Market-based Coordination of Coupled Robot Systems

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

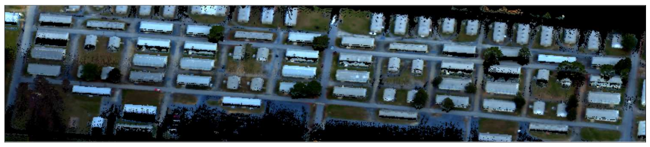
***Abstract*— Tasks such as street mapping and security surveillance seek a route that traverses a given space to perform a function. These task functions may involve mapping the space for accurate modeling, sensing the space for unusual activity, or searching the space for an object. In many cases, 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 multiple 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 where k represents the number of robots. Using this representation, the problem can be solved using a branch-and-bound approach to find an optimal route, and a route division heuristic to separate the route into k pieces. Since the branch-and-bound technique is exponential time, we present an approach to decompose the search problem into subtasks that are distributed among the robots. Using ideas from market-based approaches, we allow the robots to auction particular sections of the problem space to other robots as a way to more evenly divide the work and focus the search. Finally, we evaluate these methods on test graphs in simulation.

I. Introduction

Market-based approaches are an established paradigm for coordinating multiple robots. The approach organizes the robot team as a market economy, paying revenue for accomplishing tasks and assessing costs for consuming resources. The robots are free to bid on tasks to maximize their profits. The team mission is best achieved when the economy maximizes production and minimizes cost. Some features of the approach are that communication is opportunistic but typically not essential. The robot leader emerges through consensus. The system is robust to single points of failure, lost robots, changing tasks, and unknown or dynamic environments. A survey of the existing approaches in the domain is provided by [1].

To date, the approach has been used for missions that can be easily decomposed into independent [2] or loosely coupled [3] subtasks; however, there are no published optimality guarantees. The problem is that finding the optimal solution for the whole team is impossible by considering each robot's solution in isolation. In this paper, we pursue a new market approach. Rather than paying each robot to find the best plan for itself, we pay them to find the best plan for the whole team. We do this by partitioning and distributing the multi-robot problem or solution space, where each partition is disjoint yet preserves robot couplings within the partition. A point in the partition represents a complete solution for the entire team. We assign partitions to robots. The robots compute solutions in parallel using combinatorial optimization techniques and auction parts of their search space on which other robots can bid. The robots use these auctioned partitions to augment their search spaces. Finally, robots can also circulate their solutions to other robots as a way to terminate the other robots' solution searches.


(a)

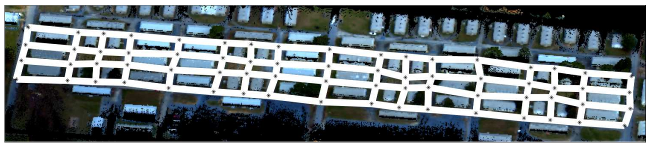

(b)

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.

We have extended these ideas toward the multi-robot coverage problem with incomplete prior information. Many tasks, such as robotic surveillance and patrol, require a robot to visit points in an environment to accomplish a goal. We model these tasks as coverage problems where a robot is required to visit most locations in a given area. We assume a map of the environment is available. Because we seek optimality guarantees on the solution path for coverage efficiency, we choose to use a graph representation as the foundation of our solution approach.

In a 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 direction. Another example is a Voronoi diagram where the paths are

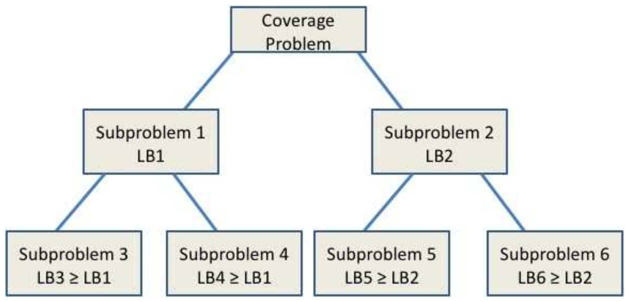


Fig. 2. This shows a search tree generated using branch-and-bound where each subproblem has a lower bound value and the parent nodes always have equal or lower bounds than their children.

edges in the graph and the path intersections are nodes. This is one way to generate paths with optimality bounds for some of the problems in continuous space coverage [4].

