$d(c) = 1$" and the one whose degree is lowest is selected for "$d(c) > 1$".

TABLE 2. Possible combinations of agent's location and next steps

1	Agent is covered completely($d(c) = 0$)											
	a											
2	Agent is covered on three sides ($d(c) = 1$)											
	a		b		c		d					
3	Agent is covered on two sides($d(c) = 2$)											
	a		b		c		d		e		f	
4	Agent is covered on one side($d(c) = 3$)											
	a		b		c		d					
5	Agent is free on all sides($d(c) = 4$)											
	a											

3.1.2 Planning from a Surrounded Cell

Dijkstra's Algorithm finds the shortest path from each node to all the others on a graph [42]. It is conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959. This algorithm is often used in routing and as a subroutine in other studies such as [43].

An extension of *Dijkstra's Shortest Path Algorithm* is the *Wave front Algorithm,* which finds the shortest path from a starting node to the nearest of several same-type target nodes [44]. It simply operates with the expansion of a wave generated from the source until it hits the target.

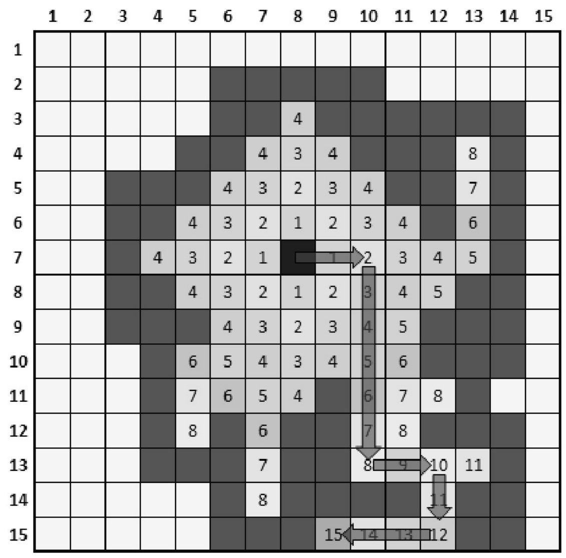
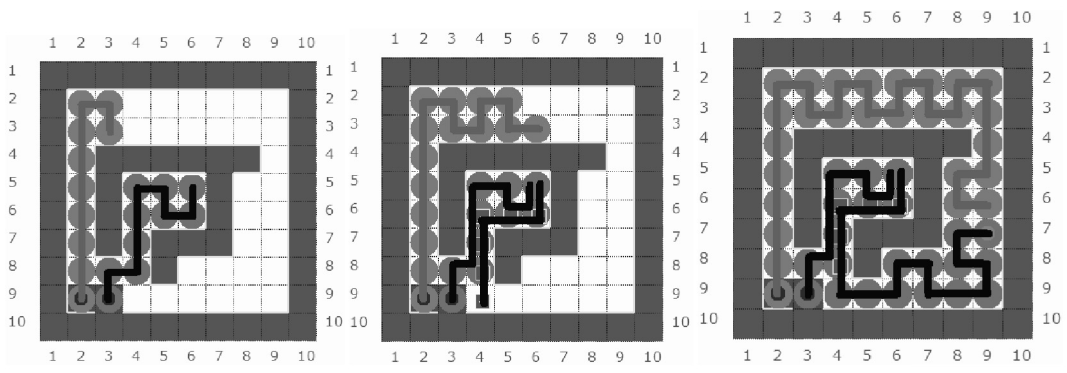


FIGURE 8. The agent, starting from cell $\mathbf{c}_{8,7}$ follows the red line and finds the target at cell $\mathbf{c}_{9,15}$ after 15 steps.

An implementation of the Wave front Algorithm is illustrated in Fig. 8. In the example, the agent (blue cell) is located at cell $c_{8,7}$ initially, while the target cell (green cell) is$c_{9,15}$. At the beginning, the agent has no information about the location of the target. In order to find the shortest path, it generates a wave from its location. The wave is expanded at every time step until the target is hit. In the figure, the numbers on the cells represent the position of the wave at each time step. The distances of the same-numbered cells to the agent are equal. Whenever the target is found, a reverse list of parent-child relations is formed from the target to the agent. For the example; the parent cell for the target is $c_{10,15}$, the parent for cell $c_{10,15}$ is $c_{11,15}$and so on. The list of parents is the shortest path from the agent to the target. According to the list, the first step will be through cell $c_{9,7}$ for this example.

An example scenario is depicted in Fig. 9, where the shortest path algorithm is used to solve the bottleneck when the agent is surrounded by obstacles or covered cells. The algorithm finds the shortest path from the agent to the nearest uncovered cell. During this process, the agent uses the set of C (current global cover map) and finds its way only among the already identified cells. When no empty cell can be identified, the algorithm stops since the whole terrain has been covered.

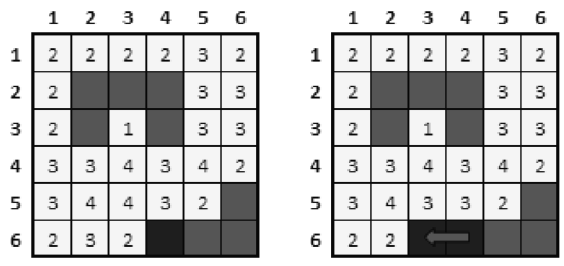


(a) Initial state (b) 9th step (c) Last position

FIGURE 9. (a) Initially, agent $\mathbf{a}_1$ starts from cell $\mathbf{c}_{2,9}$ and constructs a blue line, agent $\mathbf{a}_2$ starts from cell $\mathbf{c}_{3,9}$ and constructs a black line. (b) At the 9th step, agent $\mathbf{a}_2$ is surrounded by obstacles and covered cells, it executes the Wavefront Algorithm and finds the shortest path to the nearest uncovered cell (cell $\mathbf{c}_{4,9}$). (c) After coverage, agent $\mathbf{a}_1$ and $\mathbf{a}_2$ stop at cell $\mathbf{c}_{9,6}$ and cell $\mathbf{c}_{9,7}$ respectively.

3.1.3 An Example Run of the Algorithm

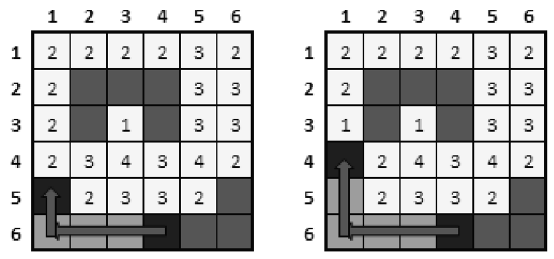
In Fig. 10-12, the algorithm is explained step by step through an example including factors influencing the decision of the agent. The number in a cell corresponds to that cell's degree.



(a) Initial state (b) First step

FIGURE 10. (a) The agent is located at cell $\mathbf{c}_{4,6}$ initially. The downwards direction is closed and the right-side is an obstacle. The degrees of the left and upper cells are 2 and 3 respectively. (b) The agent selects the lower one and moves to cell $\mathbf{c}_{3,6}$ at first step.

