Multi-robot Boundary Coverage with Plan Revision

Kjerstin Williams (*kj@robotics.caltech.edu*) and Joel Burdick(*jwb@robotics.caltech.edu*)
Division of Engineering and Applied Science
California Institute of Technology, Pasadena, CA 91125

Abstract— **This paper revisits the multi-robot coverage problem in which a group of k robots must inspect every point on the boundary of a 2-dimensional environment. We focus on the case in which revision of the original inspection plan may be necessary due to changes in the robot team size or the environment. Building upon prior work, which presented a graph-based approach to path planning for this problem, we present a graph representation of the task that is greatly reduced in complexity and a path revision algorithm appropriate for addressing such changes.**

I. INTRODUCTION

In the boundary coverage problem, detailed in Section II, a group of k robots is required to completely inspect the boundary of all 2-dimensional objects in a specified environment. Additionally, one would practically seek inspection plans which balance the inspection load, as much as possible, across the cooperating robots. Though this paper addresses the boundary coverage problem in the abstract, such tasks have many practical inspection, surveillance, and security applications [1]. We introduced the boundary inspection problem and a graph-based methodology for its solution in recent work [1]. This paper extends that work and overcomes some of its limitations. In particular, we introduce three innovations: 1) a new graph representation of the task that is greatly reduced in complexity; 2) a modular restructuring of the path planning algorithm that allows decentralization of much of the path planning computation; and 3) a path-revision approach which exploits the reduced complexity of the graph representation and the modularity of the path planning algorithm to allow for plan revision in response to mid-task changes in robot team size and/or changes in the environment.

Such changes can occur in a number of practically important situations. The number of robots cooperating in the inspection task might change during the inspection task due to robot failure, the retasking of robots to another chore, or the addition of robots to the inspection team. In such cases, the remainder of the inspection task should be replanned to adapt to the characteristics of the newly resized inspection team.

This paper introduces an improved graph-based task representation and an improved planning algorithm that can handle such situations requiring task replanning. The replanning capability is a necessary step in making the approach more robust to failure and uncertainty and introducing complementary or collaborative subtasks to the problem.

The boundary coverage problem considered in this paper and in reference [1] (a deliberative, complete approach) and the related boundary coverage problems considered in [2], [3],

and [4] (nondeterministic, swarm-based approaches) are distinct from the large body of literature on *freespace coverage*, where one or more robots must inspect or cover the freespace with a sensor or tool. References [5] and [1] provide a relevant review of freespace coverage research.

Section II summarizes the boundary coverage problem and our modeling assumptions. Section III describes a new method for the construction of an equivalent graph representation of the multi-robot inspection task. Section IV reviews the constructive heuristic used to find a solution to the boundary coverage problem, an NP-hard edge-covering graph problem called the k-Rural Postman Problem. In Section V, we identify modularity in the algorithm and propose a method for the revision of paths mid-task in reaction to changes in team size or the environment. Simulated examples and discussion are presented in Section VI and open questions and possible extensions are summarized in Section VII.

II. THE BOUNDARY COVERAGE PROBLEM

This section reviews the critical aspects of the multi-robot boundary coverage problem introduced in [1]. The task is to be carried out in a bounded 2-dimensional environment which is populated by N objects, $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$. We assume that the boundary of each object is a piecewise smooth, closed, convex curve. We collectively refer to the boundary of the union of the objects as the "boundary" of the environment, $\partial\mathcal{O}$. We assume that the location and boundary geometry of each object to be inspected is known a priori.

We assume the inspection will be carried out by a group of k identical holonomic point robots, each equipped with an accurate scheme for localization as well as an omnidirectional "inspection sensor." The inspection sensor can measure phenomena of interest from a distance up to r_{max} from the boundary. A point p on the boundary is thus considered inspected when a robot's path intersects the imaginary line normal to the boundary at p and the distance from p to that point of intersection is less than or equal to r_{max}. The robot may also have a minimum sensing distance to the boundary, r_{min}, in which case we assume all objects in the environment are spaced at least $2r_{min}$ apart, ensuring the entire boundary is reachable.

The division of the task among the robots is determined by a centralized, supervisory agent. Once it has received its assignment, each robot calculates its inspection route, given by a series of waypoints determined by our constructive heuristic. We assume that each robot has a limited communication capability so that it may receive its task assignments and carry

out a simple leader election protocol for avoiding interference with other robots, as described in Sections IV-A and IV-B.

All robots involved in the inspection task deploy from a common "depot" location at the beginning of the inspection period, though in Section V, we relax this constraint to allow robots to start from arbitrary locations. At least one robot must inspect each point of the boundary at least once during the team's inspection tour.

To develop a set of paths whose traversal will complete the inspection task, we take the sensor's visibility constraints into account. Object $\mathcal{O}_j$'s *visibility region* consists of the set of individual robot poses in freespace for which the distance to the nearest point on the object boundary is less than r_{max}. The outer perimeter of each object's visibility region is a curve displaced a distance r_{max} normal to the object boundary, generally called an "offset curve."

