CONSTRAINT AWARE BEHAVIOR IN MULTI-ROBOT SYSTEMS

An Undergraduate Research Scholars Thesis

by

SAURABH MISHRA

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Research Advisor: Dr. Nancy Amato

May 2015

Major: Computer Science and Engineering

TABLE OF CONTENTS

ABSTRACT

Constraint Aware Behavior in Multi-Robot Systems . (May 2015)

Saurabh Mishra
Department of Computer Science and Engineering
Department of no
Texas A&M University

Research Advisor: Dr. Dr. Nancy Amato
Department of Computer Science and Engineering

In this work we present a behavioral modeling framework that accounts for a battery constraint. This framework allows for a user to model robot teams of varying configuration performing common robotic tasks such as exploration or going to user specified goals. The focus of this work is on how to model a constraint aware behavior and how assistance can be requested by and provided from a robot team. We show experimental results in simulated environments and identify trends that can be seen given a robot team configuration. We also discuss how this system can be adapted to different environments and different constraints. Our system can be setup to allow for different number of workers and helpers. The charging station, battery level and the behaviors of these agents can also be varied. We discuss the affect of these different policies on the performance of the workers. The performance is measured by the number of times the environment area is covered. In conclusion we would measure the performance based on the number of times the environment is covered by the agents.

ACKNOWLEDGMENTS

I would like to acknowledge Dr. Sam Rodriguez,Dr. Marco Morales and Dr. Nancy Amato for their support throughout the development of my research work. You have been of great help and this work wouldn't have been possible without your expertise.

CHAPTER I
INTRODUCTION

Multi-Robot coordination is required to address certain tasks. In particular, we are interested in studying tasks that may require robots to perform for a long periods of time or to be persistent. In these cases, the hardware may not allow the robots to continue performing a task until completion. For example, the robots could run out of battery or might encounter some other hardware failure. We need methods that properly handle needs for replacement, repairing, or recharging of robots in order for the task to be accomplished.

This kind of behavior has diverse applications. For example, in surveillance system robots may take turns to monitor a certain area and they need to avoid depleting their batteries. Another example is the case of a fire disaster where firefighter robots need to carry water with them to extinguish the fires and they need to keep a proper supply at all times. In these cases, when they are about to run out of supplies (battery or water), they can use the help of other robots for recharging or even get replaced by other agents while they replenish their resources.

Although this problem has been addressed in several dimensions we lack an integrated method that allows to test for different scenarios and policies. Behaviors handled are usually fixed. The method to monitor battery status are more diverse, ranging from using a fixed or adaptive threshold[23] as a trigger to predicting failures to ask for help in advance. When robots are in need to recharge, they either go to a static charging station[7] or a mobile charging station comes to meet them[24]. Approaches to charging vary from static to mobile charging locations.

Our contribution is a method that allows a robot to exchange roles for charging the robots while performing a task in a given environment. The method of charging is very flexible and applicable to a wide variety of tasks where we can tune the policies. We designed our method to be applied in different environments where the robots can perform a wide variety of tasks.

Our approach works as follows: We split the robots in two classes, workers and helpers. Workers are the robots performing the task. Helpers are additional robots with enough resources to help the worker. Workers monitor their battery levels to ensure that they can continue working and call for help when their battery is low. When a worker calls for help the nearest helper is chosen to respond. Helpers can respond in two different ways. One way is to swap its position with the worker so that the worker can get recharged at a charging station. The other way is to make a helper recharge the worker. Workers that need help can act in two different ways as well. One way is to start swapping right away. The other way is to wait for a helper to arrive before swapping. This is useful when the task requires that a robot is always present in the working environment.

CHAPTER II

RELATED WORK

The problem of handling the available energy supply in robots has been addressed from several different perspectives. There are different concerns that need to be addressed such as the method of recharging the robots, should the charging station be static or moving? The overhead of charging the robots also needs to be discussed, the robots need to maximize the amount of working time and minimize the amount of recharging time. We will discuss some of the work that has been done in this section.

There has been research on where and how to perform charging. In [18] we see a method that recharges the robots using a static docking station while [24] introduces the use of a tanker to distribute energy to other robots as more efficient than having the robots move to a charging location. In [13] a method is provided that maintains the persistency of a task by allowing the robots to carry and exchange fuel (battery) with other robots. Mobile charging units are also discussed in [3]. Recharging can be performed with automated docking and battery swap systems [22, 21].

There has also been research on finding good paths for charging. The work in [6] introduces a combined coverage and energy dependent control law that drives an agent to a fixed docking station as its energy level becomes low. This method however could lead to the environment being unmonitored when all robots are low on energy. The work in [11] describes a method to design paths for charging robots to find good rendezvous locations to meet working robots and recharge them. This work assumes that the charging robots are only used for the purpose of charging other working robots and they themselves do not need to be charged throughout the duration of the task. Similar work [4, 10] propose to find a set of meeting points for worker robots and a single mobile charging dock. The dock is assumed to have enough energy to charge the robots for the entire task.