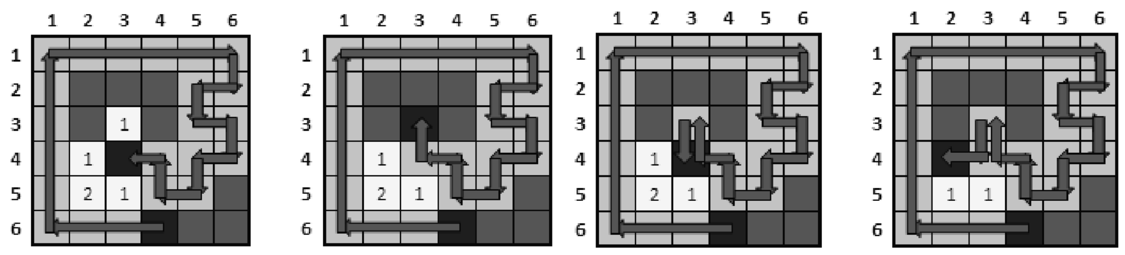
In Fig. 10, the agent is located in the cell $c_{4,6}$ at the beginning and the red cells are obstacles. For the first movement, the agent chooses the cell $c_{3,6}$ since it has two choices (cells $c_{4,5}$ and $c_{3,6}$) and cell $c_{3,6}$ has the minimum degree among them. Notice that the degrees of the cells in the neighborhood of the agent's new location after movement change.



(a) 4th step (b) 5th step

FIGURE 11. The agent is in cell $\mathbf{c}_{1,5}$ after 3 steps. It has 2 possible movement directions, cell $\mathbf{c}_{1,4}$ and cell $\mathbf{c}_{2,3}$, both with degree 2. In case of degree equality, agent selects the next cell in counter clockwise direction, starting from the top.

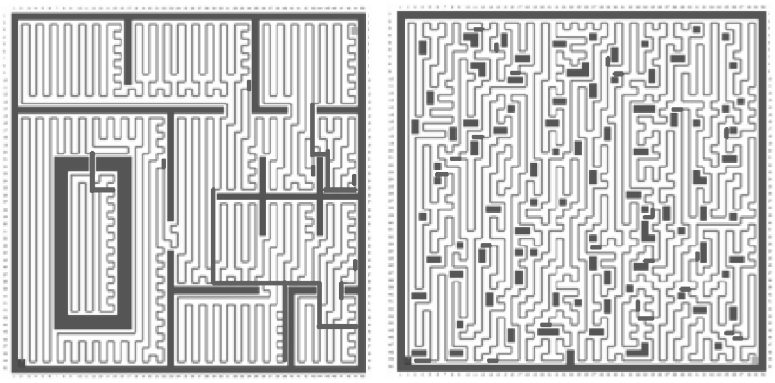
In Fig. 11, the agent is located in the cell $c_{1,5}$ after the 4th step. It has two neighbors, cells $c_{1,4}$ and $c_{2,5}$, both with degree 2. Since the rule of decision in case of equality is counter-clockwise selection, the agent selectscell $c_{1,4}$.



(a) 23rd step (b) 24th step (c) 25th step (d) 26th step

FIGURE12. (a) After 23 steps, the agent is at cell $\mathbf{c_{3,4}}$ and (b) it selects cell $\mathbf{c_{3,3}}$ for the 24th step. In this case, it is imprisoned by obstacles and covered cells. (c) It finds the shortest path to the nearest uncovered cell $\mathbf{c_{2,4}}$ at the 25th step and (d) reaches there by stepping on the previously covered cell $\mathbf{c_{3,4}}$ at the 26th step.

At the 23rd step in Fig. 12, the agent has 3 neighbors with equal degree values $(d(c_{3,3}) = 1, d(c_{3,5}) = 1, d(c_{3,3}) = 1)$ and it selects cell $c_{3,3}$. After that, at the 24th step, it is imprisoned by obstacles and a covered cell, therefore it executes the Wavefront Algorithm and selects $c_{3,4}$.



(a) Indoor-like environment (b) Outdoor-like environment

FIGURE 13. Coverage of indoor and outdoor-like environments. Blue lines shows the re-coverages.

In Fig. 13, there are two examples of Algorithm1executed on 50x50 indoor-like and outdoor-like environments. The red cells denote obstacles or the walls. The agent starts at the bottom-left corner and finishes at the top-right corner for the indoor and bottom-right corner for the outdoor (the green cell). The blue lines are the re-covered cells. For the indoor-like environment, there were 2023 cells uncovered at the beginning. 84 of them are re-covered with a total 2107 steps and 1.0415 coverage rate. For the outdoor-like environment, there were 2149 cells uncovered at the beginning and26 of them are re-covered with a total 2175 steps and 1.0121 coverage rate.

3.1.4 Efficiency of the Algorithm

As mentioned in section 2.3, there are two basic efficiency criteria used for evaluating the grid-coverage algorithms [5, 8, 45]. The first one is the *time rate* (TR - see Eq. 1), which is the ratio of the real completion time over the ideal completion time. The second one is the *coverage rate* (CR - see Eq. 3) which is the ratio of the total number of steps taken by agents over the number of target cells at the beginning. For the single agent case, both values will be the same while they differ for the multi agent case.

A rate closer to 1.0 means better efficiency in terms of minimum re-coverage and minimum time. This implies that increasing number of backtracking or re-coverage decreases the efficiency. The coverage path is accepted as optimal if TR and CR are 1.0 at the end. But it does not mean that if TR is higher than 1.0, it is not optimal. Some cases like blind alley or a closed corridor on the terrain as shown in Fig. 14 results with inevitable re-coverages which is still optimal while $TR > 1$.

