2004

A Genetic Algorithms Approach to Learning Communication and Coordination in Simulated Robots

Chris Sotzing
Colby College

A Genetic Algorithms Approach to Learning Communication and Coordination in Simulated Robots

Chris Sotzing

Senior Thesis
Advisor: Clare Bates Congdon
Department of Computer Science
Colby College

Spring 2004

Abstract

This project is motivated by an existing robot system for mapping unknown environments and attempts to improve its effectiveness through the use of genetic algorithms. Using a robot simulator, the mapping system is created using simulated robots and a simulated environment. The robots are controlled by a supervisor agent that makes the high-level decisions about tasks for individual robots to complete to accomplish the mapping effort. This research investigates the ability of adding a genetic algorithm learning component to the supervisor to improve its ability to coordinate the robotic agents.

Acknowledgments

There are a number of people who deserve a lot of thanks in regard to this project. First of all I'd like to thank the students in the Mudd lab who put up with my often painful renditions of *The Darkness*... had I known that my singing in the Linux lab would so clearly travel through the wall, I would have spared you! On that note, I should also thank *The Darkness* for helping me through those long nights in the lab. *"I believe in a thing called love..."*

On a more serious note, I'd like to thank Katelyn Mann for her help with Webots and with the challenges that arise when adding multiple robots to the Webots system. A big thank you goes out to Marc Attiyeh, Rachel Noiseux, Skyler Place and Kevin Septor for their support during the long hours in the Linux lab. And finally I'd like to thank my advisor, Clare Bates Congdon for all her help and support and for believing in me. Without her this project would not have been possible and am eternally grateful.

Peace. I'm out.

Contents

1 Introduction

The goal of my research is to develop an efficient controller for multi-robot mapping of unknown environments. I will take an existing mapping algorithm for multiple robots and add a genetic algorithm (GA) learning component in order to improve upon the algorithm and improve the efficiency of the mapping process.

1.1 Robots

Although once only the product of people's imaginations, robots have started to become commonplace throughout our lives. From the highly publicized Spirit and Opportunity Mars rovers to the industrial machines used to assemble our automobiles, almost everyone has had contact with robots in some way. It is this contact and the seemingly limitless list of tasks that robots can be applied to that originally attracted me to working in this sector of computer science.

In particular I am fascinated with the ability of robots to do things that we as humans are not able to do. For such a highly evolved organism we have a lot of limitations! For instance, take a disaster like a large nuclear or chemical spill. It would be very difficult to get scientists to ground zero to catalog the scene because of the potential for serious bodily harm. This is the perfect scenario for a robot system to take over. There are no health hazards for a robot, and in the event of some tragic accident, there is no loss of life. In addition to this benefit, the computers on board most robots can make calculations hundreds of times faster than even the fastest mathematician.

In addition to hazardous waste management, robots are also being extensively used in the military because of their ability to keep people out of dangerous combat situations. PackBot by iRobot is a good example of such a robot [3]. It is a multipurpose machine that can be used for surveillance, exploration, and possibly even weapons deployment in

the future. The robot is small enough to be carried on a soldier's back and is rugged enough to be tossed out of a building and continue to function.

The military also uses aerial robots to keep personnel out of the line of fire. Unmanned Aerial Vehicles, or UAVs, such as the Predator [1] and Global Hawk [6] are being used in combat today and can do various sorties from high-altitude reconnaissance to low-level air to ground attacks. There are a number of reasons why these robots are so valuable to the military. First, pilots are no longer put in the line of fire and are therefore safer. Secondly and most importantly, because there is no human on board the aircraft, it can perform maneuvers that were otherwise impossible. This makes these weapons far more efficient and far better than if they had a pilot aboard.

Today's robots systems can not only go places people cannot go, but they can also take on tasks that would be otherwise too tedious or too large in scope for humans to undertake. One of these tasks is the detailed mapping of unknown environments. Although this is a task that a human could easily do, it is well suited to a machine that isn't exhausted by tedium or by lengthy hours of work. In some cases robots are the only solution for mapping unknown areas, the ocean being a prime example. The oceans are the last real frontier on our planet probably because we as humans are unable to visit their depths unaided, and often not at all. This is a job for unmanned robots and it is this ability that fascinates me more than anything.

1.2 Genetic Algorithms

Machine learning and specifically genetic algorithms have interested me from the moment I learned about them. The ability for a system to evolve into a better one and to in a sense "learn" is fascinating. This is particularly exciting because it allows systems to learn the best strategies for any given environment or problem set. Genetic algorithms are an efficient way of improving a system because they don't limit themselves to enhancements that are clear to programmers without learning. (e.g. if there is a fire, robots shouldn't drive into it, etc.) That is, they can find improvements to systems that humans might overlook or not even think of.

It is this aspect of genetic algorithms that influenced me to see what kind of effect they would have on a coordinated mapping system. Although I can think of a good coordinated mapping algorithm I wanted to see if it was possible for a genetic algorithm to evolve an equal if not better system. The combination seemed to be a promising pair and I was intrigued by the thought of mixing two relatively new technologies together to form something new and different.

1.3 Overview of Thesis

This paper will document this study from concept to implementation. I will first give the background of robot mapping as well as a brief overview of genetic algorithms. This will be followed by a section describing the tools used in this study including the robots, simulator and genetic algorithm software. The system itself will then be described in terms of its creation and the addition of genetic algorithms. A results section will display my findings and then be explained in the discussion.

2 Background

As robots become cheaper and more available, they are being used for more applications and studies. One of the most common and promising uses for this new technology is the mapping of unknown environments. This has been worked on using both single and multi-robot strategies, the latter of which is a more recent development. In this chapter I'll give a brief background of the problem and discuss a specific multi-robot paper that this research built on.

2.1 History of Robotic Mapping

According to Sebastian Thrun, a major contributor to the field of mobile robotics and a co-author of the paper this study is based on, "robotic mapping addresses the problem of acquiring spatial models of physical environments through mobile robots...[and is]...generally regarded as one of the most important problems in the pursuit of building truly autonomous robots." [8]

Robot mapping dates back to the 1980's and early 1990's. Around then the field was divided into two approaches; metric and topological. The metric approach sought to represent the world with geometric figures such as grids and polyhedrons. These figures model the occupied and free spaces of the environment and robot decisions are based on these. The topological systems took a different approach in that they focused on the connectivity of different "places" or locations in the world. These places would be connected by "arcs" which included information on how to move from one place to another. [8]

The robot mapping systems in the 1990s focused on both "world-centric" and "robot-centric" algorithms. World-centric algorithms are systems in which robot sensor information is less important in relation to the global map being created. The focus is therefore on the resulting map, not the measurements that lead up to that map. Robot-centric algorithms on the other hand focus on the sensor information that robots get at

certain locations in the world. Although simpler in theory, robot-centric algorithms lost favor due to a number of disadvantages including difficulty in extrapolating measurements from multiple locations and difficulty telling similar locations apart. [8]

Since the 1990s, robot mapping systems have been mainly probabilistic techniques incorporating both single and multi-robot strategies. The following sections summarize these systems.

2.2 Multi-Robot Mapping Strategies

There are a number of groups currently studying multi-robot mapping strategies around the world despite its relative novelty in the field. One of these studies is a joint effort by three universities: Carnegie Mellon University (USA), The University of Freiburg (Germany) and The University of Bonn (Germany). In a paper entitled *Coordination for Multi-Robot Exploration and Mapping* [7] the authors, Simmons et al., describe their system for mapping unknown environments using a central controller module and a robot bidding system.