Optimizing the charging process has also been studied. In [7] we see an optimization method to place mobile docking stations to maximize the power available to working robots. The work in [23] presents a method that tries to optimize the process of charging and executing the behavior of the robots based on the foraging behavior of animals. In addition, a coverage method has been proposed, [17], that is aware of the battery levels of the robot so that the robot returns to its initial (charging) position every time it is needed in order to fulfill its coverage task. Also related, [20] provides an algorithm for generating an optimal path for a robot considering different constraints including remaining fuel through a multi-dimensional cost function. The work [9] developed a mobile recharging station capable of recharging multiple small robots.

The decision on when to recharge the robot is another issue to consider. The most common approach is to use a fixed threshold as a reference as in [1]. That method recharges the robot after some constant time (approximating a fixed battery level threshold). An alternative approach, [16], robots recharge as soon as they sense an available charging station even if their batteries are not low. The threshold method is also discussed in [23] along with an Adaptive Threshold method. Adaptive threshold evaluates if a robot can get to a goal and subsequently to a charging location with the given amount of battery.

In [8] a market based solution is applied to the Autonomous Recharging Problem. Each agent tries to minimize its cost leading to the overall good.

The placement of the charging dock is an important factor in deciding how much time is wasted in getting from a working location to a charging location. In [5] a method is shown that determines the optimal placement of a charging location based on the feedback from the working agents. The dock itself is a moving agent that optimizes its position over the course of the task.

The work in [19] shows a physical system designed to allow the robots to automatically dock themselves to a charging location. They discuss the electro-mechanical design of the system. They have a task module manager that keeps track of the battery status of the robot and forces them to recharge when the battery is low.

The issue of interference around a charging station is discussed in [12]. The robot enters into a random wander mode for a short period whenever it is unable to dock to the charging station before attempting to charge again. This behavior helps in reducing robot density commonly found at a charging station.

The approach presented here incorporates the abilities of several of the methods described above in an integrated framework. While many of the previous methods work for fixed behaviors, we can incorporate multiple behaviors such as frontier and coverage described in the later sections. We allow for different charging methods like going to a charging station, being charged by a helper, or swapping roles with the helper. We decide on when to call for help using a fixed threshold or by estimating the energy needed to keep performing a task and going to the battery station.

CHAPTER III

MULTI-AGENT SYSTEM

In this section we will briefly describe the main aspects of our roadmap-based multi-agent system which impact the overall motion of agents undergoing a scenario. This includes a description of the agents, their motion model, and the environmental model. For more information on the roadmap-based multi-agent system and applications we have studied previously, see the work in [14] [15] [2].

Mobile Agent Model and Behavior

In this work we consider scenarios consisting of a set of N agents, $A = a_1, a_2, ..., a_N$. An agent a_i is represented by positional, velocity, and acceleration values: $a_i = \{\mathbf{p}, \mathbf{v}, \mathbf{a}\}$. These values dictate the agent's motion state in the environment. Agents are equipped with a behavior rule responsible for creating a plan for the agent given it's goals and knowledge of the environment. The behaviors we describe in this work consider a battery constraint on the agent and determine when the agent's battery level require them to stop performing the actual work behavior and call for help. In the scenarios we consider here, the work behavior results in a route through the environment.

Environmental Representation

Our environmental model allows us to study both basic and potentially complex environments. The environment is composed of surfaces that represent the valid space. The agents use each surface for generating valid roadmaps and determining if their current state is in the valid space.

We utilize our roadmap-based approach to represent valid motion through potentially complex environments. The roadmap consists of a set of nodes sampled in the free/valid space of the environment. The valid space consists of the space with no obstacles. The nodes are connected using simple local planning techniques with a valid edge added to the roadmap between two nodes if the intermediate nodes lie in the valid space. A valid edge is an edge between two nodes that lie in the valid space. An agent that needs a valid path through the environment can then query the roadmap for a start and goal configuration by connecting the configurations to the nearest node in the roadmap in the same connected component. Basic graph search techniques are used to return a valid path. Local planning techniques are used to connect different valid nodes to form a graph. Local planning checks if there is a valid path between two given nodes. If there is a path then the edge is added to the graph.

Motion Model

In this work we equip our agents with force rules to allow them to move through the environment. The forces generated at each time step are used to update the acceleration, velocity, and position components. For the scenarios presented here, the agents are equipped with only a goal-based force rule, ignoring collision with other agents. The goal-based force rule guides the agent along the planned route until a final goal is reached. See the work in [14] [15] to get more details about these forces.

Battery-Constrain Aware Behavior

In our approach the team is split into workers and helpers. The split can be defined by the user or generated at random. Algorithm 1 is called iteratively for each agent a_i. Worker agents carry out specific tasks while spending their battery and eventually reaching a threshold value when they send a request for help. Meanwhile, helper agents charge their batteries at a charging location and, when ready, respond to requests for help.

