A motion planning method for mobile robot considering rotational motion in area coverage task

Yano Taiki[1,a)] Takase Hideki[1] Takagi Kazuyoshi[1] Takagi Naofumi[1]

Abstract: In recent years, utilization of mobile robot in area coverage tasks has been studied. In the case of area coverage by a single robot, it is possible to obtain a traveling route for covering the entire workspace by constructing a spanning tree on a grid graph representing the workspace. The tour route obtained by the above method is the shortest one in terms of the route length. However, it is not optimal in terms of time efficiency when we consider the rotation behavior of the robot. In this research, we focus on the rotational motion on the route. We propose a method for constructing the spanning trees on the lattice graph to derive the optimal time efficient path. Proposed method further searches the optimal tree using *compressed representation* of the graphs considering rotational motion.

Keywords: mobile robot, path planning, spanning tree

1. Introduction

In recent years, utilization of mobile robot having high mobility has been studied. There are many applications performed by mobile robot such as floor cleaning, security patrol, facility maintenance, and agricultural support. In such examples, mobile robot has to visit all points of target workspace while performing the work. Such works are called area coverage tasks. Also, the problem of finding a route in the area coverage task is called the area coverage problem.

Various studies have been carried out to derive the shortest path in the area coverage problem. These studies focus on coverage rate of the area and less duplication of the path. In particular, in the area coverage problem by a single robot, it is known that one of the shortest paths can be obtained by considering a spanning tree on the lattice graph representing the target workspace[1]. Complete coverage without duplication is achieved by this path.

On the other hand, when considering the actual traveling time of the robot on the path, it is insufficient to consider the coverage rate and less duplication of the path. Actually, due to the rotational motion on the path, there is a difference in the traveling time even among the paths having the same length. Therefore, in order to obtain the path with the shortest traveling time, a problem model considering rotational motion on the path is required. However, in a problem model using a lattice graph representation, there is a difficulty that the number of solutions to be searched explosively increases in order to obtain an optimal solution as the size of the graph increases. In order to accurately estimate the actual traveling time on the path, methods that utilize a more realistic problem model have been studied. In these methods, there is a difficulty that the path length is greatly increased due to a slightly unreached area[2]. Therefore, instead of getting a strict optimal solution, many researches using heuristic methods has been conducted[3]-[6].

The purpose of this research is to determine the optimal traveling path in terms of time efficiency while maintaining the complete area coverage. In order to achieve the purpose, we propose compressed representation of graphs focusing on rotational motion and a fast search algorithm for the optimal traveling path on a two-dimensional plane for a single robot.

In the proposed algorithm, a spanning tree on a lattice graph representing a workspace is searched in order to obtain a path covering the entire area. In this search process, the search is performed paying attention to the number of rotational motions on the path in order to minimize traveling time. In the case of a single robot, the path with the shortest traveling time can be obtained by minimizing the number of rotational motions on the path while maintaining the shortest path length. In addition, by using compressed representation of graphs which can represent multiple spanning trees with the same number of rotations and similar tree structure, it is possible to obtain an optimal solution with a realistic calculation time even for a large size problem.

The structure of this paper is as follows. First we introduce the existing research in section 2 and describe the problems. Next, in section 3, we model mathematically the area coverage problem and consider its characteristics. In section 4, we propose a spanning tree search algorithm using the model given in section 3 and discuss the computational complexity and validity of the algorithm. Section 5 will conclude the overall discussion and future works.

2. Related Work

We introduce existing researches on area coverage tasks. In the early studies, a method of randomly moving the robot within the target area has been studied[7]. Since this method does not consider the already reached area, duplication occurs in the path. There are also methods for a single robot to obtain a path covering the entire workspace without duplication in the path[1], [8].

[1] Graduate School of Informatics, Kyoto University
[a)] emb@lab3.kuis.kyoto-u.ac.jp

In these methods, the workspace is modeled as a lattice graph.