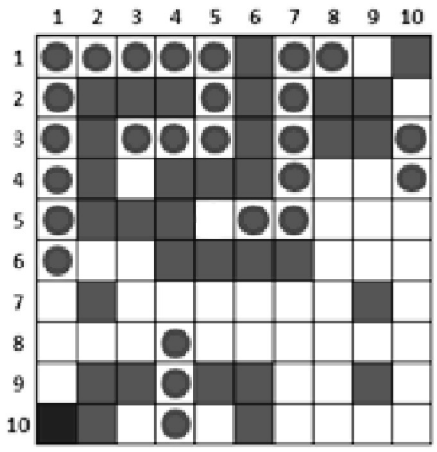
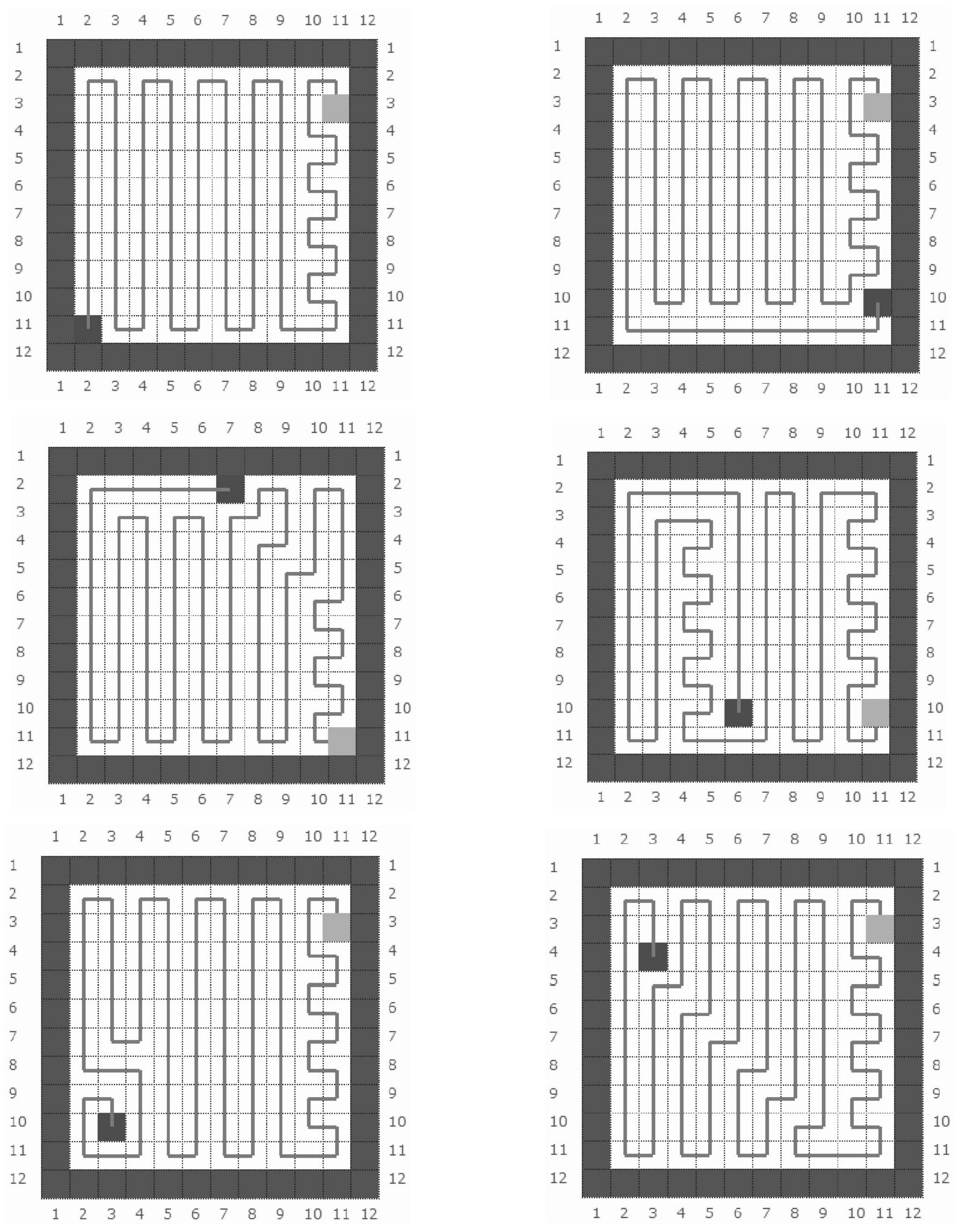


FIGURE 14. The blue cell is agent and filled red circles are to be covered more than once in any case. Therefore, the resulting **TR** and **CR** values have to be more than 1.0, even for the optimal solution.

Independent of the initial location of the agent, MGECA works close to optimal ($TR = 0$) for the single agent case if the area is convex. It is also not affected by the number of columns or rows (odd or even numbered). This is shown by trial of the algorithm with possible initial locations. Some examples of trials and solutions are



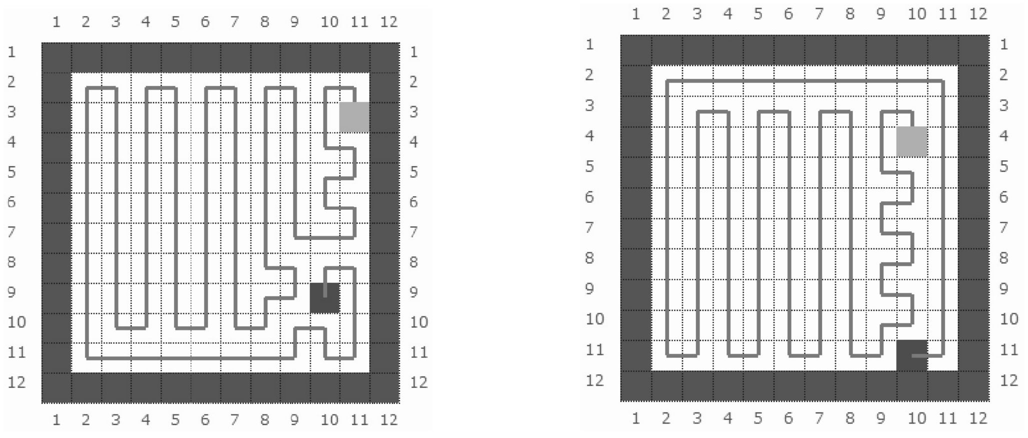


FIGURE 15. Solutions of single-agent on convex area.

Although the algorithm guarantees complete coverage, it does not promise the optimal solution. The NP-Complete structure of the problem [19] does not permit finding the optimal solution for complicated cases where the dimension of the terrain or the number of agents is too high for computation. However, the results of the solution can be compared by using easily solved problems. It is also a legitimate approximation accepting that the solution is close to optimal if TR or CR is close to 1.0.

In order to test the algorithm, we prepared two extreme worst case scenarios below. Each includes compulsory re-coverage.

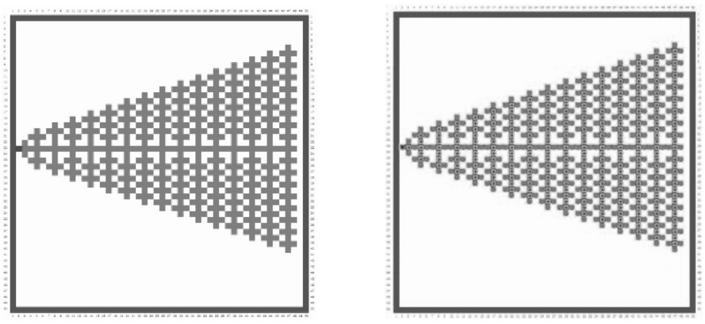


FIGURE 16. Tree-like terrain

In Fig. 16, the agent moves 1162 steps to cover 606 cells with 556 backtracking and the resulting CR is 1.917. The agent starts at the left and covers the green cells.

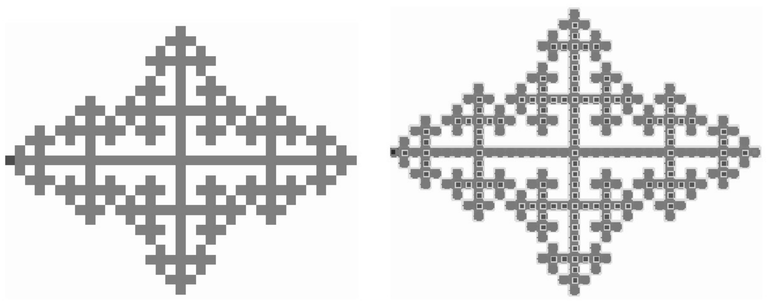


FIGURE17. Fractal-like terrain.