In the Simmons et al. system, there are two main assumptions: first, the world in which the robots are mapping is static and cannot change during the mapping process; i.e. there are no humans or other agents present in the system. Second, the robots begin in view of each other and are given their relative location. The system consists of the robots doing the mapping and a central controller module that coordinates the mapping effort.

The mapping system starts by sending the relative location information to the robots. This information is then used in conjunction with the data obtained by the onboard laser range sensors to determine exactly where the robot is in relation to its surroundings. This is accomplished by calculating three factors. First, the estimate for the robot's position is determined by calculating the maximum likelihood of the supplied and sensor data. A similar calculation is used to determine an estimation of the map (from what the robot can see from its current position). Figure 2.1 shows a diagram of the maximum likelihood estimation calculated by the Simmons et al. paper.

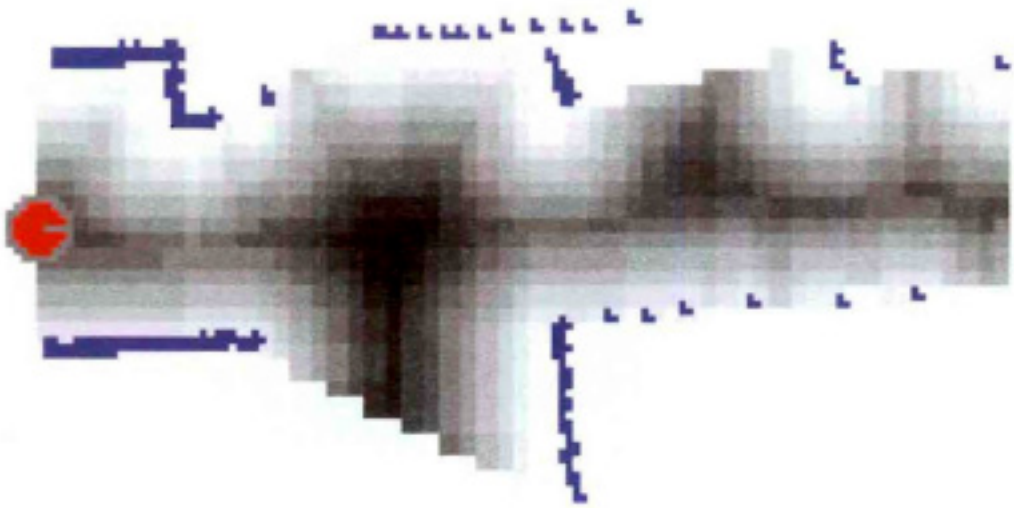


Figure 2.1: A diagram showing the result of the maximum likelihood estimation of the map in the Simmons et al. [7] paper. The darker the dots, the smaller the likelihood of there being an object present there.

The third calculation done by the robots is a posterior density calculation characterizing its "true" location [7].

Once each robot has determined its location and local map, this data is sent to the central controller module where it combines all the robot local maps into a global map of the world. Because each robots local map is created via a maximum likelihood estimate, the central controller has to use a similar algorithm to parse all the maps together with the least amount of error. Once this map has been created, the robots can begin the bidding process.

The bidding system consists of a process where robots make "bids" on sections of the world that they would like to explore and the central controller takes those bids and then assigns tasks. In order for a system like this to exist, the world has to be divided into a grid so that each bid can focus on a specific "cell" in that grid. In the Simmons et al. system, the grids had a resolution of 15 cm meaning that each cell was 15 cm square.

The bidding process begins with each robot identifying the "frontier cells" in their local maps. Frontier cells are defined as "clear" cells (cells with no obstacles within their borders) that are adjacent to unexplored cells. By definition, frontier cells must be at least 30 cm apart, which corresponds the average width of the robots used in the study. The robots make bids only on these areas because they are cells that need to be explored. A map of an environment showing the frontier cells is shown in Figure 2.2.

Figure 2.2: A view of the map created by the Simmons et al. system. Obstacle cells are black, clear cells are white, unknown cells are gray and frontier cells are denoted by gray circles.

Once the frontier cells have been identified, bids are made to the central controller by estimating the cost and information gain of each cell. The cost is calculated by computing the optimal path from the robot to the frontier cell, assuming deterministic motion. Only clear cells are considered when calculating the optimal path so that no extra time will be taken to avoid any obstacles. The information gain is found by estimating the number of unknown cells that fall within the radius of the frontier cell [7]. For efficiency a flood-fill algorithm is used, ending propagation either when a "clear or obstacle cell is found or when the distance to the frontier cell is more than the sensor range [7]. Figure 2.3 shows the information gain as a circular region around three frontier cells.

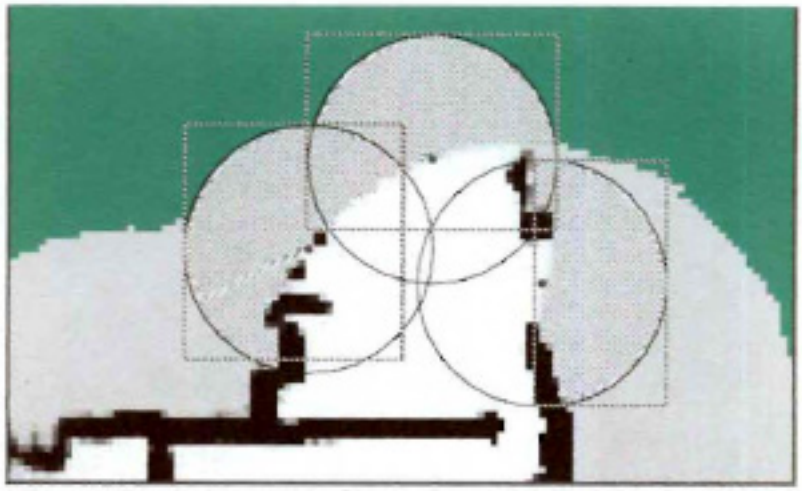


Figure 2.3: A diagram showing the information gain region for three frontier cell. Cross-hatched areas represent the information gain regions.

Once the bids are created by calculating the cost and information gain for each frontier cell, they are sent to the central controller. After waiting a short while until other bids arrive, the central controller then finds the bid with the highest "net utility" defined as information gain minus cost. The robot that made this bid is told to navigate to this frontier cell. The rest of the bids are then discounted by estimating the percentage of overlap between the information gain regions of all the bids [7]. Once the remaining bids have been decremented, the bid with highest net utility is chosen and the cycle continues until there are no bids or all bids' information gain is below a predefined threshold.

The Simmons et al. system results in a well defined map of the world that takes in to account separate rooms and other obstacles in the path. Figure 2.4 shows a map created by this system.