Especially, the method in [1]: Spanning Tree Coverage (STC) can obtain the shortest path by considering the spanning trees on the lattice graph. However, since this method does not take into account the rotational motion on the path, it is not always possible to obtain an optimal route in terms of time efficiency. The method in paper [2] aimed at minimizing the rotational motion on the path. In this method, it is difficult to obtain an optimal solution since it uses a problem model that is more realistic than a grid graph and there is a trade-off relation between the coverage rate and the path length.

There are methods of finding the routes of multiple robots by applying STC[9], [10]. In paper [10], it is shown that an area coverage problem by multiple robots is NP complete. Therefore, these methods are heuristic methods, and it is not always possible to obtain optimal solutions.

Attempts to heuristically search solutions using genetic algorithms have also been conducted for either a single robot or multiple robots[3]-[6].

The existing researches described above provide methods of deriving a complete coverage path passing through all the areas in the workspace for the single robot area coverage problem. In these researches, methods of finding a path with the shortest path length has been proposed. In case of considering time efficiency or power efficiency, heuristic methods has been proposed rather than a method for guaranteeing the optimal solution. In the case of multiple robots, it is possible to derive a path covering the entire workspace by the applying STC. In addition, it is shown that an area coverage problem in the case of multiple robots is NP complete, and methods using genetic algorithms have been proposed. However, with these methods, there is a possibility that the time required for the search becomes impractical or there is a possibility that the solution is not optimum.

3. Area Coverage Problem Considering Rotational Motion

In this section, we define the area coverage problem by a single robot. Then we give a mathematical model of the problem and explain its characteristics.

3.1 Area Coverage Task

First, the following five assumptions about the target space of the area coverage task are given.

- The target space can be divided into equal-sized square cells.
- The square cells are arranged in a lattice pattern.
- The target space is connected, and there is always a route between two arbitrary cells.
- Square cells can be divided into four equal-sized subcells.
- The size of the subcell is the size that the robot can cover in a unit time.

This means that the target workspace can be represented as a lattice graph by considering the center of each square cell as a vertex. In addition, it shows that we can obtain a shortest path passing through the whole space from the spanning trees on the lattice graph. **Fig. 1** shows an example of target workspace. The example workspace contains 36 square cells. 6 square cells are blocked by obstacles, and 30 square cells are free. These cells correspond to vertices of lattice graph. There are edges only between vertices correspnding to adjacent free square cells.

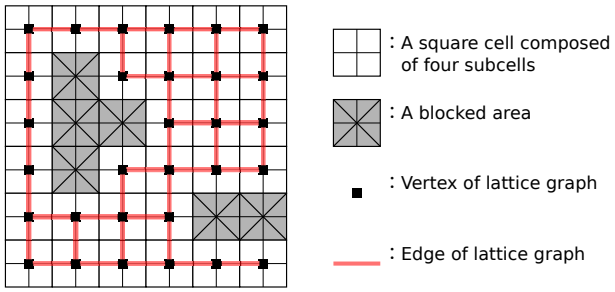

Fig. 1 Example of the problem.

Next, the following two motions are given as the behavior of the target robot: 90 degrees rotation and rectilinear movement. The rotational motion performed by the robot is 90 degrees rotation to the right or left without rectilinear movement. The rectilinear movement performed by the robot is an operation of moving straight a certain unit distance. This distance is equal to the length of a robot and also the distance between two adjacent subcells. In addition, the time required to perform a unit of rotational motion is shorter than the time required to perform a unit of rectilinear movement.

Under these assumptions, the area coverage problem is defined as a problem of finding a closed path passing through all subcells for given target workspace.

The optimal solution in this problem is the traveling path that gives the minimum coverage time. In this paper, the coverage time represents the time required to complete the area coverage task. The coverage time is determined by the number of rectilinear movements and the number of rotational motions on the path.

3.2 Formulation of The Problem

The area coverage problem described in Section 3.1 is modeled as a graph problem to obtain a spanning tree on a given lattice graph. First, notation of constants and variables are given as shown in **Table 1**. The problem model is given as follows by using the defined notation.

- `Instance:` Lattice Glaph $G = (V,\ E)$
- `Solution:` Spanning Tree $T = (V,\ E_t)$
- `Optimization Goal:` $\min\{Cost(T)\}$

Each of the instances, solution and optimization goal are described in detail below.

