A Comparison of Algorithms for Teams of Robots[*]

David A. Gustafson
Computing and Information Science Department
Kansas State University
Manhattan, KS, U.S.A
dag@cis.ksu.edu

Venkata Prashant Rapaka
and Scott DeLoach
Computing and Information Science
Department
Kansas State University
Manhattan, KS, U.S.A.
sdeloach@cis.ksu.edu

Abstract - *The effective use of teams of robots to accomplish large tasks is an important goal of robotic research.. An important part of achieving this is to understand the trade-offs between various approaches to team organizations and communications. The goal of our research was to understand and compare, in a real-world environment, the strengths and weaknesses of a variety of common approaches to multiple-robot algorithms.*

We have built a robot simulation environment. The behavior of our simulated environment will be validated by comparison with the behavior of our real robots. We have used the standard search-and-tag problem for our initial investigation in which a number of targets that must be located in an unstructured environment. A group of robots will explore the environment and will report the location of the targets. The efficiency of the approaches will be compared in terms of time required to locate the first target and the average time to locate the targets that were found..

Our initial experiments revealed interference among the robots, the targets and the obstacles. The effect of this interference was larger than expected. Our future research will involve attempting to model this behavior. An initial model is discussed.

Keywords: Robotics, Cooperative Algorithms, Swarm Algorithms.

1 Introduction

There are many tasks, such as search and rescue and surveillance, that would benefit from having multiple robots. For these tasks and others, the effective use of multiple robots is an important goal of robotics research. However, the study of the behavior of groups of autonomous robots accomplishing an important task is difficult. Some multiple robot algorithms will work well in small groups but will not scale to larger groups. Some algorithms will suffer with interference from other robots. Other algorithms may be affected by obstacles. Much research has assumed perfect knowledge such as accurate location information and perfect recognition of targets. An important part of achieving this goal of the effective use of multiple robots is to understand the effect of errors and the trade-offs between various approaches to multiple-robot algorithms in a realistic environment. The goal of our research is to understand these effects better in order to improve the design of multiple-robot algorithms and to predict the effect of errors on these algorithms.

One common approach to multiple-robot algorithms is a swarm approach that involves little or no cooperation among the robots [1], [4], [3]. This approach draws from nature and is the apparent approach of large colonies of animals, such as ants or termites. The key idea is that each robot has an individual algorithm that if accomplished by the individual robot will also accomplish the global task. This is the idea of emergent behavior. The overall effect of the group of robots is beyond the behavior of each individual.

Another approach is a cooperative multiple-robot algorithm. In cooperative algorithms, robots communicate information to other robots [2]. This may be simple sharing or could include centralized or distributed planning. In the simple cooperative approach that we used, each robot was assigned a part of the arena as its initial area to search. As each robot moved through its area, it built a map (occupancy grid) and broadcast its location and its findings to the other robots. Each robot tried to search unexplored space in its area and then the unexplored area in adjacent areas.

2 Simulator

We built a robot simulation environment. A number of master student projects were involved in building the different parts of the simulator. Our goal was to make as realistic a simulation as possible. The behavior of this simulated environment will be validated by comparison

[*] **0-7803-8566-7/04/$20.00 © 2004 IEEE.**

with the behavior of our real robots running the same robotic code. Our real robots include Nomadic scouts, Activmedia pioneers and ATs, and locally-built all-terrain robots.

The simulator is distributed. One process runs the environment. Each robot has a process that runs the robotic code which calls functions in the hardware simulation code that converts those commands to the standard requests known to the environment. These requests are sent to the environment which accomplishes the specified actions. For example, a motion request will result in the position of the robot changing in the environment. Currently the move is done correctly. In the future, traction of the surface and slippage will be included in the calculation.

No confirmation of the new position of the robot is returned to the robot since that location confirmation does not happen in the real world. This is a critical difference for realistic simulation of the world. If we were simulating a situation where there was a super controller that viewed the whole arena and was able to tell each robot where it was and where to go, we could create a process with a view of the whole arena that could send messages to each robot. However, we would still require that process to obtain location information from vision processing.

