Fuzzy Counter Ant Algorithm for Maze Problem

Mohit Ahuja*, Baisravan HomChaudhuri†, Kelly Cohen‡ and Manish Kumar§

University of Cincinnati, Cincinnati, Ohio, 45221

This effort explores the effectiveness of adding a layer of fuzzy logic to a group of swarming multi agent robots for exploration and exploitation of an unknown obstacle rich environment represented by a 2D maze problem. The generalized maze problem has been considered as an interesting test bed by various researchers in AI and neural networks. Using a cooperative multi agent robot system reduces the convergence time considerably as compared to a single agent. For the multi agent case, a robust and effective decision making technique is required that prevents a robot from moving to a region already explored by some other robot. In this paper, we present a counter ant algorithm (modified ant colony optimization algorithm) based on a fuzzy inference system which enables multiple agents in path planning along the unexplored regions of a maze in order to find a solution rapidly. Simulation results demonstrate the effectiveness of this approach.

I. Introduction

AUTONOMOUS intelligent multi-agents have a widespread application in different fields including homeland security. Several applications of such autonomous robots include exploration, mine detection, border patrol, ISR missions etc. A major focus of these robots is its ability to plan its path and execute the task assigned to it in a cooperative manner. It is desired that the autonomous robot should have some adaptive and intelligent decision making capability for navigation in unknown terrains. Intelligent behavior of a robot may be achieved as a result of its capability to learn and adapt1-6. Navigation and obstacle avoidance tasks may generally be solved using artificial neural networked based strategies7. Neural networks have learning properties which makes them ideal candidates for robot motion planning. However, a neural network application with standard feed forward back propagation algorithm is unsuccessful for a class of problems representing “maze-like” features8,9. Werbos and Pang9 showed that this problem may be solved by a cellular simultaneous recurrent neural network (SRN) structure, of fixed architecture, repeated at each node. They also show that the cellular neural network approach will not work with a feed forward design, an argument also derived from Minsky et al10. However, in these applications the architecture used is fixed, regardless of maze size or complexity. The price of this is of course, is the repetition of the entire network at each node of the maze11 that will lead to high convergence time.

Exploration in an unknown environment is an important task for autonomous robots. Exploration is the act of searching or traveling a terrain for the purpose of discovery. The problem deals with the use of a robot to maximize the knowledge over a particular area. The exploration problem arises in mapping and search and rescue situations, where an environment might be dangerous or inaccessible to humans. A maze is a puzzle in a form of a complex branching passage through which the solver must find the route. In order to solve the maze using a single robot, the robot should search the whole maze to find a solution. This is because the environment is unknown with no apriori knowledge of the solution. Multi agent robot system can be used to solve the maze problem to increase efficiency. During the exploration, each robot should ideally search different locations so that the environment as a whole is mapped at a faster rate. Exploration of an agent in a region already been explored by another will result in overlap of the region’s map and hence decrease efficiency along with long elapsed time. After an agent reaches the end point (the solution) of the maze, information exchange should enable the other agents to reach the maze exit in near minimum time.

*Graduate Student, Department of Aerospace Engineering, University of Cincinnati, Student Member, AIAA
† Graduate Student, Department of Mechanical Engineering, University of Cincinnati, Student Member, AIAA
‡ Associate Professor, Department of Aerospace Engineering, University of Cincinnati, Associate Fellow, AIAA
§ Assistant Professor, Department of Mechanical Engineering, University of Cincinnati, Member, AIAA
A typical maze problem solving requires two separate modes. First, the agents need to explore the whole region for solution in the most optimized way. Second, after any agent finds the solution, efficient algorithm is required to make the phase transition of other agents from exploration to follow. Some algorithms to solve a maze problem will include, random walk\textsuperscript{1}, follow the wall algorithm\textsuperscript{2}, Pledge Algorithm, Tremaux’s algorithm, etc. In random walk, the agent will have equally likely probability to move to any path sensed by it. This method is very inefficient and results in exploration of the same region over and over again. For a simply connected maze, the “follow the wall” algorithm, often referred to as the left hand rule or right hand rule, the agent follows a path where the same wall is always on one of its sides will find the exit or return to the start. This method works well when the beginning is the entry point of the maze but is not suitable when the starting position of the agent is any point within the maze. The pledge algorithm is used to find a solution out of the maze but does not help in exploring the maze. To use the pledge algorithm, an exploration algorithm is required that will be followed by pledge algorithm and a well defined mode change is required. Tremaux’s algorithm is an efficient method for finding the way out of a maze. It requires marking a path that it has already followed, and is guaranteed to work for all mazes with well defined passages. When the agent arrives at a marked junction, the algorithm enables the agent to take up an unmarked path if possible. If it is not possible to pick an unmarked passage then the agent should take a marked path, marking it again. The agent shouldn’t pick a path marked twice. If there is no exit, this method will take the agent back to the start. Tremaux’s algorithm effectively implements a depth-first search\textsuperscript{3}. This method works well for single robot cases but is problematic in multi robot group.