In Fig. 17, the agent starts at the left-most cell and covers the green cells on the fractal-like terrain. The red-filled circles are covered more than once (inevitably 4 times at the top of the branches). The agent moves 472 steps to cover 254 cells with 218 backtracking and the resulting CR is 1.858.

3.2. Multi Agent Coverage with MGECA

The aim of multi robot coverage algorithms is to cover the entire area with minimum backtracking in minimum time while also endeavoring to balance the number of cells each agent covers.

Our multi agent coverage algorithm also guarantees total coverage, decreases the coverage completion time depending on the number of agents and distributes the uncovered cells among the agents at an acceptable level.

The basic property of the algorithm is that all the agents apply the single coverage steps autonomously. However, there are two new cases that we face when we apply the single coverage algorithm. The first is that the agent might be surrounded completely by covered cells or agents after the decision of the preceding agents although it is not imprisoned at the beginning of the turn. For this case, the agent solves the shortest path to the nearest uncovered cell. The agents are permitted to occupy the same cell with another agent in the next step. The second problem is that the nearest cell may change as the other agents move. The solution is choosing only the first cell on the reverse list of parent-child relations and resolving the shortest path at every step in case of imprisonment. The main steps are presented in Algorithm.

ALGORITHM 2. The Multiagent Grid Coverage Algorithm.

```
1.  fFor i=1 to M
2.    for j=1 to N
3.      if $c_{i,j} \in G$
4.        $d(c_{i,j}) \leftarrow (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})$    //Find the degree of each cell
5.        status$(c_{i,j}) \leftarrow UNCOVERED$
6.      else if $c_{i,j} \in A$//A, the set of agents
7.        $s \leftarrow s + 1$//Increment agent index,
8.        $L(a_s) \leftarrow (i,j)$//Set current location (CL) of agent$a_s$ to location (L) of $c_{i,j}$
9.        status$(i,j) \leftarrow COVERED$
10. while completed = FALSE//Main loop
11.   for s=1 to n(S)
12.     $c_{i,j} \leftarrow L(a_s)$
13.     if $d(c_{i,j}) > 0$ // Cell has an uncovered neighbour
14.       $\Omega(c_{i,j}) = \{c_{i-1,j}, c_{i+1,j}, c_{i,j-1}, c_{i,j+1}\}$ //Neighbors of cell $c_{i,j}$
15.       $L(a_s) \leftarrow \arg\min_{c \in \Omega(c_{i,j})} d(c)$//The neighboring cell with the lowest degree.
    Choose in counterclockwise direction starting search from $c_{i,j-1}$ in case of multiple
    minimums
16.       status$(i,j) \leftarrow COVERED$
17.       Update Degree Of Neighbors$(i,j)$
18.     else // $d(c_{i,j}) = 0$
19.       If Find Shortest Path (Nearest Empty Cell)=TRUE
20.         $L(a_s) \leftarrow first\ cell\ through\ location\ of\ nearest\ empty\ cell$
21.       else if Find Shortest Path (Nearest Empty Cell)=FALSE
22.         completed =TRUE
```

At the beginning of execution, the agents tend to move to the top of the terrain if they are not located at the borders or near an obstacle. Next, they move to the left side of the terrain and start covering the area vertically from left to right. Generally, the execution ends at the right side. The reason for this left-to-right coverage behavior is the counter-clockwise selection rule. If the rule were to be applied clockwise, then the tendency of movement would be vertically from right to left.

In case of imprisonment of a cell, the shortest path algorithm is solved to find the nearest uncovered cell. As explained in section 3.1.2, a wave is createdfrom the agent and it is expanded until it hits a target. The distances of all the cells on the wave are the same as the wave expands. The important thing with the execution of the shortest path algorithm for the multi agent case is that the target cellsare searched (clockwise) on each cell on the wave. This clockwise selection prevents the imprisoned agent to select the sametarget cell with its free neighbor agent when it is moving side by side with that free agent.

In Fig. 18, two different cases of algorithm execution are presented on 50×50 terrains. There are 4 agents on the left one and 2 agents on the right. The blue lines represent the backtracked cells.

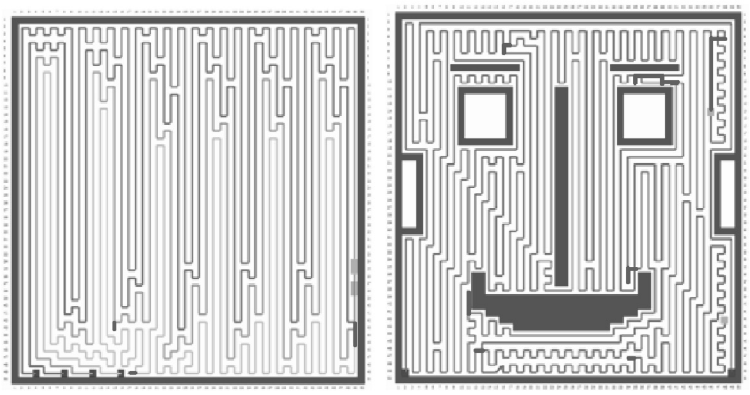


FIGURE 18. Multiagent execution. The paths of each agent are shown in different colors.

4. Evaluation and Results

4.1. Simulation

Here in this section, the performance of the single-agent and multi agent coverage executions of MGECA is evaluated and the results are discussed.

Tests are performed by using software (downloadable at [46]) which we developed. The software allows for constructing terrains with different dimensions, placing obstacles or agents on the terrain and executing MGECA. The obstacles or the agents can be located manually or automatically. In order to be able to compare the results with the studies [7, 8] (explained in Section 2), we used exactly the same three types of 100×100 terrains (empty, outdoor and indoor) with different numbers of obstacles and agents as shown in Fig. 19.

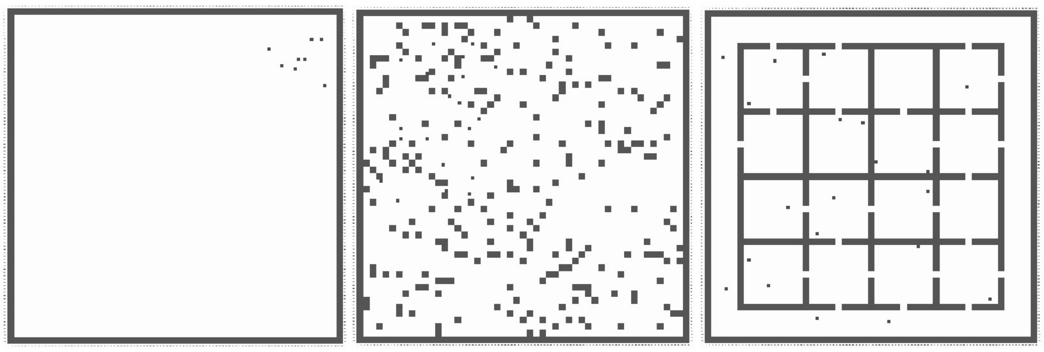

(a) Empty terrain (b) Outdoor-like terrain (c) Indoor-like terrain

FIGURE 19. Example scenarios for each type of terrain. The red zones are obstacles and the blue squares are the agents. (a) 8 agents are clustered with 30% on the empty terrain. (b) 20 agents are clustered with 60% on the outdoor-like terrain and (c) 14 agents are clustered with 100% on the indoor-like terrain. The obstacles on the outdoor-like terrain are located randomly with 10% density. The terrains are the same as the those in [8] for comparability.

