Performance Based Task Assignment in Multi-Robot Patrolling

Charles Pippin
Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, Georgia 30332
pippin@gatech.edu

Henrik Christensen
Center for Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, Georgia 30332
hic@cc.gatech.edu

Lora Weiss
Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, Georgia 30332
lora.weiss@gtri.gatech.edu

ABSTRACT

This article applies a performance metric to the multi-robot patrolling task to more efficiently distribute patrol areas among robot team members. The multi-robot patrolling task employs multiple robots to perform frequent visits to known areas in an environment, while minimizing the time between node visits. Conventional strategies for performing this task assume that the robots will perform as expected and do not address situations in which some team members patrol inefficiently. However, reliable performance of team members may not always be a valid assumption. This paper considers an approach for monitoring robot performance in a patrolling task and dynamically reassigning tasks from those team members that perform poorly. Experimental results from simulation and on a team of indoor robots demonstrate that in using this approach, tasks can be dynamically and more efficiently distributed in a multi-robot patrolling application.

Categories and Subject Descriptors

I.2.9 [**ARTIFICIAL INTELLIGENCE**]: Robotics—*workcell organization and planning*; I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Multiagent systems*

Keywords

Multi-robot cooperation, task assignment, auction algorithms, trust, performance monitoring, ROS

1. INTRODUCTION

The area of multi-robot systems has been an active area of research for many years, due in no small part to the ability for a team of robots to operate more efficiently and be more robust to failure than a single robot. However, there are still many challenges related to the interaction between the robots themselves. In conventional multi-robot systems approaches, each team member explicitly operates as part of a team and is generally trusted. However, future robotic teams may have different quality levels and operational capabilities. For instance, a team member may have trouble cooperating due to communication errors, or because they are busy performing other tasks, or even because of conflicting goals [1]. Robot teams may need to consider which team members are trustworthy and dynamically adjust their teaming and task assignment strategies accordingly.

The multi-robot patrolling problem is a surveillance task that uses multiple robots to visit every important location in a known environment [4]. Each location in the environment must be visited repeatedly, with the problem being to minimize the time in-between visits. This problem is interesting from a multi-robot research perspective, because it presents challenges in optimization and task assignment, cooperation, communication and reliability. This problem also has useful implications for real world scenarios, including those in the surveillance, security, and search and rescue domains.

Cooperation is important in this task, as it is necessary for the robots to work together to improve the efficiency of the system as a whole. An effective multi-robot patrol team should be able to visit points more efficiently and with greater reliability than a single robot. However, reliability is also important, particularly in security applications. For instance, if robots on the team do not perform as expected, the system should degrade gracefully.

The performance criterion considered in this paper is the *refresh time*, which is the time gap between any two visits to the same location[1]. The maximum refresh time reflects the bounds on the effectiveness of a robot team in detecting events in the environment [13]. If a robot fails to perform its assigned tasks or visits locations too infrequently, this will affect the performance of the team. To mitigate such performance issues, other robots could be assigned some of these tasks, thereby decreasing the maximum refresh time. This paper presents a dynamic approach for observing which team members patrol poorly, and can be used to more effectively perform patrol task allocation by reassigning those locations with the greatest refresh time to other team members using a bidding mechanism between the better performing team members.

[1]In the literature, this is also referred to as the *idle time* of a node.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAC'13, March 18-22, 2013, Coimbra, Portugal.
Copyright 2013 ACM 978-1-4503-1656-9/13/03 ...$15.00.

This paper is organized as follows. In Sections 2 and 3 we present the related work and review the problem formulation for the multi-robot patrolling task. In Section 4, we discuss the use of a performance monitor and a task reassignment approach. In Sections 5 and 6, we present results of experiments using this approach, performed in simulation and on a team of indoor mobile robots. Finally, in Section 7, we conclude and present future work.

2. RELATED WORK

The problem of cooperative patrolling by a multi-robot team has received considerable attention recently in the robotics literature [13, 16, 7, 18]. This problem is similar to the well known 'Art Gallery' problem [10], in which each location in an art gallery must be viewable by a guard. However, here the tasks are to repeatedly visit the locations in the environment and to minimize the amount of time in-between visits.