The intersection of multiple objects' visibility regions are called *multiview regions*: from every pose within such a region the robot is a distance less than r_{max} from more than one point on $\partial\mathcal{O}$, and thus may inspect these points simultaneously. Our routing strategy takes advantage of multiview regions when they arise, as simultaneous sensing of multiple object boundaries may reduce overall travel. A *boundary segment* is an interval of $\partial\mathcal{O}$. A continuous sequence of poses within a visibility region is a *visibility path* for a boundary segment if and only if every point in the boundary segment will have been viewed when a robot has assumed every pose in the visibility path. This terminology is illustrated for a sample environment in Fig. 1(a).

III. Constructing a Graph Representation for the Boundary Coverage Problem

The planning procedure begins with the construction of a graph representation of the inspection task. The graph construction procedure introduced in this section is motivated by the procedure found in [1], but contains a few key differences that will provide significant improvements in overall computational performance of the algorithm.

The graph's edges come in two varieties: required inspection edges, E_R, and connectivity edges, E_C. Each required edge represents an equivalence class of visibility paths, one of which the robot must traverse in order to inspect the associated boundary segment. This representation allows flexibility in navigation implementations, as individuals can avoid one another by taking different paths while "traversing" the same graph edge. The end-points of each edge are graph vertices, which physically correspond to terminal points for the equivalence class of visibility paths where a robot may transition locally from a path represented in one edge to a path represented in another. To simplify the construction of the graph representation, we consider a single, easily defined "preferred visibility path" for each edge. The cost function assigns each edge a weight equal to the length of the preferred path along that edge, though we note that other cost functions may be used.

The final result of the construction is an undirected, connected graph $G = (V, E, c : E \to \mathbb{R}^+)$, where V is the set of the graph's vertices, E is the set of its edges, and the cost function c assigns weights to the edges $E_R \subseteq E$. The remaining edges $E_C = E \backslash E_R$ represent a collection of paths that provide paths to the depot location and between disconnected inspection regions.

A. Determining E_R: Edges Required for Inspection

Let S_{max} denote the set of offset curves that are displaced a distance r_{max} normal to each object boundary. Observe that these curves represent the perimeter of each object's visibility space. Let P denote the union of the regions in freespace that are bounded by these curves. The region P will consist of one or more disjoint regions, $P_i, i \in \{1, \ldots, m\}$, termed *inspection regions*.

Let r_{min} denote the closest distance to an obstacle that yields adequate inspection performance (we assume that $r_{min} < r_{max}$, and r_{min} may approach 0). Let $S_{min} = \{S_{1,min}, \ldots, S_{N,min}\}$ denote the offset curves displaced a distance r_{min} normal to each object boundary. Thus, for object $\mathcal{O}_j$, traversal of a visibility path enclosing the entire object and contained within the region bounded by S_{min} and S_{max} will result in the inspection of every point on $\partial\mathcal{O}_j$.

In choosing paths that meet these constraints, we also wish to exploit the multiview regions' potential for reducing the path length of the robots' inspection tours, combining visibility paths for multiple objects. We also note that the length of the visibility paths for those boundary segments not visible from a multiview region would be minimized by following the boundary as closely as possible (i.e., the curve defined by the local component of S_{min}). To construct visibility paths and their corresponding edges which both exploit multiview regions and follow boundary segments closely outside those regions, we consider each disjoint inspection region P_i in turn. First we construct the components of G that lie in the multiview regions, then we construct edges corresponding to the inspection of the remaining boundary segments.

1) Edges inside multiview regions: Let M_i denote the union of the multi-view regions in the inspection region P_i. One convenient choice for the preferred visibility paths through M_i is the local components of the *Generalized Voronoi Graph (GVG)* [6] of the nearby object boundaries. Using these paths, robots are routed through multiview regions with a preference for paths whose constituent poses are equidistant from the boundaries currently being inspected.