The instance of problem is the grid graph G reflecting the ter-

Table 1 Definition of constants and variable symbols.

Notation	Definition
G	Input graph. $G = (V, E)$
V	Set of vertices.
E	Set of edges.
T	Spanning tree. $T = (V, E_t),\ E_t \subseteq E$
$Cost(T)$	$Cost(T) = 4C_S\|\|V\|\| + C_R R_T$
C_S	Unit amount of the cost for rectilinear movement by robot.
C_R	Unit amount of the cost for 90 degree rotation movement by robot.
R_T	Sum of the number of 90 degree rotation movement in a path corresponding to tree T.

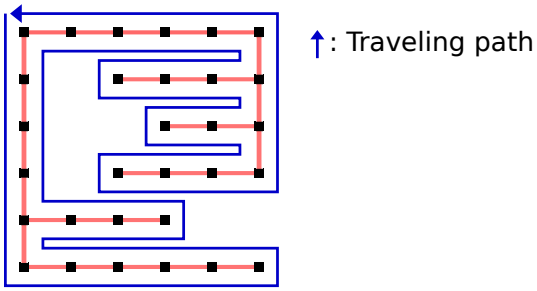

Fig. 2 Examples of the solution for the problem of Fig. 1 and the robot traveling path.

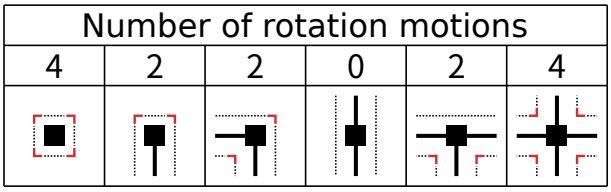

Fig. 3 Classification of vertices on lattice.

rain data of the target area as shown by the red line and the black square in Fig. 1. The vertices of the lattice graph correspond to the square cells on the target area. The edges represent the connection between square cells, and edges exist only between square cells that can pass through.

The solution of this problem is a spanning tree of the graph. As shown in **Fig. 2**, it is possible to derive the traveling path of the robot along the circumference of the spanning tree. Therefore, if a spanning tree can be obtained on the graph, it is possible to derive a traveling path covering the entire target area.

The optimization goal of this problem is minimizing the cost of the spanning tree obtained as a solution. The cost of a tree is calculated as the sum of two costs, a cost proportional to the number of vertices ($4C_S||V||$) and a cost proportional to the number of rotations on the robot path ($C_R R_T$). The former cost is corresponding to the length of the traveling path. Since one vertex in the spanning tree corresponds to four subcells on the robot path, four times the number of vertices is given as a value corresponding to the path length. The latter cost is corresponding to the number of rotations performed by the robot on the path. The rotaion is defined with a 90 degrees rotation as the unit. The number of rotations around each vertex is determined by the degree of the vertex and the connection pattern of the edge. **Fig. 3** shows all patterns of vertex degrees and edge connections. From left to right in the figure, the vertex of degree 0 to 4 are displayed. For each vertex, the robot path is shown with dotted lines, and the rotations are shown as red corners. As shown in the figure, the number of rotations is 0 to 4 depending on the pattern of degree and edge connections.

3.3 Characteristics of The Problem

Consider the number of solutions to the problem described in Section 3.2. First, consider a lattice graph with $n * n$ vertices as an input. In [11], the number of spanning trees of the lattice graph with $n * n$ vertices is approximated as $\exp(n^2 Z_L)$ when Z_L is a finite nonzero constant. Therefore, the number of spanning trees increases exponentially as the size of the target workspace

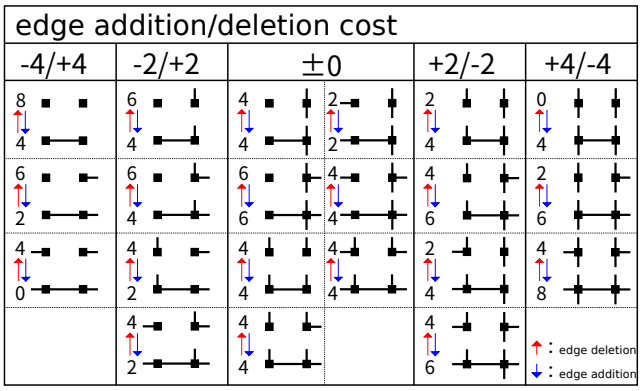

Fig. 4 Classification of edges on lattice.