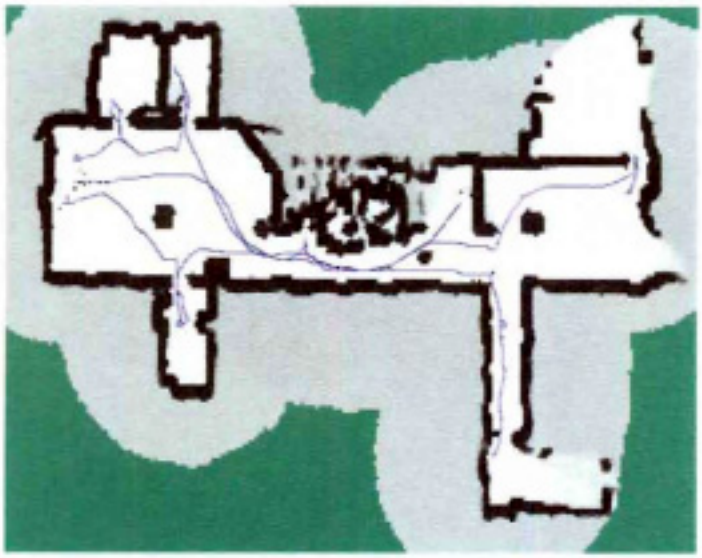


Figure 2.4: A map created by the Simmons et al. system. Robot tracks are marked in blue and the circular gray areas represent the information gain areas mentioned above.

2.3 Genetic Algorithms

A genetic algorithm (GA) is an approach to solving problems that is inspired by the biological process of evolution and natural selection. That is, over the length of the experiment, solutions tend to get better and better following the "survival of the fittest" principle. To start, for any given problem a set of random solutions are created. This set of solutions is called the "population". The solutions can be of any type but because this study uses a solution that consists of a set of rules, we will use this type while explaining the algorithm.

As mentioned before, the initial population is a set of randomly created strings, each containing data for a number of rules for the system being evolved. Each string in the population is evaluated and given a score by a "fitness function" to determine its "fitness". In biological terms, a fitness is basically a success rating for an organism. The higher the fitness, the more likely the organism will survive. Genetic algorithms function in a similar fashion. After each string is evaluated and given a fitness, the population "reproduces" to create a new population. The strings with the higher fitnesses are more likely to pass on their data to the next generation and in theory as the generations

progress, the solutions to the problem get better and better. A diagram illustrating the genetic algorithms process is shown in Figure 2.5.

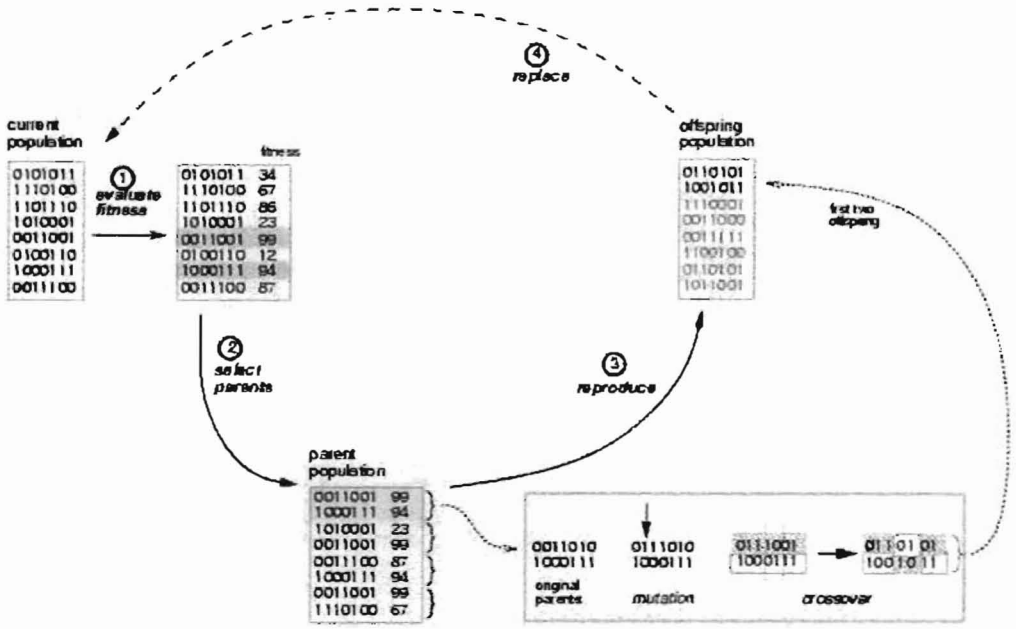


Figure 2.5: Diagram showing the process of a genetic algorithm.

The genetic algorithm mimics the biology that inspired it in another way as well in that there are other factors that affect what rules or "genes" end up in the next generation. "Mutations" and "crossovers" can also have an affect. A mutation is a spot change where a bit of the string is randomly modified. In the case of a binary string, a 1 might be randomly changed to a 0, thereby changing the rule completely. A crossover is a similar modification where portions of two strings swap to create two new strings, as shown in the bottom right of Figure 2.5. Both mutations and crossovers can be controlled in the algorithm in terms of their frequency, and often different combinations of these can have significant changes on results.

3 Tools

The goals of this study were twofold: first, we wanted to see if it was possible to recreate a simplified version of the Simmons et al. system in a simulated environment. In doing so, we wanted to see if it was possible to take a similar bidding system and attain similar results for the mapping system created by Simmons et al. We then wanted to see if the addition of a genetic algorithm learning component could improve the effectiveness of the central controller. The goal was to recreate and then improve upon the coordinated multi-robot mapping system created by Simmons et al.

There are a number of tools that were used in this experiment to help discover the effect of genetic algorithms on a coordinated robot mapping system. Each one played an important and specific role in the creation of the project and their details are listed below.

3.1 Khepera Robot

In order to reach these goals, a number of tools were used in this study. First, a robot was needed to mimic the system created in the Simmons et al. paper. The Khepera robot [5] was chosen for a number of reasons. One of the first and most influential reasons for choosing the robot was its cost. Being a miniature robot, it is relatively inexpensive compared to its larger counterparts. This size also played into the choice of robot in that we had limited lab space and robot size was important. Had we decided to implement the mapping system outside of the simulator, the size of the testing area would be directly related to the size of the robot and therefore a smaller robot was preferred. Because we had decided to work in a simulated world we wanted a robot that could be easily represented in a simulator. The Khepera robot satisfied this requirement well in conjunction with the Webots simulator. (Section 3.2)

The Khepera robot includes two wheels and 8 infrared sensors. The wheels are independent of each other and can go forwards and backwards at a speed determined by

the user. Sensors can be set as either light or distance sensors. Figures 3.1 and 3.2 show a sensor diagram from and a photo of the Khepera robot.

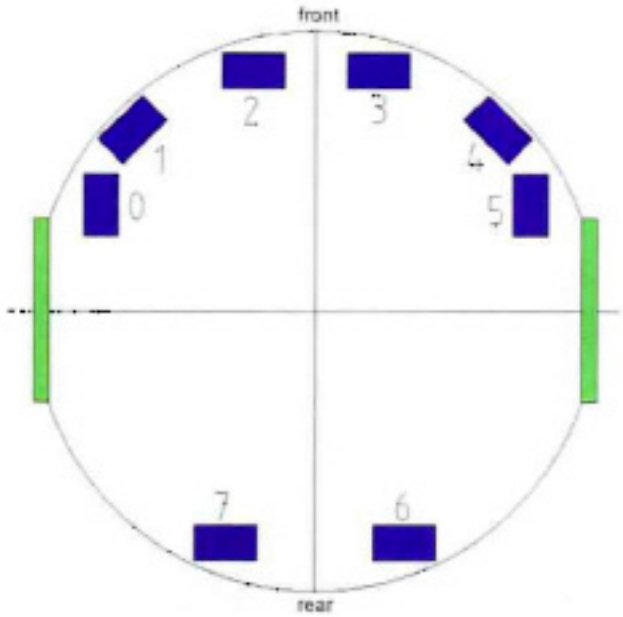


Figure 3.1: Diagram of the Khepera robot showing infrared sensors (in blue) and wheels (in green).