Any sensor reading command is converted to requests to the environment which responds with a value based on the objects in the environment. For example, a sonar request will result in the environment calculating the distance to objects in the cone of the sonar ping. Currently, the environment sends back the value calculated in a 2D model. In the future, 3D will be used and randomness will be introduced to better represent real sonar behavior.

The two sensors used in this experiment were a non-directional Boolean heat sensor that returns a 1 if there is a heat source within a fixed distance and a directional, Boolean sonar that returns a 1 if there is an object within the directional cone of the forward-looking sonar. These two were chosen as the simplest sensors that could allow the search-and-tag task to be accomplished.

The user can see the activity in the simulator using viewers. Each viewer is also run as a separate process. This allows various viewers to be connected to the system. World-view or god's-eye viewers can be used to see the whole action and robot-centric viewers can be used to give the user the image that would be seen from a camera mounted on the robot. Each viewer can be set either at a fixed place or on a robot. A special night-vision viewer has been created to simulate what would be seen through a night-vision scope.

The units of the simulator's environment are not tied to the size of the robots or the motion of the robot. These relationships are determined by the environmental XML file and the hardware simulator developed for the particular robot being simulated.

3 The Initial Experiment

The goal of our research was to compare and understand the strengths and weaknesses of a variety of approaches. The initial experiment was a comparison between a simple swarm approach and a simple cooperative approach. We have used the standard search-and-tag problem in which there are a number of items that must be located in a large unstructured environment. A number of robots explore the environment and report the location of each of the items. The efficiency of the approaches was compared in terms of time required to locate the first item and average time to locate the items located. Simulations were stopped when it appeared that little progress was being made on tagging targets.

Our swarm approach was a variation of foraging behavior often used in other work, e.g. [5], [6]. Each robot started near the center of the arena and was given an assigned area (actually a quadrant). The robot picked a random point in that quadrant and moved toward that point. The robot first moved along the y-axis and then moved along the x-axis to the destination point. When the robot arrived at the selected point, it would select a new random point and move to the new location. When it sensed a heat source (target), it scanned the vicinity using the sonar (the heat sensor is non-directional) and headed for the nearest object. After it investigated the object and tagged it, the robot backtracked to the path it was traversing and selected a new destination point. This seemed to be the simplest swarm algorithm that had a reasonable chance of success at tagging objects.

Note that the swarm robots must keep track of their own location by "dead-reckoning". As it turned out, this was not always done well and robots tended to drift into other quadrants.

The simple cooperative approach was to similarly divide the space into assigned quadrants. Each robot started in the same location as in the swarm algorithm. As each robot moved, it broadcast its location, obstacles, and tags to the rest of the robots. Using that information, each robot had an occupancy grid that it used to identify unexplored areas in its own territory and in adjacent territories. So, if an obstacle blocked a robot from exploring part of its territory, the adjacent robot would eventually explore that part.

Note that the reported x,y position of the robots was calculated by the robots and was not as accurate as desired. Again, the individual robot had no easy way to verify its actual location and so may have been incorrect about where it was in relation to the map.

4 Testing Setup

A 200 by 200 unit arena was selected. Each robot and each target had a diameter of 2 units. With straight motion, the current simulated robot would take about 500 time steps to move across the length of the arena. Although a smaller arena would be reasonable for a small number of robots, targets and obstacles; this size seemed reasonable for the range of configurations we wanted to test.

The robots were placed systematically. The center point of the arena was 0,0. The first four robots were placed at points +/- 10, +/- 10. That is, one robot was at 10,10 ; another at 10, -10 ; etc. The four additional robots were placed at points +/- 40, +/- 40. This alllowed the robots to move initially without interference with the other robots.