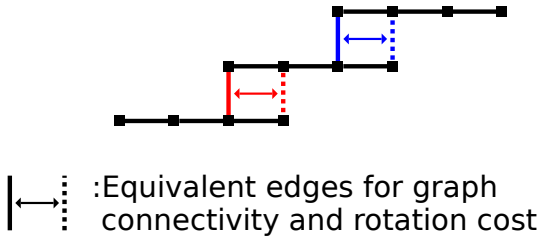

Fig. 5 Choices of edges with cost +2/−2.

increases and search for a optimal tree becomes difficult.

Next, let us consider the types of edges of the lattice graph. The edges can be classified depending on the types of the two vertices connected by the edge. Let us consider the amount of change in the rotation cost caused by addition or deletion of the edges to the lattice graph. The edge classification by the cost is shown in **Fig. 4**. In the figure, edges are classified into 5 types according to the edge addition/deletion cost: −4/+4, −2/+2, 0/0, +2/−2, and +4/−4. Each entry represents edge addtion/deletion between two states: the state where the edge does not exist between 2 vertices and the state where the edge exists between 2 vertices. In the figure, the absent state is placed in the upper side and the present state is placed in the lower side. The numbers displayed next to each state represent the rotation cost of the 2 vertices in the state. The addition cost and deletion cost are the difference between the numbers for the absent state and the present state.

Next, let us consider the characteristics of edges in terms of the rotation cost and graph connectivity. **Fig. 5** shows the combinations of edges that can be replaced with addition and deletion of edges while preserving the connectivity of the graph and the number of rotations. The solid lines represent existing edge and the dotted lines are the possible replacements of the existing edge of the same color. It is possible to construct several trees with same rotation cost by selecting edges in the tree. Therefore, it is considered that these combinations of edges can be omitted in searching for an optimal tree.

In addition, combinations of edges shown in red or blue in Fig. 5 can include edges with +4/−4, +2/−2, or +0/−0 rotation cost. Since these edges have negative deletion costs, they should not present in a spanning tree with the smallest rotation cost. When at least one edge is necessary to maintain graph connectivity, the search space is reduced by considering the each combination as one group without selecting an edge.

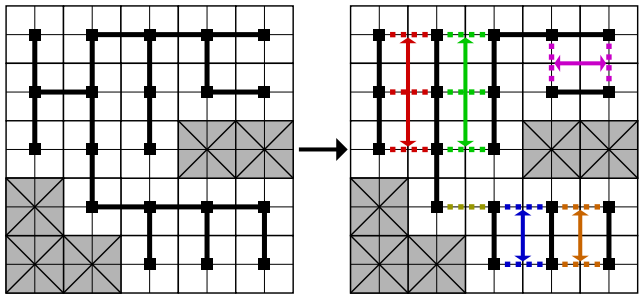

Fig. 6 Compressed representation of graphs.

4. Proposed Algorithm

In this section, we propose a spanning tree search method for area coverage problem by a single robot. First, we propose the compressed representation of graphs that can reduce the amount of computation. The representation utilize the characteristics of the rotation cost described in the section 3.3. Next, we explain the optimal tree search method using the features of the compressed representation.

4.1 Compressed Representation of Graphs

In the area coverage problem, the evaluation criterion of the solution is the sum of the rotation cost and the cost due to the number of vertices. Since the number of vertices included in the spanning tree is always equal to the number of vertices of the input graph and C_R is smaller than C_S, the parameter to be considered in optimizing the solution is the sum of the number of rotations. Therefore, let us consider the compressed representation $F = (V,\ E_f)$ corresponding to a certain tree T. E_f is subset of E_t, and also the addition cost of $e \in E_f$ is less than 0. The compressed representation F is obtained by repeating the elimination of the edge with 0 or less deletion cost on T. The edges on the compressed representation are only edges with less than 0 addition cost. The rotation cost of the compressed representation is determined by considering the types of each vertex in F according to Fig. 3 or considering sum of the addition cost of existing edges in F according to Fig. 4. In addition, it is possible for one compressed representation to represent multiple trees.

