An Efficient Algorithm for Environmental Coverage with Multiple Robots

Ling Xu and Anthony (Tony) Stentz
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
{lingx, axs}@cs.cmu.edu

***Abstract*— Tasks such as street mapping and security surveillance seek a route that traverses a given space to perform a function. These task functions may involve mapping the space for accurate modeling, sensing the space for unusual activity, or searching the space for an object. In many cases, the use of multiple robots can greatly improve the performance of these tasks. We assume a prior map is available, but it may be inaccurate due to factors such as occlusion, age, dynamic objects, and resolution limitations. In this work, we address the NP-hard problem of environmental coverage with incomplete prior map information using k robots.**

To utilize related algorithms in graph theory, we represent the environment as a graph and model the coverage problem as a k-Rural Postman Problem. Using this representation, we present a graph coverage approach for plan generation that can handle graph changes online. Our approach proposes two improvements to an existing heuristic algorithm for the coverage problem. Our improvements seek to equalize the length of the k paths by minimizing the length of the maximum tour. We evaluate our approach on a set of comparison tests in simulation.

I. Introduction

Many tasks, such as street sweeping, mail delivery, and robotic surveillance and patrol, require a team of robots to visit points in an environment to accomplish a goal. These goals usually entail effectively mapping, sensing, or searching the space. In this work, we address the coverage problem where multiple robots are required to visit most locations in a given area. We assume a map of the environment is available. This is a reasonable assumption since information of most outdoor spaces can be obtained via satellite images. However prior maps can differ from the actual environment due to factors such as occlusions, age, and lower map resolutions. Even with an otherwise accurate map, dynamic conditions such as the presence of people or vehicles can diminish the effective accuracy of prior information. Therefore, an additional goal for the coverage problem we are addressing is to efficiently replan when changes occur in the environment.

In robotics, there is related work in the area of continuous space coverage. For this problem, the robot must pass a detector over all points in the space in order to complete a task [1][2][3][4]. These methods work well for open spaces, but are not suited for network environments such as roads and intersections.

In graph theory and operations research, researchers have tackled this coverage problem by representing the environment as a graph, and using routing algorithms such as the traveling salesman or postman problems to generate solution tours. In the graph representation, nodes in the graph denote locations in the environment and edges in the graph are the paths between the locations. For example, the map in Figure 1(a) is converted into a graph shown in Figure 1(b) by changing each street intersection into a node and each street into an edge. Each edge has a cost assigned to it where the cost can represent measurements such as Euclidean distance between locations, terrain traversability, travel time, or a combination of several metrics. Additionally, each edge is undirected meaning it can be traversed in either directions.

Fig. 1. (a) The map of an urban environment is shown where the box-like shapes represent buildings and the spaces between the buildings are roads. (b) We represent the prior map as a graph where edges (white lines) denote the roads and vertices (stars) denote the road intersections.

For our problem, we seek k coverage paths that visit a designated edge subset of the graph where k is the number of robots. This coverage problem can be modeled as an arc-routing problem. We focus on two types of arc-routing problems: the k-Chinese Postman Problem and the k-Rural Postman Problem. The k-Chinese Postman Problem (k-CPP) seeks k paths that visit all the edges of the graph at least once, and the k-Rural Postman Problem (k-RPP) seeks k paths that visit a predefined subset of graph edges at least once.

Two versions of the k-CPP exist: one is a direct extension of the 1-CPP [5] where the objective is to minimize the sum of the k tour lengths. Polynomial algorithms [6][7] exist that produce optimal solutions to this problem. However, for many practical applications, optimizing the total tour length may not be ideal since the optimal solution may assign one robot to do all the work. A second version of the k-CPP is known as the Min Max k-CPP (MM k-CPP) where the goal is to minimize the maximum cost path as a way to reduce the total time spent and to equalize the work among the k robots. The MM k-CPP is NP-hard. While an optimal solution does exist [8], it is not computationally efficient

enough for practical problems. As a result, a number of heuristics have been developed to provide more efficient solutions to the MM k-CPP problem [9][10][11].