Figure 3.2: Front view of the Khepera robot showing infrared sensors 1 through 5. Photo courtesy of [5].

Although not used in this study, Khepera robots also have the ability to be expanded to include accessories such as cameras, grippers and additional input / output devices. This

expandability makes it a very powerful tool and though we aren't using these features, they are worth noting.

3.2 Webots

Webots [2] was chosen as the simulation environment. The simulator creates an extremely realistic model of the Khepera robot as well as the environment that it is in. A screen shot of the simulator is shown in Figure 3.3.

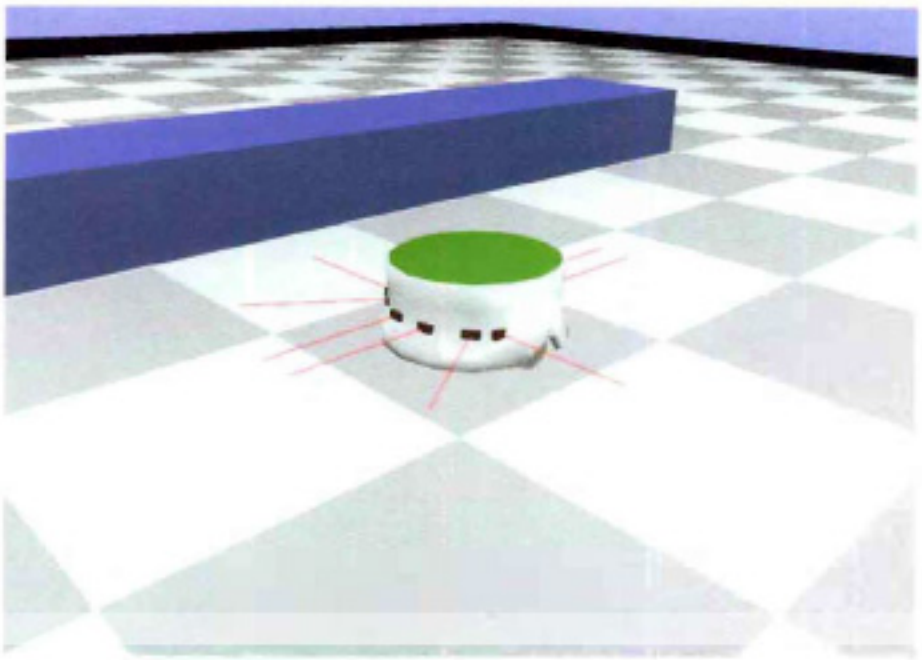


Figure 3.3: A simulated Khepera robot in the Webots simulator. Red lines show sensor vectors.

There are number of characteristics that make Webots the ideal simulator for this study. First and foremost, there is direct support for the Khepera robot and code generated for the simulator can be put onto the robot with little or no modification. Webots also does a convincing job at simulating noise and other real world environmental features that might be experienced in a real-world situation. These include sensor noise, wheel slippage and the requirement of robots to communicate via simulated radio emitters and receivers. These factors are important should the system ever be applied to real robots in a real environment.

In addition to the factors mentioned above however, the most important factor that makes Webots the perfect system for this study is its supervisor feature. In addition to the robots in the world, the system allows for another agent called a supervisor to be added. This supervisor controls the world and can move / modify aspects of the environment that the robots themselves have no control over. For instance, the supervisor allows automatic teleportation of robots, should the experiment call for such a feature. (This feature is used in this study and it will be explained in detail later on in the paper.) This feature of Webots is essential for this kind of experiment because the robot needs to be moved back to the starting point after each fitness analysis.

This supervisor ends up being an essential part of the system in that it acts as both the supervisor in the sense mentioned above, but also as the central controller for the mapping system. Without this functionality, the experiment would not have been possible and it is for this that Webots was chosen as the simulator.

3.3 Genesis

The GENEtic Search Implementation System, or Genesis, is a system for easily incorporating genetic algorithm solutions into programs. Genesis allows users to have a powerful genetic algorithm component in their programs and it allows for simple user control of this learning system. It is a very modular system that allows users to test multiple experiments with different genetic algorithms, combinations of mutation and crossover rates, as well as number of other parameters. Genesis makes the inclusion of these learning components into programs easy by only requiring the user to supply an evaluation function which returns the fitness for any given point in the search space [8].

Genesis was designed to make genetic algorithms easy to apply to almost any system and because of this it was the right program for this study. In addition, because both Genesis and the Webots/Khepera interface are written in the C programming language, incorporating the former into the latter was very simple.

4 System

In this study there were two main objectives, as mentioned in the goals section: first, recreate a simplified version of the Simmons et al. system, and second add the genetic algorithms component to evaluate the contributions of a learning system.

4.1 Create the system

In reconstructing the Simmons et al. system we simplified a number of aspects so that once the genetic algorithm was added, it would be easier for the system to learn.

4.1.1 Simplifications

One of the first simplifications we made was to limit the size of the world to a known size. Although what was in the world was unknown, the central controller and subsequently the robots would know how many cells the world was made up of. This idea of a world made up of "cells" was taken a little further by completely limiting the world to a grid interface. That is, each cell in the grid could essentially be "on" or "off"; clear or obstacle. This way no cell could be partially open and also contain a wall. This allowed the world to be stored as a two-dimensional array that could be easily updated and accessed for bidding purposes. The array consisted of integers representing the different states of each cell; unexplored, obstacle, clear and frontier cell. Figures 4.1 and 4.2 show a world in the simulator and the corresponding two-dimensional array.