A theoretical analysis of the patrolling problem is provided by Chevaleyre [4]. The results showed that the problem could be solved with a Traveling Salesman Problem (TSP) approach. This is extended to the multi-robot case by spacing each of the robots evenly along the path [4, 5]. Chevaleyre also showed that it makes sense to partition the graph in some cases, particularly when there are long corridors or edges separating clusters of nodes. Experiments were performed in simulation using existing real robot architectures and realistic simulation environments in [7, 16].

An approach for reassigning tasks from poorly performing team members was presented in Parker's L-ALLIANCE framework, in which a robot monitored a peer robot and took over a task from when the time for completing the task exceeded a threshold [11]. Pippin and Christensen presented an approach to monitoring robot performance as compared to the performance of the team for determining when a robot's performance could be considered out of control [14]. Lewis and Weiss [8] developed a collection of metrics to measure the performance achieved when collaboration is allowed among vehicles, including the gain obtained over the base capability of the robots operating independently.

In other works, market-based auction methods were applied to the patrolling problem as an approach to the initial node assignment [6, 9]. Auction methods are a class of decentralized algorithms that solve the multi-robot task allocation problem by splitting computation across multiple nodes and iteratively performing task assignments [3]. These algorithms serve as a mechanism for distributed task allocation and generally do not explicitly consider individual team members' performance characteristics. However, individual robots on a team may have varying levels of performance. In this work, we update the multi-robot patrol task with a mechanism for monitoring task performance and reassigning patrol locations using an auction based mechanism.

3. MULTI ROBOT PATROLLING

Many recent approaches to the patrolling task represent areas in the environment with a topological map (a graph) [15]. The nodes in a graph represent areas of interest in the environment, and edges in the graph represent traversable paths between two locations. Applying the notation from the literature, we can refer to the graph as $G(V, E)$, where $V = 1 \ldots n$ is the set of nodes and E is the set of edges. A weight is associated with each edge, $e_{i,j}$, representing the distance between each edge. The graph is assumed to be metric and undirected. Let $R = 1 \ldots r$ be the team of robots to assign the set of nodes in each of the r graph partitions. When the patrol task begins, there is an initial startup time for all robots to navigate to their assigned starting nodes in the graph and to begin patrolling. Robots patrol simultaneously and repeatedly along the graph, visiting their assigned patrol nodes, according to a given strategy [4].

Chevaleyre presents two main classes of patrolling strategies, the cyclic strategy and partition based strategies [4]. In the cyclic based strategies, a single closed path, s, is generated that visits all of the nodes in the graph at least once. In the single robot case, a robot travels this closed path indefinitely. In the worst case, the amount of time for a robot to visit a node twice while following this strategy is equal to the length of s. Calculating the closed path is known to be *np-hard*, and this problem is closely related to the *Travelling Salesman Problem* [4].

In the multi-robot case, the simplest approach is to space the robots along the closed path such that during the patrol they maintain a constant distance between them [4, 5]. Cyclic strategies have known optimality bounds and are preferred when the graph does not contain long edges that connect clusters of nodes [4]. From a reliability perspective, when one robot malfunctions, the remaining $(r - 1)$ team members can simply space themselves evenly over the patrol cycle and continue patrolling. However, there are situations in which robots may have degraded performance, but continue to function. In these situations it would be desirable to allow the poorly performing robot to continue to perform a subset of its original patrol path.

Graph partition approaches divide the graph into subsets of nodes and assign these nodes to individual robots on the team. Pasqualetti et al. present optimality bounds for three major types of partition based patrol graphs: cycles, trees, and chains, and remark that the selection of the roadmap may not be unique for an environment and that the performance can vary based on the choice of the graph structure [13]. For the partitioning case, a cyclic graph can be transformed into an acyclic roadmap using min-max path cover approaches or a chain partition approach. For acyclic graphs, a tree based approach can be used. For the purposes of this paper, we convert a cyclic roadmap of the environment into a chain partition, using the approximation algorithm described in [12]. This algorithm is easy to compute and has known optimality bounds. This approach is described further in the next section.

4. APPROACH

In this section, we present our approach to performance monitoring and task reassignment of robots performing the patrol task. We assume that the a map of the environment is provided in advance, and that further a topological map is generated from this map and provided to each of the robots. At startup, each of the r robots on the team partitions this graph and assigns the r^{th} partition to itself, calculating a closed cycle over the partition. There is an initial startup time for each robot to navigate from its starting location to the first node in its closed path. Once each robot begins patrolling, it is observed by a central monitor process which keeps track of the maximum refresh time of each robot's assigned nodes. If a robot's performance exceeds a threshold, based on the performance of its team members, one of the poorly performing robot's nodes is offered to the rest of the team and re-assigned using an auction based protocol.