In this work, we seek coverage paths for multiple robots that visits a designated subset of edges in the graph. When the number of robots is one, this coverage problem can be modeled as an arc-routing problem: the Rural Postman Problem (RPP). The RPP seeks a coverage tour that traverses a required subset of the graph edges using the extra graph edges as travel links between the required edges. A common framework to address this NP-hard problem [5] is the branch-and-bound algorithm [6] which creates a search tree to partition and explore the solution space as shown in Figure 2. By partitioning the coverage problem into several subproblems, each subproblem provides a lower bound for its branch of the search tree. The algorithm always chooses the branch with the best lower bound to search first, and the optimal solution is one that has the lowest cost of all the branches. Note that branch-and-bound partitions the space in the same way advocated by our market approach.

When the number of robots is more than one, this coverage problem can be modeled as a Min Max k-Rural Postman Problem (k-RPP) where k tours, one for each robot, are sought that together visit a required subset of the graph and the goal is to minimize total traversal time by minimizing the maximum cost path. Heuristics for this problem include cluster first, route second approaches [7][8][9]. While these methods are computationally efficient, they do not necessarily provide high quality solutions. In order to obtain better solutions, we use the optimal algorithm for the RPP to find a minimum cost solution for the entire problem first, and then divide the optimal route among the k robots using a $(2 - \frac{1}{k})$-optimal approximation algorithm [10]. This route first, cluster second approach is extended from [11], where it was found to provide higher quality solutions for the k-Chinese Postman Problem compared to the cluster first, route second approach used.

If the size of the k-RPP problem is large, finding the solution can be expensive. To address large problems, we introduce two contributions. Our first contribution is a bounded algorithm for the k-RPP problem that uses a parallel branch-and-bound approach to find the initial route [12]. This approach divides and assigns the search space to the robot team for parallel computation. Including the optimal algorithm as part of the route first, cluster second approach results in bounded solutions for the k-RPP. Since the routing step is the more expensive portion of the algorithm, we focus on minimizing the computation time for the search since it will, in turn, minimize the total computation time for the k-RPP problem. Our second contribution is an auction strategy between the robots that allows the the robots to circulate partitions between one another as a way to focus the search. Rather than having each robot work independently on a particular part of the solution space, we can exploit the communication network to share information between robots to find the solution faster. Finally, we evaluate the parallel branch-and-bound approach and the auction strategy in simulation.

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

II. Approach

Our approach consists of two parts. First is the parallelizing of the branch-and-bound algorithm that splits the search space between k robots. For this parallelization, the robots only share the solutions generated at the end of their individual branch-and-bound processes. The second part of our coverage algorithm expands the communication between the robots through the use of auctions. During the branch-and-bound process, the robots can auction newly generated branches on which the remaining robots can bid. In this manner, robots can share the search space using the auction framework.

A. Parallel branch-and-bound

The parallel branch-and-bound approach is a way to assign partitions of the branch-and-bound search tree to multiple robots for computation. While we are framing this problem as multi-robot coordination, this approach also works for a cluster of computers or processors. Although many different auction algorithms could be used, we start by awarding partitions based on availability only. This first scheme is essentially equivalent to a parallel branch-and-bound framework where the search tree is divided into a few partitions that can be evaluated at the same time (Figure 3) on different robots or processors. By dividing the computation among several robots, the optimal solution can be obtained more quickly.