Since the initial locations can affect the performance, for better evaluation, exactly in the same manner as in [8], we run 100 runs on each type of the terrain for randomly located 2, 8, 14 and 20 agents and collected the values of completion time, total steps, targets covered, re-coverages made, TR values and total coverages of minimum and maximum covering agents. The rate of obstacles on the outdoor-like terrain is 10% and they are located randomly on the terrain for each replication. The width of the doors and walls on the indoor-like terrain and the size of the obstacles on the outdoor-like terrain is$2D$. In order to detect the effect of grouping, the agents are spread randomly on the terrain with clustering 30%, 60% and 100% of the dimensions. After the completion of 100 runs, the averages of each value are calculated and the maximum and minimum CR values are selected. The numbers of agents, target cells and obstacles for each type of terrain is listed on Table 3.

TABLE 3. Experimental results of MGECA in different conditions

Terrain Type	Terrain Properties	# of agents	Clustering	Ideal Targets per Agent (Eq. 2)	TIME		TOTAL			CR		Coverage	
					Time of Completion	TR (Eq.1)	Total Steps	Total Re-coverage	CR (Eq.3)	Min (Eq. 5)	Max (Eq. 5)	Min	Max
Empty	9602 total, 9602 targets, 0 obstacles, % 0 obstacles	2	30	4801	4837	**1, 0075**	9673	69	**1, 0072**	1, 0004	1, 0242	4789	4815
		2	60	4801	4832	**1, 0065**	9664	59	**1, 0061**	1, 0002	1, 0206	4793	4811
		2	None	4801	4826	**1, 0052**	9651	47	**1, 0049**	1, 0002	1, 0237	4793	4810
		8	30	1200	1261	**1, 0508**	10092	498	**1, 0519**	1, 0187	1, 1045	1151	1240
		8	60	1200	1267	**1, 0558**	10135	550	**1, 0574**	1, 0170	1, 1079	1143	1247
		8	None	1200	1267	**1, 0558**	10141	559	**1, 0583**	1, 0320	1, 0970	1244	1149
		14	30	685	768	**1, 1212**	10750	1167	**1, 1218**	1, 0437	1, 2507	618	745
		14	60	685	766	**1, 1182**	10732	1156	**1, 1207**	1, 0554	1, 2317	619	739
		14	None	685	777	**1, 1343**	10884	1314	**1, 1373**	1, 0933	1, 2361	618	747
		20	30	479	567	**1, 1837**	11363	1782	**1, 1860**	1, 0662	1, 2536	406	542
		20	60	479	568	**1, 1858**	11359	1784	**1, 1863**	1, 0870	1, 3119	401	545
		20	None	479	568	**1, 1858**	11375	1805	**1, 1886**	1, 0995	1, 2973	406	547
Outdoor	9602 total, 8644 targets, 960 obstacles, % 10 obstacles	2	30	4321	4700	**1, 0877**	9399	821	**1, 0957**	1, 0411	1, 1210	4263	4379
		2	60	4321	4725	**1, 0935**	9449	870	**1, 1014**	1, 0578	1, 1337	4270	4373
		2	None	4321	4668	**1, 0803**	9335	751	**1, 0875**	1, 0333	1, 1450	4264	4378
		8	30	1079	1338	**1, 2400**	10708	2192	**1, 2574**	1, 1115	1, 3660	1003	1118
		8	60	1079	1350	**1, 2512**	10801	2251	**1, 2633**	1, 0856	1, 3641	997	1149
		8	None	1079	1330	**1, 2326**	10640	2109	**1, 2472**	1, 0902	1, 4012	977	1142
		14	30	616	839	**1, 3620**	11750	3285	**1, 3881**	1, 1612	1, 6034	533	662
		14	60	616	831	**1, 3490**	11645	3159	**1, 3723**	1, 1807	1, 5884	530	681
		14	None	616	836	**1, 3571**	11680	3193	**1, 3762**	1, 1596	1, 5361	527	685
		20	30	431	626	**1, 4524**	12529	4075	**1, 4820**	1, 1782	1, 8255	336	514
		20	60	431	630	**1, 4617**	12606	4140	**1, 4890**	1, 2031	1, 9328	329	523
		20	None	431	632	**1, 4664**	12646	4164	**1, 4909**	1, 2077	1, 9227	339	513

Indoor	9602 total, 8164 targets, 1440 obstacles, % 8, 4 obstacles	2	30	4081	4469	**1, 0951**	8138	798	**1, 1087**	1, 0468	1, 1496	4027	4111
		2	60	4081	4453	**1, 0912**	8141	765	**1, 1037**	1, 0512	1, 1337	4035	4106
		2	None	4081	4465	**1, 0941**	8138	793	**1, 1080**	1, 0595	1, 1347	4033	4105
		8	30	1020	1347	**1, 3206**	10777	2686	**1, 3320**	1, 1680	1, 4875	906	1164
		8	60	1020	1320	**1, 2941**	10559	2459	**1, 3036**	1, 1533	1, 3993	901	1164
		8	None	1020	1287	**1, 2618**	10302	2187	**1, 2695**	1, 1180	1, 3875	924	1126
		14	30	582	895	**1, 5378**	12540	4484	**1, 5566**	1, 3273	1, 8423	748	436
		14	60	582	865	**1, 4863**	11988	3990	**1, 4989**	1, 2600	1, 8923	452	740
		14	None	582	818	**1, 4055**	11460	3335	**1, 4105**	1, 1575	1, 8643	473	716
		20	30	407	691	**1, 6978**	13829	5773	**1, 7166**	1, 4723	1, 9523	231	582
		20	60	407	657	**1, 6143**	13142	5077	**1, 6295**	1, 3547	1, 9316	298	647
		20	None	407	627	**1, 5405**	12554	4444	**1, 5480**	1, 2518	1, 9818	287	557

We make the following observations with the examination of the results in Table 3:

- In each of three cases, the time for completion decreases but the rates increase as the number of agents increases.
- The TR and CR values of the algorithm increases with increasing number of agents.
- The algorithm works best for the empty terrain case. The ratios are almost 1.0 and optimal for 2 agents.
- The results are similar for the outdoor and indoor terrain cases.
- There is no significant difference between the TR and CR rates for all cases.
- Since the agents have to make inevitable backtracking when they finish covering each room, the smallest TR value for indoor-like terrain is around 1.09.
- On the empty and outdoor cases, the values of 30, 60 and 100 percent clustering are almost identical. Apart from the outdoor-like terrain, clustering seems to affect the results when the number of agents increases.
- The difference between minimum and maximum TR values is significant. This shows that the initial locations of targets and obstacles affect the results.

As a summary, the values show that the algorithm performs close to optimal when the number of agents is small and the time of complete coverage decreases as expected when the number of agents increases.

4. 2. Comparison with Existing Algorithms

TABLE 4. Test Values of MGECA (the proposed method), MFC and MSTC.