In this effort we solve the exploration problem in a maze using modified ant colony optimization algorithm (counter ant algorithm)\textsuperscript{4} and fuzzy logic\textsuperscript{5} modeling. The ant colony optimization algorithm is a probabilistic technique for solving computational problems which can be reduced to finding potential paths through graphs. In the ant colony optimization technique, the probability of an ant to follow a particular path increases with the increase of deposited pheromone concentration (a chemical substance excreted by the ants). More pheromone concentration indicates that more ants have followed the path. In an exploration problem, it is desired not to explore a region that has already been explored by any other agent. A modified version of ant colony optimization, Counter Ant Algorithm (CAA), deals with robot’s collaborative behavior based on repulsion than attraction to pheromone concentration i.e. the more the concentration of pheromone, the less the probability of choosing that particular path. The robot’s reaction consists henceforth in avoiding paths more covered by this chemical substance. Fuzzy logic can be used in modeling this type of ant’s behavior. This is because most of the animal’s or insect’s action can be stated as collection of If-Then rules expressed using natural language. Studies of this behavior often provide a verbal description of both field observation and interpretation. Rozen and Margaliot\textsuperscript{6} provide enough insight of fuzzy modeling to develop a mathematical model for the foraging behavior of the ants.

In this paper, the simulation results of solving the maze problem with counter ant algorithm using fuzzy logic modeling are shown. A brief description of the methodology is presented in the next sections along with the fuzzy inference system. The maze considered here changes randomly in each simulation. The scalability of the approach is also investigated. The developed methodology will later be extended to dynamically changing maze along with its implementation in an experimental multi agent system to solve a laboratory based maze problem.

II. Methodology

The task is defined as follows: Given a maze of the type shown in Fig. 1 (which may vary in size, number of dimensions or the configuration of the obstacles), find an appropriate algorithm which provides a common decision support system between multi-agents and moves agents from their initial position (start point) to a target position (end point), while minimizing the convergence time which the robot takes to find the solution. Initial work on the problem is done by simulating the movement of one agent through the maze using a technique which was similar to the “Carrot or Stick” approach of reinforcement learning. This technique can be explained as assigning rewards and punishment for the outcome of a task. This approach has been used by various researchers for developing algorithms in the field of evolutionary robotics\textsuperscript{7}. A similar approach was used for the maze problem where the reinforcement mechanism is based on the numerals “1 or -1”. The area of maze is being divided into square cells with each cell having an initial value 0. The number of these cells depends on the size of maze required. Whenever the robot makes advances for exploration, the value of each cell being advanced is replaced with number 1 whereas if the same move leads to a blocked path, the numeral value of the all the cells, which are part of that path, are being
replaced by -1. These values are stored in the form of matrix of size similar to the number of cells. This matrix is called Agent Value and it acts as memory for the further exploration of maze in a way that if the robot or agent happens to come to the same cell again, then the value -1 would prevent the robot repeating the same “mistake” again and hence reduces the convergence time.

Figure 1. The maze problem: find the path which will lead the agent 1 and 2 from the present position to end point.