Now, we will detail the coverage algorithm for distributing the solution space for the k-RPP problem among k robots. The initial problem consists of a single graph that contains required and optional edges. The first step is to separate the problem into multiple subproblems. To accomplish this, one robot is designated the team leader, and the algorithm for the leader is shown in Alg. 1. The leader first divides the initial problem into k subproblems using an abbreviated

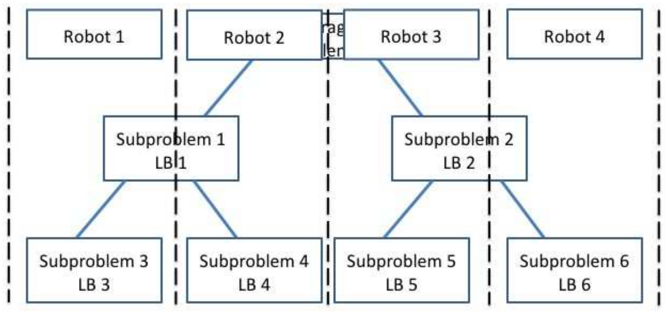


Fig. 3. This represents the parallel branch-and-bound algorithm where the search tree is divided into four partitions and distributed to the robots.

version of branch-and-bound (shown in Figure 3), and sends the first $(k-1)$ subproblems out as shown in lines 1 and 2. Next it works on its own subproblem while waiting to receive solutions from all the other robots (line 3). When all the solutions are received and the leader has solved its own subproblem, all the solutions are sorted based on cost, and the one with the lowest cost is the optimal solution (line 4). Finally, the solution is split into k routes using an algorithm by Frederickson, Hecht, and Kim [10] (line 5). Their algorithm operates in the following manner. It begins with a tour $s...s$ where s is the depot vertex. Next, it partitions the tour into k sections with similar cost: $\{(s...v_1)$, $(v_2...v_3)$, ..., $(v_k...s)\}$ where v_i is a vertex along the path. Finally the path segments are modified to begin and end at s by adding a shortest path segment from v_i to s or from s to v_i when needed.

```
  Input: $s$: depot,
         $k$: number of robots,
         $G = (V, E)$ where $E = \{E_R, E_T\}$ where $E_R$
         and $E_T$ are the required and travel set of edges,
         respectively
  Output: $P_{1...k}$
1 $G_1, G_2, ..., G_k$ = SplitProblem($G$, $s$, $k$);
2 SendToOtherRobots($G_1, G_2, ..., G_{(k-1)}$);
3 $S_k$ = ProcessProblem($G_k$);
4 $S_{min}$ = SortSolutions($S_1, S_2, ..., S_k$);
5 $P_{1...k}$ = FHK($G$, $S_{min}$, $s$);
6 return $P_{1...k}$
```

Algorithm 1: k-RPP algorithm for the leader robot

The remaining robots that are not the leader receive their individual subproblems, and work on them using the same ProcessProblem method as the leader. To process their problems, each robot uses the branch-and-bound algorithm to generate a tree that searches through combinations of optional edges using the assigned subproblem as the root. For each branch of the tree, the algorithm generates two subproblems by including and excluding an optional edge. Next, each subproblem in the branch-and-bound tree is solved using the polynomial-time Chinese Postman algorithm [13] [14]. The cost of the Chinese Postman solution is the lower bound on the cost of the solutions for its corresponding branch. Once a solution is found, its optimality is proven by comparing its value to the lower bounds of the remaining branches of the tree. To keep computation costs low, we condense the optional edges into optional path segments. The details of this process can be found in [15]. Once a robot finds a solution to their partition, it broadcasts the solution to the other robots. This broadcast enables the other robots to terminate their searches if the broadcasted solution has a lower cost than any of the subproblems in their search trees.

The k-RPP algorithm is dominated by the branch-and-bound algorithm which has a complexity of O($|V|^3 2^t$) where t is the number of optional path segments and $|V|$ is the number of vertices in the graph. The algorithm returns a solution with an approximation factor of $(2-\frac{1}{k})$-optimal.

B. Auction strategy

In the parallel branch-and-bound framework, the robots are constrained to a particular part of the search space depending on the initial assignment. With a single branch-and-bound tree, the algorithm always processes the lowest cost subproblem in the entire tree first, but with the separated search trees, some robots may be wasting their computation on unpromising parts of the search space since they do not have access to other parts of the space.