# of Agents	Clustering	EMPTY						OUTDOOR						INDOOR					
		MGECA		MFC		MSTC		MGECA		MFC		MSTC		MGECA		MFC		MSTC	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
2	30	**1, 01**		1, 02		1, 10		1, 09		**1, 01**		1, 09		1, 09		**1, 02**		1, 09	
		4789	4815	4730	4877	5048	5269	4263	4379	4268	4379	4574	4695	4027	4111	4015	4171	4230	4468
2	60	**1, 01**		1, 02		1, 13		1, 09		**1, 01**		1, 11		1, 09		**1, 03**		1, 13	
		4793	4811	4719	4885	5095	5445	4270	4373	4265	4381	4627	4778	4035	4106	3994	4194	4290	4621
2	100	**1, 01**		1, 02		1, 15		1, 08		**1, 01**		1, 12		1, 09		**1, 02**		1, 14	
		4793	4810	4723	4886	5161	5529	4264	4378	4268	4376	4525	4842	4033	4105	4014	4171	4166	4663
8	30	**1, 05**		1, 16		3, 13		1, 24		**1, 17**		3, 24		1, 32		**1, 20**		3, 19	
		1151	1240	837	1396	38	3752	1003	1118	788	1260	18	3500	906	1164	849	1225	12	3262
8	60	**1, 06**		1, 18		2, 89		1, 25		**1, 18**		2, 93		1, 29		**1, 18**		2, 98	
		1143	1247	902	1414	77	3462	997	1149	789	1274	59	5158	901	1164	846	1202	44	3042
8	100	**1, 06**		1, 16		2, 68		1, 23		**1, 15**		2, 80		1, 26		**1, 17**		2, 84	
		1244	1149	953	1391	127	3210	977	1142	871	1243	76	3016	924	1126	839	1199	90	2905
14	30	**1, 12**		1, 22		5, 38		1, 36		**1, 23**		5, 51		1, 53		**1, 32**		5, 47	
		618	745	431	836	2	3685	533	662	451	760	3	3392	436	748	439	768	2	3192
14	60	**1, 12**		1, 19		8, 00		1, 35		**1, 21**		5, 12		1, 48		**1, 27**		5, 14	
		619	739	522	815	8	3387	530	681	481	745	13	3156	452	740	453	741	11	2999
14	100	**1, 13**		1, 20		4, 38		1, 36		**1, 20**		4, 46		1, 40		**1, 24**		4, 31	
		618	747	511	824	25	3002	527	685	463	741	26	2784	473	716	445	725	23	2517
20	30	**1, 18**		1, 27		7, 54		1, 45		**1, 32**		7, 80		1, 69		**1, 49**		7, 81	
		406	542	307	609	1	3612	336	514	281	567	2	3362	231	582	242	608	1	3188
20	60	**1, 19**		1, 25		7, 02		1, 46		**1, 28**		7, 11		1, 61		**1, 39**		7, 02	
		401	545	332	599	4	3364	329	523	285	552	5	3066	298	647	271	566	5	2866
20	100	**1, 19**		1, 25		5, 84		1, 47		**1, 27**		6, 20		1, 54		**1, 32**		5, 75	
		406	547	319	599	9	2796	339	513	294	547	12	2674	287	557	277	540	11	2348

We compared the results of MGECA with Multi Robot Forest Coverage (MFC) and Multi Robot Spanning Tree Coverage (MSTC) of which experimental results are presented by Zheng and Jain in [8]. In their evaluation, they used the same parameters (i.e., size, shape and type of the terrain, the rate of the obstacles, the rate of clustering, the shape of rooms for outdoor-like terrain and the number of total target cells) and collected the same statistical values (i.e., the time and rate of completion and the values of best and worst agents). In Table 3, "TR (Eq. 1)", "$max\ CT = \max\{t_a\}_{a=1}^{|A|}$" and "$min\ CT = \min\{t_a\}_{a=1}^{|A|}$" values of the algorithms are presented.

From Table 4, we observe the following:

- The results of MGECA are the best on the empty terrain case while MFC works best for outdoor and indoor cases.
- The rates of MSTC are significantly the worst almost for all cases compared to MGECA and MFC.
- The difference between values of 30%, 60% and 100% clustering is not so high for MFC and MGECA while it is too high for MSTC. This shows that MFC and MGECA are more consistent compared to MSTC and affected less from clustering.
- The difference between minimum and maximum values is too high for MSTC. The smallest for most cases is MGECA. This shows that MGECA distributes the cells to the agents better.

The cover time of MGECA is close to the existing algorithms for most cases and it is consistent with the rates not increasing quickly with the increasing number of obstacles and the agents. It is also possible to increase the performance of MGECA by adding new rules for different types of terrains, but we prefer to solve the problem with simple rules and not to modify it for special cases. In addition to the numerical results, there are also some other criteria that affect the performance of an algorithm. The properties which distinguish MGECA are explained in section 5 below.

5. Advantages of MGECA

5.1 Online

The most important property and advantage of MGECA is that it works online since the agents of MGECA make their decisions at each step based on the degrees of the cells surrounding the agents.

5.2 Fine Resolution

The terrain resolution is an important characteristic of grid coverage algorithms, which affects the rate of unnecessary coverage (see Fig. 20). The resolution of MGECA is better than the other algorithms, since the area is decomposed to $D \times D$ cells, whereas it is $2D \times 2D$ for MFC and MSTC. An example of grid decomposition with $D \times D$ and $2D \times 2D$ approximations is shown in Fig. 20.

Suppose that zones A and B are to be covered and the total area is $20D \times 20D$ while the size of the coverage tool of the agent is $D \times D$. When the zones are decomposed into $2D \times 2D$ large squares for MFC and MSTC, the number of total small squares is 92 for Zone A and 56 for Zone B. After the decomposition to $D \times D$ cells for MGECA, it is 77 for Zone A and 38 for Zone B. These numbers mean that the agents in MFC and MSTC have to move 15 steps more in Zone A and 18 steps more in Zone B than the agents in MGECA.

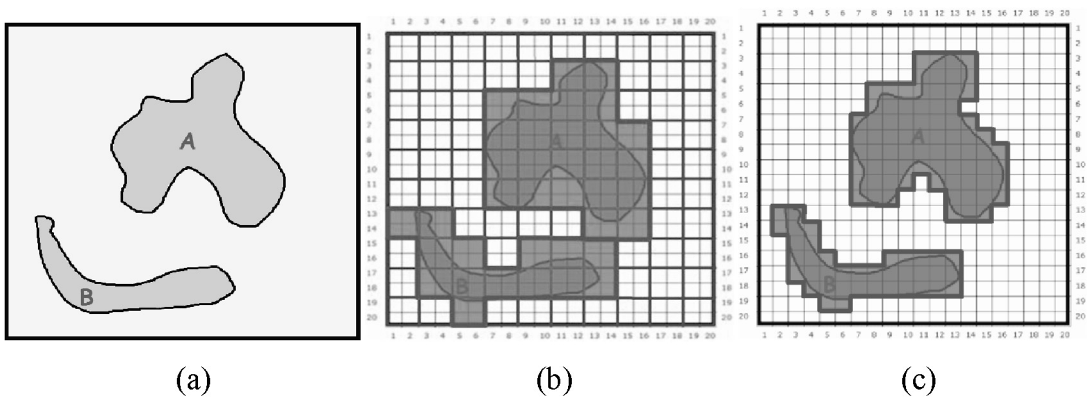


FIGURE 20. The coverage area with zones A and B is in (a). The $\mathbf{2D \times 2D}$ decomposition of the zones is in (b) and $\mathbf{D \times D}$ decomposition which is used by MGECA is in (c).

