Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference

Multirobot Task Allocation with Real-Time Path Planning

Bradley Woosley[1] **and Prithviraj Dasgupta**[2]
[1]Computer and Electronics Engineering Department, University of Nebraska-Lincoln
[2]Computer Science Department, University of Nebraska, Omaha

Abstract

We consider the multi-robot task allocation (MRTA) problem in an initially unknown environment. The objective of the MRTA problem is to find a schedule or sequence of tasks that should be performed by a set of robots so that the cost or energy expended by the robots is minimized. Existing solutions for the MRTA problem mainly concentrate on finding an efficient task allocation among robots, without directly incorporating changes to tasks' costs originating from changes in robots' paths due to dynamically detected obstacles while moving between tasks. Dynamically updating path costs is an important aspect as changing path costs can alter the task sequence for robots that corresponds to the minimum cost. In this paper, we attempt to address this problem by developing an algorithm called MRTA-RTPP (MRTA with Real-time Path Planning) by integrating a greedy MRTA algorithm for task planning with a Field D*-based path planning algorithm. Our technique is capable of handling dynamic changes in a robot's path costs due to static as well as mobile obstacles and computes a new task schedule if the original schedule is no longer optimal due to the robots' replanned paths. We have verified our proposed technique on physical Corobot robots that perform surveillance-like tasks by visiting a set of locations. Our experimental results show that that our MRTA technique is able to handle dynamic path changes while reducing the cost of the schedule to the robots.

Introduction

Multi-robot task allocation (MRTA) is an important yet challenging problem in robotics with applications in several domains such as unmanned search and rescue, robotic surveillance, mapping and exploration, automated landmine detection, robotic vacuum cleaning, etc. The MRTA problem consists of a set of robots that have to perform a set of tasks. We consider a class of MRTA problems called ST-SR-TA (Gerkey and Matarić 2004), where ST stands for single-task robots, i.e., each robot is able to execute at most one task at a time, SR means single robot tasks, i.e., each task requires one robot to be completed, and TA means time-extended assignment, problems where the information to allocate tasks to robots arrives over time. The problem is complicated because of several reasons - the set of tasks is usually not known beforehand and tasks arrive dynmically. Additionally, the environment is initially unknown and robots have to dynamically plan their path to a task as they detect obstacles on their sensors. The objective of the MRTA problem is to find a schedule or sequence of tasks to be performed by each robot so that the overall cost or energy expended by the robots to perform the set of tasks is minimized. MRTA is known to be an NP-hard problem and several researchers have proposed solutions using heuristics including market-based algorithms, learning-based techniques, vehicle routing-based approaches, etc.. However, much of this research has focussed on determining an efficient task allocation among robots without directly considering the effect of updated path costs between tasks originating from changes in robots' paths due to dynamically detected obstacles while moving between tasks. Dynamically updating path costs is an important aspect to find a consistent solution to the MRTA problem because changing path costs can alter the task sequence for robots that corresponds to the minimum cost. In this paper, we describe an algorithm from MRTA-RTPP (MRTA with Real-time Path Planning) that dynamically combines task and motion planning on multiple robots. In our proposed algorithm, robots dynamically update their path costs upon encountering static as well as mobile obstacles while navigating between task locations using the Field D* algorithm. Updated path cost information is simultaneously integrated with the MRTA algorithm to compute a new task schedule if the original schedule is no longer optimal due to the robots' replanned paths. We have verified our proposed technique by implmenting it on physical Corobot robots that perform surveillance-like tasks by visiting a set of locations and shown that it successfully replans the robot's path on encountering obstacles and navigates the robot along a reduced cost path to visit the locations.