4.1 Graph Partitioning

To obtain the initial graph representation of the environment we perform a series of pre-processing steps. We begin with the bitmap image file which represents a map of the environment generated from a mapping process and use the EVG-THIN software from Beeson [2] to generate a Voroni graph. This is the same approach used by Portugal and Rocha [15]. Next, we perform additional pre-processing on the graph to prune short leaf nodes and to merge nodes that are close together.

From this graph, we calculate the minimum spanning tree (MST) to remove cycles in the graph. Next, we compute an open tour of the edges that visits all of the vertices (nodes) in the graph by starting with a leaf in the MST and visiting each branch of the tree, shortest edges first. Finally, we use the chain partition algorithm to divide the path among the r team members. This determines the initial assignment of nodes to robots.

4.2 Performance Monitor

In this paper, we adopt the performance metric of maximum refresh time. When the refresh time of any robot's assigned nodes exceeds a threshold on this metric, we seek to re-assign some of that team member's nodes to other team members. In our approach, we assume the existence of an external monitor that can fully observe the visits to each node. Each robot self reports node visits to the monitor which tracks the refresh time for each node. At each time step, the monitor can calculate the node with the maximum refresh time for each robot. We set the amount of time in between performance monitoring periods to be the expected maximum refresh time for the patrol partition.

The monitor compares each robot's maximum refresh time, defined as the maximum refresh time of all nodes assigned to that robot, against the average refresh times of all currently trusted team members. Specifically, we define a control threshold at one standard deviation, σ, above the average max refresh time for the team. When a robot exceeds this threshold, the monitor marks this robot as untrusted and performs a task reassignment.

The max refresh time for a robot is the maximum refresh time for all nodes assigned to robot r. Let I_k^r be the set of the refresh times at the previous k node visits for a robot, r. Let I_n^r denote the refresh time of a node visited by robot r and being the nth visit by r to any node assigned to it. The running max refresh time, $M_k^r = \max(I_{k..n}^r)$, is the observed maximum refresh time for a robot over the window $(n, n-1, \ldots, n-k)$, where $n > k > 0$. The threshold for the max refresh time, $\theta_{maxrefresh}$ is defined as the average running max refresh time over all robots, plus one standard deviation:

$$\theta_{maxrefresh} = \overline{M_k} + \sigma \tag{1}$$

We define a patrol period as the expected amount of time to perform a patrol of the maximum partition plus a constant factor. This factor is included to capture the additional time needed to navigate due to the non-holonomic motion of the robot and related to time spent navigating around obstacles. At the end of each patrol period, the monitor checks whether $M_k^r > \theta_{maxrefresh}$ for each robot. In that case, a robot is considered to be performing poorly and is marked as *untrusted*. The monitor then selects one node to be reassigned from all *poorly performing* robots, using the process described in the next section.

4.3 Auction Based Task Reassignment

We use a market-based auction algorithm to perform task reassignment from the *poorly performing* robots to the remaining robots on the team. This approach reassigns a single node from the set of *poorly performing* robots during each monitor period. The pseudocode for this process is shown in Fig. 1. Recalling that the joint patrol partitions form a single patrol chain, we seek to exchange those nodes that are the ends of a robot's chain partition. These nodes comprise the leaf nodes of a partition. Let L denote the set of leaf nodes belonging to all known *poorly performing* robots. Then, our approach is to reassign a node from L to another team member that is currently performing well and considered to be a *trusted* performer.

This approach varies from the general auction approach in the literature in that tasks are initially assigned according to the graph partitioning algorithm described in Section 4.1. Here, auctions are only used to reassign tasks. A central auctioneer performs the auction, announces the task winner and reassigns the task. The first step in the process is to calculate the set L over the partitions. A separate auction is announced to all team members for each node, $n \in L$, by sending an *Announce Auction* message with the node identifier.

4.3.1 *Bid Calculation*

Upon receiving the *Announce Auction* message, each robot calculates a bid for adding the new, candidate node to its patrol partition as follows. The candidate node is added to the list of the existing nodes in the robot's partition, along with any intermediate nodes along the minimal path to the candidate node. Next, the minimum spanning tree (MST) of the subgraph is calculated and an open tour that visits all of the nodes in the MST is generated by visiting each of the branches in the MST, shortest branches first, as described in [13]. The candidate max refresh time of the new partition, $c_{\max(i)}$, is calculated from the new tour by computing the path distance along the tour for a round trip: $c_{\max(i)} = PathDist(tour) * 2$. Finally, $C_{\max(i)}$ is submitted as the bid for this robot.