In many applications, it is unnecessary to cover all of a given space – usually a part of the graph is required. In this case, the k-RPP is the appropriate model. To our knowledge, only one algorithm exists for the k-RPP. The algorithm is a heuristic algorithm introduced by Easton and Burdick [12]. The k-RPP algorithm consists of four steps: first, it divides the graph into k clusters using the method introduced by Ahr and based on the farthest-point clustering algorithm [13]. Next, it connects the edges in each cluster by creating a spanning tree between the disconnected components. Then, it utilizes the 1-CPP algorithm to find a route for each cluster. Finally, the algorithm refines each cluster by removing extraneous edges at the end of the tour once all the required edges have been visited. The extra edges are replaced by the shortest path to the depot from the final required edge in the tour. The k robots are assumed to start from the same depot. This approach has been extended to dynamic settings where the graph and number of robots can change [14].

In this paper, we present a k-RPP algorithm with two major improvements over the prior algorithm. The first improvement, to the clustering step, enables a better division of the graph edges into k partitions. The second improvement, to the routing step, allows for shorter tours of the clusters by using the optional edges to travel between required edges. These two improvements to the algorithm help generate more evenly distributed paths for faster coverage of the environment.

The rest of the paper is organized as follows. In Section II, we introduce and describe each component of the coverage algorithm. In the next section, we explain the set of tests we conducted. The results are presented and discussed in Section IV. Finally, we conclude and list future directions for this work.

II. Coverage Algorithm

A. k-Rural Postman Problem

The Easton and Burdick algorithm with our improvements is shown in Algorithm 1. We will give an overview of the algorithm, and then go through the two improvements in more detail. First, let us define a couple of terms. We call the required set of edges *coverage* edges, and the optional set of edges *travel* edges.

In the clustering step of the k-RPP algorithm, the algorithm calls k-means [15] to find k clusters of the coverage edges (line 2). The next step ensures each cluster is connected by first creating a new graph for each cluster that consists of nodes that denote the connected coverage components, and edges that represent the shortest cost paths between each component. A spanning tree of this new graph is generated (line 5) and the shortest paths that correspond to edges in the spanning tree are added to each cluster (line 6). Next, we add the remaining edges in the original graph to the cluster as travel edges (line 7). Once the clusters are finalized, the algorithm calls the 1-CPP algorithm [16] to find the optimal tour of each cluster (line 8). Finally, the k paths are returned (line 10).

Input: k, number of robots
$G = (V, E)$ where E = {R, T} where R is the required set of edges and T is the travel set
Output: $P_{1...k}$

```
1  G' = (V, R);
2  E_{1...k} = K-means(G', k);
3  for i ← 1 to k do
4  |  G_i = (V, E_i);
5  |  (V, E_ST) = SpanningTree(G_i, G);
6  |  E_i = E_i + E_ST;
7  |  E_i = {E_i, E-E_i};
8  |  P_i = 1-CPP(G_i);
9  end
10 return P_{1...k}
```

Algorithm 1: k-RPP Algorithm

The following introduce and explain the two improvements that we made to the previous algorithm.

1) K-means Clustering: To cluster the edges in the graph, we first use the farthest point clustering method to find the set of k edges that represent each cluster. These k representative edges are computed sequentially such that representative edge R_i is maximizing its distance from all the previous edges $R_1 ... R_{i-1}$.

Input: k, number of robots
G, connected graph where each edge has a cost value
Output: $E_{1...k}$, k clusters of edges

```
1  c_{1...k} = FindRepEdges(k);
2  c'_{1...k} = [];
3  converge = false;
4  loops = 0;
5  while converge == false do
6  |  ClusterEdges(G, c_{1...k}, E_{1...k});
7  |  RecomputeCentroids(c_{1...k}, E_{1...k});
8  |  if |c_{1...k} − c'_{1...k}| < ϵ then
9  |  |  converge = true;
10 |  end
11 |  if loops == τ then
12 |  |  converge = true;
13 |  end
14 |  c'_{1...k} = c_{1...k};
15 |  loops++;
16 end
17 return E_{1...k};
```

Algorithm 2: K-means Clustering Algorithm

In many situations, this set of representative edges may not accurately represent the edges in the graph. To address this problem, we use the k-means clustering method which iterately generates the representative edges or cluster centroids to maximize the similarity within each cluster. This allows the centroids to better represent the layout of the coverage edges. As shown in Algorithm 2, the representative edges are found using the farthest point method (line 1). Next, the edges in the graph are assigned to the closest cluster according to a distance function (line 6). Using this set of

clusters, the centroids are recomputed (line 7). This process is iterated until the centroid values converge (lines 8-10) or the maximum number of loops is reached (lines 11-13).