Our construction strategy facilitates a transition between the exploitation of multiview regions and close boundary-following along the offset curves S_{min}. Consider the subset of these curves contained within an inspection region P_i and denote the region defined by the convex hull of these curves $P_{i,min}$. The perimeter of $P_{i,min}$ consists of segments of these boundary-following curves connected to their neighboring curves in the inspection region by line segments tangent to both curves.
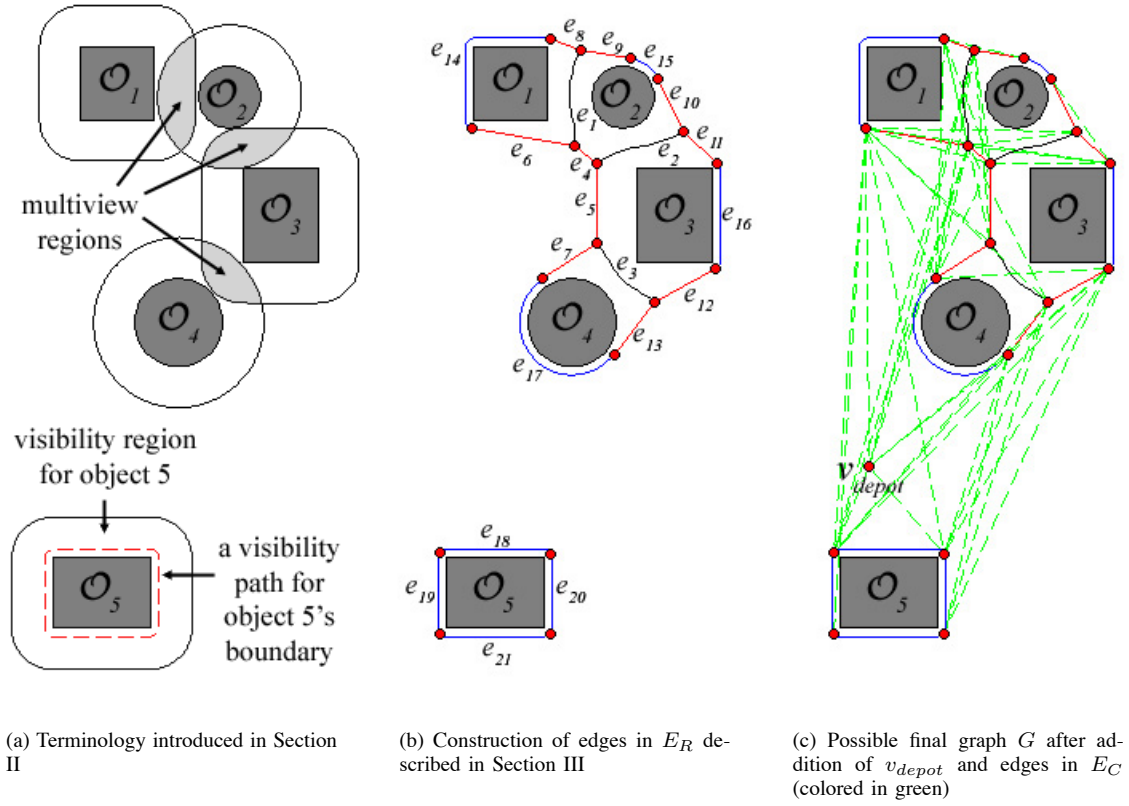
1717

(a) Terminology introduced in Section II

(b) Construction of edges in E_R described in Section III

(c) Possible final graph G after addition of v_{depot} and edges in E_C (colored in green)

Fig. 1. Illustration of terminology introduced in Sections II and III

Next find the *truncated multiview region*, $M_{i,min} = M_i \bigcap P_{i,min}$. Edges within these truncated multiview regions are established by adding to G all edges and vertices of the GVG contained within $M_{i,min}$ and assigning to each edge a weight equal to the length of the corresponding GVG path. Edges only partially contained in $M_{i,min}$ are truncated with a vertex at their intersection with the boundary of $M_{i,min}$. Edges e_1, e_2, and e_3 in Fig. 1(b) are examples of this type of edge addition.

2) Edges outside the multiview regions: After constructing the edges in the multiview regions, we must complete the set of required inspection edges, so that G contains edges representing visibility paths contained within the region bounded by S_{min} and S_{max} and enclosing each object. Consider every object O_j whose associated visibility region is partially or fully contained in P_i. By the construction in Section III-A.1, each of the objects associated with a multiview region already has at least one edge associated with its inspection. We connect the end vertices of these edges such that a cycle in G corresponding to the inspection of ∂O_j is formed as follows.

If a straight line between two existing vertices lies in freespace and contributes to the object's circumnavigation, an edge is added representing that direct path as the preferred visibility path. In Fig. 1(b), edges e_4 and e_5 illustrate this type of edge addition.

To complete the cycle for each object, we add edges connecting the GVG end vertices to exterior tangent points on the boundary-following curve $S_{j,min}$ (the offset curve for O_j), examples of which are seen in edges e_6 through e_{13} in Fig. 1(b). By construction, the remaining single-degree vertices lie on $S_{j,min}$, the preferred visibility path for boundary following. Edges representing these connecting segments of $S_{j,min}$, like edges e_{14} through e_{17} in Fig. 1(b), are added to G.

If P_i incorporates no multiview regions, then P_i contains a single object, O_j. The preferred visibility path for O_j is the boundary-following curve $S_{j,min}$. In this case, a vertex v is placed on the preferred visibility path and a single self-looping required graph edge of weight equal to the length of the curve $S_{j,min}$, originating from and returning to v, is added to G. The vertex must be placed to specifically guarantee mutual visibility with at least one other vertex in G. Alternatively, the edge can be subdivided to guarantee visibility; in our implementation, we place four vertices at points tangent to $S_{j,min}$ from distant points in the four cardinal directions. In Fig. 1(b), edges e_{18} to e_{21} illustrate these types of edges.

These two steps complete the construction of the edges $E_R \in G$ whose traversal is required for inspection to be complete.

B. Determining E_C: Edges Providing Connectivity

The group of required edges associated with boundary inspection in each inspection region P_i form a distinct connected subgraph of G. The possibly disjoint components of this construction must be connected to allow for robot transit to other parts of the graph.

A vertex, v_{depot}, is next added at the depot location. We then induce a *complete graph* on the vertices of G, adding edges connecting each vertex in G to every other vertex in G if such an edge does not already exist. These *induced edges* are assigned a weight equal to the distance between those vertices. If the path associated with an induced edge passes through an obstacle, it is excluded. The induced edges comprise the set of edges connecting edges, E_C. The graph G is now connected, i.e, a path exists between every vertex in the graph.