Related Work

The problem of multi-robot task allocation (MRTA) has been investigated using different techniques (Gerkey and Matarić 2004; Mataric, Sukhatme, and Ostergaard 2003), and, recently with market-based approaches (Dias et al. 2006). One of the earliest systems using for MRTA was

the M+ system (Botelho and Alami 1999). In (Gerkey and Matarić 2004) a widely accepted taxonomy for MRTA problems is provided. The problems are classified along three dimensions: (a) single task robots (ST) vs. multi-task (MT) robots, related to the parallel task performing capabilities of robots, (b) single robot (SR) task versus multi-robot (MR) tasks, related to the number of robots required to perform a task, and, (c) instantaneous assignment (IA) versus time extended assignment (TA), related to the planning performed by robots to allocate tasks. Mataric et al. compare performance of robots teams using auction-based strategies for coordination and commitment. Their results show that the least time is required to complete all tasks (put out all alarms) when the robots are allowed to coordinate their plans with each other, as well as to dynamically change their plans (Mataric, Sukhatme, and Ostergaard 2003). The *traderbots* approach by (Dias 2004) uses multi-round, single-item auctions for dynamic task allocation across multiple robots, while in (Jones et al. 2006) the *traderbots* approach is augmented using the Skill, Tactics, Play (STP) approach for coordinated teamwork. The MRTA problem has also been approached as an exploration problem of matching a set of robots to a set of targets using an algorithm called *PRIM-ALLOCATION* (Lagoudakis et al. 2005). Zlot and his team have also used auction-based algorithms for multi-robot task allocation (Zlot 2006; Jones, Dias, and Stentz 2011). The MRTA problem has also been combined with techniques from multi-agent coordination and optimization techniques such as negotiation (Viguria, Maza, and Ollero 2007), reinforcement learning (Schneider et al. 2005), vector regression learning (Jones et al. 2006), Hungarian algorithm (Liu and Shell 2011), and dynamic vehicle routing (Bullo et al. 2011) to improve the performance of the robots and deal with uncertainty. Our proposed algorithm extends these directions of MRTA algorithms by combining MRTA with an online path planning algorithm to accommodate dynamically changing task schedules.

Real Time Path Planning for MRTA

We consider a set of mobile robots that are deployed in a bounded, polygonal environment. Each robot knows the boundaries of the environment but is not aware of features such as obstacles or about the presence or location of other robots inside the environment. The environment's map is divided into a grid-like structure with the size of each grid-cell corresponding to the footprint of a robot. Each robot has a laser sensor for detecting obstacles and other robots, an indoor GPS device for localization, and is capable of wireless networked communication with other robots and a base station.

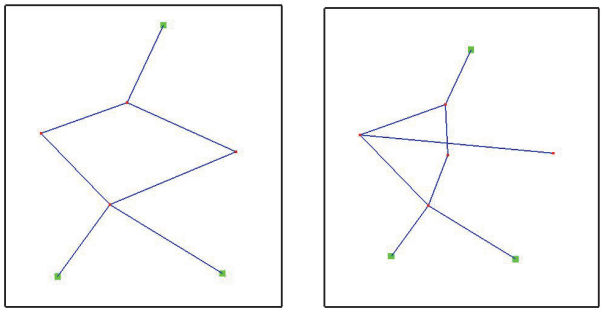
The multi-robot task allocation problem consists of assigning a set of tasks to a set of robots so that the overall cost of performing the tasks by the robots is minimized. Tasks are distributed spatially within the environment and can arrive dynamically. The spatial and temporal distribution of tasks are not known *a priori* by the robots. Also environment features such as obstacles have to be discovered in real time by the robots while navigating in the environment. The MRTA problem in such as scenario is known to be NP-hard. The actual tasks performed by robots are domain specific, such as, reaching a specific location to perform surveillance over a specific area, range clearing, object manipulation, etc. To focus on the performance of the robots in determining and executing an appropriate task schedule, we have assumed that performing a task by a robot corresponds to the robot visiting the location of the task (within a certain radius). A schedule of tasks is given by the set of waypoints corresponding to the locations of tasks that the robot has to perform. The objective of the MRTA problem then is to determine the lowest cost path of its current schedule. This problem is non-trivial as the schedule of tasks gets dynamically updated as new tasks arrive, and, obstacles get discovered along previously determined paths as the environment is being explored while performing tasks (navigating between waypoints). Our proposed approach uses a widely adopted real-time path planning algorithm for robots called Field D* and integrates it with the dynamically updated schedule calculated by the MRTA algorithm using the MRTA-RTPP algorithm, as described in the following sections.