4.3.2 *Winner Selection and Task Reassignment*

The auction approach seeks to reassign a node from one of the poorly performing (*untrusted*) robots to a robot that is performing well by selecting from among the bidders that will result in the smallest candidate refresh time. Intuitively, this reassigns a node from a poorly performing robot to one that is comparatively underutilized by assigning it to the robot that would still have the smallest candidate path. Note that this is different from assigning the node to the robot with the smallest marginal cost for adding the candidate node. The latter could result in robots that disproportionately grow their patrol paths.

The auction algorithm selects the bid with the minimum max refresh time from the set of bids received from all trusted robots for all $n \in L$. To ensure iterative improvements, the monitor also keeps track of the max observed

```
 1: for all $p : PoorPerformers$ do
 2:     for all $n$ : leaf nodes in $Partition_p$ do
 3:         $a \leftarrow AnnounceAuction(n, R)$
 4:     end for
 5:     $C \leftarrow ReceiveBids(A) \forall R \in Trusted$
 6:     $w \leftarrow Min(MaxRefresh(A))$
 7: end for
 8: $AnnounceWinner(w, n)$
 9: if Receive $ACK_w$ then
10:     $ReassignTask(p, n)$
11: end if
```

Figure 1: ReassignTask() pseudocode.

refresh time during the current patrol cycle and will not award a node to any candidate bid that exceeds this value. The winning bid is sent as an *Announce Winner* message to the winning bidder. The auctioneer waits to receive an acknowledgement message from the winning bidder before performing the task reassignment from the original robot. This is necessary to ensure that at least one robot is still including this node in its partition.

When a robot receives the *Reassign Task* message, it removes the node from its current partition, and recalculates the MST and open tour for the new partition. It is assumed that a robot will relinquish the node when this message is received. However, even if it does not, this will still result in an improved refresh time for the node because the winner robot will also cover that node.

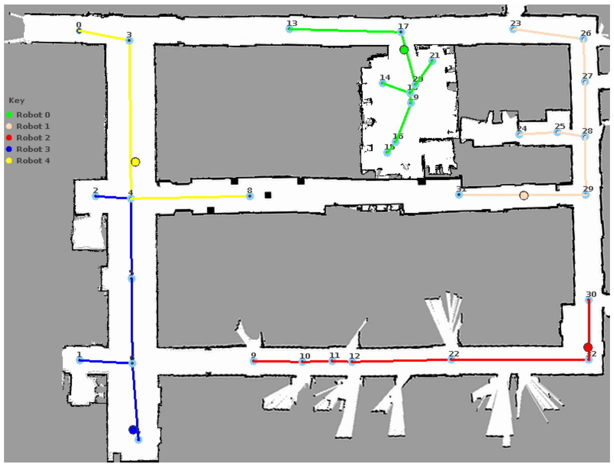


Figure 2: Using the chain partition algorithm, the graph is initially partitioned into approximately equal tours for each robot.

5. EXPERIMENTS

A set of experiments were performed to demonstrate that the use of the reassignment approach can improve the performance of the team by re-assigning tasks away from poorly performing team members, thereby reducing the overall max refresh time of nodes in the graph. Each of the r robots is given a map of the environment and the full patrol graph. At startup, each $robot_i$ locally partitions the graph into r separate partitions and assigns the i_{th} partition to itself.

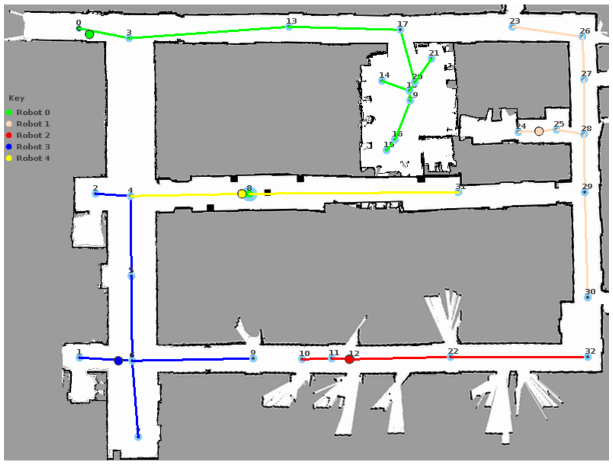


Figure 3: The partitions have been updated after several task reassignment operations as a result of poor performance by robot 2.