Figure 2(a) shows the simulation result for single agent exploring the maze. As the agent starts moving from cell to cell, the value of matrix gets updated with the value of corresponding cell in the maze so that at every junction, where the agent has to take a decision among the directions available, it can consult values of immediate cells in those directions. Preference is always given to direction having cell value ‘0’. If any of the direction doesn’t have zero value, then the agent moves in the direction where the value of cell is ‘1’, which denotes that the way ahead is blocked and updates the values of those cells as ‘-1’. Here the value ‘-1’ denotes the punishment so that if the agent happens to come to same cell again it should not repeat advancing to same cell again. Fig 2 (b) shows the final agent value matrix representing the values of each cell of the same maze. It can be easily seen on observation that matrix cells having value ‘1’ in figure 2 (b) denotes the solution of the maze from the initial point of agent 1 to the final point.
Once the result for single agent has been obtained, next stage was to scale this concept to two agents so as to develop a common decision support system between them and also to reduce the convergence time. Same concept was scaled by adding another agent in the matrix so that both agents can explore the maze in most optimized manner by avoiding the repetition of the already explored path. But there arose some cases when both the agents happened to came across each other. In such cases the agents were unable to find the immediate cells having cell value ‘0’ because of which they had to chose the cells having value ‘1’ in either directions and hence the value of those cells were updated to ‘-1’, therefore closing that path for future advances as well. Fig 3 shows one of such cases. In this figure it can be seen that even if the agent ‘1’ or agent ‘Red’ goes straight, due to the cell value ‘1’, the ‘2’ agent or the ‘blue’ agent will have to choose a path having cell value ‘1’, thus updating the value of that particular cell to ‘-1’. In any of the cases, one of the two agents will get confined in that particular region only and hence will not be able to explore the maze further.
Thus for the purpose of developing a decision support system we need a better organized and more structured algorithm that could overcome the problems faced in initial algorithm. Therefore we decided to incorporate a hybrid algorithm which has the key points of old approach and the “Counter Ant Algorithm” [4]. Ant colonies, and more generally social insect societies, are distributed systems that in spite of the simplicity of their individuals, present a highly structured social organization. As a result of this organization, ant colonies can accomplish certain complex tasks. For mass foragers such as ants, it is important that all foragers reach a consensus when faced with a choice of paths. While travelling the ants deposit a chemical substance called pheromone along the path and the concentration of pheromone increases with the increase in the number of ants travelling through the same path. When faced with a choice of path, the probability of choosing a path is higher where the concentration of pheromone is high. In this process the ant colony as a whole is able to obtain the optimum path from the nest to the food. This positive feedback is counteracted by negative feedback due to pheromone evaporation in the paths other than the optimum path.

For our exploration strategy, we use the Counter Ant Algorithm [4] where the agent chooses its path according to the less pheromone concentration i.e. probability of choosing a path with lesser pheromone concentration is higher than the other paths. This method enables the agents not to explore a region that is been already explored by some other agent and in the process reduces convergence time.

Given two paths the function suggested by Goss et al. [13] and Deneuborg et al. [14] is,

\[
P_{n,k}(L,R) = \frac{(k + L)^n}{(k + L)^n + (k + R)^n}
\]

Where, \(P_{n,k}(L,R)\) is the probability of choosing the left branch than right and “n” and “k” are the degree of non linearity and the degree of attraction attributed to an unmarked branch respectively. \(L\) and \(R\) represent the concentration of pheromone in left and right branch respectively. Researchers in [6] have shown how fuzzy logic modeling can be used to model the probabilistic ant’s foraging behavior and have shown in their simulation results, the effectiveness of the model.

While solving the maze problem, we use the counter ant algorithm technique and model the probability model using fuzzy logic technique for its simplicity, robustness and its ability to incorporate “IF-THEN” rules. Similar to the “Carrots or Stick” approach, the pheromone concentration has been replaced by numeral value ‘1’ i.e. as the agent moves from one cell of maze to another, instead of depositing a chemical, it will update the value of cell with value ‘1’. However, unlike the old approach, the concept of ‘-1’ is being replaced by numeral ‘3’ which depicts the action of ants depositing more pheromone on the blocked path so as to avoid that path in future. Preference will be given to the path having less pheromone concentration. Now, given two branches with pheromone concentration \(L\) and \(R\), the input to the fuzzy inference system (FIS) is the difference between the pheromone concentrations of the two branches,

\[
D \text{ (the input)} = L - R
\]
The rules are:

- If $D$ is positive (i.e. concentration $L$ is higher than $R$) Then $P(L,R) = 0$
- If $D$ is negative (i.e. concentration $R$ is higher than $L$) Then $P(L,R) = 1$

Here, $P(L,R)$ is the probability of choosing “$L$” (left) than “$R$” (right).