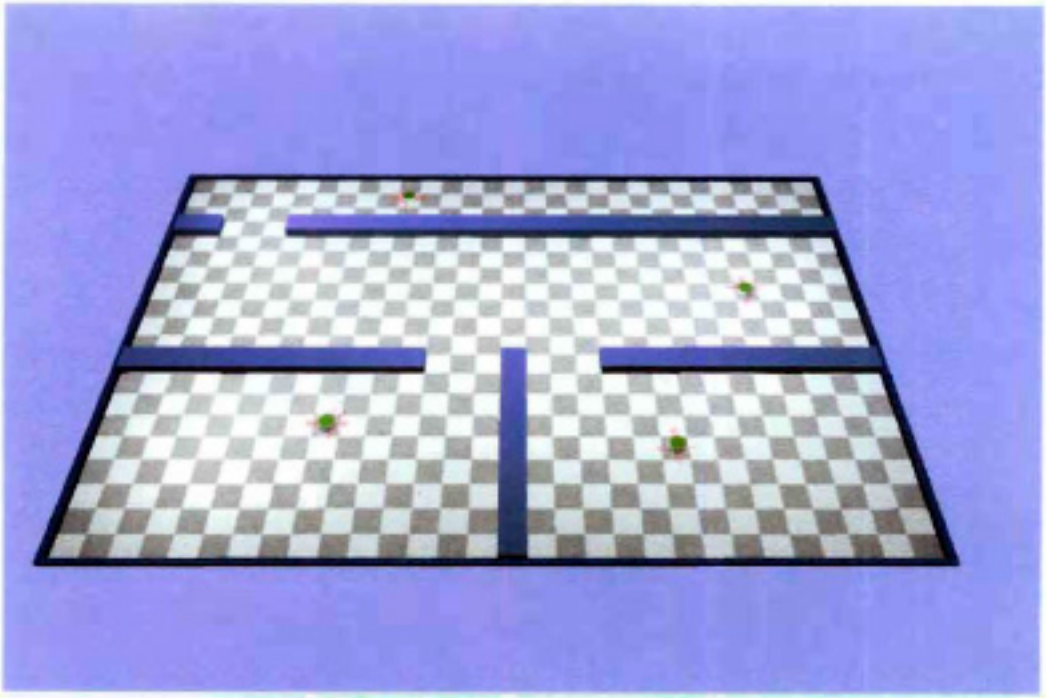


Figure 4.1: A Webots screenshot showing the grid world. This particular map is a simulation of the Colby College Linux Lab in the Mudd building.

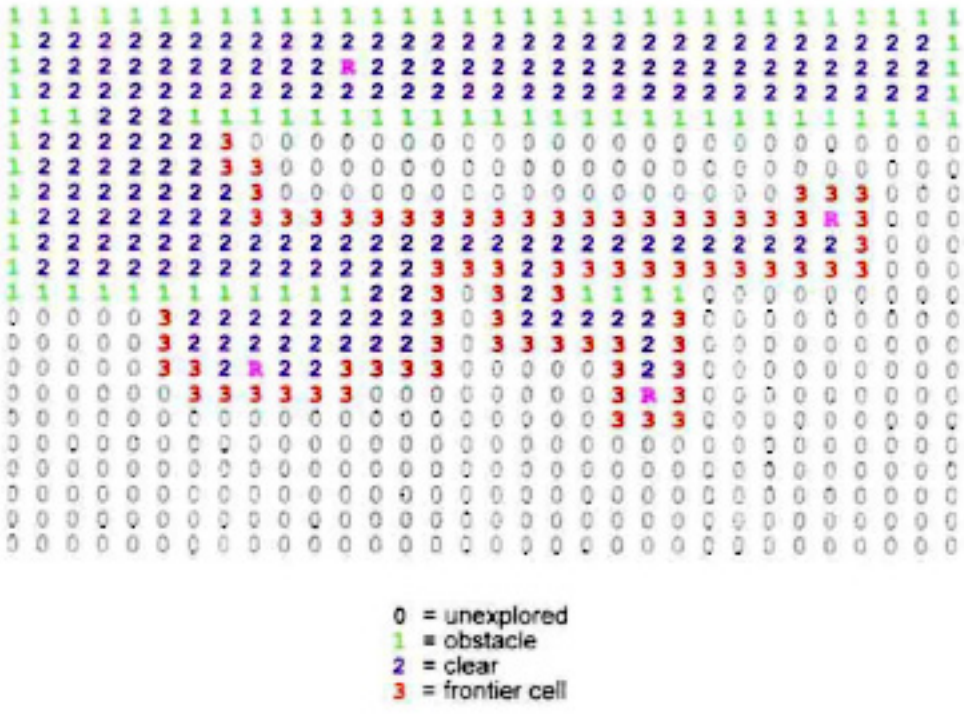


Figure 4.2: The two-dimensional array representation of the map. This particular map is of the world represented in Figure 4.1

This mapping simplification was important due to the fact that in this study the robots were only allowed to move one cell at a time, in one of four cardinal directions. This was done to eliminate as much confusion as possible during the mapping process and it would pose a problem if there were cells that were only partly clear. Getting the robots to move one cell at a time was far more difficult that it may seem. The Webots simulator, in an attempt to create the most realistic environment for testing, includes a lot of "noise" in the system. For instance, it is virtually impossible to have the robot move perfectly straight because Webots will simulate motor speed inconsistencies that would happen in a real environment. This in combination with the fact that the Khepera robots don't have an on board compass or GPS, made programming the robots to move one cell at a time a challenge. To overcome this, we attempted to override the noise and simulate a GPS system that would help keep the robot on the right track.

4.1.2 Robot Movement

In a normal Khepera program, the robot will evaluate its surroundings, move one "step" and then restart the process. In this scenario, a step is a predefined "click" of the loop and usually is a very small distance so as to allow the robot to make very minute changes to its vector. The first thing we did to get by this was to figure out how many steps it took to go from one cell to another (cardinal direction; no diagonals). This was determined to be ~64 steps and therefore the robots were programmed to run for 64 steps before they stopped and re-evaluated their surroundings. The 64 steps however was only the closest estimate of how many steps it took to get to the next cell; it was not perfect and if unchecked, after a number of moves, the robot was completely off kilter and no longer moving from one cell to the next. A system had to be devised to correct this.

To solve this translation problem, the central controller doubled as a supervisor that corrected the robots location when it was done with a move. By knowing the size of the world, the number of cells in that world and the exact location of the robot, it can figure out which cell it should be in and "teleport" it to the correct location. This teleportation ends up only being a tiny amount, but it also allows the robot to always start its next move from the exact center of the cell it is in.

The translation issue was only half of the difficulty in getting the robots to move one cell at a time. Turning the robot dealt with the same limitations. In order to solve this we followed the same technique and discovered that it took ~32 steps to make a 90° turn. Therefore for each turn, the robots were programmed to begin turning and not stop until 32 steps had passed. This left us in a similar situation where the robot was close but not exactly facing the right direction. Once again, the central controller became a supervisor and corrected the robots rotation by finding which cardinal direction it was closest to and setting the rotation to that degree. By using this system of supervisor correction the robots were essentially "tricked" into moving one cell at a time. This simplification would be helpful with the introduction of the genetic algorithm later on in the study.

4.1.3 Bidding

Our system starts off much the same as the Simmons et al. system. Robots are placed in the middle of the world in view of each other. They first analyze their surroundings and update their maps. Now that the maps are updated, the robots can begin the bidding process. Each robot makes a bid for each frontier cell in its map*. The bidding process is similar to the Simmons et al. system but is slightly simplified once again to prevent confusion once the learning component is added.

The bidding system, like that of Simmons et al., consists of two factors; the cost and the information gain of each frontier cell. The cost is virtually the same in that it is a calculation of the distance (Manhattan Distance) from the robot to the frontier cell. A difference however is that because our world is a grid-based world, the distance is not an estimate but the actual calculated distance. The information gain however is not quite as complex as that of the previous system.

In the Simmons et al. system, the information gain was a complicated calculation that basically estimated how many "unknown" cells would be discovered at any given frontier

* For the first bidding cycle of the system, the robots only have a local map, but this gets updated to the global one by the next cycle.

cell. Our information gain calculation was far simpler because of the grid world interface and the two-dimensional array representation. For any frontier cell, the number of adjacent cells that were unexplored were counted and returned. Although simplified, this information gain serves the same purpose and relays the same information that the information gain in the Simmons et al. study did.

Once the cost and the information gain for a given frontier cell was known, the bid was created and sent to the central controller. In addition to sending bids for all the frontier cells in their map, the robots also sent their updated map so that the central controller could update the global map for the next round of bidding.