As a result, adding more communication between the processes can lead to more efficient computation since it can help direct the search to the more favorable parts of the space for all the robots. Translating this idea to parallel branch-and-bound, when several robots are working on disjoint subproblems at the same time, if communication is added between the robots, when one robot finds a new partition, it can auction the partition and lower bound to the other robots. In some cases, the auctioned partition can have a lower bound that is better than those of the other robots. In these cases, the robots with less promising partitions can bid to win the more lucrative partition. For example, in Figure 4, when robot 1 finds a new partition (subproblem 7), it broadcasts the lower bound (LB7) to the remaining robots. The other robots compare the new partition to their current partition. Robot 4 finds that subproblem 7 has a better lower bound that its current partition, so it bids for subproblem 7, wins the bid, and directs its search to the new partition.

The second improvement to the algorithm is for the robots to continue to bid on new problems even if they have found a solution. This prevents the idling of robot or processor resources, and is another way to distribute the work more evenly among the robots.

We now discuss the modified coverage algorithm that adds communication between the robots. Within the ProcessProblem method, as each of the robots generates their search tree, the robots will occasionally auction a newly added branch or subproblem in their tree to the other robots. If another robot want this subproblem, it will indicate this to the sender. The sender can decide which robot to give the subproblem to based on a heuristic – for this work, the sender uses a simple heuristic of first come, first serve. The only complication

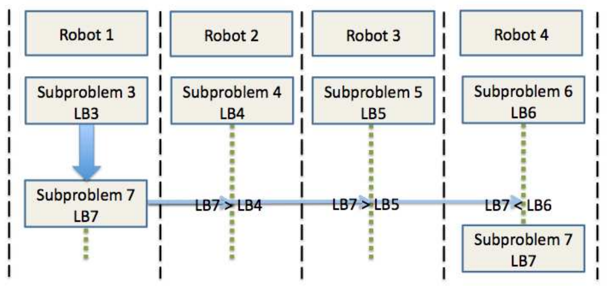


Fig. 4. This demonstrates the communication strategy where each robot is working on a part of the search tree. With communication, robot 1 is able to auction part of its search, subproblem 7, to other robots. In this case, robot 4 decides to accept subproblem 7 from robot 1.

that can arise is if the sender itself is currently processing the subproblem (this can occur if the subproblem had a low cost and low cost subproblems are processed first). In this situation, the sender does not send the subproblem out.

Finally, while the algorithm currently operates with one robot assigned as the leader, it can be easily modified to be fully distributed where each robot is sent the initial problem. The robots would each call the SplitProblem method and work on the subproblem that corresponds with their id. The best solution for each robot would be sent to the other robots and each robot can then sort and find the optimal solution.

III. Testing Framework

Our testing evaluated the performance of the parallel branch-and-bound algorithm and the auction strategy. We ran the tests on a single machine with four processing units, each of which contained a 2.67GHz Intel Core i5 processor, and they all shared 8GB of RAM. To simulate a team of robots, we represented each robot as a computer process running an instance of the algorithm on separate cores. One process was designated the leader. Each process communicated with the others through the network layer using the TCP protocol.

There were two main sets of tests: one evaluating the performance of the parallel branch-and-bound algorithm without the auction strategy, and the second evaluating the parallel branch-and-bound with the auction strategy. To describe the testing framework, we will first explain the test graphs, then the procedure for each test, and finally present the metrics for performance evaluation.