In a maze, at most there can be four choices for an agent to move as shown in the fig 4.

![Figure 4. The choices of paths for an agent](image)

As a result, we have six FIS (Fuzzy Inference System) functions i.e. FIS$_{12}$, FIS$_{13}$, FIS$_{14}$, FIS$_{23}$, FIS$_{24}$ and FIS$_{34}$ (Where subscript 12 etc. denotes Fuzzy Inference System for choosing path 1 or 2). All the FIS are similar having one input i.e. the difference of concentration between the two paths, and one output, the probability of choosing a path over the other. As an example, for FIS$_{12}$, the input is the difference in immediate cell values for paths 1 and 2. The output will be the probability of choosing 1 than 2. Since the input can be negative or positive, the membership functions should monotonically increase from zero and reach the saturation level. For our case we use sigmoid functions. The output used is Sugeno i.e. one or zero. Below shown are the FIS and the membership functions.

![Figure 5. The Fuzzy Inference System (MATLAB – Fuzzy Logic Toolbox)](image)
Hence the probabilities of choosing the paths are as:

Path 1 = mean \([P_{12}, P_{13} \text{ and } P_{14}]\)

Path 2 = mean \([(1 - P_{12}), P_{23} \text{ and } P_{24}]\)

Path 3 = mean \([(1-P_{13}), (1-P_{23}) \text{ and } P_{34}]\)

Path 4 = mean \([(1-P_{14}), (1-P_{24}) \text{ and } (1-P_{34})]\)

Output =Max \([ \text{Path 1, Path 2, Path 3, Path 4}]\)

Stagnation condition is a common case that can arise when all the paths, surrounding the agent, has already been explored by other agents. The pheromone concentration in all those paths will be high and equal. This condition is taken care of by incorporating the pheromone evaporation where the concentration of pheromone will decrease with each time step. The pheromone evaporation model can be considered as,

\[
Pheromone \text{ Concentration (t+1)} = Pheromone \text{ Concentration (t)} - s^* \text{(Time step)}
\]

The parameter “s” determines the pheromone evaporation rate which depends upon the size of maze.

### III. Results and Discussion

Final simulation results have been shown in Figs. 7-8. Fig. 7 gives a demonstration of the effectiveness of the decision making algorithm developed above. At the decision point, marked in the maze below, the ‘Blue’ agent had two ways to go i.e. it could have followed the ‘Red’ agent but as per the algorithm developed it decided to avoid the already explored path and chose to explore the other path so as to minimize the convergence time. Now considering the stagnation case when the ‘Red’ agent would come back to the marked decision point, due to pheromone evaporation process the cell value on marked point will be less as compared to the present cell of the agent due to which probability of choosing the marked cell will be more as compared to going back. Fig. 8(a) provides the demonstration of the final simulated solution to the desired maze problem. In the figure 8(b), the red line corresponding to the numeral values less than ‘1’, indicates the solution from initial position of agent ‘Red’ to the final exit point. On the other hand, the blue line corresponds to numeral values less than ‘1’, which indicates the solution from initial position of agent ‘Blue’ to final position.
In order to analyze the successful behavior for a cooperative decision support system between two agents, the algorithm being developed was tested for different maze sizes. The number of steps involved in finding the solution as well as getting both the agents out of the maze is preserved. In order to obtain a statistical insight into the effectiveness of this algorithm, the MATLAB simulation is run 250 times. Each time a random, unique and different maze pattern is generated. The algebraic mean of all 250 runs is then plotted against the maze size. Fig. 9 shows the plot of number of steps and maze sizes for two agents. The maze size increases from 5x5 to 50x50 and the total number of steps to find the convergence (i.e. maze solution) increases sharply.
In Figure 10, the algorithm is being repeated for different maze sizes and different number of agents. The steps being involved in finding the solution for pioneer agent is observed. The mean of the 250 iterations is then plotted against the number of agents for each maze size. This figure of merit provides insight into the algorithm’s performance by observing the trends concerning increase in number of agents for a particular maze size. Furthermore, the simulation is being run for five group sizes varying from a single agent to five agents, and the maze size varying for 15x15 to 50x 50. The total number of steps taken to get the pioneer agents agent out of the maze are counted and then the average value is taken by dividing each value by the total number of agents. Fig. 10 shows the plots of the average steps to the number of agents. It is interesting to see from the figure that, as the number of agents increases for every particular maze size, the average steps travelled by a single agent to find the solution decreases sharply as one would expect i.e. with the increase in number of agents in a group the average contribution of an agent decreases sharply.
It is interesting to see the performance of the developed algorithm for different maze sizes and different number of agents in a group. Ordered pairs are being formed by running each group of agents for each maze size. Each pair was run for 250 times in random pattern maze generation. The total number of steps required for pioneer agents to find the solution is counted for each run and the mean of those iterations is taken as an output number. Such results for the ordered pairs are then plotted against number of agents for each maze size. Fig. 11 shows the plots of mean number of steps for a group and the number of agents in that group, ranging from 2 to 10, for different maze sizes varying from 8 x 8 to 40 x 40. Looking at these plots, observe that for each maze size the total number of steps decreases sharply with increase in number of agents, which reaches a minimum value and then increases with an increase in number of agents. Thus adding an agent to the group increases the efficiency to explore the maze, by exploring larger area of the maze, and decreases the convergence time for pioneer agent. However, for a constant maze size, as the number of agents increases, the total number of steps involved in order to get them all to the solution also increases because as the number of agents increase there is more repetition of already explored paths in order to avoid collisions and also because of the direction decisions of each agent.