The central task monitor is available and has full visibility into the arrival of robots at nodes. Robots send node visit messages to the monitor using the network interface. The monitor keeps track of the refresh time for each node, as well as the maximum refresh time for all nodes.

In these experiments, a subset of the robots are marked as *poor performers*. The performance for this type of robot is affected by randomly adjusting the maximum forward velocity of the robot after each visit to a patrol node. The robots that perform normally have a maximum speed of 0.25ms^{-1} and the maximum speed of the *poor performers* is determined by sampling from a normal distribution with $\mu = 0.15\text{ms}^{-1}$ and $\sigma = 0.10\text{ms}^{-1}$.

5.1 Experiments in Simulation

Three different experiment types were performed:

naive strategy The robots patrol the set of nodes in the initial partition. A subset (1 or 2) of the robots on the team are marked as *poor performers*.

auction strategy A central monitor observes the performance of the robots on the team and reassigns tasks using the auction based approach as described in Section 4. A subset (1 or 2) of the robots on the team are marked as *poor performers*.

all perform The robots patrol the set of nodes in the initial partition. None of the robots are *poor performers*.

Simulations were run with 3, 5, and 8 robots on a team. Each simulation ran for 2 hours of simulated time. For the 3 and 5 robot teams, 5 experiments of each type were performed, while 2 experiments were run for each 8 robot team type, resulting in over 80 hours of simulated patrols. The experiments used the Stage multi-robot simulation environment [19], shown in Fig. 4. The open-source Robot Operating System (ROS) architecture [17] was used to implement the robot messaging, low level control and behaviors.[2] Each

[2]The setup was patterned after the ROS patrol simulation from: http://www.ros.org/wiki/isr-uc-ros-pkg#patrol.

robot uses the ROS navigation stack for navigation, localization, and obstacle avoidance. Each robot also runs a custom *Patrol* behavior which implements the graph chain partition algorithm, and repeatedly navigates to the nodes in the robot's patrol path. This behavior also implements the auction protocol and calculates the robot's bids. The simulation also includes a central monitor node which listens for task completion messages and includes the performance monitoring and task reassignment components. Robots communicated with the central monitor by sending messages using UDP broadcast over the local network.

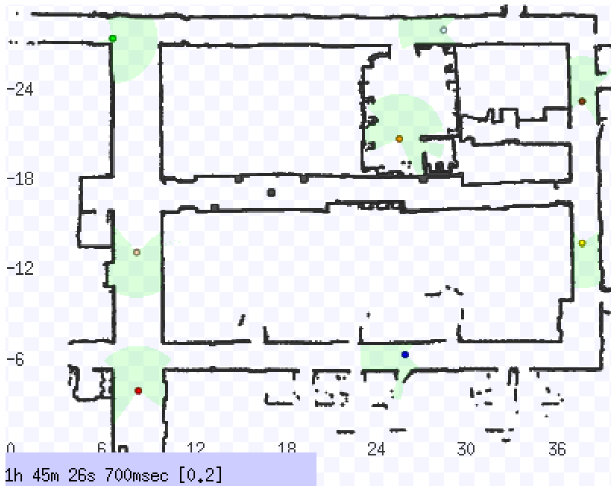


Figure 4: The Stage multi-robot simulator is shown with eight robots patrolling the environment. The robots in the simulation run the ROS navigation stack and participate in auctions for task reassignment.

For each experiment, we track the *running max refresh time* of the overall patrol. This value represents the maximum refresh time of any node in the environment, computed over a window of the last τ seconds. We set the value for τ to be greater than twice the expected time to complete a patrol, to prevent cycling of the value, due to the out and back nature of the open tour in each partition.

5.2 Robot Experiments

A second set of experiments was performed using the TurtleBot indoor mobile robot.[3] The robot has a bumper sensor and a single axis gyroscope. The robot also uses a Kinect sensor, which includes an infrared laser projector and corresponding infrared camera which generate range data of the scene for indoor distances up to 6 meters. The turtlebot carries a netbook laptop which runs Linux and the same ROS libraries and behaviors used in the simulation experiments.