An example of a compressed representation is shown in **Fig. 6**. Removing edges with 0 or less deletion cost from the left spanning tree gives a compressed representation shown by black edges and vertices on the right side. This representation expresses $3 * 3 * 2 * 2 * 2 = 72$ spanning trees by selection of existing edges in the each combination of edges indicated by the dotted lines and the arrow.

Here we consider the direction of each vertex in terms of the shape of the edge connection on the compressed representation. A vertex whose degree is 2 and whose two edges are connected vertically is a vertical vertex and a vertex whose two edges are connected horizontally is a horizontal vertex. All other vertices are multi direction vertices having both vertical and horizontal directions. Consider the horizontal or vertical regions composed of adjacent vertices with the same direction. The multi direction vertices overlap with the horizontal region and the vertical region. Let the length in the horizontal or vertical direction of horizontal/vertical region respectively be the length of the region and the width in the perpendicular direction be the width of the region. Let us consider the difference between the compressed representation with the minimum rotation cost and the current compressed representation as a set of vertices whose directions are not matched. In the algorithm of the next section, the operation to obtain the compressed representation with the minimum rotation cost is considered as an operation to replace the vertex direction of the partial region on the compressed representation. The rotation cost of the compressed representation is determined by the number of line segments constituted by the connecting edges and the vertices in the same direction and the connection edges between the regions. The selection of the replacement region is performed so as to decrease the number of line segments on the compressed representation while paying attention to the connection edges between the region. An example of region replacement is shown in **Fig. 7**. By replacing the vertex direction in the region, the edges existing in the region change, and the shape of the connection in the overlapping part of the region changes. In addition, in Fig. 7, the number of line segments which affect the rotation cost has been reduced from 7 to 6 by replacement. The number of rotations has decreased from 22 to 12. The algorithm described in the section 4.2 searches for solutions using the features of this compressed representation.

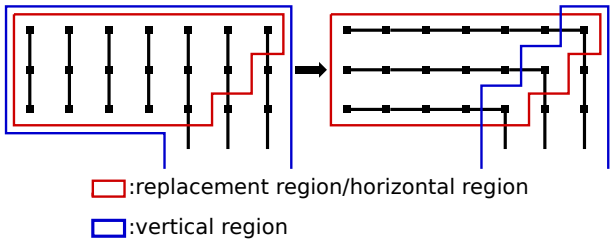

Fig. 7 Example of region replacement.

Algorithm 1 Algorithm to derive a spanning tree with the minimum rotation cost.

Input: lattice graph $G = (V, E)$
Output: spanning tree with the minimum rotation cost T

```
L ⇐ GET_LINES(G)
while L ≠ null do
    l' ⇐ l ∈ L
    L ⇐ L − {l'}
    d ⇐ DIRECTION(l')
    (R, cost, reg_num) ⇐ SEARCHCHANGEREGION(l', d)
    if (cost < 0) or (cost = 0 and reg_num < 0) then
        G ⇐ CHANGEDIRECTION(G, R, d)
        L ⇐ GET_LINES(G)
    end if
end while
T ⇐ CONNECTREGS(G)
```

4.2 Search Algorithm for Spanning Tree with Minimum Rotation Cost

The proposed algorithm performs a search for replacement region for each line segment on the lattice graph. If this replace-

Algorithm 2 Algorithm for deriving the replacement region with the largest decrease in rotation cost