The solution of the $D \times D$ decomposition with MGECA is shown in Fig. 21. The agent starts from the blue cell, follows the brown line, backtracks on the red line and finishes at the red cell. Total backtracking on Zone A is 4 and it covers the area in 81 steps while it covers Zone B in 42 steps with 4 backtracking. The execution results show that MGECA covers the area for the cases on the example in shorter time than MFC and MSTC even if they cover the area optimally (92 steps for Zone A and 56 steps for Zone B) without any backtracking.

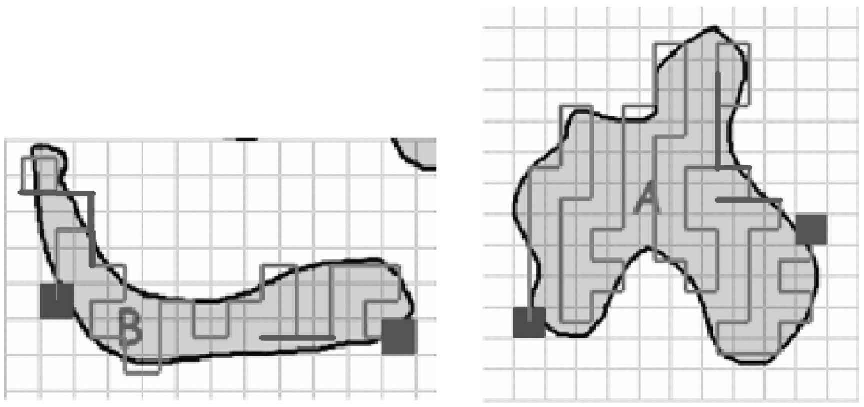


FIGURE 21. MGECA coverage solution for Zones A and B. Backtracked cells are shown with a red line.

5.3 Failsafe

The behavior of the algorithm in case of failure of any agent is also another important factor that is to be determined when comparing the coverage algorithms. In spite of the MFC results with the smallest TR values for indoor and outdoor-like terrains, it is not robust in the presence of failure of any agents [8]. However, MGECA and MSTC are still robust in case of agent failure. MGECA, especially, still continues working and guarantees complete coverage if all the agents except one fail.

5.4 Performance

Since there is not any standard test environment for coverage algorithms yet, we compared the results of MGECA with the results of MFC and MSTC [8]. The performance of MGECA is the best for empty terrain and it distributes the area better to the agents since the difference between the max and min values are smallest. In other scenarios, MGECA performs comparably to MFC and MSTC.

5.5 Dynamic Terrain

The terminating condition of MGECA is the completion of the coverage of all cells on the terrain and it is checked by the agents by executing the Wavefront Algorithm. This means that the algorithm stops when any of the agents cannot find any uncovered cell on the terrain. This provides the algorithm with the capability of adding new target cells or changing the status of the covered cells to “uncovered”. MGECA will be still working if ever any change on the terrain occurs.

5.6 Initial Location

The execution of MGECA is completely independent from the initial locations of the agents. It is not necessary to locate initially the agents near the border, near an obstacle or near another agent. The agents are not supposed to know about the initial locations of other agents as it is necessary for the agents in BSA-CM [6].

6. Conclusion and Future Work

In this paper, we proposed an area coverage algorithm which is suitable for planning paths for mobile agents assigned to cover an area. The algorithm is based on observing the cells in the neighborhood, selecting the cell with smallest degree (a parameter which indicates the amount of uncovered neighbor cells) and finding the shortest path to the nearest uncovered cell when there is not any uncovered cell around. The basic advantages of MGECA are that (i) it works online, (ii) it decomposes the terrain with a fine resolution, (iii) it is failsafe, (iv) the numeric results are comparable to the existing Multi Forest Coverage and Multi-Robot Spanning Tree Coverage algorithms, (v) it responds to the changes in the environment during execution (vi) and the agents are not limited to the initial locations.

We evaluated MGECA and compared it with other algorithms on a testbed (available online [46]) which we developed. The infrastructure and testbed environment are open to development and testing new coverage algorithms and are suitable for more complex multi agent operations.

Since the agents are generally expected to perform extra tasks (e.g., observing with another sensor, communicating, operating its additional tools) depending on its duty while moving, the algorithm’s structure has to suffice the requirements. The framework of MGECA is suitable for adding new properties for the agents. As future work, we plan to add new decision strategies for both path planning and execution of additional tasks.

It is also possible to add learning, bidding and more cooperating abilities to the agents in the MGECA framework.

The representation of terrain is another subject that we plan to handle in the future. We plan to increase the resolution and decompose the terrain into hexagonal grids, adapt MGECA for that and compare the results with the squared decomposition. Since there would be some places of higher importance that are to be covered with priority in some realistic applications, this need has to be satisfied inside the algorithm. Generally, this problem is solved by assigning weight values to the cells in an approximate cellular decomposition approach. We will also try to adapt MGECA for weighted terrain for both squared and hexagonal cells.

The main aim of MGECA was to solve the coverage problem in minimum time with minimum backtracking while running online. It is also possible to improve the online results by optimizing with some offline heuristic operations.

As a result, we believe that MGECA will contribute to multi agent path planning studies and that it is applicable for path planning of unmanned and autonomous systems for its simple, fast, online and improvable structure.

References

[1]. A. Zelinsky, R.A. Jarvis, J. C. Byrne and S. Yuta, Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot, *Proceedings of the International Conference on Advanced Robotics*, (1993), 533–538.

[2]. Y. Gabriely, E. Rimon, Spanning-tree based coverage of continuous areas by a mobile robot, *Annals of Mathematics and Artificial Intelligence*, **31**, (2001), 77–98.

[3]. E. Gonzales, M. Alarcon, P. Aristizabal, C. Parra, BSA: A Complete Coverage Algorithm. *International Conference on Robotics and Automation*. (2005), 2040–2044.

[4]. E. Gonzales, A. Suarez, C. Moreno, F. Artigue, Complementary Regions: a Surface Filling Algorithm, *International Conference on Robotics and Automation*, (1996), 909-914.

[5]. X. Zheng, S. Koenig, D. Kempe, S. Jain, Multi-Robot Forest Coverage for Weighted and Unweighted Terrain, *IEEE Transactions on Robotics*, **26**(6), (2010), 1018-1031.

[6]. E. Gerlein, E. Gonzales, Multirobot Cooperative Model Applied to Coverage of Unknown Regions, Multi-Robot Systems, Trends and Development, Dr Toshiyuki Yasuda (Ed.), (2011).

[7]. N. Hazon, G. A. Kaminka, Redundancy, Efficiency and Robustness in Multi-Robot Coverage, *International Conference on Robotics and Automation*, (2005), 735-741.

[8]. X. Zheng, S. Jain, S. Koenig, D. Kempe, Multi-robot forest coverage, *International Conference on Intelligent Robots and Systems*, (2005), 2318–2323.