In order for k-means to work for a graph where the centroid values may not lie on a node of the graph, we introduce a distance function (shown below) where the distance from the centroid (c) to an edge (e) is the cost (w) of the shortest path (SP) from the graph node closest to the centroid to the edge plus the Euclidean distance from the centroid to the closest node.

$$\begin{aligned} d(c,e) &= w(SP(ClosestNode(G,c),e)) \\ &\quad +|ClosestNode(G,c)-c| \end{aligned} \tag{1}$$

2) Use of Travel Edges: In the prior k-RPP algorithm [12], after clustering all the edges, the 1-CPP algorithm generates a path for each cluster, and the path is refined. We implemented an improvement to these two steps that enables the removal of the refinement step by having the 1-CPP find the optimal plan for each cluster of coverage edges. In order to generate an Eulerian tour, the 1-CPP algorithm must decide which edges in the graph to traverse twice; these edges would be doubled in the solution. To maintain a low cost path, it is important to double the minimum cost set of edges. Therefore, if we allow travel edges as well as coverage edges to be doubled, the doubled set may have a lower cost than if we only consider coverage edges.

B. Dynamic k-RPP Algorithm

Dynamic changes occur when the environment differs from the original map. For the coverage problems we are addressing in this work, most changes to the environment are discovered when the robots are actively traversing the space. These online changes are typically detected when the robots are not at the starting location, but at middle locations along the coverage path when some of the edges have already been visited. Because it is not necessary to revisit those edges, the visited edges in the previous plans are converted to travel edges.

To replan with the robots starting at a different location from the depot or goal location, we extend an idea based on [17] that we used with the 1-RPP [18]. To account for different start and end locations, we add an artificial edge to each of the k clusters that has one endpoint at the current robot location and the other endpoint at the depot. This edge is assigned a large cost to ensure it is not doubled in the solution. After the k tours are found, the artificial edge in each tour is removed and the paths are adjusted to start at the current robot positions.

To illustrates the online coverage algorithm, in Figure 2, we show a k-CPP solution of the example graph shown previously. In Figure 3, four robots (blue, red, cyan, and magenta) traverse the initial paths until one of them (blue) discovers an edge in its path is missing. The robot does not know about this discrepancy until it reaches the adjacent node. The algorithm then sets the traversed edges in the four paths to be travel (Figure 4) and the problem becomes a k-RPP. The replanned coverage tours are shown in Figure 5.

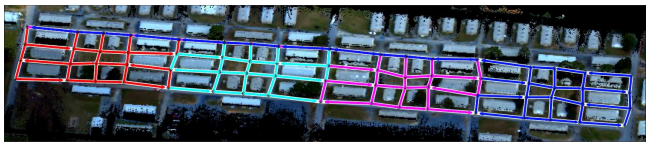


Fig. 2. Different colored lines represent the k-CPP solution and are superimposed on the graph from Figure 1(b). Robots 1, 2, 3, and 4 are blue, red, cyan, and magenta, respectively.

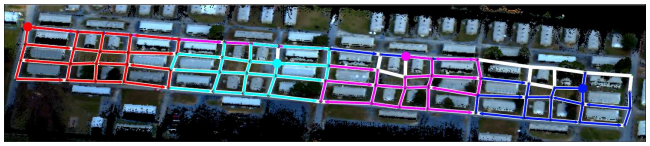


Fig. 3. During traversal of the four tours, robot 1 (blue) discovers that an edge in the path is missing and ends the traversal. The circles represent the current locations of the robots.