The experiments were performed using a team of three TurtleBots in the same office environment, shown in Fig. 5, that was mapped for use in the simulations. Three experimental runs were performed with a central monitor performing the auction strategy and a single *poor performer* on the team. In the first two runs, after observing multiple patrol cycles, the monitor observed the *poor performer* on the team and used the auction-approach to reassign one of its nodes to other team members. Those runs were ended after the successful node reassignments.

[3]http://turtlebot.com

The third run was executed for approximately 90 minutes, with all three robots patrolling continuously during that time. The robot trajectories during this run are shown in Fig. 8. After several initial cycles, the monitor auctioned and reassigned nodes from the *poor performer*, *robot 0* to *robot 1*. Later, after several more patrol cycles, the monitor reassigned another set of nodes, this time from *robot 0* to *robot 2*. At this point, no more nodes were reassigned as the *running max refresh times* across the team were similar.

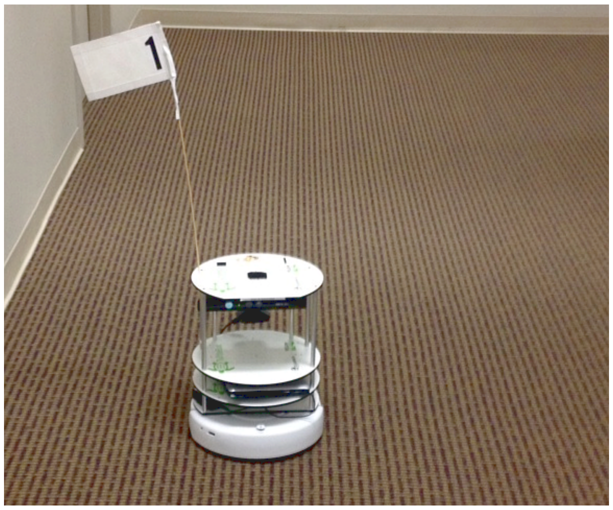


Figure 5: A TurtleBot shown patrolling in the hallways of an office building during the robot team experiments.

6. RESULTS AND DISCUSSION

In our first result, we compare the *running max refresh time* for each of the three types over the full patrol time. An example result from experiments with three robots is shown in Fig. 6. In the case where all robots perform as expected, the *running max refresh time* is approximately constant, and is close to the calculated value for the max partition patrol time (it is slightly greater than the calculated value, due to the non-holonomic motion and obstacle avoidance behaviors of the robot.) For both the naive and auction cases, the *running max refresh time* values vary, due to the random sampling of the velocity for the *poor performer* robot. However, after some initial task reassignment, the performance of the auction method improves over that of the naive approach. The auction approach reassigned multiple nodes from the *poor performer* robot to robots with neighboring partitions. After the neighbors were assigned these tasks, the *poor performer* is well below maximum refresh time. Subsequent reassignments shifted tasks from the *poor performer's* neighbors to their neighbors.

The results for different team sizes were averaged over all of the experimental runs, and are shown in Fig. 7. In all experiments, the auction based approach to task reassignment resulted in better performance than the naive approach to patrolling which does not consider individual robot performance. In the set of experiments with 1 *poor performer* out of 8 robots, the auction approach unexpectedly resulted in better performance than the all perform case. We attribute this result to the obstacle avoidance behavior of the robot.

Here, there are fewer nodes for each robot to visit, and inefficiencies in robot motion are more noticeable. In this case, a robot that was modeled as a good performer had difficulty navigating a narrow doorway on the right side of the environment, and this caused the robot to slow down on this leg of its tour, resulting in unexpected poor performance and increasing the max refresh time for the entire patrol. The auction based method reassigned one of this robot's assigned nodes to a neighbor robot and this resulted in a decreased the maximum refresh time.

The experiments with the team of robots demonstrated the use of this approach in a real-time patrolling scenario in an office environment. Here the poor performance dimension of varying speed was artificially introduced. However, this approach could be applied more generally to other performance dimensions.