Algorithm 1 ConstraintAwareBehavior

```
Input: agent $a_i$
1: if $a_i \rightarrow$ IsWorker() then
2:     $a_i \rightarrow$ ExecuteWorkerBehavior()
3: else if $a_i \rightarrow$ IsHelper() then
4:     $a_i \rightarrow$ ExecuteHelperBehavior()
5: end if
6: if NumOfHelpers $>$NumOfInitialHelpers then
7:     SetExtraHelpersAsWorkers()
8: end if
```

Execute Worker Behavior

Algorithm 2 describes the general worker behavior that executes its assigned task while keeping track of its battery levels. The task could be anything such as executing a simple patrolling behavior where each agent patrols a certain area of the environment. The battery level of the agent decreases at each time step. The battery is dependent on the agent's movements, a moving agent would spend its battery faster than a stationary agent, see Section Battery Level for a detailed explanation about the modeled battery. Once a worker gets to low battery, it calls for help. The low battery status can be determined in two ways. The first method uses a pre-defined threshold value for the low battery. Once an agent reaches this threshold, it immediately calls for help. The second method is smarter way of deciding whether the battery is low or not. Using this method a worker calculates whether it can get to its next goal and then to a charging station from that goal with its current battery level. If the battery level is too low to execute that action then the worker calls for help. It is possible that a worker may not receive help from a helper due to the shortage of helpers. This can happen when the number of workers is much larger than the number of helpers. To handle this case, the agents have a second threshold value called the $CriticalThreshold$. Once the battery reaches below this threshold, the worker moves to the nearest charging location to recharge itself and becomes a helper. If the number of helpers in the environment becomes more than the initial number of helpers, the extra helpers are sent back to the environment as workers.

Calling for Help

As shown in Algorithm 3, first the worker finds the closest helper that is available (with a recharged battery). Then, it triggers the appropriate behaviors in workers and helpers depending on the type of help needed by the worker which can be a BasicSwap, WaitAndSwap or a Recharge.

In the BasicSwap help method, the worker becomes a helper and heads for the charging station while the a helper becomes a worker and goes to replace the worker in its task.

In the case of WaitAndSwap the worker waits for a helper agent to arrive at its location before moving to a charging location. This ensures that the environment is never left empty. Since the

Algorithm 2 ExecuteWorkerBehavior

```
Input: agent $worker_i$
1: if $worker_i \to$ HasNewGoal() then
2:     $worker_i \to$ FindPathToNewGoal()
3: end if
4: if $worker_i \to$ HasNoBehavioralRoute() then
5:     ComputeBehavioralRoute()
6: end if
7: if $worker_i \to$ BatteryMethod() == “threshold” then
8:     if $worker_i \to$ BatteryLowerThanThreshold() then
9:         $worker_i \to$ CallForHelp()
10:    end if
11: else if $worker_i \to$ BatteryMethod() == “AdaptiveThreshold” then
12:    if !$worker_i \to$ CanReachGoalAndCharingLocation($worker_i \to$NextGoal(),
       NearestCharingLocation($worker_i \to$NextGoal())) then
13:        $worker_i \to$ CallForHelp()
14:    end if
15: end if
16: if $worker_i \to$ BatteryLowerThanCriticalThreshold() then
17:    $worker \to$New_Goal = NearestChargingLocation($worker \to$GetCurrentPosition())
18:    $worker_i \to$ SetAsHelper()
19: end if
```

Algorithm 3 CallForHelp

```
Input: agent $worker$, $help\_tag$
1: $helper = worker \to$ FindClosestAvailableHelper()
2: if $help\_tag \to$ BasicSwap then
3:     $worker \to$New_Goal = NearestChargingLocation($worker \to$GetCurrentPosition())
4:     SwitchHelperWorker($worker$,$helper$)
5: else if $help\_tag \to$ WaitAndSwap then
6:     $helper \to$New_Goal = $worker \to$ GetCurrentPosition()
7:     $helper \to$IsHelpingWorkerToWaitAndSwap = true
8:     $helper \to$HelpingAgent = $worker$
9: else if $help\_tag \to$ Recharge then
10:    $helper \to$New_Goal = $worker \to$ GetCurrentPosition()
11:    $helper \to$IsHelpingWorkerToRecharge = true
12:    $helper \to$HelpingAgent = $worker$
13: end if
```

worker waits for a helper, it does not carry out any other task and hence its battery reduces at a slower rate. Once the helper reaches the worker, the worker can move to the nearest charging location. Once it has recharged its battery, the worker becomes helper. This could be helpful in a patrolling scenario where leaving the environment is not an option.

In the case of Recharge, when a worker needs to be charged, we model the case where the helper carries a charged battery and replaces the battery in the worker. This could be useful when the robots are heterogenous and there is a need for a particular robot to be present in the environment at all times. The robot can be charged by another robot and it will continue its task.

Execute Helper Behavior

Algorithm 4 describes the helper behavior. The helper initially waits at a charging location until someone calls for help. Once the helper is called for help, its new goal would be set appropriately by the call for help depending upon the nature of the help call. The helper can help the worker in three ways, BasicSwap, WaitAndSwap and Recharge, see section Call For Help. In the case for WaitAndSwap and Recharge, Algorithm 3 assigns the helper with a worker as its “HelpingAgent”. The ExecuteHelperBehavior then checks to see if the helper gets close to the worker. Once the helper is close enough to the worker, WaitAndSwap or Recharge can be executed. This algorithm also ensures that if a helper is at a charging station then it recharges its battery and becomes available to help once the battery is fully charged. UpdateReachedGoal updates the goal of the agent at each time step. It is used to ensure that the agent has reached the goal.