Path Planning with Field D* Algorithm

Field D* (Ferguson and Stentz 2006)is a grid-map based real-time path planning algorithm that allows a robot to plan its path towards its goal and refine the plan after encountering obstacles along the initially planned path. A principal advantage of D* is that it allows to plan a smooth trajectory for the robot through the grid cells by using interpolation, instead of constraining the robot's movement to cells that are at orientations of $\pm\pi$ or $\pm\frac{\pi}{2}$ from its current position. The basic D* algorithm starts by calculating the cost to reach a neighboring cell node from the node at the goal location. It then makes a virtual edge to each neighboring node that would give a low cost path towards the start location. It then repeats the same calculation for the node with the lowest cost so far, and continues until the node with the lowest cost corresponds to the start node. At this point in time, given the information the robot knows, the planed path is the lowest cost path from the start point to the goal point. The algorithm then calls for the robot to begin executing the planned path. To do this, the robot follows the set of node pointers from the start node to the goal node. If the robot's sensors detect an obstacle in front of the robot while the robot is tracing out the path between nodes, the robot's motion stops and it marks the cell that it was attempting to reach as an obstacle and updates the path to the goal. After the update is complete, the robot continues along the new path, which may require it to back track to a node it recently visited and then leave from it to the next node. In the Field D* algorithm, the basic D* algorithm is augmented using interpolation to determine which point along any side of the current grid would be the most efficient point to go to. Those points are stored in the map and when the robot traverses the path given by its plan, it selects the closest point on the edge of the current cell as the next point to navigate to instead of navigating to one of the cell's corners.

Issues with D* Algorithm During the implementation of the Field D* algorithm, there were several challenges arising from the way in which obstacles are detected and marked on the map. The problems encountered and proposed solutions are mentioned below:

- *No unexplored path from current location:* During initial simulation tests of the Field D* algorithm, it was noticed that if there was a case when the robot followed a wall of obstacles into a corner then the robot would stop moving. It was determined that when this happened, the previous paths that had already been explored prevented the robot from finding a new path back to the goal because old paths were not re-planned over if the robot had to move in the opposite direction from how it was already moving. This problem was fixed by checking if the next grid square that the robot was supposed to go to did not exist, meaning, if there was no path found to the goal, all planned paths were removed from the map, but the detected obstacles stayed. After the paths were removed, the path to the goal was re-planned completely from scratch.
- *Avoiding frequent replanning using long-range laser data:* The original obstacle detection in the Field D* algorithm only detected an obstacle that was directly in-front of the robot. This meant that if there was a wall that the robot could not pass, it would drive up to the wall, detect that it was an obstacle, re-plan the path by trying to move to the next grid-square to the right or left of the path. This method of finding obstacles works, but takes a considerable time to compute the re-planned the path. To get around this time consuming problem, we exploited the longer range of the laser sensors on the robot. The locations of obstacles perceived from the laser data are directly incorporated into the map of the environment maintained by the robot. This allows the robot to perceive multiple obstacles at once, requiring fewer re-plans and fewer obstacle hits.
- *Trapped on an obstacle:* When the robot gets into tight spaces, which are about two to three times the size of the robot, as the robot would try to navigate out of the area, it would detect the walls bordering the area it had and incorrectly mark nodes as obstacles, even marking the node which the robot is currently on as an obstacle. There is no way a robot can be sitting on an obstacle, so when this case is detected, the current node is unmarked as an obstacle and the path is re-planned.
- *Working with multiple goal points:* The D* algorithm and its variants have been designed to work with a single start point and a single end point. This works very well in the case where the robot needs to get from the start position to an end position once, but when there are multiple waypoints that the robot needs to visit, there needs to be a method by which the robot can stop on its current path, either when it reaches the goal or when a higher level algorithm such as MRTA decides that a different node is a better end point. For integrating the Field D* algorithm with the MRTA algorithm, when the MRTA algorithm determines a new goal location for the robot, the robot aborts its movement towards it previous goal, removes the old path information while retaining information about the obstacles it has found, and then plans the path from its current location to the new goal.