In this paper, the tasks being reallocated are patrol areas. However, it should be noted that this approach to monitoring and task reallocation is not limited to graph partitions in the multi-robot patrolling problem. More generally, this approach can be applied to any domain that requires task allocation amongst multiple robots with the possibility of variance in performance across team members. Another particularly interesting aspect is the notion of trust and reputation. Here, it is assumed that the team members have the intent to perform tasks, but perform them poorly, due to errors in navigation, control or hardware. Once a robot's performance moved below the θ threshold, it was no longer considered trustworthy for assigning new tasks. However, we observed that in some situations, an individual robot's performance decreases when it takes on additional tasks for the benefit of the team. An improved trust model should take this into account. Our ongoing work considers situations in which robots maintain models of trust about team members that they have interacted with, and share those models within the community in a decentralized manner. Furthermore, while the auction based task assignment mechanism is decentralized, in this paper, the auctioneer and monitor were implemented on a centralized node. However, the monitor and auctioneers could also be distributed. This is also a subject of our ongoing work.

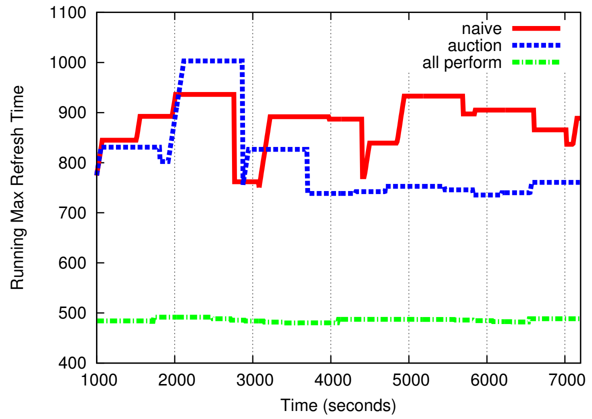


Figure 6: Results are shown for a team of 3 robots with 1 poor performer and compare Max Refresh Times for the naive and task re-assignment approaches and the case in which all robots perform as expected.

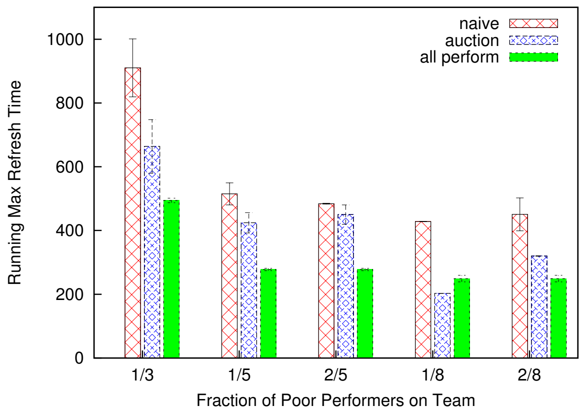


Figure 7: The auction based strategy is compared to the naive strategy when there are *poor performers* on the team and to the case where all robots perform as expected. The error bars represent one standard deviation.

7. SUMMARY

This paper presents a method for recognizing which robots are performing poorly in a multi-robot patrolling task. Both simulated and robot experimental results using this approach were presented. The experiments showed that a monitoring approach can be effective for detecting poorly-performing team members. In addition, a task reassignment mechanism can be effective for more efficiently allocating patrol tasks, when compared to the naive approach which doesn't monitor individual robot performance. This may prove useful in situations in which multi-robot teams are dynamically formed or when not all team members are likely to perform effectively over time. The results show that by observing the performance characteristics of individual robots, tasks can be allocated more efficiently than the approaches which do not consider performance.

Future work will consider learning mechanisms relevant to task performance, as well as to study the problem using a more distributed approach to task monitoring and reassignment, and also consider the case of limited communications. Finally, we plan to perform additional experiments using our team of TurtleBot indoor mobile robots.

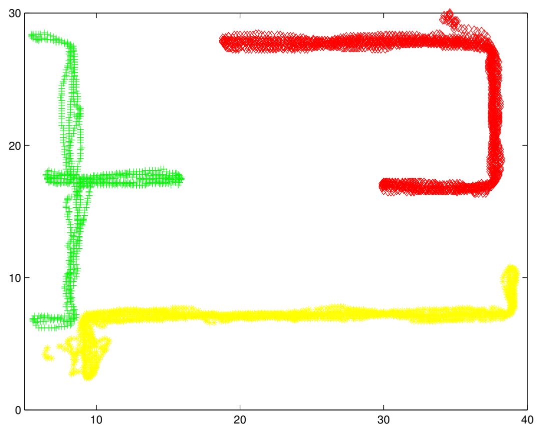
8. ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers for their helpful comments. We also appreciate the helpful comments from Fabio Pasqualetti regarding that author's related work on graph partitioning for the patrolling problem. This work was internally funded by the Georgia Tech Research Institute.