The scalability of the solution can be observed from two different perspectives i.e. scalability in terms of maze size and scalability in terms of group size. It can be seen from Fig. 11, that all groups are capable of finding the solution for any maze size. Considering the fact that as the maze size increases, the number of cell also increases, because of which the complexity as well as combined time taken by all the agents to get out of the maze also increases. Thus after a particular time period, the pheromone concentration gets totally evaporated leaving no trailing edge behind and thus increasing the total number of steps involved. Also with increase in number of agents in a group the probability of repetition of already explored paths also increases, which also contributes to increase in total number of steps as well as time taken.

Figure 10. Plot of Average Steps to Number of agents
Figure 11. Results showing the scalability of the algorithm
IV. Experimental Validation

The fuzzy ant maze exploration system enables multi agents to explore an unknown terrain in an unsupervised way. In the experimental setup presented in Figure 12, the off-the-shelf robots\textsuperscript{15} are being used as agents to explore a generalized maze test bed. The maze being setup in the lab with the aim that we can easily adapt its pattern so as to match the simulation conditions where the maze pattern changes in a random order. The robots being used in the lab can emulate the performances of the simulated agents using multiple sensor arrays for both long and short range object detection. These robots include an array of 9 Infrared Sensors for obstacle detection as well as 5 Ultrasonic Sensors for long range object detection. The algorithm is run at a remote workstation, which is connected to all the robots using a wireless communication card. Data from the sensors is collected and sent back to the remote workstation using a linux based interface named “player”. This interface provides a means to interpret that data and utilize the same to take the required decision for that respective robot, from which it came. Fig. 12 shows the experimental setup in lab where in 3 robots are trying to find the solution of the maze utilizing the above discussed algorithm.

![Figure 12. Actual Experimental setup in the lab](image-url)
V. Conclusions and Future Work

This paper presents the fuzzy ant methodology developed for exploration and exploitation of mazes. The results are based on MATLAB simulation and an overview of the experimental system is described. Different figures of merit have been plotted to provide into the behavior of the algorithm for different scenarios. Also these figures of merit have been useful in providing important information about the optimal number of the agents in a group required for solving the maze problem. The scalability of the algorithm has been shown for different maze sizes as well as different agent groups. Future work includes modification of the algorithm to ensure improved scaling characteristics (maze size as well as number of agents). Moreover, a comparison will be made with alternative techniques to gauge the effectiveness of the developed algorithm.

Acknowledgments

The authors would like to acknowledge the fruitful discussions with Mr. Elad Kivelevitch and the support of Mr. Curtis Fox in preparation of the experimental system.

References