4.2 Add Genetic Algorithms

Once the initial system was set up, the genetic algorithms component could be included. As mentioned above, the GA string is a set of rules. How the GA string goes from a binary string of numbers to a set of rules is demonstrated in Figure 4.3.

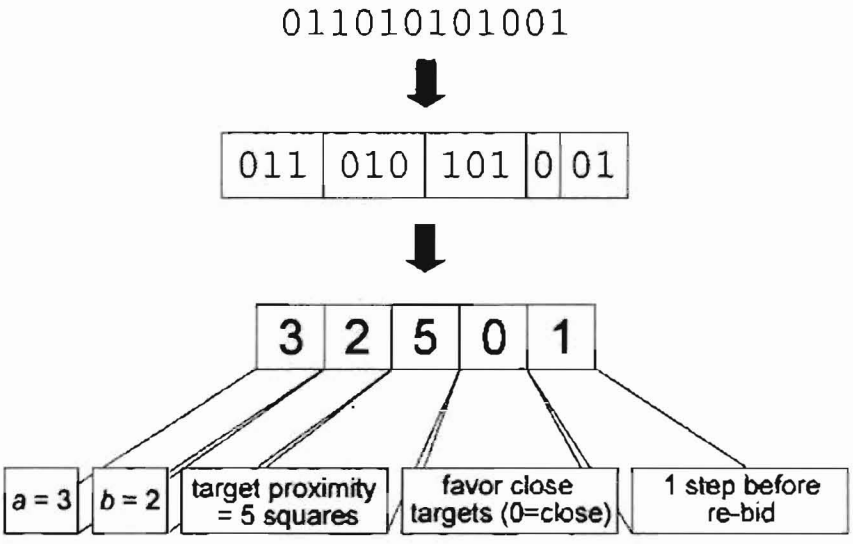


Figure 4.3: A diagram showing how the GA string is converted from bits to rules

In this system there are 5 rules consisting of 12 bits that together determine how the central controller will react to the robot bids. The first 6 bits encode for two variables, a & b, that control the weight of both the cost and the information gain of each bid. Each of the two variables is represented by a 3-bit section and can code for a weight of 0-7 used to construct a bid value using the following formula:

$$\text{bid value} = a * \text{information gain} - b * \text{cost}$$

The next 3 bits represent how close simultaneous targets of different robots can be, in terms of the number of squares. The range is 1-8 squares using Manhattan distances; the effect of this rule is to eliminate some bids from consideration. This also prevents robots from in a sense "over mapping" an area. It is a waste of resources to have all the robots in one part of the world when there is a lot of unexplored space elsewhere. The third rule is represented by 1 bit and contains the information on whether robots should favor targets that are near or distant to their current location; the effect of this rule is to mitigate ties. The fourth rule of the string contains information about how far the robot should move towards the target before it must stop its current task and re-bid; 1, 2 or 3 steps towards the target and then re-bid, or go all the way. This determines how often the central controller will interrupt the robots during a task and enables the system to be more flexible. This is a change from the Simmons et al. system and it was hoped that this kind of flexibility would allow for more efficiency.

As mentioned earlier, the genetic algorithms were implemented using Genesis [4]. To ready the experiment, the GA string data was set in Genesis including the bit-string length, number of rules, number of bits per rule, etc. The system began with a random population of GA strings, or rule sets, being created. For string, the fitness was calculated by sending it to the Webots program as shown in Figure 4.4.

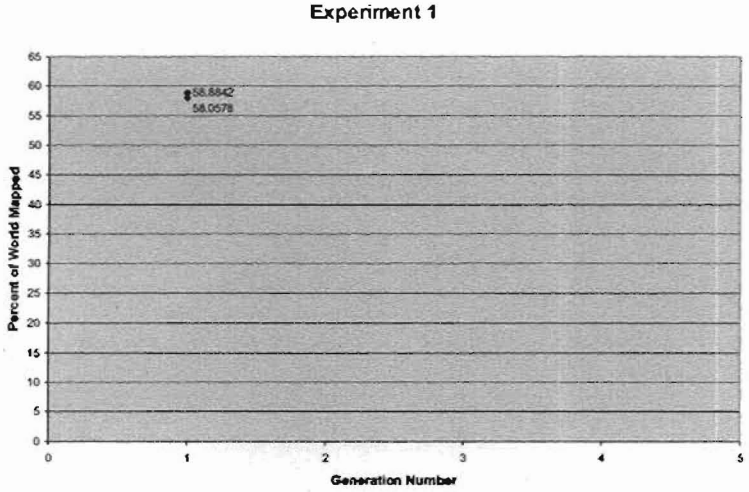
Figure 4.4: A diagram showing the relationship between the genetic algorithm (Genesis) and Webots.

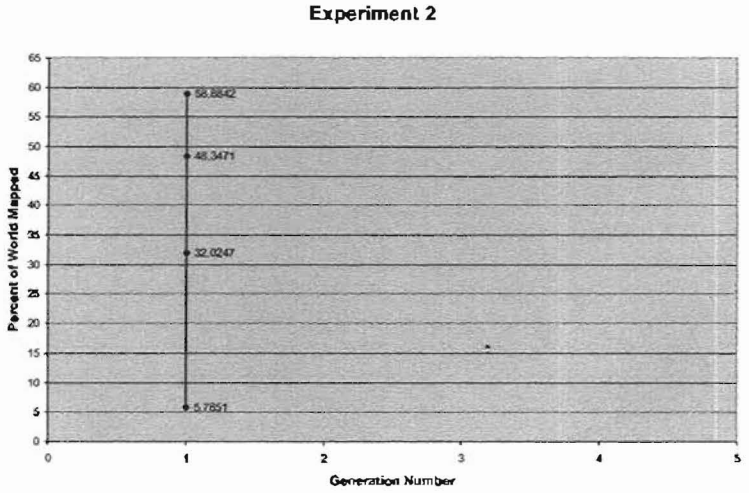
There, the robots would map the world and the central controller would make decisions based on the parameters specified in the GA string. After a predetermined number of robot moves, the mapping side of the system would stop, calculate the percent of the world that had been mapped and return this percentage back to Genesis as the fitness.

The fitness returned to Genesis is the key to the genetic algorithms learning system. The better the fitness, the more likely the string that created that fitness will pass its characteristics on to the successive generation of strings, resulting and better and better solutions. In this study, because the fitness was represented as the percent of the world that was mapped, the higher the fitness value, the better. In order to make the learning as specific as possible, the percentages were multiplied by 10,000 to compensate for fractions of percents.

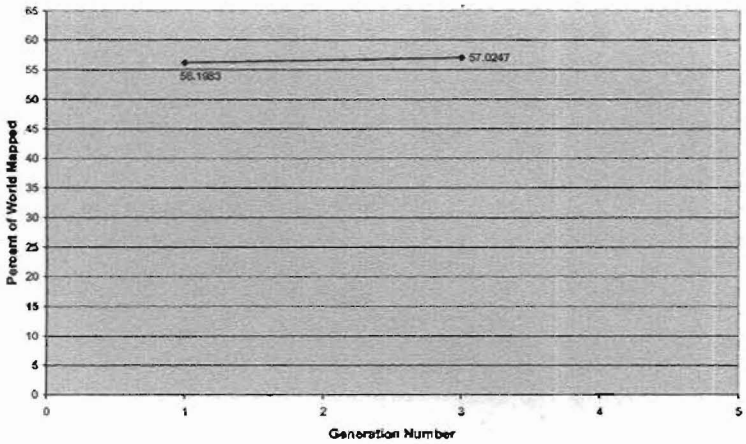
5 Results

To test the system, a total of 10 experiments were run. For each experiment, the initial population of strings was 20 and the number of trials was 50. Since different mutation and crossover rates were not being tested, they were set to about mid range, .01 and .6 respectively. Each robot was given a total of 50 steps before the mapping effort stopped and the percent mapped was returned to Genesis. For each experiment, all the new best fitness values were recorded and as well as the generation that they were found in. The results of the 10 experiments are shown below.