Battery Level

The battery is defined as a function of the speed of the agent. The battery reduces by a certain fixed amount plus the speed of the agent. This shows somewhat realistic model of a real battery. When the agent does not move, the speed will be zero. This will make the battery reduce at a slower rate which is essentially true since the robot will not be doing any work. This feature was added to support the wait and swap switch behavior.

Algorithm 4 ExecuteHelperBehavior

```
Input: agent $helper_i$
if $helper_i$ $\rightarrow$ HasNewGoal() then
    $helper_i$ $\rightarrow$ FindPathToNewGoal()
end if
if $helper_i$ $\rightarrow$ IsHelpingWorkerInWaitAndSwap() then
    if HelperReachedCloseToWorker(HelpingAgent) then
        $worker$ = HelpingAgent
        $worker$ $\rightarrow$New_Goal = NearestChargingLocation($worker$ $\rightarrow$GetCurrentPosition())
        SwitchHelperWorker($worker$,$helper$)
    end if
else if $helper_i$ $\rightarrow$ IsHelpingWorkerToRecharge() then
    if HelperReachedCloseToWorker(HelpingAgent) then
        $worker$ = HelpingAgent
        $helper$ $\rightarrow$RechargeWorker($worker$)
        $helper$ $\rightarrow$ New_Goal = NearestChargingLocation($helper$ $\rightarrow$GetCurrentPosition())
    end if
end if
$helper_i$ $\rightarrow$UpdateReachedGoal()
if $helper_i$ $\rightarrow$ IsAtChargingStation() then
    $helper_i$ $\rightarrow$ RechargeBattery()
else if $helper_i$ $\rightarrow$ !IsAtChargingStation() then
    $helper_i$ $\rightarrow$ New_Goal = NearestChargingLocation($helper$ $\rightarrow$GetCurrentPosition())
end if
```

Algorithm 5 BatteryStatus

```
Input: agent $a_i$
BatteryLevel -= DefaultDepletionRate + MovingDepletionRate($a_i$.GetSpeed())
return BatteryLevel
```

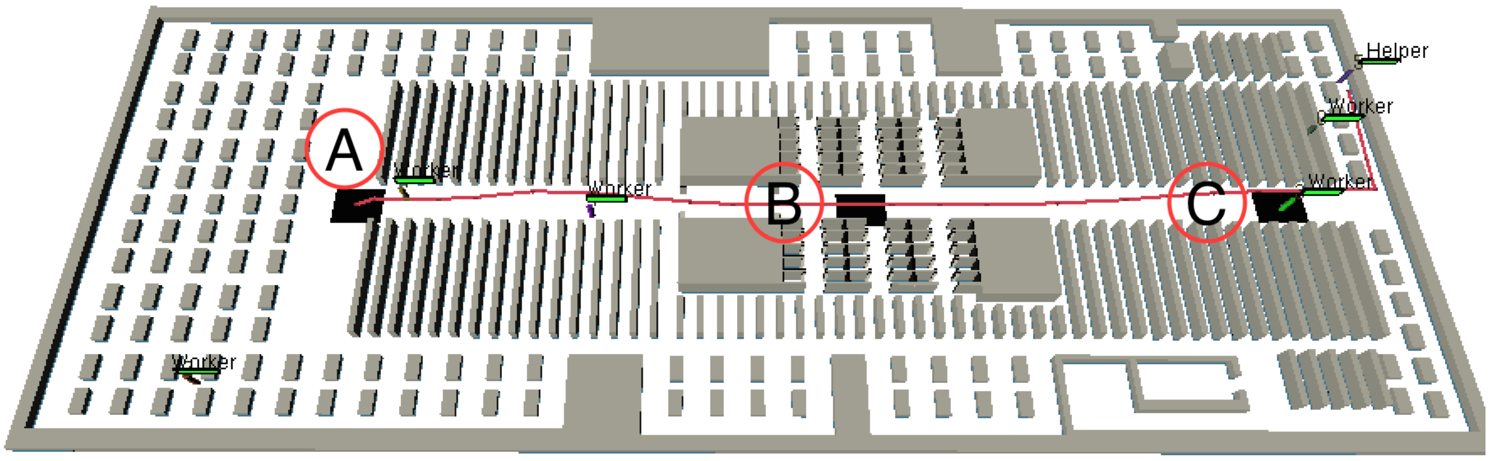
Algorithm 6 Adaptive Threshold

```
Input: agent $worker$
DistToGoal = $worker$ $\rightarrow$FindDistanceToGoal($a_i$);
DistToChargingFromGoal = $worker$ $\rightarrow$FindDistanceFromGoalToCharging$a_i$);
if DistToGoal+DistToChargingFromGoal >DistAllowedByBattery() then
    return true
else
    return false
end if
```