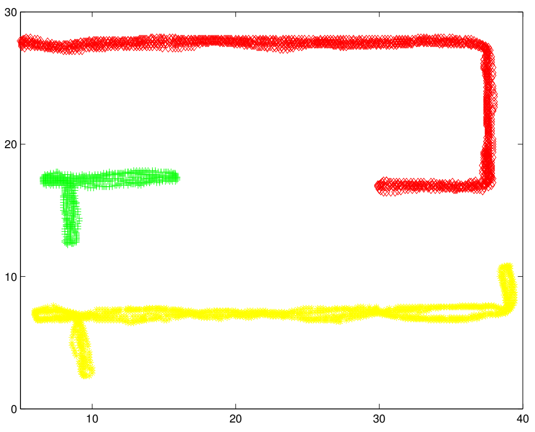
9. REFERENCES

[1] R. C. Arkin. *Behavior-Based Robotics*, chapter 9. Cambridge, Mass., MIT Press, 1998.

[2] P. Beeson, N. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *Robotics and Automation (ICRA)*, pages 4373 – 4379, April 2005.



(a) Initial Partition



(b) After Task Reassignments

Figure 8: A set of trajectories is shown for a team of three TurtleBot indoor robots while patrolling an office environment, with 1 poor performer on the team (the leftmost, green trajectory). (a) After initial startup, the robots patrolled and partitioned the environment using the chain partition algorithm. (b) After observing multiple cycles, the central monitor auctioned and reassigned tasks away from the poor performer to others.

[3] D. P. Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, 20(4):133–149, 1990.

[4] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pages 302 – 308, Sept. 2004.

[5] Y. Elmaliach, N. Agmon, and G. Kaminka. Multi-robot area patrol under frequency constraints. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 385 –390, April 2007.

[6] K.-S. Hwang, J.-L. Lin, and H.-L. Huang. Cooperative patrol planning of multi-robot systems by a competitive auction system. In *ICCAS-SICE, 2009*, pages 4359 –4363, aug. 2009.

[7] L. Iocchi, L. Marchetti, and D. Nardi. Multi-robot patrolling with coordinated behaviours in realistic environments. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2796 –2801, Sept. 2011.

[8] A. S. Lewis and L. G. Weiss. Intelligent autonomy and performance metrics for multiple, coordinated UAVs. *Integrated Computer-Aided Eng.*, 12(3):251–262, July 2005.

[9] T. Menezes, P. Tedesco, and G. Ramalho. Negotiator agents for the patrolling task. In J. Sichman, H. Coelho, and S. Rezende, editors, *Advances in Artificial Intelligence - IBERAMIA-SBIA 2006*, volume 4140 of *Lecture Notes in Computer Science*, pages 48–57. Springer Berlin / Heidelberg, 2006.

[10] T. Michael. *How to Guard an Art Gallery and Other Discrete Mathematical Adventures.* Baltimore: The Johns Hopkins University Press, 2009.

[11] L. E. Parker. ALLIANCE: An architecture for fault tolerant multi-robot cooperation. In *IEEE Transactions on Robotics and Automation*, volume 14, pages 220–240, 1998.

[12] F. Pasqualetti, A. Franchi, and F. Bullo. On optimal cooperative patrolling. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7153 –7158, Dec. 2010.

[13] F. Pasqualetti, A. Franchi, and F. Bullo. On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *Robotics, IEEE Transactions on*, 28(3):592 –606, June 2012.

[14] C. Pippin and H. Christensen. Performance based monitoring using statistical control charts on multi-robot teams. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 390 –395, July 2012.

[15] D. Portugal and R. Rocha. MSP algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1271–1276, New York, NY, USA, 2010. ACM.

[16] D. Portugal and R. Rocha. On the performance and scalability of multi-robot patrolling algorithms. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 50 –55, Nov. 2011.

[17] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. *In Proceedings of the Open-Source Software workshop at the International Conference on Robotics and Automation (ICRA)*, 2009.

[18] E. Stump and N. Michael. Multi-robot persistent surveillance planning as a vehicle routing problem. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pages 569 –575, Aug. 2011.

[19] R. Vaughan. Massively multi-robot simulation in Stage. *Swarm Intelligence*, pages 189–208, 2008.