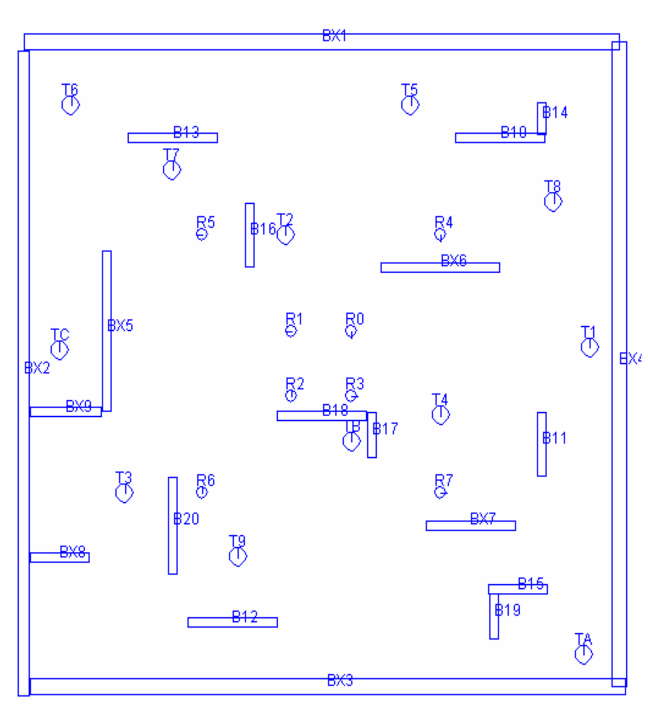

Figure 1. Full test environment

Twelve targets were arbitrarily placed in the arena. They were fairly evenly distributed among the quadrants (see figure 1) and between near and far from the center. They were numbered T1…T9,Ta.Tb, Tc. They were introduced in numerical order. That is, the first scenario had targets T1 to T4. The second scenario had targets T1 to T8, and the third scenario had all the targets. The first four were placed one per quadrant. The next four were placed mainly in the 1st and 2nd quadrants. The last four were mainly in quadrants 3 and 4.

Sixteen obstacles were created, number BX5, ... B20. Note that BX1 – BX4 were the outer walls. Some obstacles were adjacent to other obstacles to form L shaped walls. The obstacles were also introduced in numerical order. Thus, the first scenario with obstacles had obstacles BX5 to BX8, etc. The obstacles were detected by the sonar but were transparent to the heat sensor.

5 Results of the Initial Experiment

The two algorithms were run on identical environments. A variety of situations were tried with each. It was not possible to make sufficient runs for a statistically valid analysis. However, with the knowledge obtained a more elaborate set of experiments are being developed.

5.1 Swarm Algorithm

The first sequence of comparisons was the swarm algorithm with no obstacles. Four robots were tested with 4 targets, 8 targets and 12 targets. Then eight robots were tested with the same sequence of targets. The results are shown in table 1.

Table 1: swarm - average number of targets found by 4 robots / 8 robots

	4 targets	8 targets	12 targets
4 robots	2	1.5	3.75
8 robots	0	0	2.6

The decrease in performance from 4 targets to 8 targets is best explained by distracting heat sensing caused by targets, such as TB. This target was very close to the starting positions of the initial four robots. However, there was an obstacle between the robots and the target. The result was that two or three (varied between trials) of the robots would sense this target and then run into each other. Once a robot ran into another robot in the presence of a target, it was very common for the two to continue to interfere with each other.

The increase in number of targets found with 12 targets was expected. However, the percentage of targets found actually decreased. Interestingly enough, some of the nearest targets were not found. Not being able to distinguish between targets and obstacles created problems. Also, sonar is not perfect and the simulator was designed to mimic real sonar. One robot, R3, always missed target T4. It would sense the heat of that target but when rotating to locate the target with sonar, the robot would be too far away to catch that target in its sonar cone.

The decrease in performance from 4 robots to 8 robots with four and eight targets is hard to explain. No increase in interference between robots was obvious but performance went down in both number of targets found. Since there were random destinations chosen, this inconsistency may be explained by the low number of trials. A smaller number of trials were run due to the length of time to run a trial. In the situation of 12 targets, the number of targets fournd went down but those found, were found sooner. One new type of interference occurred with 12 targets. If two targets were close, after one target was tagged and then the heat turned off, the robot would be close to the previously tagged robot but would sense the heat from the other target. The result was numerous attempts to re-tag the already tagged target.