For our testing, we used rectilinear graphs. Rectilinear graphs are graphs where the vertices are generated uniformly in the plane. Edges connect one vertex to its four closest neighbors, and are vertical and horizontal connectors that represent the Euclidean distance between the vertices. In other words, these graphs consist of a regular pattern of nodes and edges or arcs. We chose rectilinear graphs because they are similar to the networks used in many coverage applications. For example, in street coverage, the graph layout is similar to a grid where all streets (edges or arcs) meet at intersection points (nodes), and most intersections are four ways. For our testing, three graph sizes were used; the sizes are $|V| = \{10 \times 10, 14 \times 14, 17 \times 17\}$. For each graph, a random set of edges were selected to be arcs. To keep runtimes reasonable, graphs taking longer than ten minutes of computation were excluded from the test set.

A. *Parallel branch-and-bound without auctions*

The parallel branch-and-bound tests ran each algorithm on fifteen distinct rectilinear graphs (five for each graph size). The edge costs for each graph were random numbers between 1 and 50. We varied the number of robots k from 1 to 4. Each combination of graph and k value was run five times.

B. *Parallel branch-and-bound with auctions*

For the second set of tests, our goal was to assess the performance of the parallel branch-and-bound algorithm with auctions between the robots. The graphs used were the same as the ones used in the first set of tests, and k ranged from 2 to 4, and, as before, each test combination was run five times. To avoid heavy network traffic, we limited the robots to only auction a portion of their subproblems. More specifically, subproblems were only auctioned on a time interval that was randomly chosen between zero and one second.

C. *Metrics*

During each call to the coverage algorithm, we computed the maximum size of the search trees for each robot, computation time, and cost and variance of the k final paths.

IV. Results

Before presenting the results, we explain our notation and data organization. Information about the graphs is shown in Table I. The first two columns display the average number of nodes and edges for each graph size. The last two columns show the average number of required and optional edges, respectively.

TABLE I

$\|V\|$	$\|E\|$	$\|E_R\|$	$\|E_O\|$
100	180	102.6	77.4
196	364	242.6	121.4
289	544	360	184

For the results, the size of the search tree is represented by the number of branches in the largest search tree generated by the k robots, and the computation time is the total k-RPP time. Because these values can be drastically different for different graphs, we scaled them in the following manner. We computed the size of the search tree and computation time for a single robot working on the same problem; these values are the baseline for each graph. For each of the results, we characterize improvement as the baseline value minus the value obtained with k processes normalized by the baseline value. As a result, in the Figures 5, 6, 7, and 8, the amount of improvement is shown along the y-axis – the higher the value, the better. Additionally, in these figures, the x axis of denotes the set of fifteen test graphs and the y values are the improvement averages over the five trials for each graph.

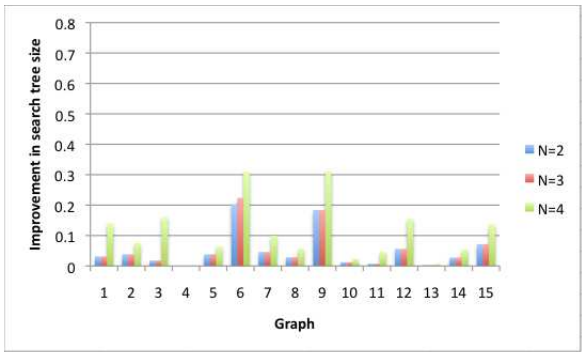


Fig. 5. The improvement in the number of branches in the search tree for the coverage algorithm with no auctions. The number of processors/robots ranged from 2 to 4.

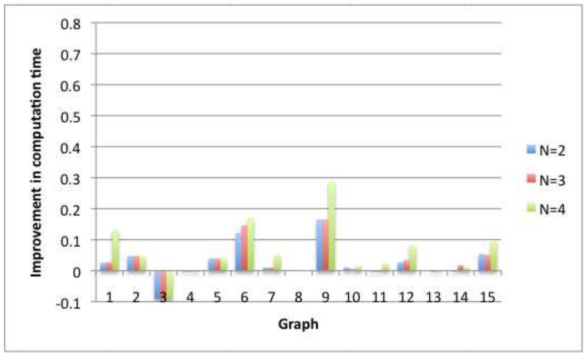


Fig. 6. The improvement in the total computation time for the coverage algorithm with no auctions. The number of processors/robots ranged from 2 to 4.