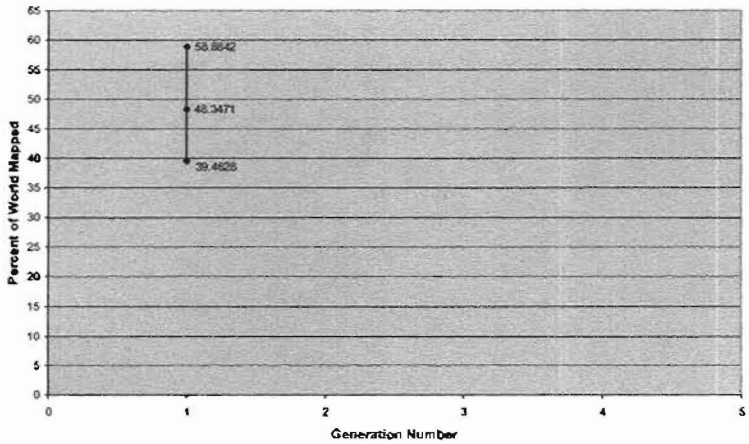




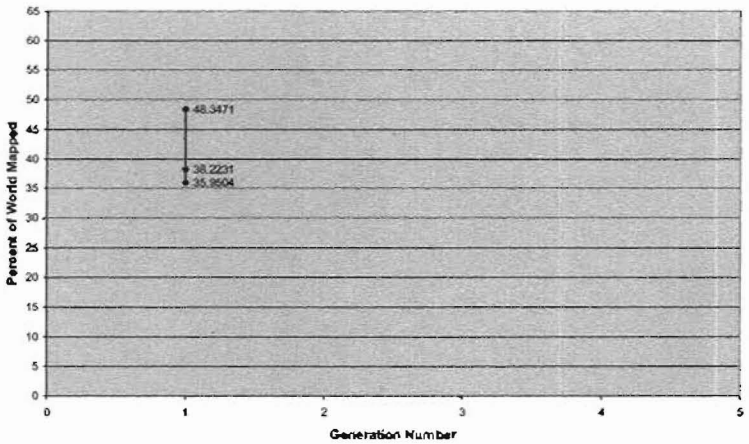
Experiment 3



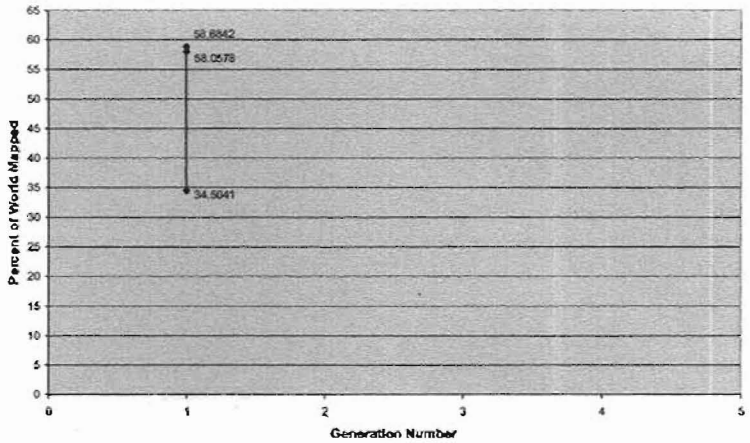
Experiment 4



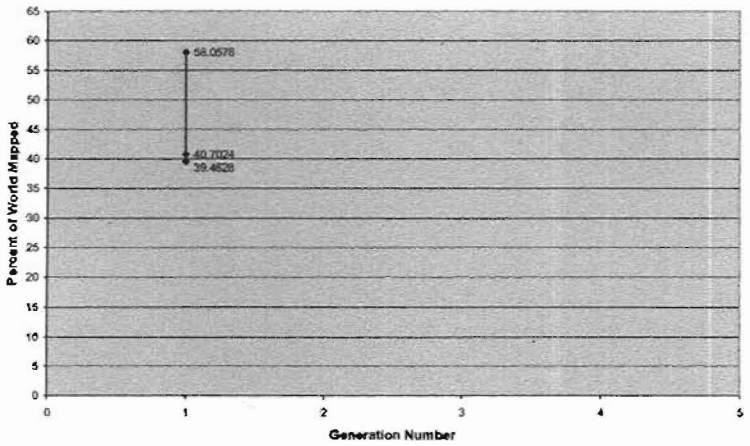
Experiment 5



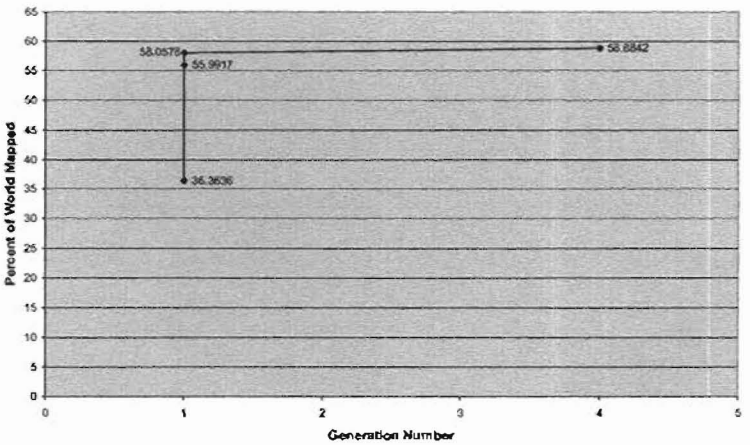
Experiment 6



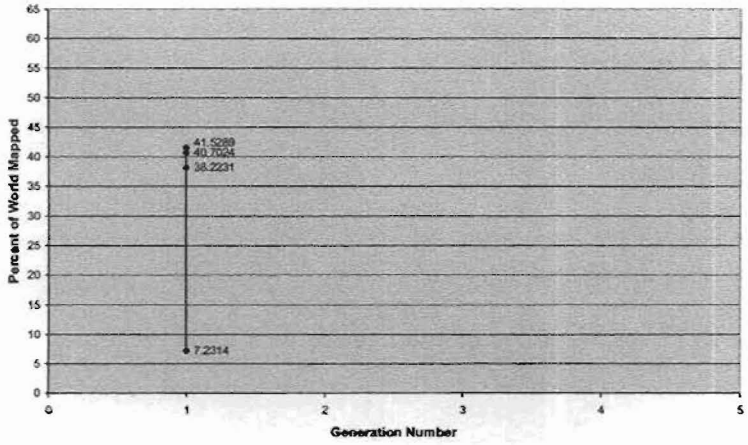
Experiment 7



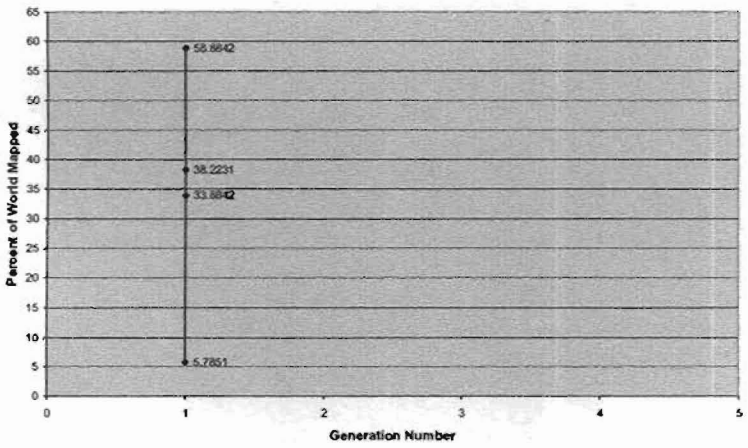
Experiment 8



Experiment 9



Experiment 10



On almost all of the 10 experiments, the system mapped about 60 % of the world. Although this number seems low, it is important to keep in mind that this is the percent of the world that was mapped in the *number of steps allotted.* As it turns out, when each robot is given 50 steps, 60 % of the world is about the most that can be mapped. In this respect, the study was a success in that the system successfully organizes multiple robots to map an unknown world.

Despite the aforementioned achievement, the study was not a complete success. This was because although the system was able to coordinate multiple robots to map an unknown environment, there was little or no learning in the process. On almost every

experiment the best solution was found in the first generation, *before* any reproduction could take place. This means that the best solutions were being randomly created. If one or two solutions were randomly created this would be acceptable, but since almost all of the best solutions were being randomly created this means that there was no learning.

This is most likely due to the short length of the GA string and the relatively small number of rules contained in that string. There were too many combinations of rules that resulted in good solutions. Had more rules been incorporated resulting in a longer GA string, there might have been more learning.

Despite the lack of learning demonstrated in most experiments there were some interesting patterns in the strings that resulted as shown in table 5.1.

Table 5.1: The best GA strings for each of the 10 experiments.

Experiment	Best GA String	% Mapped
1	2 7 2 0 0	58.8842%
2	2 7 2 0 1	58.8842%
3	3 6 3 0 2	57.0247%
4	3 7 2 0 1	58.8842%
5	4 7 2 0 0	48.3471%
6	1 5 2 0 3	58.8842%
7	1 7 0 0 1	58.0578%
8	2 7 2 0 0	58.8842%
9	7 1 3 0 1	41.5289%
10	1 4 2 0 0	58.8842%

In almost all of the experiments, the second parameter (weight of *b*) was far larger than the first parameter (weight of *a*). Both these parameters play important roles in the bidding function as shown in the formula given in Section 4.3. The value *a* determines how much to count the information gain in the fitness function and the value *b* determines how much to *decrement* the cost in the fitness function. These results show a tendency to pay less attention to the cost (as shown by the large *b* value) and subsequently pay more attention to the information gain. The only experiment that had a

GA string that didn't follow this pattern was experiment 9 which had a best percentage of only 41.5 %. Although it is perplexing why this experiment didn't evolve a better string, it might have had to do with a bad initial population of solutions or that the GA wasn't run for a long enough number of generations.

The target proximity values, represented by the third parameter, varied but tended to stay low which allowed robot targets to be closer to one another. This is most likely because the robots started out close together. Because of this, if the target proximity was a value higher than the Manhattan distance between the two robots, the initial bids would be too close to each other and one of the robots' bids would not be accepted, leaving it idle. The fourth parameter, which told the system to favor close or far targets if there were more than one good possibility, always favored close targets. This is to be expected in that if there are two targets with the same information gain, and one is far closer than the other, there is no point in going the extra distance to the far one. This might have had some effect on the high values of b in that if the robots were already favoring close targets, it was in their best interest to discount b as much as possible.