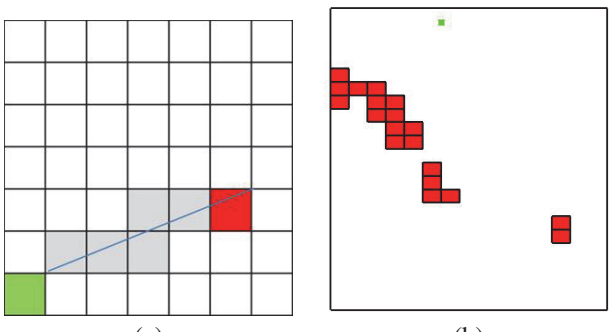


(a) (b)

Figure 1: (a) Initially assigned paths to three waypoints (tasks) for three robots, (b) Updated path with newly added waypoints.

Multi-Robot Task Allocation

As mentioned before, each task is represented as a waypoint for the robot to navigate to. The initial set of waypoints or tasks is provided to each robot. Note that the order of the tasks could be different depending on the robots initial location. For example, the initially allocated path for three robots is shown in Figure 1(a). In this figure, green squares, red dots, and blue lines represent the robots, waypoints, and the path, respectively. When the robots reach a waypoint within a previously defined boundary, the next waypoint is provided to the robot. The robot's path planning algorithm, D*, then generates the optimal plan to get to the next waypoint. The cost incurred by a robot to move between any two points in the environment is assumed to be proportional to the Euclidean distance between the points. Finding the shortest schedule through a set of waypoints is similar to the traveling salesman problem, which is known to be NP-hard, and can be solved using an approximation in polynomial time. A



(a) (b)

Figure 2: (a) Example for Bresenham line, (b) Obstacle grid map generated by Laser sensor

wrinkle to this problem arises from the fact that the order to visit the waypoints could be altered while the robot is trying to reach the waypoint. The first case happens when an obstacle lies on the path generated by D* and the robot discovers the obstacle while traveling towards its current goal point. In such a scenario, the robot has to make a detour around the obstacle and move on a new path different from the original path. Consequently, the cost to its earlier goal point gets recomputed. Secondly, if any new waypoints are added to the set of waypoints while a robot is moving between two waypoints the cost of the schedule changes to accommodate the new waypoints. For example, Figure 1(b) shows the updated path by adding new waypoints.

Obstacle Map Generator

One of the critical components for robot navigation is to figure out where the obstacles in the environment are. For the D* path planning algorithm, the obstacle information needs to be provided represented on the grid map corresponding to the environment. As the robot moves, the grid map needs to be continuously updated using the laser scan data. We have used the Bresenham line algorithm (Bresenham 1965) to update the grid map continuously. The Bresenham line algorithm is an algorithm which determines which points in an n-dimensional raster should be plotted in order to form a close approximation to a straight line between two given points. We consider these two given points as the robot's current position and end point of each laser beam, with $n = 2$. We define two variables in each cell called hitBeam and missedBeam respectively. The value of hitBeam is increased by one at the cell where the obstacle is detected while the value of missedBeam is increased by one at the cell where the Bresenham line passes. Figure 2(a) shows the two points and Bresenham line. In the figure, the green cell is where the robot is currently located, which corresponds to the start cell for the Bresenham line. The red cell is occupied by the obstacle, which corresponds to the end cell for the Bresenham line. So, the value of hitBeam is increased at the red cell. The gray cells are where the Bresenham line passes and the value of missedBeam is increased at those cells. The way to decide if a cell is occupied by any obstacle is to compute the value of the *hitBeam residue* in the cell given by $(\text{value of } hitBeam - \text{value of } missedBeam)$. If the residue is larger than a certain threshold, the cell has an obstacle, otherwise, it is a free cell. Figure 2(b) shows an example of the obstacle grid map generated using the Breshenham line technique.