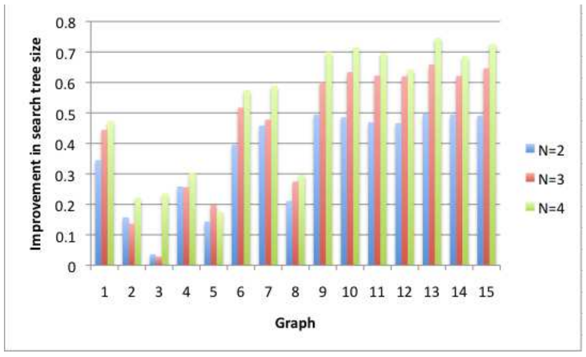


Fig. 7. The improvement in the number of branches in the search tree for the coverage algorithm with auctions. The number of processors/robots ranged from 2 to 4.

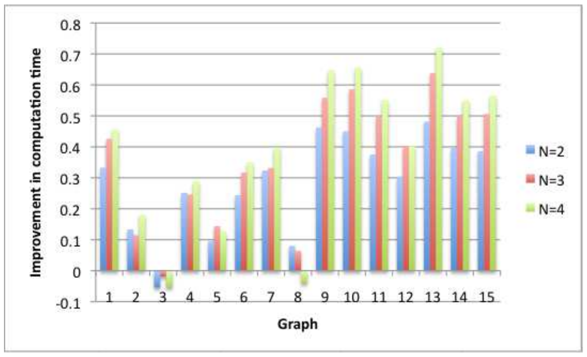


Fig. 8. The improvement in the computation time for the coverage algorithm with auctions. The number of processors/robots ranged from 2 to 4.

A. *Parallel branch-and-bound without auctions*

The results show that dividing the work between multiple processes helps reduce the size of the maximum search tree by decreasing the number of branches in the tree (Fig. 5). Since the solution space is initially split into k sections and the optimal solution resides in one of these sections, in most cases, the work is not distributed evenly among the robots. However, based on the results, it is clear that increasing the number of processes or robots can reduce the size of the maximum individual search trees such as for graphs 6 and 9. In terms of total computation time, for most of the graphs, using more robots leads to a decrease in runtime. However, there is an overhead in the parallel k-RPP algorithm since it consists of splitting the initial problem, and includes some delays due to communication. For graph 3 in Fig. 6, the computation time gets worse when more than one robot is used to parallelize the search effort. For this particular graph, because the runtime of search using a single robot is fairly low at 1.5 seconds, the overhead costs are relatively high in comparison resulting in worse performance with more robots. Since most of the coverage problems that are solved using branch-and-bound are computationally more expensive, the overhead of adding communication is negligible as shown by the remaining graph instances.

B. *Parallel branch-and-bound with auctions*

Adding the auction strategy to the parallel k-RPP algorithm enables the processes or robots to share the work better by distributing problems more evenly. Overall, the addition of the auctions improves the efficiency of the search algorithm by reducing the number of branches as shown in Figure 7. Compared with the same algorithm run without auctions, the performance for all instances of k improves when information is shared. In some instances, such as graph 1, the relative differences between the improvements for the different values of k shrinks. Furthermore, in a lot of cases, the use of auctions enables a team of robots to perform much better than a larger team size. For example, for graph 12, two robots using the auction strategy performs roughly twice as well as four robots without auctions. Similar to the no auctions case, there is less improvement with graph 3 due to the shallow search tree. With more communication between the robots, there is a higher overhead which is reflected in the computation times in Figure 8. These overhead costs can mask the improvement which occurs with graphs 3 and 8.