An interesting factor of these results is that there are a number of different strings that resulted in the same fitness, or percent of the world that was mapped. These strings are marked in blue in Table 5.1. Although the strings are different, there are a number of important similarities. First of all in all 6 strings, the target proximity was all equal to 2, meaning that at any point no two target cells could be with in 2 cells of each other. Also, in all six cases the system favored close targets, as mentioned above. The biggest difference lies in the first two parameters. However, though the numbers are different, the relationship of a low a value and a high b value isn't. This meant that the cost of each bid was decremented a lot and the information gain was the prime focus. Although the specific numbers are different, the effect on the bidding outcome is similar.

6 Conclusions and Future Work

The study successfully created a multi-robot mapping system however the results of adding learning are inconclusive. This was mainly because of the small search space and the fact that because of this a genetic algorithm really can't help. In order for learning to make a difference, a much larger search space would have to be created and subsequently far more parameters in the GA string. Some of these new parameters could be different strategies for obstacle avoidance, more complicated bidding function, etc. A larger more complicated world for the robots to search would also help.

There were a number of aspects of this study that I did not pursue due to time constraints. First and foremost, the system should be evaluated using more than two robots. Group behavior would be better analyzed and coordination results would be far stronger. Another detail of the study that was left out was the ability to test each rule set on multiple worlds. At the moment only one world is mapped to determine the fitness and the more maps the rule set can be tested on, the more rugged and efficient the final evolved rule set will be. Once these aspects are included, the result of the system would have to be compared to the non-learning version of the system.

This study also lays the foundation for a myriad of future work. One of the first things that would be interesting to do would be to implement the system without the limitation of the grid world. This system limits the robot to moving from one grid location to another. It would be extremely interesting to remove this limitation and see how the system fares in a more realistic, "grid-less" environment. This would tend to be the case in a real world environment and therefore to be able to easily function in such an environment would be a system virtue.

Another addition to this project would be to implement the system on real robots, rather than just on simulated ones. Due to the fact that Khepera robots were used in this system, the robot code could be directly sent to the robots, but new questions would arise in the sense that there would be no supervisor to correct rotation and translation. In order for

this adaptation to work, the grid limitation would first have to be removed. This would also bring up the question of whether a supervisor is more or less advantageous than a system where robots communicate between each other only.

The idea of having the robots communicate without the use of a supervisor also brings up the issue of specialized robot roles. That is, the system could also evolve specialized roles for robots so that each one doesn't do the same task. It might be the case for instance, that without a supervisor, one or more robots eventually do not do any of the mapping themselves, but instead coordinate the mapping effort. This would be particularly interesting because it would show the evolution of a administrative hierarchy in the robots and would relate directly to real-life behavior.

7 References

[1] Aeronautical Systems (www.uav.com)

[2] Cyberbotics (www.cyberbotics.com)

[3] iRobot Corporation (www.irobot.com)

[4] J.J. Grefenstette. A user's guide to GENESIS. Technical report, Navy Center for Applied research in AI, Washington, DC, 1987. Source code updated 1990.

[5] K-Team Corporation (www.k-team.com)

[6] Northrop Grumman (www.grumman.com)

[7] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, Håkan Younes. Coordination for Multi-Robot Exploration and Mapping. *American Association for Artificial Intelligence.* 2000.

[8] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann, 2002.