One advantage to this revision strategy is that it supports the case where the robots start at different depots. Instead of each robot starting at the same depot, different robots can have different depots. Morever, this method can be easily extended to handle changing team sizes due to robot failure or new robots entering the scene by excluding the artificial edge corresponding to the failed robot or not having artificial edges for the new robots, respectively.

We considered using the the revision algorithm given in [14] which, when replanning, uses the next edge in the current path of each robot as the representative edges for each new cluster. While this method works well on graphs where the clusters are far apart, it does not perform as well when the clusters are close together, particularly in situations when some of the robots are at the same location. In this case, the representative edges of the clusters corresponding to those robots would be the same, and the resulting clusters would not be well distributed.

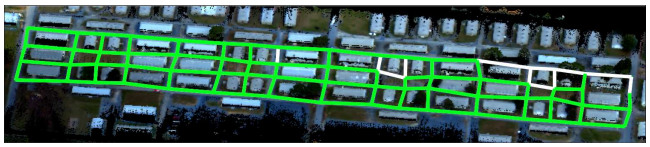


Fig. 4. To prevent visiting edges that have been already traversed, the graph is modified by setting the traversed edges to be travel edges, represented by green lines.

Fig. 5. New paths are replanned for the updated graph. These four paths start at the nodes where the previous traversals ended.

III. Comparison Tests

The goal of our testing is to compare the improved k-RPP algorithm against the original k-RPP algorithm. For the tests, we wanted to assess the two algorithmic improvements, k-means clustering and use of travel edges, individually. To do this, we compared four combinations of the algorithm: the original algorithm, the original algorithm with k-means clustering, the original algorithm using travel edges, and the original algorithm with both improvements (our algorithm). For the tests, we varied four different parameters: algorithm combination, size of graph, starting node s, and number of robots k. The test simulated k robots executing k tours starting at node s.

We conducted two sets of tests: static and dynamic k-RPP tests. The first test set evaluated the algorithm combinations on four graphs. The second test set demonstrated the re-planning capability of the algorithm when handling online changes. We will first explain the procedure for each test, and then present the metrics for comparing the two heuristics. The code ran on a machine with a 2.53GHz Intel processor and 4GB of RAM.

A. Static k-RPP Tests

The first set of tests ran each algorithm on four different graphs: three rectilinear graphs and one real-world road network dataset. We ran five trials on each graph with each trial having a random set of travel edges. Each set of travel edges constituted half of the edges in the graph. The results were averaged over the five trials. The number of robots, k, ranged from one to ten. For the rectilinear graphs and each k value, we varied the starting node over all the nodes in the graph to avoid bias. Therefore, each of the algorithm combinations was called $k \times 3 \times 5 \times |V|$ times where $|V|$ is the number of nodes in the graph. For the road network, because the graph is so large, instead of varying the starting node over all nodes in the graph, we sampled a set of fifty distinct nodes. The samples were consistent for the four methods. In total, this test yielded $k \times 5 \times 50$ plans for each combination.

B. Dynamic k-RPP Tests

For the second set of tests, our goal was to evaluate the efficiency of the algorithm in replanning online when changes occur. For these tests, we add a new parameter: a set of changes. Additionally, we constrained k to be ten. At the beginning of the test, the graph is connected and has no travel edges. Using the coverage algorithm, k tours are computed. The test simulates the robots executing the tours starting at node s. If along the execution, the next edge to be traversed is in the change set, that edge is considered blocked. It is removed from the graph, and the graph is updated to reflect the visited edges and current robot locations. The algorithm plans a new set of tours using the updated graph. The execution is simulated again until another edge along the new path is found to be blocked or the traversal is completed. This test consisted of $3 \times 5 \times |V| + 5 \times 50$ replans for each combination.

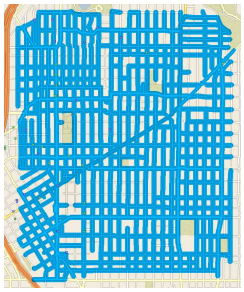


Fig. 6. This is the road network we used as our real-world example. This network covers an area of 1.5 miles by 2 miles. The edges and nodes included in our graph are highlighted in blue.