A sample graph representation of our example environment from Figs. 1(a) and 1(b) is illustrated in Fig. 1(c). The differences between the graphs resulting from this method of construction and those constructed using the method in [1] are discussed in Section VI.

IV. A GRAPH ALGORITHM SOLUTION TO THE BOUNDARY COVERAGE PROBLEM

As shown in [1], once the graph representation of the task has been constructed, the k-robot boundary coverage problem can be posed as the *k-Rural Postman Problem (kRPP)* graph problem. This problem is concerned with finding a set of $k \geq 1$ tours $T = \{T_1, ..., T_k\}$ within an undirected, connected weighted graph $G = (V, E, c : E \rightarrow \mathbb{R}^+)$ such that each edge in a required subset of edges $E_R \subseteq E$ is traversed in at least one tour and is NP-hard.

We present an extension to the constructive heuristic given in [1], reformulated in a modular way. This allows for revision of the planned paths that arise from unforseen incidents, such as a change in team size or a change in environment. There is also a clear division between the initial, centralized task division step and the rest of the path planning process, which is decentralized, allowing each robot to plan its own route.

A. Modular Constructive Heuristic for solving the kRPP

1) Partition the required edges E_R: We begin by partitioning the required edges E_R into k groups, one for each robot, $F_1 \cup ... \cup F_k = E_R$, using a farthest-point clustering method similar to that used in the "Cluster Algorithm" heuristic for the min-max k-Chinese postman problem presented by Ahr and Reinelt [7]. The method is based on an algorithm which aims to minimize the maximum intra-cluster distance [8].

Given a set of k *representative edges* $(f_1, ..., f_k)$, f_i is the first edge assigned to the cluster F_i that will eventually form the ith robot's tour. The $|E_R| - k$ remaining edges $e \in E_R$ are assigned to the group F_i that minimizes the distance through G between e and f_i.

Initially, the k representative edges are chosen such that the first representative edge, f_1, is the edge $e \in E_R$ farthest from v_{depot}. Subsequent representative edges f_i are chosen such

that the sum of the minimum distance through G to each of the existing representative edges $f_1, ..., f_{i-1}$ is maximized.

We note that after this partitioning step, which is carried out by a centralized, supervisory agent, the remainder of the planning procedure can take place online, with each robot calculating its own route.

2) Include edges for connectivity: Edges are added to each group to create k connected and, thus, traversable subgraphs of G, each of which must also include the depot vertex. The minimum spanning tree method described in [1] is used to select and add edges to those in F_i, yielding a connected subgraph G_i.

3) Compute a tour of each subgraph G_i: A single-postman tour T_i is computed on each subgraph $G_i, i = \{1, ..., k\}$ using Edmonds' and Johnson's Chinese Postman algorithm [9]. Each tour $T_i \in \{T_1, ..., T_k\}$ originates and terminates at v_{depot}, and has length $C(T_i) = \sum_{e \in T_i} c(e)$.

4) Refine tours: Finally, for each tour T_i, sequences of edges that are retraced in the course of a tour are replaced with the shortest path in G, if a shorter path between the end vertices of the sequence exists. When a robot has traversed every required edge in its tour, it returns to the depot via the shortest path through G.

Additional path refinement steps may be added as desired.

B. Path Planning Using the Graph Algorithm

The k robots' inspection paths are crafted based on the graph solution to the kRPP. Because edges in G specify paths through freespace, the tours found by the graph algorithm each define a series of waypoints. Each robot visits its tour's waypoints in sequence. When robots must pass within close proximity of one another, priority is established through a simple leader election process, in which priority is given to the robot with the largest distance left to travel. Other robots wait until the leader proceeds out of their proximity. See [1] for more details.

V. PLAN REVISION

Plan revision may become necessary during the course of task execution due to any number of events: the robot team size may change as one or more robots may fail or be retasked, or additional robots may be deployed to assist those already carrying out the task. Robots may encounter new objects that must be inspected, or the inspection assignment may be changed thereby changing the structure of G. The first step in robustly handling these practical issues is to devise a method for plan revision in the course of task execution.

To revise the robots' paths mid-task, we can appeal to the modular structure of the planning heuristic summarized in Section IV-A to partially reuse prior calculations with modified inputs based on the current state of inspection.

1) Repartition the unvisited required edges: Let k' denote that number of cooperating robots that remain after one or more robots has failed, been retasked, or added. Similarly, let E_R' denote the set of unvisited required edges at the time of revision. This set of edges includes edges that have not yet

been visited in the original plan at the time of plan revision, as well as newly required edges that might arise from changes in the environment's geometry. The edges in E'_R must be repartitioned among the k' robots so that the inspection task can be completed.

The supervisory agent partitions the edges in E'_R into k' groups, $F'_1 \cup ... \cup F'_k = E'_R$ using the farthest point clustering algorithm described in Section IV-A.1. The k' representative edges used as the basis for partitioning are chosen based on the robots' current positions. For robot i already engaged in the inspection task and currently traversing the jth edge of its tour, the representative edge f'_i will be edge $j + 1$ in the robot's current tour. Representative edges for robots new to the task, which are deployed from the depot, are chosen from E'_R such that the sum of the minimum distance through G to the existing representative edges is maximized.

As in the initial planning process, after the robots are assigned their respective groups of required edges, the remainder of the planning procedure can take place online.