Input: l: line segment and d: direction of l
Output:
region: the replacement region that maximizes turn cost reduction, *cost*: decrease in cost of rotation, and *reg_num*: decrease in the number of connected graphs

```
1: region ⇐ l
2: tmp_reg ⇐ region
3: cost ⇐ 0
4: tmp_cost ⇐ 0
5: reg_num ⇐ 0
6: tmp_r_num ⇐ 0
7: loop
8:     next_vertices ⇐ NextVertices(tmp_reg, d)
9:     if next_vertices = null then
10:        break
11:    end if
12:    (add_vertex, add_cost, add_r_num)
                    ⇐ MinAddCost(tmp_reg, next_vertices, l, d)
13:    tmp_reg ⇐ tmp_reg ∪ {add_vertex}
14:    tmp_cost ⇐ tmp_cost + add_cost
15:    tmp_r_num ⇐ tmp_r_num + add_r_num
16:    if (tmp_cost < cost) or
           ( tmp_cost = cost and tmp_r_num < reg_num) then
17:        region ⇐ tmp_reg
18:        cost ⇐ tmp_cost
19:        reg_num ⇐ tmp_r_num
20:    end if
21: end loop
```

ment region decreases the rotation cost or reduces the number of connected graphs while preserving the rotation cost, the algoritm is performed to sequentially replace the direciont of discovered region.

Algorithm 1 shows an algorithm for finding the spanning tree with the minimum rotation cost from the input graph G. The symbols used in the algorithm L is the set of segments on the lattice graph, l' is the vertex set representing one line segment, d is the variable representing the direction of a vertex, R is a set of vertices representing the replacement region. In the first line of the proposed algorithm, we first acquire L by the function Get_Lines. Get_Lines is a function that simply returns the set of all vertical line segments and horizontal line segments in the graph. Next, line 3-4 extracts l' from L. In line 5 we get the direction d of l' by the function Direction which returns the direction of the line segment. In line 6, the replacement region search is performed in a direction orthogonal to d for the given line segment l' by the function SearchChangeRegion shown in algoritm 2. Details of the search will be described later.

In line 7, it is first judged whether the amount of change in rotation cost due to region replacement is smaller than 0, or whether the amount of change is 0 and the number of connected graphs is decreasing. If these conditions are satisfied, the processes of line 8 and 9 are performed. In line 8, the function ChangeDirection replaces the direction of the vertices in the replacement region with the direction orthogonal to d and remove or add edges to match the shape of edge connectivity with the vertex directions.

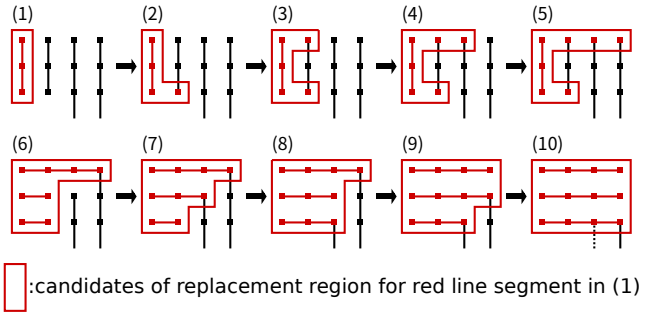

Fig. 8 Example of replacement region search.

In this replacement of the vertex direction, the vertex corresponding to the end point of the line segment in the region may overlap the adjacent region, so it is set as the vertex in both the vertical and horizontal directions. However, when there are multiple adjacent line segments consisting of one vertex, these vertices are collectively taken as the line segment in the d direction. Also, all edges extending from the vertex other than the end point of the line segment to the outside of the region are removed from the graph because the deletion cost is all less than 0. The above operations are repeated until set L becomes empty. Finally, we obtain a spanning tree from the compressed representation by the function ConnectRegs. If the compressed representation F is disconnected at this stage, additional costs of adding edges between disconnected graphs are always 0 or positive. Therefore, add the edge with +0 presence cost until the number of connected components no longer decreases. At this point, if there is still an disconnected graph, add the edge with +2 addition cost, then add the edge with +4 addition cost until the number of connected components no longer decreases. By this processing, a connected graph is obtained. Finally, closed paths in the graph are eliminated. Now, since the deletion cost of the edges constituting the closed path is 0 or positive, by removing edges with deletion cost 0 and eliminating the closed path, a spanning tree with the minimum rotation cost can be obtained.