[9]. I. A. Wagner, M. Lindenbaum, A. M. Bruckstein, Distributed covering by ant-robots using evaporating traces, *IEEE Transactions on Robotics and Automation*, **15**(5), (1999), 918–933.

[10]. Y.H. Choi, T. K. Lee, S. H. Baek, S. Y. Oh, Online Complete Coverage Path Planning for Mobile Robots Based on Linked Spiral Paths Using Constrained Inverse Distance Transform, *International Conference on Intelligent Robots and Systems,* (2009), 5788-5793.

[11]. P. Doherty, J. Kvarnström, F. Heintz, A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems, *Autonomous Agents and Multi Agent Systems.* **19**(3), (2009), 232-377.

[12]. H. Sahin, L. Guvenc, Household robotics: autonomous devices for vacuuming and lawn mowing [Applications of control], *Control Systems IEEE,* **27**(2), (2007), 20-96.

[13]. L.P. Kalra, J. Gu, An autonomous self-contained wall climbing robot for non-destructive inspection of above-ground storage tanks, *Industrial Robot: An International Journal,* **34**(2), (2007), 122 – 127.

[14]. T. Palleja, M. Tresanchez, M. Teixido, J. Palacin, Modeling floor-cleaning coverage performances of some domestic mobile robots in a reduced scenario, *Robotics and Autonomous Systems.* **58**(1), (2010),37-45.

[15]. E.J. van Henten, J. Hemming, B.A.J. van Tuijl, J.G. Kornet, J. Meuleman, J. Bontsema, E.A. van Os, An autonomous robot for harvesting cucumbers in greenhouses, *Autonomous Robots* **13**, (2002), 241-258.

[16]. H. Najjaran, A. Andrew, Landmine detection using an autonomous terrain-scanning robot *Industrial Robot: An International Journal* **32**(3), (2005), 240–247.

[17]. E.H. Spafford, D. Zamboni, Intrusion detection using autonomous agents, *Computer Networks,* **34**(4), (2000), 547-570.

[18]. J. S. Jennings, G. Whelan, W. F. Evans, Cooperative search and rescue with a team of mobile robots, *International Conference on Advanced Robotics*, (1997), 193-200.

[19]. D. P. Ariel, Robotic Coverage, Mathematical Foundations of AI, Lecture 2, (2008).

[20]. H. Choset, Coverage for robotics-a survey of recent results, *Annals of Mathematics and Artificial Intelligence* **31**, (2001), 113-126.

[21]. S. Hert, S. Tiwari, V. Lumelsky, A terrain-covering algorithm for an AUV, *Autonomous Robots*, **3**, (1996), 91–119.

[22]. V .J. Lumelsky, S. Mukhopadhyay, K. Sun, Dynamic Path Planning in Sensor-Based Terrain Acquisition, *IEEE Trans on Robotics and Automation*, **6**(4), (1990), 462-472.

[23]. L. Jean-Claude, Robot Motion Planning, Kluwer Academic Publishers, (1991).

[24]. H. Choset and P. Pignon, Coverage path planning: The boustrophedon cellular decomposition, International Conference on Field and Service Robotics, (1997).

[25]. A. Noa, H. Noam, A.K. Gal, Constructing spanning trees for efficient multi-robot coverage, International Conference on Robotics and Automation, (2006), 1698–1703.

[26]. Y Gabriely, E. Rimon, Competitive online coverage of grid environments by a mobile robot, *Computational Geometry: Theory and Applications*, **24**(3), (2003), 197-224.

[27]. G. Chalkiadakis, C. Boutilier, Sequentially optimal repeated coalition formation under uncertainty, *Autonomous Agents and Multi Agent Systems,* **24**(3), (2012), 441-484.

[28]. S. Ichikawa, Characteristics of object-searching and object fetching behaviors of multi-robot system using local communication, International Conference on Systems, Man, and Cybernetics, (1999), 775–781.

[29]. X. Deng, C. H. Papadimitriou, Competitive distributed decision-making, *Algorithmica*, **16**(2), (1992), 350–356.

[30]. G. Dudek, M. Jenkin, E. Milos, D. Wilkes, Robotic exploration as graph construction, *IEEE Transactions on Robotics and Automation,*7(6), (1991), 859-865.

[31]. R. Menezes, F. Martins, F. E. Vieira, R. Silva, M. Braga, A model for terrain coverage inspired by ant's alarm pheromones, *ACM Symposium on Applied Computing*, (2007), 728–732.

[32]. A. Felner, Y. Shoshani, Y. Altshuler, A. Bruckstein, Multi agent Physical A* with Large Pheromones, *Autonomous Agents and Multi Agent Systems*, **12**(1), (2006), 3-34.

[33]. D. Spears, W. Kerr, W. Spears, Physics-based robot swarms for coverage problems, *International Journal on Intelligent Control and Systems*, **11**(3), (2006), 124–140.

[34]. P. Fazli, A. Davoodi, A. K. Mackworth, Multi-robot repeated area coverage,*Autonomous Robots*, **34**(4), (2013), 251-276.

[35]. P. Menzel, F. D. Aluisio, Robo Sapiens: Evolution of a New Species, MIT Press, (2001).

[36]. T. Balch, The case for randomized search, *IEEE International Conference on Robotics and Automation,* (2000).

[37]. Z.L. Cao, Y. Y. Huang, E. L. Hall, Region Filling Operations with Random Obstacles Avoidance for Mobile Robots, *Journal of Robotics Systems*, 5(2), (1988), 87-102.

[38]. Y. Altshuler, A. B. Bruckstein, Static and expanding grid coverage with ant robots: Complexity results, *Theoretical Computer Science*, **412**(35), (2011), 4661-4674.

[39]. K.S. Senthilkumar, K. K. Bharadwaj, Multi-robot exploration and terrain coverage in an unknown environment, *Robotics and Autonomous Systems*, **60**(1), (2012), 123-132.

[40]. N. Hazon, F. Mieli, G. A. Kaminka, Towards Robust On-line Multi-Robot Coverage, *International Conference on Robotics and Automation*, (2006), 1710-1715.

[41]. N. Agmon, N. Hazon, G. A. Kaminka, The giving tree: Constructing trees for efficient offline and online multi-robot coverage, *Annals of Mathematics and Artificial Intelligence*, **52**(2-4), (2008), 143–168.

[42]. E.W. Dijkstra, A Note on Two Problems in Connexion with Graphs. Numerische Mathematik,**1**, (1959), 269-271.

[43]. S. Moshe, Dijkstra's algorithm revisited: the dynamic connexion, *Control and Cybernetics*, **35**(3), (2006), 599-620.

[44]. M.G. Zhang, B. J. Cheng, X. F. Li and M. Y. Wang, A fast algorithm of shortest path ray tracing, *Chinese Journal of Geophysics*. **49**(5), (2006), 1315-1323.

[45]. S.G. Shuzhi and F. Cheng-Heng, Complete Multi-Robot Coverage of Unknown Environments with Minimum Repeated Coverage, *International Conference on Robotics and Automation*, (2005), 715-720.

[46]. A software platform for multi agent area coverage. (2013)
http://kovan. ceng. metu. edu. tr/~sinan/mgeca/ Last access: 27 December 2013.