2) Include edges for connectivity: At the time of path revision, because the robots will not necessarily start their new tours from the depot vertex, a new effective depot vertex must be specified for each robot. For robot i already engaged in the inspection task and currently traversing the jth edge, the end vertex toward which the robot is moving will serve as the new depot $v'_{depot,i}$. For robots new to the task, which are deployed from the depot, $v'_{depot,i} = v_{depot}$ (this need not be the original depot, but for simplicity we assume it is). This step is executed for the groups $F'_1, ..., F'_k$, each connected to its respective $v_{depot,i'}$, as described for $F_1, ..., F_k$ and the single v_{depot} in IV-A.2.

3) Compute a tour of each subgraph G'_i: This step is executed for the subgraphs $G'_1, ..., G'_k$, as described for $G_1, ..., G_k$ in IV-A.3, yielding tours $T'_1, ..., T'_k$.

4) Tour refinement: Because the new tours T'_i terminate at $v_{depot,i}$, if $v_{depot,i} \neq v_{depot}$, once every required edge in a tour has been visited, the rest of the tour is replaced with the shortest path to the original point of deployment, v_{depot}. As in IV-A.4, for each tour T'_i, sequences of edges that are retraced in the course of a tour are replaced with the shortest path in G, if a shorter path between the end vertices of the sequence exists.

VI. Simulation Results and Discussion

The simulations presented below are aimed at showing both improved graph complexity (as compared to [1]) and our algorithm's ability to replan after a change to the team or task.

A. Reduction in Graph Complexity

To generate a typical graph representation with the technique previously presented in reference [1], the user selects a subdivision step size parameter directly affecting the number of vertices $|V|$ in G and an edge-weeding parameter directly affecting the number of edges $|E|$ in G. In that construction technique, edges represent straight paths through freespace and

the waypoints in the tours that are calculated all correspond to vertices. In the construction presented in this paper, waypoints along the paths represented by graph edges are simply a parameter associated with each edge. These waypoint lists contribute only to the storage space required for the graph and do not add to its complexity. A graph generated using the method in Section III will have far fewer vertices than a graph generated with typical values of the user-selected parameters as presented in [1]. In the example graphs in Figure 2, generated for typical values of these user-selected parameters, we observe that the new implementation offers reduction in $|V|$ by an order of magnitude while offering an arbitrarily fine spatial resolution of waypoints. Such savings in $|V|$ were consistently seen in the examples explored.

This reduction of complexity greatly improves the running time of the constructive heuristic. The most expensive steps in the algorithm are heavily dependent on the number of vertices in the graph, especially those incident to the edges in E_R:

1) the all pairs shortest path calculation used in the partitioning step (Section IV-A.1) has time complexity $O(|V_R|^3)$ [7], where V_R is the set of vertices incident to E_R.
2) the minimum spanning tree calculation used in the connectivity step (Section IV-A.2) has time complexity $O(|E_{F_i}| log |V_{F_i}|)$ [10] for each group F_i.
3) the single postman tour calculation (Section IV-A.3) has time complexity $O(|V_{G_i}|^3)$ [9] for each subgraph G_i.

For the values of k used in simulations ($k \leq 10$), team size did not play a large role in the running time. Using the new graph representation for the example environment shown in Figure 2(b) and running on a Pentium-4 processor (3GHz CPU), the constructive heuristic runs to completion in 7.79 seconds for $k = 5$. Using the implementation in [1] results in the graph of Figure 2(a), and a time to completion of 39.0 seconds. To further examine the explosion in running time for relatively small changes in parameters, we tested the construction presented in [1] with subdivision step size parameters of 1 and 0.5, which had $|V| = 2237$ and $|V| = 4155$, respectively, keeping all other parameters fixed. Their respective running times were 17 minutes 6.0 seconds and 96 minutes 13.3 seconds. The vast improvement in the running time offered by use of the new graph representation enables us to tackle the problem of path revision mid-task.

B. Revision: Change in Robot Team Size

To illustrate the path revision method, consider the case of robot failure mid-task.