C. Test Graphs

Rectilinear graphs are graphs where the vertices are generated uniformly in the plane. Edges connect one vertex to its four closest neighbors by straight line segments that represent the Euclidean distance between the vertices. In other words, these graphs consist of a regular pattern of nodes and edges. We chose rectilinear graphs because they are an approximation to the road networks used in many coverage applications. For example, in street coverage, the graph layout is similar to a grid where all streets (edges) meet at intersection points (nodes), and most intersections are four ways. For our testing, three graph sizes were used; the sizes are $|V| = \{10 \times 10, 14 \times 14, 17 \times 17\}$.

The real-world example we used for testing is the road network of an urban neighborhood obtained from a dataset gathered by Newson and Krumm [19]. The network is shown in Figure 6. We converted the network into a graph with 769 vertices and 1130 edges.

D. Metrics

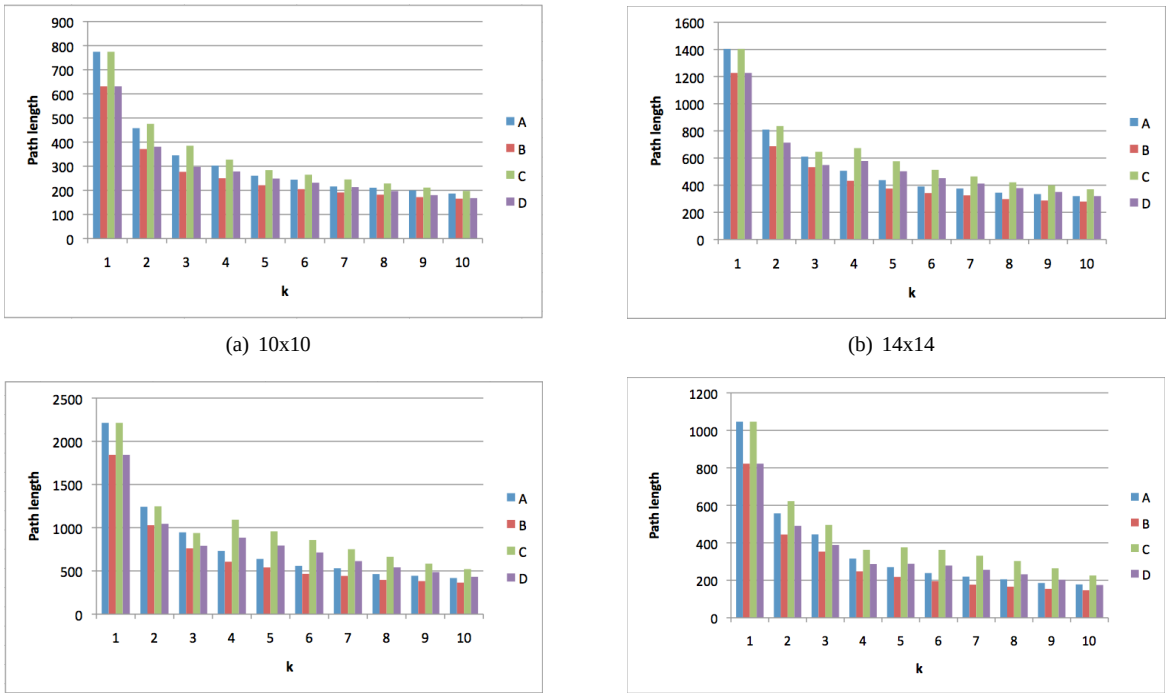
During each call to the coverage algorithm, we computed the maximum path length, computation time, number of travel edges, number of coverage edges, and number of connected components of required edges. We limited the number of iterations of the k-means algorithm to 100 to maintain efficiency.

IV. Results

Before presenting the results, we explain our notation and data organization. For the rectilinear graphs, the average degree for each node is 3.7, and the standard deviation is 0.5. For the road network, the average degree is 2.96 and the variance is 1.0.