Table 2: swarm – average time steps for 1st target and all targets found

#robots	4 targets	8 targets	12 targets
4 robots	4053/9332	3122/6357	5304/11556
8 robots	0/0	0/0	2402/3259

The same sequence was tried for scenarios with 8 obstacles and then with 16 obstacles. Both the 4 and 8 robot versions were slightly less effective with 8 obstacles but completely failed for the 16 obstacles.

5.2 Cooperative Algorithm

The cooperative algorithm was run with only the 4 robots scenarios. The 4 robots were initially run in the scenarios with 4, 8 and 12 targets and no obstacles. The number of targets found in each is shown in table 3.

Table 3: cooperative - number of targets found by 4 robots

#robots	4 targets	8 targets	12 targets
4 robots	3	4	5

The cooperative algorithm was better than the swarm algorithm in terms of number of targets found in each scenario. However, in the coperative scenarios, the percentage of targets found still decreased as the number of targets increased.

The similarity in numbers between the three scenarios of 4, 8, and 12 targets (no obstacles) indicates that any advantage of the cooperative algorithm would come later in an execution with the systematic coverage of the whole environment.

Table 4: cooperative – time steps for 1st target and average for all targets found

#robots	4 targets	8 targets	12 targets
4 robots	4870/9789	4885/10968	4933/11753

The cooperative performance stayed almost identical for the situation with 8 obstacles, with the 1st target found at almost the same times. However, the performance decreased significantly for the environments with 16 obstacles. In those environments, only one target was found at approximately 15, 000 time steps.

6 Conclusions

Initially we expected to be comparing larger groups of robots with these algorithms. As we did the experiments, we discovered that the behavior was very unpredictable. Adding more targets should have made the task easier. However, interference between targets and between obstacles caused very big changes in the behavior of the robots.

For algorithms to scale up in terms of number of robots, number of targets, and number of obstacles, it necessary for the algorithm to have some characteristics related to discrimination between other robots, obstacles, and targets. There seems to be a tradeoff between sensing at a distance and confusion about which item was sensed. We maintained one pattern of robot placement, target placement, and obstacle placement. We need to investigate the sensitivity of the algorithm to the pattern of items. A discrimination measure might lead to the ability to predict the effect of a pattern of items on the ability of the algorithm to perform. This discrimination measure will help to develop better understanding of the dynamics of swarm and cooperative algorithm. Part of this understanding might be a probabilistic model of the behavior of these algorithms in complex environments.

We also intend to improve our simulator to enhance the performance and allow larger and longer executions of complex environments.

Initial Model of Interference

Our experience with the multiple-robot algorithms suggests a model that should include factors such as the discrimination power and the localization accuracy of the robots. Much work has been done on models for cooperative and swarm behavior, e.g. [7]. However, most models assume perfect localization, perfect recognition, etc.

Although assuming perfection can simplify the study of the algorithms, it is not realistic with robots. We feel that we must study the effects of errors early in the process of comparing multiple-robot algorithms. For example, an algorithm that has potential for strong interference between robots will not scale up well unless the robots can be spatially separated early in the algorithm. The model will need to include both probabilities of distances between elements and terms relating discrimination capabilities.

References

[1] E. Boonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, c1999

[2] D. Gustafson and E. Matson, “Taxonomy of Cooperative Robotic Systems”, Proc. IEEE SMC Conf, Washington DC, Oct 2003

[3] B. Hallam et al, *From Animals to Animats 7,* MIT Press, c2002

[4] J. Kennedy and R. Eberhart, *Swarm Intelligence*, Academic Press c1999

[5] V. Kumar and F. Sahin, “Foraging in Ant Colonies applied to the Mine Detection Problem”, Proc. IEEE Int Wkshp on Soft Computing in Industrial App, June 2003

[6] V Kumar and F Sahin, “A Swarm Intelligence based Approach to the Mine Detection Problem”, IEEE SMC 2003

[7] K. Sugawara, M. Sano, I. Yoshihara, K. Abe, and T. Watanabe, “Foraging Behavior of Multi-robot System and Emergence of Swarm Intelligence” 1999 pp257-262