Next, the processing of function SearchChangeRegion shown in algorithm 2 is described in detail. Function SearchChangeRegion selects the vertex to be included in the replacement area from the vertices existing in the direction orthogonal to the given line segment l. Therefore, in line 8, the function NextVertices obtains the set of vertices *next_vertices* adjoining the current replacement region *tmp_reg* in the direction orthogonal to the direction d. From *next_vertices*, the algorithm select a vertex with the largest decrease of rotation cost by adding vertex to the replacement region. The function MinAddCost in line 12 is a function for selecting the additional vertex. The change in the rotation cost when the vertex is included in the replacement region is determined by the direction of the eight neighbor vertices. At this time, the multiple direction vertices outside the replacement region is handled as the vertex of direction d. This makes it possible to perform a search using the features of compressed representation on the general lattice graph. Also, if there are multiple vertices whose rotation cost reduction amounts are equal, vertex nearer to the line segment l is selected. This makes it possible to determine the minimum necessary replacement region. An example of

searching a replacement region is shown in **Fig. 8**. In this example, the search is started from the red line segment in (1). The above processing is repeated until *next_vertices* becomes empty. In each iteration, the replacement region with the largest amount of decrease in rotation cost and the largest decrease in the number of connected graphs is stored in the process of lines 16-20. *Region*, *cost* and *reg_num* at the end of the loop express the region where the reduction amount of rotation cost is the largest, the reduction amount of rotation cost, and the decrease amount of the number of connected graphs, respectively. As a result, (8) is selected as a replacement region.

4.3 Time Complexity of the Algorithm

The proposed algorithm always finishes calculation for any input. Throughout this algorithm, there is no process to increase the rotation cost. Also, the number of connected components does not increase in a section where there is no change in rotation cost in algorithm processing. Therefore, a loop of processes in which the rotation cost or the number of connected components are not increasing. Therefore, this algorithm always finishes calculation.

Next, we consider the computational complexity of the algorithm. For each process in the algorithm, let us consider the maximum repetition of the process independently. It can be thought that the algoritm consists of three hierarchical processes and the function ConnectRegs is executed only once at the end. The three hierarchical processes are process A for searching a replacement region for a line segment, process B for applying process A to each line segment in set L, and process C for updating L and executing process B each time region replacement is performed. Consider the maximum number of calculation steps for each process on the subgraph of the lattice graph with the number of vertices $n * n$. For processing A, the worst case is the case of finding the region by adding $n - 1$ edges to each n vertex of a line segment. Therefore, the maximum number of steps is $n^2 - n$. Next, process B is repeated up to the number of elements in set L, which is the number of line segments on the graph. It is always smaller than $2n^2$, which is the case that one vertex is regarded as a horizontal line segment and a vertical line segment with 0 length. Therefore, maximum number of calculation steps of process B is $O(n^2)$. Finally, execution count of process C is the number of region replacements performed in the algorithm. The maximum number of regions that can exist simultaneously on the $n*n$ graph is $n*n$, and the region once replaced is never returned to the state before replacement. Therefore, The number of region replacement is at most a constant multiple of the number of regions. Thus, the maximum number of steps of process C is $O(n^2)$. The total number of steps in the algorithm is obtained by multiplying the number of steps of process A, B and C. As a result, the number of steps in the three processes is at most $O(n^6)$. With respect to the function ConnectRegs, addition of up to $n^2 - 1$ edges and removal of $(1/2)(n^2 - n) - (n^2 - 1)$ edges are performed in the worst case, so the number of steps of calculation is $O(n^2)$. The total calculation amount takes the sum of these. Therefore, this algorithm is guaranteed to finish calculation with $O(N^3)$ for the number of vertices N.

4.4 Optimality of the Solution

First, consider the difference S between certain compressed representation and an optimal compressed representation giving an optimal solution. Now, the difference between the compressed representations is taken as a set of vertices whose directions are different between the two. Also consider a set of vertices which are subsets of S and which are adjacent to each other on the graph and whose directions coincide with each other as a region R_S included in S. At this time, the following property holds.