Moving Robots Avoidance

While dynamically planning the path between waypoints for multiple robots, it is important for robots avoid each other. Because the environment and its obstacles are initially unknown, it is infeasible to calculate the robots' trajectories beforehand. To address this problem, each robot tracks other robots using its laser sensor and GPS. When the trajectories of two robots are about to intersect each other, the robots select a movement to avoid colliding with each other. We consider three behaviors for robots, viz., STOP, GO and DETOUR. For example, if two robots approach each other head on while moving along their respective paths, then the robot with the lower id stops (STOP behavior) while the robot with the higher id is given precedence to move around the stopped robot using the DETOUR behavior. On the other hand, if two robots are imminent to have a head-side collision, the robot with the higher id stops while the robot with the lower id continues to move along its trajectory using the GO behavior. In terms of cost or energy expended by robots to perform one of the above behaviors, STOP incurs more cost than GO, while DETOUR incurs more cost than STOP, i.e., cost(DETOUR) $>$ cost(STOP) $>$ cost(GO). The above technique ensures that the robot with the lower id performs the behavior that incurs lower cost.

Algorithm 1 shows the MRTA-RTPP algorithm that integrates MRTA with real time path planning using Field D* algorithm.

TaskAllocation($Robot_{Pos}$)
  d = compute distance between robot's current position and current goal waypoint;
  if $d <$ *waypoint hit boundary threshold* **then**
    set current goal waypoint visited;
  end
  forall the $w_i \in$ *not visited waypoints* **do**
    Use D* to compute shortest path between Robot's location and w_i;
    $w_{target} = \arg_{w_i} \min(cost(w_i), cost(goal_{prev}))$;
  end
MRTA-RTPP()
  $completedBool = \{false_0, , false_n\}$; //missions completed or not for all robots
  $target = \{0_0, , 0_n\}$ // set of targets for all robots
  while *(any of completedBool is false)* **do**
    forall the $i \in$*robots having false completedBool* **do**
      ith target = **TaskAllocation**(ith robot position);
      if *ith target* $\neq$ *target(i)* **then**
        target(i) = ith target;
        ComputeShortestPath(target(i));
      end
      if *no more target* **then**
        set $completeBool(i)$ = true;
      end
    end
  end

Algorithm 1: MRTA-RTPP algorithm

Experimental Results

For verifying the MRTA-RTPP algorithm we have run several experiments using Corobot robots. The Corobot robot has four wheels, 2 GB RAM, 80 GB HDD and runs Windows XP. One Hokuyo laser range sensor with a range of 40 cm to 4 m was mounted on the robot and used to detect obstacles. Each robot has Wi-Fi capability and can communicate with other robots over an ad-hoc wireless network. For localizing each robot, we attached a Hagisonic Stargazer RS kit on board each robot. This unit provides 2-D coordi-

(a) (b) (c) (d)

Figure 3: (a) Photograph of Corobot robot used for experiments, (b)-(d) Three environments, each with the same five waypoints (tasks) but different distributions of obstacles used in our experiments. The green circle marked 'S' is the robot's start location and the red, numbered circles show the different waypoints.

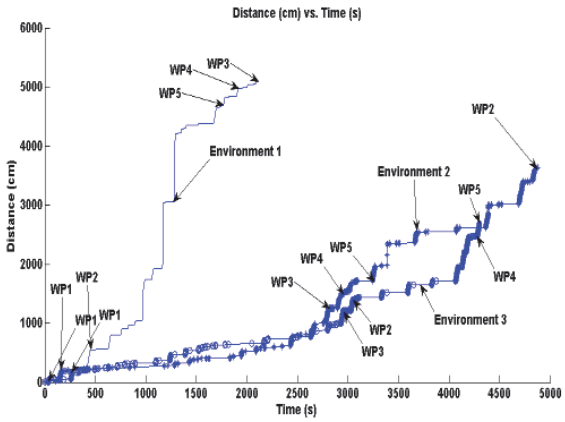


Figure 4: Time taken and distance traveled by a single robot to reach the set of five waypoints in the three different test environments.

nates of the robot's position within the reference frame of the environment. It is capable of localizing the robot within ± 2 cm of its actual location. The footprint of the robot is 40 cm $\times$ 40 cm. A photograph of one of the Corobot robots used in our experiments is shown in Figure 3(a). The default speed of the robot was set to 5.2 cm/sec. The testing arena used for the robots has an area of 4×2 m^2. Five waypoints (tasks) were placed within it at arbitrary locations and obstacles were placed at different locations to create three test environments used for our experiments, as shown in Figure 3(b)-(d).