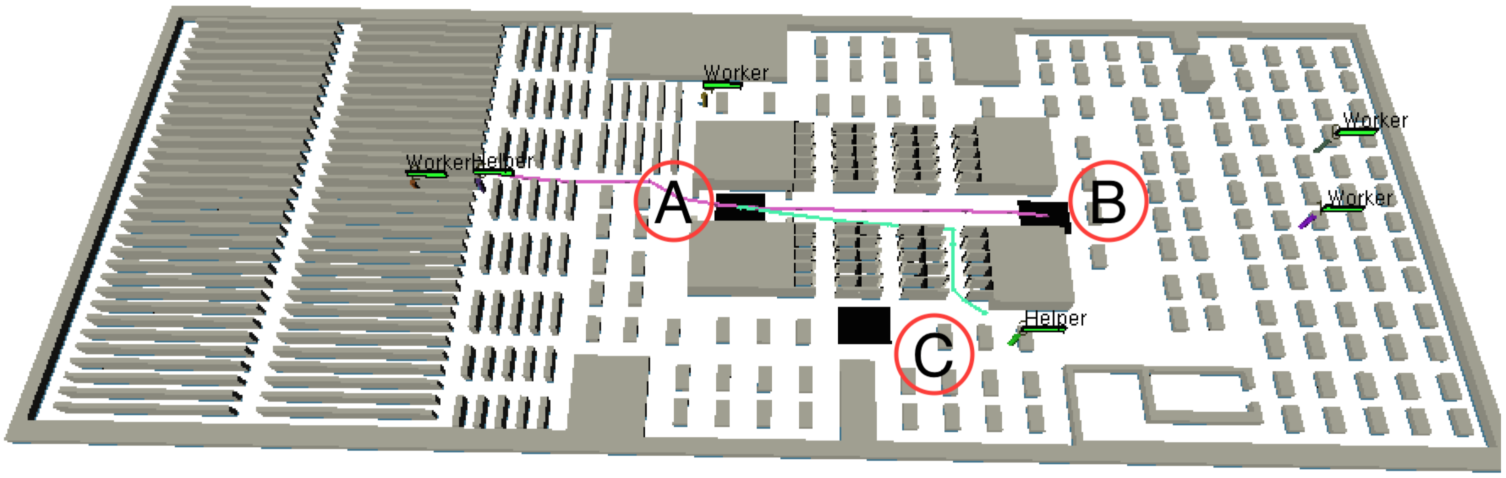
Algorithm 6 describes a method where the robots are able to smartly calculate whether they can reach a goal and then to a charging station. The helpers first calculate the distance to their goal and then the distance to a nearest charging location from that goal. If the distance is greater than the distance that their battery would allow them to cover, then the robots call for help immediately without continuing to the goal.

CHAPTER IV

EXPERIMENTS



(a) Evans 1



(b) Evans 2

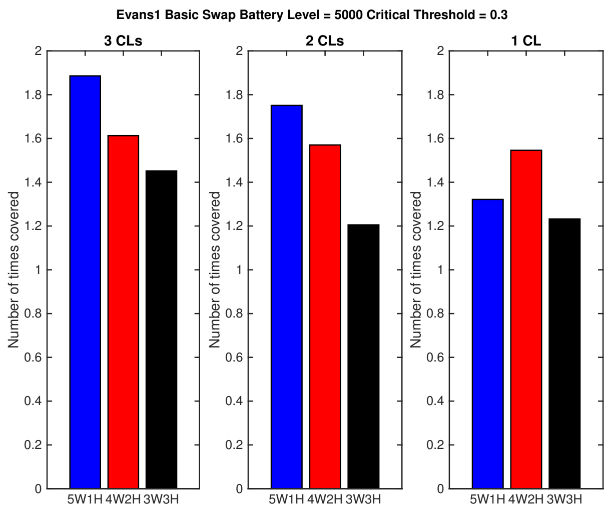
Fig. IV.1. Evans library simulated model (3rd floor) (a) current configuration and (b) potential configuration with same numbers of shelves and desks.

In this section we will present some example scenarios consisting of different robot teams, battery configurations, behavior constraints, and environmental configuration to show how persistent a robot team can perform a task. This type of analysis can be useful when deciding what kind of robot team is needed to perform a task given behavioral requirements, agent constraints, and number of agents needed to be able to continually perform the desired behavior. We set the battery level to 5000 units. The depletion rate for the battery is an amount that will be subtracted from the battery at each time step that includes a base amount plus an additional amount depending on if the agent is moving. The base amount we used in this experiment was 2.2 and the additional movement amount was 0.5. The CriticalThreshold was varied from 0.05 to 0.3. Charging agents would increase their battery level while at a charging station by five times the base amount or 11 per time step.