- Regardless of the order of replacement, the compressed representation obtained by replacing the region R_S one by one is equivalent to the compressed representation in which entire S is replaced at once.

Assuming that the following property holds, the algorithm can obtain the compressed representation that gives the optimal solution by replacing the replacement region found by the algorithm one after another.

- The proposed replacement region search algorithm finds all areas R_S included in S.

In order to show that the solution obtained by this algorithm is optimal, we prove the above property. The region R_S is a region composed of line segments in the same direction. The search for replacement region is performed for each line segment on the compressed representation toward the direction orthogonal to the line segment. Therefore, if the length of the longest line segment in the region R_S is equal to the length of the region R_S, it is possible to search a range including the whole R_S. In this search, when there are k replaceable line segments within the search range, the connection pattern with the adjacent region in which the rotation cost is the smallest is examined for each case where the number of replace lines is from 1 to k. Therefore, it is possible to find a region with the largest decrease in rotation cost within the search range.

In addition, a search is performed from a segment that is not the longest line segment, and when a partial region of the region R_S is found as a replacement region, the remaining region is also found as a replacement region. Likewise, when the length of the longest segment of the region R_S is different from the length of the region, a subset of the region R_S is found as a replacement region, and the remaining regions are replacement regions. Therefore, the proposed region search algorithm can always find the region R_S, and by substituting the found replacement region one after another, it is possible to obtain the solution with the minimum rotation cost.

5. Conclusion

In this paper, we proposed compressed representation of graphs and a spanning tree search algorithm for mobile robot area coverage task considering turns on the path. Proposed algorithm gives an optimal area coverage path in terms of coverage time by deriving a spanning tree with minimum turn cost on the workspace. In the case where the number of vertices of the lattice graph representing the workspace is N, the algorithm has order of N^3 time complexity. Therefore, it is possible to solve the area coverage problem within a realistic time scale using the algorithm.

For future work, we intend to perform quantitative evaluation

and propose multi-robot area coverage algoritms.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number 18K18024.

References

[1] Gabriely, Y. and Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot, *Annals of Mathematics and Artificial Intelligence*, Vol. 31, No. 1, pp. 77–98 (2001).

[2] Bochkarev, S. and Smith, S. L.: On minimizing turns in robot coverage path planning, *Proc. of Int'l Conf. on Automation Science and Engineering (CASE)*, pp. 1237–1242 (2016).

[3] Schfle, T. R., et al.: Coverage path planning for mobile robots using genetic algorithm with energy optimization, *2016 International Electronics Symposium (IES)*, pp. 99–104 (2016).

[4] Jimenez, P. A., et al.: Optimal area covering using genetic algorithms, *Proc of Int'l Conf. on advanced intelligent mechatronics*, pp. 1–5 (2007).

[5] Yakoubi, M. A. and Laskri, M. T.: The path planning of cleaner robot for coverage region using Genetic Algorithms, *Journal of Innovation in Digital Ecosystems*, Vol. 3, No. 1, pp. 37 – 43 (2016).

[6] Kapanoglu, M., et al.: A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time, *Journal of Intelligent Manufacturing*, Vol. 23, No. 4, pp. 1035–1045 (2012).

[7] Màntaras, L., et al.: Generation of Unknown Environment Maps by Cooperative Low-cost Robots, *Proc. of the 1st Int'l Conf. on Autonomous Agents*, pp. 164–169 (1997).

[8] Ryu, S.-W., et al.: A search and coverage algorithm for mobile robot, *Proc. of 8th Int'l Conf. on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 815–821 (2011).

[9] Hazon, N. and Kaminka, G. A.: Redundancy, Efficiency and Robustness in Multi-Robot Coverage, *Proc. of Int'l Conf. on Robotics and Automation*, pp. 735–741 (2005).

[10] Zheng, X., et al.: Multi-robot forest coverage, *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3852–3857 (2005).

[11] Shrock, R. and Wu, F. Y.: Spanning trees on graphs and lattices in d dimensions, *Journal of Physics A: Mathematical and General*, Vol. 33, No. 21, p. 3881 (2000).