Additionally for the results, we use a labeling system to denote each of the combinations (Table I). Label C is the original k-RPP algorithm. Label D is the algorithm using travel edges in the 1-CPP method. Label A is the algorithm using k-means clustering. Finally Label B is our k-RPP algorithm which contains the two improvements.

Results from static k-rpp tests are presented in Figure 7. The figure contains four subfigures, one for each graph type. Each subfigure plots the length of the maximum path for each combination averaged over all starting nodes, trials,



(a) 10x10

(b) 14x14

(c) 17x17

(d) Road network

Fig. 7. This graph shows the average path length of the four algorithm combinations for the 10x10(top left), 14x14(top right), 17x17(bottom left), and road network(bottom right) graphs with 50% travel edges.

and k from one to ten. Next, we show the results from the dynamic tests. Figure 8 shows the maximum path length for each algorithm and graph with k equal to ten while Figure 9 compares the runtimes of each replan. Table II lists the average number of travel edges, coverage edges, and connected coverage components for each graph.

A. Static k-RPP Tests

For the set of static tests shown in Figure 7, our k-RPP algorithm (B) outperforms the other algorithms in generating a path with a lower cost. Note that both of the improvements boosts the original algorithm in finding lower cost paths with k-means clustering being more effective. While the algorithm performs well on the three rectilinear graphs, it does the best on the road network data producing paths that are much shorter in length than the paths generated by algorithm C (the original k-RPP algorithm). This arises from the road network having a greater variety in the degree of its nodes resulting in more distinct coverage clusters, and k-means is better at finding these clusters. Finally, we computed the variance in the path costs for each graph and k. On average, algorithms A and B have path variances of 0.18, and algorithms C and D have variances of 0.32. This shows that our approach is more successful in generating paths with similar costs, and that the lower variance is due to the k-means method.

TABLE I

Label	K-means	Travel edges
A	Yes	No
B	Yes	Yes
C	No	No
D	No	Yes

TABLE II

$\|V\|$	Avg Travel	Avg Coverage	Avg Components
100	91.47	95.02	3.41
196	175.836	194.738	4.70
289	279.49	271.12	6.20
769	592.88	542.75	9.90

B. Dynamic k-RPP Tests

In the dynamic tests shown in Figure 8, our k-RPP algorithm (B) finds the lowest paths as well. Because each algorithm produces different paths initially, the revised graph at each replan will vary between the four heuristics. As a result, algorithm B does not perform as well as in the static case since the updated graphs will have different sets of coverage and travel edges. This test shows that in spite of the variations in graphs, our k-RPP still performs the best.

Figure 9 shows that the original k-RPP algorithm has the lowest runtime while our k-RPP algorithm has the highest runtime. Of the two improvements, k-means is the more expensive. However, for the sizes of the graphs we used, the computation times for the improved algorithm are reasonable. Finally, Table II shows that, on average, the graphs used for the dynamic tests were roughly the same as the graphs used for the static tests.

Fig. 8. This graph shows the average path length of the four algorithm combinations for the dynamic tests with five replans for each set of edge removals.

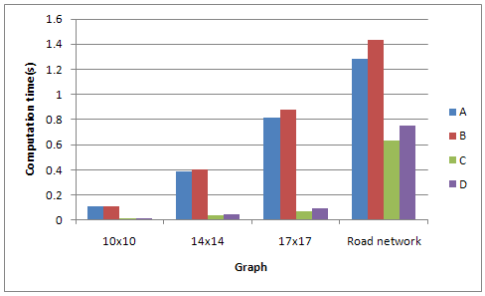


Fig. 9. This graph shows the computation time for each algorithm in the dynamic tests.

C. Discussion

The results show that our k-RPP algorithm does find shorter maximum paths which translates to a shorter traversal time for different team sizes of robots. Our static tests show that our algorithm can produce paths that are lower than those generated by the original algorithm as well as paths that are more similar in cost. Moreover, this algorithm is robust to graph discrepancies as the dynamic tests demonstrate. Of the two improvements, k-means is more effective as the value of k increases while using travel edges is better when k is small.