For our first example, the environment shown in 2(b) is used, again with sensing parameters $r_{min} = 5, r_{max} = 25$. For $k = 3$, routes planned and executed successfully are shown in Figure 3(a). Figure 3(b) illustrates the progress made at the time the robot carrying out the route shown in blue fails. The complete post-revision paths are shown in Figure 3(c). A second example, with sensing parameters $r_{min} = 5, r_{max} = 50$, is shown for $k = 3$ in Figure 4.
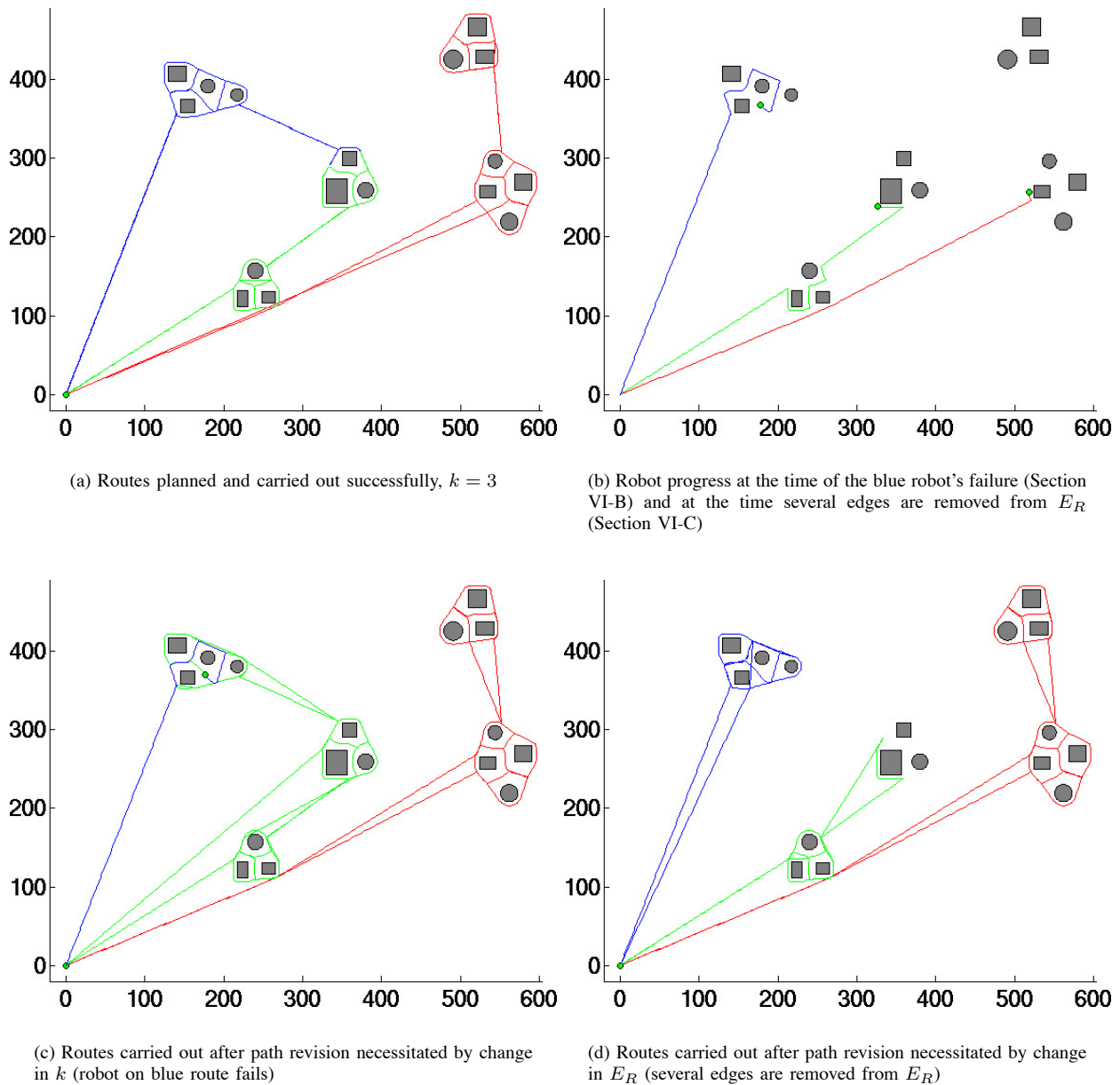
(a) Routes planned and carried out successfully, $k = 3$

(b) Robot progress at the time of the blue robot's failure (Section VI-B) and at the time several edges are removed from E_R (Section VI-C)

(c) Routes carried out after path revision necessitated by change in k (robot on blue route fails)

(d) Routes carried out after path revision necessitated by change in E_R (several edges are removed from E_R)

Fig. 3.   Illustration of examples of path revision presented in Sections VI-B and VI-C for a sparse environment with several inspection regions

In the first example, we note that the edges "abandoned" by the failed robot are reassigned to the robot executing the route drawn in green, while in the second, the edges are divided among both remaining robots. This is due to the nature of the partitioning process and to the structure of the environment: in the first example, at the time of failure, every required edge in the blue route is closer to the robot following the green route than the robot following the red. In contrast, the robots in the second example are routed through a more closely packed environment and the remaining robots share the abandoned edges. The repartitioning results in such cases depend more heavily on the robots' positions at the time of failure.

We note that in the example of Fig. 3, the differences in

the pre- and post-revision red route are due to the fact that the route is replanned from a new "depot" point, even while the edges yet to be inspected remain the same. This replanning is not necessary, but is inexpensive when used with the new graph construction method and, as in this case, can improve upon the previously planned path. Selecting whether to use the original tour or the revised tour in cases such as this is a feature that will be included in a future implementation.