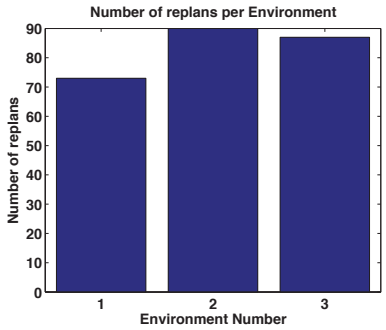
We conducted different experiments to verify the correct operation and quantify the performance of the MRTA-RTPP algorithm. For each experiment, we used 2 Corobot robots used. Robots had to avoid static obstacles and avoid each other (mobile obstacles) while planning their paths. In the first set of experiments, we observe the time required by the MRTA-RTPP algorithm to visit the five waypoints in the three different test environments. In all the environments, the robot starts with the waypoints ordered w_1 through w_5. However, the positions of the obstacles in the three test environments requires the robot to use the Field D* algorithm replan its path. For example, in environment 1 shown in Figure 3(b), the direct path between w_2 and w_3 is blocked. The MRTA-RTPP algorithm determines that a lower cost schedule results if, after visiting w_2, w_5 and w_4 are visited before visiting w_3. Figure 4 shows the time taken and distance traveled by one robot to visit the five waypoints in environments $1-3$ using the MRTA-RTPP algorithm. The horizontal lines along each of the three curves represents time required by the D* algorithm to recompute a path towards a newly provided waypoint determined by the MRTA algorithm when the robot encounters an obstacle while moving towards its previous waypoint. Note that more horizontal lines occur when the order of waypoints gets revised indicating that replanning is triggered when obstacles are encountered while moving towards waypoints in the order given in the original schedule.

In our next set of experiments, we quantified the number of path replans made by the MRTA-RTPP algorithm for the different environments, as shown in Figure 5(a). We observe that the number of replans is the lowest for environment 1 where the plan is revised once after visiting w_2, while it is the highest for environments 2 and 3 where the plan is revised twice, after visiting w_1 and again after visiting w_3, as the robot traverses the boundary of the obstacle segregating w_2 from the rest of the waypoints.

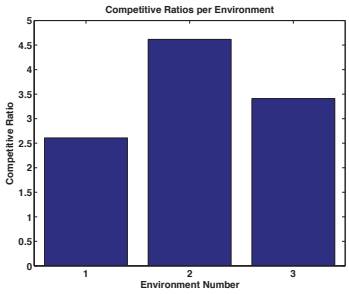
In our final set of experiments, we measured the performance of the MRTA-RTPP algorithm in terms of path cost with respect to an offline algorithm that is provided a map of the environment along with the location of obstacles on the map. The Hungarian algorithm (Kuhn 1955) was used as the offline algorithm for determining the task schedule. The performance is measured by the competitive ratio (CR) of MRTA-RTPP that is given by:

$$CR = \frac{\text{Path cost incurred by MRTA-RTPP}}{\text{Path cost incurred by offine algorithm}}$$

Figure 5(b) shows that CR is the lowest (MRTA-RTPP performs closest to optimal cost) in environment 1 where the plan is revised only once after visiting w_2 and the extra path traveled by the robot (to unsuccessfully visit w_3 after w_2) is the least. In contrast, the CR is higher in environments 2 and 3. We observe that the CR and the number of replans in the three environments are correlated with each other because more number of replans result in more unsuccessful paths which increase the CR. However, the CR for environment 3



(a)



(b)

Figure 5: (a) Number of path replans done after encountering obstacles by the MRTA-RTPP algorithm for the three environments. (b) The competitive ratio of the path cost calculated by the MRTA-RTPP algorithm and the optimal path cost calculated offline with prior information of obstacles, for the three environments.

is considerably lower than that of environment 2, although the number of replans done in both the environments are comparable. The reason for this is that, in environment 2, due to the location of w_2 behind the obstacle, the robot ends up tracing the boundary of the obstacle twice, once while going from w_1 to w_3 and again while going from w_5 to w_2, resulting in higher path cost and consequently, higher CR than environment 3. The algorithm also operated correctly with multiple robots and the same set of waypoints. Overall, our experiments show that the MRTA-RTPP operates correctly and appropriately revises the task schedule while replanning the robot's path in different environments.