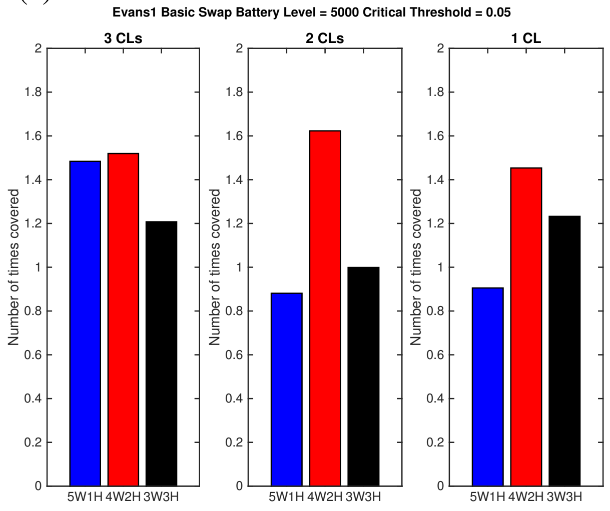
Two different environments configurations are shown in Figure IV.1. These are modeled after the 3rd floor of the Evans Library at Texas A&M University, College Station. The charging location for each configuration is shown using a black box. In the experiments we tested different set of workers and helpers with different number of charging location in the Evans1 and Evans2 environment. The number of workers was varied from 3 to 5 and the number of helpers was varied from 1 to 3. The number of charging locations in each case was also varied from 3 to 1. The location of the 3 charging locations is marked in the Figure IV.1(a) and Figure IV.1(b) as A, B, and C. In the case with 2 charging locations, location B was removed from Evans1 and location C was removed from Evans2. In the case with one charging location, locations A and C were removed from Evans1 whereas locations B and C were remove from Evans2.

In this set of experiments we tested different set of workers and helpers with different number of charging location in the Evans1 and Evans2 environment. The behavior tested here was the Frontier behavior. Frontier behavior is used to systematically cover the environment and it allows us to specify a metric that measures the the number of times the given roadmap was covered by the agents. Using this metric we can analyze the performance of the workers. We added a new threshold to accommodate for agents dying in the case where there were not enough number of helpers. When a worker's battery level reaches this new threshold, the worker moves to a charging location to get recharge its battery. Once its battery is recharged, it comes back and resumes its task. This threshold is varied by the $CriticalThreshold$ value as shown in the graphs. Figure IV.2(a) and IV.2(b) show results from the Evans1 environment with different critical thresholds. You can see that with a higher critical threshold, less agents die due to the battery running out and hence more area is covered with more workers. Since the number of agents dying is decreased, more workers are available for the entirety of the task hence, more workers directly relates to more area being covered in the environment. With a lower critical threshold the workers barely have any chance of going to a charging location themselves and coming back. This would mean that with more workers and less helpers we would expect more agents to die and hence less area to be covered as shown in the results. From the graphs we can see that the area covered in case with 3 workers and 3 helpers remained almost the same as our previous results but the area covered in the other two cases reduced significantly. We are able to achieve these results in both the Evans environments.

Another environment configuration is shown in Figure IV. This is a simulated model of an office building. The charging locations are marked with black boxes. The results from this environment are similar to the results from the Evans building environment. However, in this environment more area is covered by the agents. The office environment also shows a clearer trend in more workers performing better than fewer as compared to the Evans environment. The Evans environment was much more complex and large compared to the Office environment. Office environment allows agents to get to charging locations even with very low battery left. This leads to less agents dying early and hence more area is covered. This shows that our method works in different types of environment with similar results. Covering robots deciding to go for charging earlier rather than stay to work for longer time allows the workers to perform better since there are less agents that die. Figure IV.4 provides the results from this environment.