Note that in addition to the path revision method of Section V, we also explored a method for post-failure path revision which preserved the partitioning calculated in the initial execution of the constructive heuristic. This method repartitioned only the unvisited required edges in F_i initially assigned to
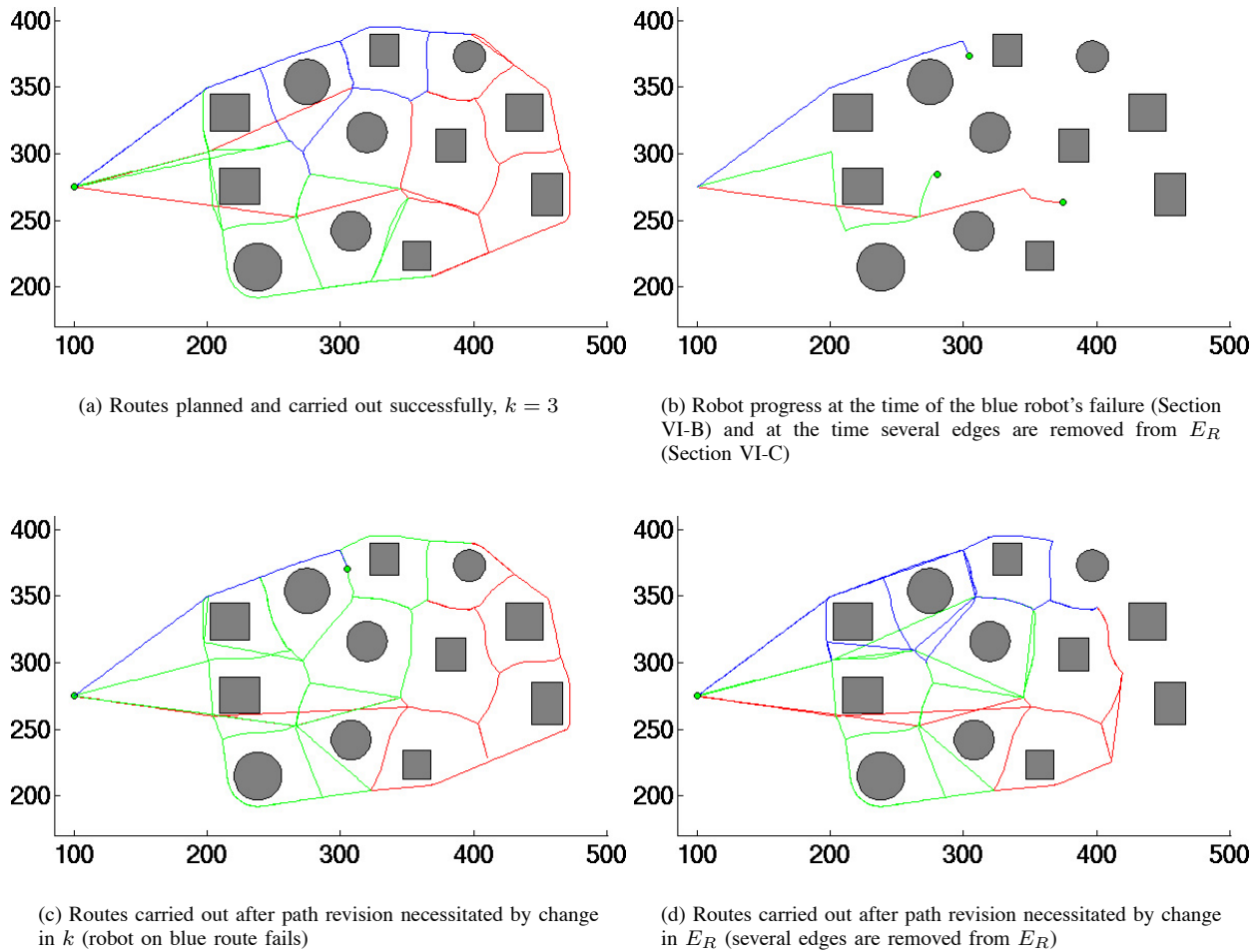
(a) Routes planned and carried out successfully, $k = 3$

(b) Robot progress at the time of the blue robot's failure (Section VI-B) and at the time several edges are removed from E_R (Section VI-C)

(c) Routes carried out after path revision necessitated by change in k (robot on blue route fails)

(d) Routes carried out after path revision necessitated by change in E_R (several edges are removed from E_R)

Fig. 4.   Illustration of examples of path revision presented in Sections VI-B and VI-C for a more densely packed environment with a single inspection region

the failed robot i. By reducing the number of edges being partitioned, the time complexity of the repartitioning step taken during path revision was significantly reduced. This method did not lend itself to extensions such as revision after addition of robots or changes to G, however, and when the savings in running time this method offered were overwhelmed by the savings in running time due to the new graph representation, we pursued the more computationally intensive but more flexible method presented in Section V instead.
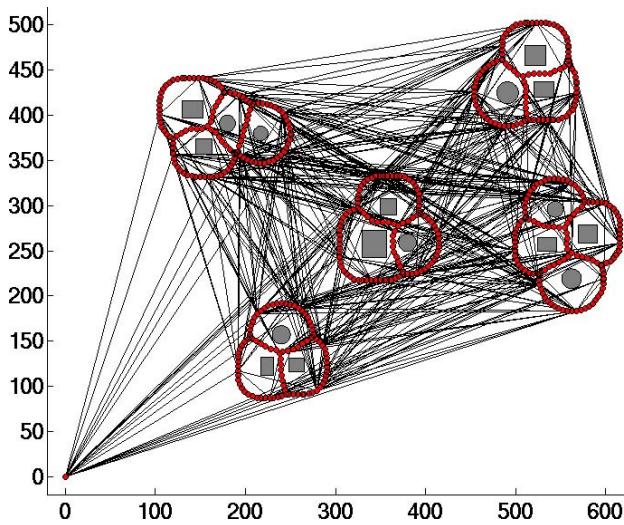
C. Revision: Change in Inspection Task