Conclusions and Future Work

In this paper, we have described an algorithm that integrates an MRTA algorithm with a dynamic path planning algorithm. Our experimental results show that the proposed algorithm works successfully in different environments. In the future, we plan to consider more complex task schedules with multiple visits to waypoints by different robots. Improvements to the replanning technique to reduce the time required to recompute a path and to determine the optimal schedule by the MRTA are also being investigated. Finally, we are planning to implement the proposed algorithm on teams of heterogeneous robots.

Acknowledgements

This research has been supported by the U.S. Office of Naval Research grant no. N000140911174 as part of the COMRADES project.

References

Botelho, S., and Alami, R. 1999. M+: A scheme for multi-robot coopeation through negotiated task allocation and achievement. In *Proceedings of the International Conference on Robotics and Automation*, 1234–1239.

Bresenham, J. 1965. Algorithm for computer control of a digital plotter. *IBM Systems Journal* 4(1):25–30.

Bullo, F.; Frazzoli, E.; Pavone, M.; Savla, K.; and Smith, S. 2011. Dynamic vehicle routing for robotic systems. In *Proceedings of the IEEE*, volume 99. 1482–1504.

Dias, M. B.; Zlot, R.; Kalra, N.; and Stentz, A. 2006. Market-based multirobot coordination: a survey and analysis. In *Proceedings of the IEEE, Special Issue on Multirobot Systems*, volume 94, 1257–1270.

Dias, M. 2004. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamics Environments*. Ph.D. Dissertation, The Robotics Institute, Carnegie Mellon University.

Ferguson, D., and Stentz, A. 2006. Using interpolation to improve path planning: The field d* algorithm. *J. Field Robotics* 23(2):79–101.

Gerkey, B., and Matarić, M. 2004. A formal analysis and taxonomy of task-allocation in multi-robot systems. *The International Journal of Robotics Research* 23(9):939–954.

Jones, E.; Browning, B.; Dias, M.; Argall, B.; Veloso, M.; and Stentz, A. 2006. Dynamically formed heterogeneous robot teams performing tightly coordinated tasks. *Proceedings of the International Conference on Robotics and Automation*.

Jones, E.; Dias, M.; and Stentz, A. 2011. Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous robots* 30(1):41–56.

Kuhn, H. 1955. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly* 2:83–97.

Lagoudakis, M.; Markakis, V.; Kempe, D.; Keskinocak, P.; Koenig, S.; Kleywegt, A.; and Tovey, C. 2005. Auction-based multi-robot routing. In *Proc. of the International Conference on Robotics: Science and Systems*, 343–350.

Liu, L., and Shell, D. 2011. Assessing optimal assignment under uncertainty: An interval-based algorithm. *The International Journal of Robotics Research* 30(7):936–953.

Mataric, M.; Sukhatme, G.; and Ostergaard, E. 2003. Multi-robot task allocationin uncertain environments. *Autonomous Robots* 14:255–263.

Schneider, J.; Apefelbaum, D.; Bagnell, D.; and Simmons, R. 2005. Learning opportunity costs in multi-robot market based planners. In *Proceedings of the International Conference on Robotics and Automation*, 1151 – 1156.

Viguria, A.; Maza, I.; and Ollero, A. 2007. Set: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. In *Proceedings of the International Conference on Robotics and Automation*, 3339–334.

Zlot, R. 2006. *Complex Task Allocation for Multi-robot Teams*. Ph.D. Dissertation, Carnegie Mellon University.