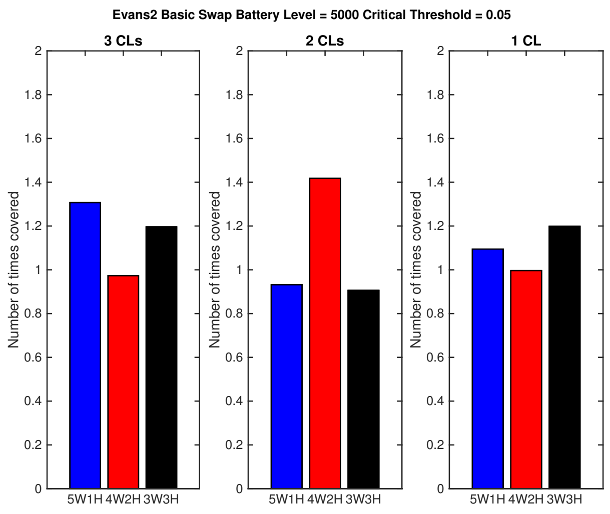


(a) Evans1 results with critical threshold = 0.3

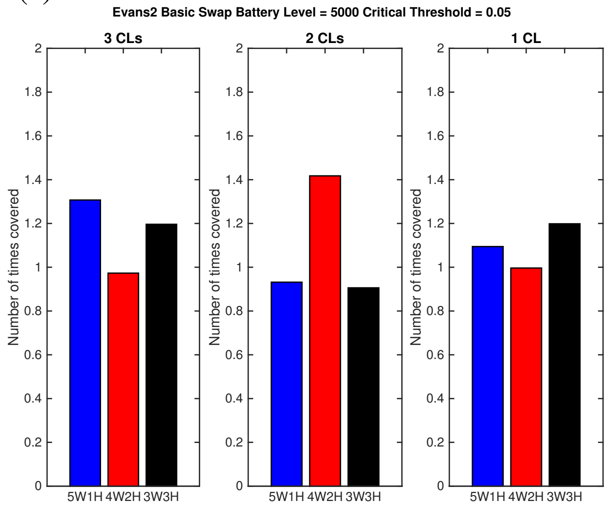


(b) Evans1 results with critical threshold = 0.05

Fig. IV.2. Evans Library Building Results

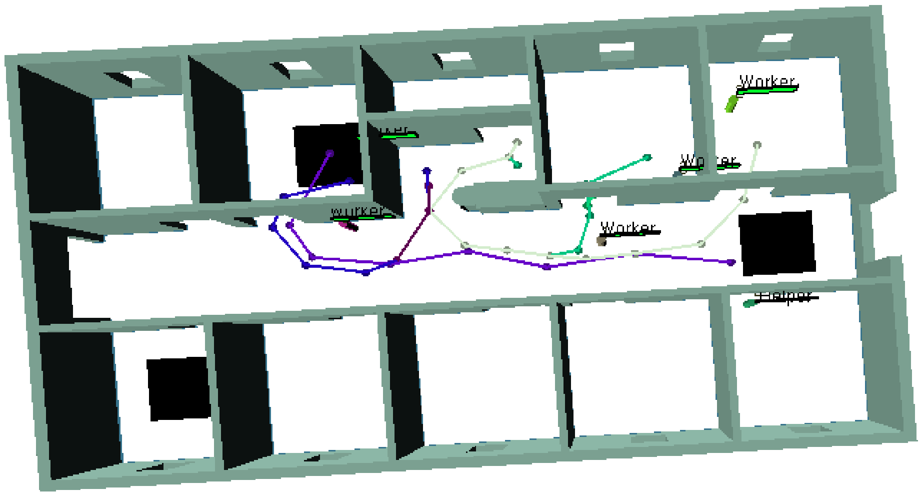


(a) Evans2 results with critical threshold = 0.3

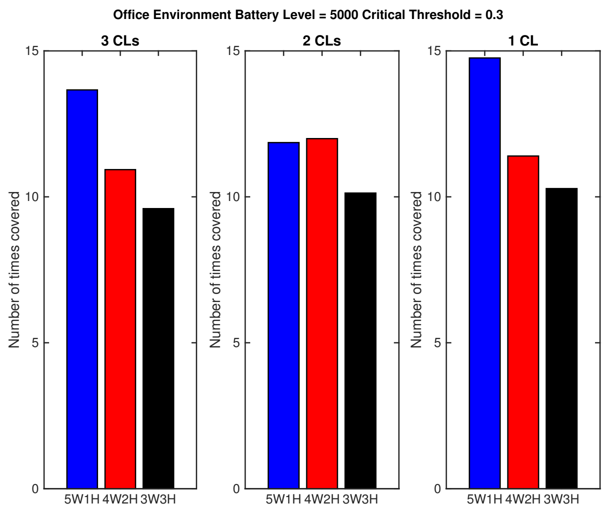


(b) Evans2 results with critical threshold = 0.05

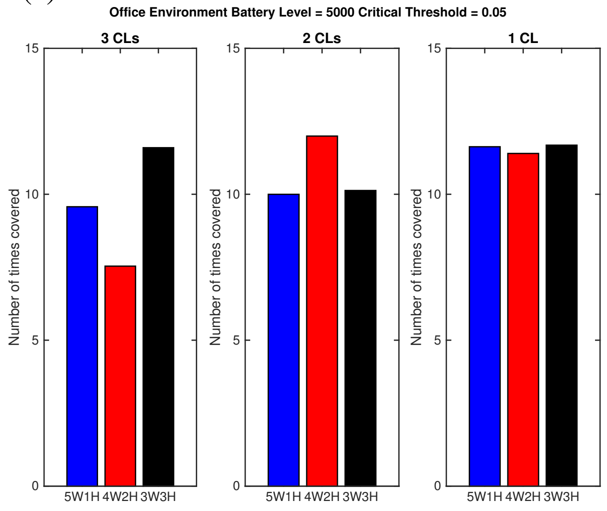
Fig. IV.3. Evans Library Building Results



(a) Office Environment



(a) Office results with critical threshold = 0.3



(b) Office results with critical threshold = 0.05

Fig. IV.4. Office Environment Results

CHAPTER V
CONCLUSION AND FUTURE WORK

We have presented a behavioral modeling framework that accounts for a battery constraint. This framework allows for a user to model robot teams of varying configurations performing common robotic tasks such as exploration or going to user specified goals. Our modeling framework allows us to identify some expected trends when a robot team is configured in certain ways. It also provides insight into the importance of the location of charging locations and how they can impact the success of a robot mission. Finding the best set of charging locations is an interesting avenue of future work, especially in more complex environments with constraints on the available placement of these locations. This type of framework can be extended to a variety of constraints and our pluggable behavioral system allows a user to easily change the work that a robot team is expected to perform.