Path revision can be a useful tool when information is acquired mid-task. Consider an example where, initially, every boundary point in our example environment must be inspected. The examples illustrated in Figures 3(a) and 4(a) are used again. Mid-task, again at the point illustrated in Figures 3(b) and 4(b), the planning system receives a command to ignore some of the boundary segments (for example, information about these segments comes from other sources, and thus they no longer need to be inspected). Fig. 3 illustrates an example in which, mid-task, the supervisory agent instructs
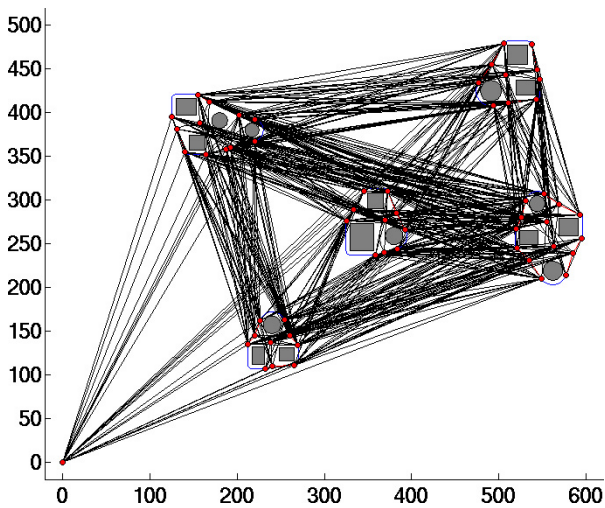
the team to ignore the boundary segments in the centermost inspection region. Fig. 4 illustrates a case in which, mid-task, the supervisory agent instructs the team to ignore three rightmost objects. The edges associated with these objects are thus removed from E_R, and the traversal of the remaining unvisited edges in E_R is divided among the k robots by the path revision algorithm.

The complete post-revision paths, no longer incorporating inspection of these objects, are shown in Figures 3(d) and 4(d). In such cases, the revision of the robots' paths allows them to bypass now-unnecessary inspection and complete the task more quickly.

In future work, we will explore how new or unexpected environmental information obtained by a robot may be integrated into the global graph representation; combined with our plan revision method, this will be a step toward a more sensor-based approach, relaxing our perfect localization assumption and allowing us to consider undertaking tasks with incomplete environmental information.

(a) Graph generated with technique presented in [1] for $r = 25$, subdivision step size= 5, weeding parameter = 50; $|V| = 582, |E| = 1044$

(b) Graph generated using method in Section III with $r_{min} = 5, rmax = 25$; $|V| = 57, |E| = 648$

Fig. 2. Comparison of graph construction methods

VII. CONCLUSIONS

This paper introduced an improved graph-based representation of the boundary coverage problem and utilized the considerable computational savings offered by the new representation to explore plan revision. We see the path revision capability presented here as a step towards making the graph-based approach more robust to failure and open to the addition of complementary or collaborative subtasks. The use of multiple robots offers the opportunity for transient, but potentially unplanned, collaborations between teammates, a task extension we intend to explore in future work. For example, a robot

with a complementary sensory suite might detour from its assigned task to aid another robot that has found an interesting target. Plan revision will be a necessary component in the implementation of such extensions. Plan revision necessitated by changes in the underlying graph representation will also be a necessary component in future work addressing changes to the known environment as unexpected features are detected during an inspection tour. With the ability to replan taking into account changes to the team and to the task itself with only a minor delay in task execution, we move a significant step closer to online planning. We intend to develop this application of graph methods to multi-robot tasks further, continuing to move towards online planning, exploring the opportunity for robot collaboration, and pursuing practical implementation of these methods.

REFERENCES

[1] K. Easton and J. Burdick, "A coverage algorithm for multi-robot boundary inspection," *Proc. 2005 IEEE International Conference on Robotics and Automation*, 2005.
[2] N. Correll and A. Martinoli, "Modeling and analysis of beaconless and beacon-based policies for a swarm-intelligent inspection system," *Proc. 2005 IEEE International Conference on Robotics and Automation*, 2005.
[3] ——, "Collective inspection of regular structures using a swarm of miniature robots," *Proc. of the Ninth Int. Symp. on Experimental Robotics ISER-04*, 2004.
[4] Y. Zhang, E. K. Antonsson, and A. Martinoli, "Evolving neural controllers for collective robotic inspection," *Proc. of the 9th Online World Conf. on Soft Computing in Industrial Applications*, 2004.
[5] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.
[6] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.
[7] D. Ahr and G. Reinelt, "New heuristics and lower bounds for the min-max k-chinese postman problem," in *Algorithms-Esa 2002, Proceedings*, ser. Lecture Notes in Computer Science, 2002, vol. 2461, pp. 64–74.
[8] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, no. 2-3, pp. 293–306, 1985.
[9] J. Edmonds and E. L. Johnson, "Matching, euler tours, and the chinese postman," *Mathematical Programming*, vol. 5, pp. 88–124, 1973.
[10] J. Kruskal, "On the shortest spanning subtree of a graph and the travelling salesman problem," *Proc. American Math. Society 7*, pp. 48–50, 1956.