C. *k-RPP route division results*

For the k-RPP problem, we are trying to minimize the maximum or worst cost path for k robots. The goal of MinMax is to better distribute the work and reduce the total

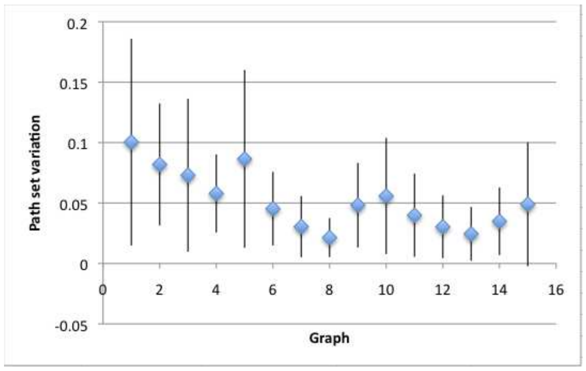


Fig. 9. The normalized value and standard deviation of the variation in cost of the k final paths. Each value represents the average over k where k ranges from 2 to 4.

traversal time for the team. As a result, we computed the path variation of the k final paths. Figure 9 shows the variation of the paths normalized by the mean of the paths. This value is averaged over all values of k. As shown, the variation is fairly low for the different graphs. By using an optimal algorithm, the solution includes the minimum set of optional edges. Furthermore, the route division process is bounded which ensures few extraneous edges are added. As a result, the final solution is a set of paths that are similar in cost.

V. Discussion

Our results indicate that parallelizing the branch-and-bound algorithm, and employing auctions between the processes can solve large coverage problems more efficiently. Distributing the search tree among k processes alone may not result in major improvements in efficiency since the promising solutions may not be distributed evenly among the partitions. In some cases, the size of the search tree with more processes is the same as for a single process as shown by graphs 4, 10, and 11 in Figure 5. However, for some problems such as graphs 6 and 9, distributing the initial problem can significantly reduce the branching factor in the search process.

When auctions occur between the processes, the efficiency significantly increases for all values of k. The auctioning of subproblems during the search process allows the robots to share the whole search space. In this manner, the k robots act more like a single robot since they have access to and can evaluate solutions outside their initial assignment. In most instances, auctioning subproblems helps reduce the amount of computation performed by each robot. However, in some instances, the sharing of subproblems can lead to certain branches which may look promising but are dead ends to be distributed among all the processes leading to larger search trees and high computation times such as the case of graph 5 with four robots in Figures 7 and 8.

VI. Conclusion

This work comprises a first implementation of an optimal market solution for coupled robot systems where a combinatorial optimization technique, the branch-and-bound algorithm, is parallelized for a team of robots and combined with an auction framework between the robots to solve large coverage problems more efficiently. As the results show, the market-based approach helps divide the solution space of the coverage problem among multiple robots, but allows the robots to still access the entire space through the use of auctions. These techniques enable multiple robots to generate a bounded solution for the k-RPP through working together. For future work, we plan to investigate different auction and bidding strategies for more efficient searches.

Acknowledgment

This work was conducted in part through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. Additionally, this work was partially sponsored by the ONR Contract Number N00014-09-1-1031.

References

[1] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, 94(7):1257 –1270, 2006.

[2] R. M. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-Robot Exploration Controlled by a Market Economy. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3016 – 3023, May 2002.

[3] E. G. Jones. *Multi-Robot Coordination in Domains with Intra-Path Constraints*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2009.

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

[5] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc Routing Problems, Part II: The Rural Postman Problem. *Operations Research*, 43(3):399–414, 1995.

[6] E. L. Lawler and D. E. Wood. Branch-And-Bound Methods: A Survey. *Operations Research*, 14(4):699–719, 1966.

[7] 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.

[8] 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.

[9] L. Xu and A. Stentz. An Efficient Algorithm for Environmental Coverage with Multiple Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2011.

[10] 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, Oct. 1976.

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

[12] B. Gendron and T. G. Crainic. Parallel Branch-and-Bound Algorithms: Survey and Synthesis. *Operations Research*, 42(6):pp. 1042–1066, 1994.

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

[14] E. Minieka. *Optimization Algorithms for Networks and Graphs*. M. Dekker, New York :, 1978.

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