From the timing results, we can see that k-means clustering is more costly to use than including travel edges. Despite being more costly, replans still take less than a second on the rectilinear graphs and one and a half seconds on the large road network. However, if computation is limited, one strategy is to always include the travel edge improvement since it is inexpensive; when planning offline, the algorithm would use both improvements, and when replanning online, it would exclude the k-means clustering. Additionally, we chose to limit the number of iterations of k-means at 100, but this number could be set to be lower to decrease the computation time.

V. Conclusion and Future Work

We present an improved algorithm to the multi-robot coverage problem with an incomplete prior map. Our two improvements strive towards minimizing the maximum length path by dividing the work more evenly among the robots. As shown in the results, the improved algorithm aids in reducing the path lengths for both static environments and environments where it is necessary to replan online. Because there is a trade-off regarding computation time, we recommend using one or both improvements depending on the particular time constraints of the application. For future work, we plan to extend this approach to mixed graphs (graphs with directed and undirected edges). Additionally, we plan to test this approach on physical robots in dynamic settings.

Acknowledgment

This work was sponsored by ONR Contract Number N00014-09-1-1031.

References

[1] Howie Choset. Coverage for Robotics - A Survey of Recent Results. *Annals of Mathematics and Artificial Intelligence*, 31:113 – 126, 2001.

[2] Xiaoming Zheng, Sonal Jain, S. Koenig, and D. Kempe. Multi-Robot Forest Coverage. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 3852 – 3857, Aug. 2005.

[3] Ioannis Rekleitis, Vincent Lee-Shue, Ai Peng New, and Howie Choset. Limited Communication, Multi-Robot Team Based Coverage. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3462–3468, New Orleans, LA, April 2004.

[4] Shuzhi Sam Ge and C. Fua. Complete Multi-Robot Coverage of Unknown Environments with Minimum Repeated Coverage. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 715 – 720, 2005.

[5] M Guan. Graphic Programming Using Odd and Even Points. *Chinese Mathematics*, 1:273–277, 1962.

[6] A. Assad, W. Pearn, and B. L. Golden. The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases. *American Journal of Mathematical and Management Sciences*, 7:63 – 88, 1987.

[7] Lei Zhang. Polynomial Algorithms for the k-Chinese Postman Problem. In *IFIP World Computer Congress on Algorithms, Software, Architecture - Information Processing*, pages 430–435, 1992.

[8] Dino Ahr. *Contributions to Multiple Postmen Problems*. PhD thesis, Heidelberg University, 2004.

[9] Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation Algorithms for Some Routing Problems. *Annual Symposium on Foundations of Computer Science*, pages 216 –227, 1976.

[10] Dino Ahr and Gerhard Reinelt. New Heuristics and Lower Bounds for the Min-Max k-Chinese Postman Problem. In *Algorithms ESA 2002*, volume 2461 of *Lecture Notes in Computer Science*, pages 7–19. 2002.

[11] Dino Ahr and Gerhard Reinelt. A Tabu Search Algorithm for the Min-Max k-Chinese Postman Problem. *Computers and Operations Research*, 33(12):3403–3422, 2006.

[12] K. Easton and J. Burdick. A Coverage Algorithm for Multi-Robot Boundary Inspection. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 727 – 734, Apr. 2005.

[13] Teofilo F. Gonzalez. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science*, 38:293 – 306, 1985.

[14] K. Williams and J. Burdick. Multi-Robot Boundary Coverage with Plan Revision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1716 –1723, May. 2006.

[15] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[16] Jack Edmonds and Ellis Johnson. Matching, Euler Tours, and the Chinese Postman. *Mathematical Programming*, 5(1):88–124, 1973.

[17] Harold Thimbleby. The Directed Chinese Postman Problem. *Journal of Software Practice and Experience*, 33:2003, 2003.

[18] Ling Xu and Anthony Stentz. A Fast Traversal Heuristic and Optimal Algorithm for Effective Environmental Coverage. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[19] Paul Newson and John Krumm. Seattle Road Network Data. http://research.microsoft.com/en-us/um/people/jckrumm/MapMatchingData/data.htm, 2009.