In the future we would also like to look into techniques using which the agents would be able to decide the best arrangement of the charging locations in the environment and the optimal ratio of workers to helpers needed to perform the task. Another focus would be to model other constraints on the agents. One such constraint is water, we can model the behaviors with both water and battery constraints. The agents would have to manage both the constraints and work to make sure the working time is maximized.

REFERENCES

[1] D. Austin, L. Fletcher, and A. Zelinsky. Mobile robotics in the long term-exploring the fourth dimension. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 613–618 vol.2, 2001.

[2] O. B. Bayazıt, J.-M. Lien, and N. M. Amato. Swarming behavior using probabilistic roadmap techniques. In *Swarm Robotics*, pages 112–125. Springer, 2005.

[3] A. Birk. Autonomous recharging of mobile robots. In *In: Proceedings of the 30th International Sysposium on Automative Technology and Automation. Isata Press*. Citeseer, 1997.

[4] A. Couture-Beil and R. Vaughan. Adaptive mobile charging stations for multi-robot systems. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1363–1368, Oct 2009.

[5] A. Couture-Beil and R. Vaughan. Adaptive mobile charging stations for multi-robot systems. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1363–1368, Oct 2009.

[6] J. Derenick, N. Michael, and V. Kumar. Energy-aware coverage control with docking for robot teams. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3667–3672, Sept 2011.

[7] A. Drenner and N. Papanikolopoulos. Docking station relocation for maximizing longevity of distributed robotic teams. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2436–2441, May 2006.

[8] B. Kannan, V. Marmol, J. Bourne, and M. Dias. The autonomous recharging problem: Formulation and a market-based solution. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3503–3510, May 2013.

[9] A. Kottas, A. Drenner, and N. Papanikolopoulos. Intelligent power management: Promoting power-consciousness in teams of mobile robots. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1140–1145. IEEE, 2009.

[10] Y. Litus, P. Zebrowski, and R. Vaughan. A distributed heuristic for energy-efficient, multi-robot, multi-place redezvous. 25(1):130–135, 2009.

[11] N. Mathew, S. Smith, and S. Waslander. A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3497–3502, May 2013.

[12] A. Munoz, F. Sempé, A. Drogoul, and R. G. Leclerc. Sharing a charging station in collective robotics. *LIP6 research reports*, 2002.

[13] T. D. Ngo and H. Schioler. Sociable mobile robots through self-maintained energy. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 3, pages 2012–2017, Oct 2006.

[14] S. Rodriguez, J. Denny, A. Mahadevan, J. Vu, J. Burgos, T. Zourntos, and N. M. Amato. Roadmap-based pursuit-evasion in 3d structures. In *24th Intern. Conf. on Computer Animation and Social Agents (CASA)*, 2011.

[15] S. Rodriguez, J. Denny, T. Zourntos, and N. M. Amato. Toward simulating realistic pursuit-evasion using a roadmap-based approach. In *Motion in Games*, pages 82–93. Springer, 2010.

[16] F. Sempé, A. Muñoz, and A. Drogoul. autonomous robots sharing a charging station with no communication: a case study. In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, volume 5, pages 91–100, 2002.

[17] I. Shnaps and E. Rimon. On-line coverage of planar environments by a battery powered autonomous mobile robot. In *Workshop on the Algorithmic foundations of Robotics (WAFR)*, 2014.

[18] M. Silverman, D. Nies, B. Jung, and G. Sukhatme. Staying alive: a docking station for autonomous robot recharging. In *IEEE International Conference on Robotics and Automation*, pages 1050–1055, May 2002.

[19] M. C. Silverman, B. Jung, D. Nies, and G. S. Sukhatme. Staying alive longer: autonomous robot recharging put to the test. *Center for Robotics and Embedded Systems (CRES) Technical Report CRES*, 3:015, 2003.

[20] R. Simpson, J. Revell, A. Johansson, and A. Richards. Multi-cost robotic motion planning under uncertainty. In *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ*, 2014.

[21] K. Suzuki, P. Kemper Filho, and J. Morrison. Automatic battery replacement system for uavs: analysis and design. 65:563–586, 2012.

[22] K. Swieringa, C. Hanson, J. Richardson, J. White, Z. Hasan, E. Qian, and A. Girard. Autonomous battery swapping system for small-scale helicopters. In *IEEE International Conference on Robotics and Automation*, pages 3335–3340, May 2010.

[23] J. Wawerla and R. Vaughan. Near-optimal mobile robot recharging with the rate-maximizing forager. In F. Almeida e Costa, L. Rocha, E. Costa, I. Harvey, and A. Coutinho, editors, *Advances in Artificial Life*, volume 4648 of *Lecture Notes in Computer Science*, pages 776–785. Springer Berlin Heidelberg, 2007.

[24] P. Zebrowski and R. Vaughan. Recharging robot teams: A tanker approach. In *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, pages 803–810, July 2005.