Najla Ahmad
May 22, 2018

Capstone Proposal

1. Domain Background

In a multi-agent system, agents work together to complete tasks. For example, an example of agents working together is a Sensor Web, such as one the one developed for NASA[1], where agents work in coalitions to determine if a weather phenomenon is occurring or not. Another example of agents working together is given by Shoham and Leyton-Brown (2009)[2]. They researched a sensor network, in which each unit has sensing capabilities and limited processing power. Individually, each sensor builds a local view of what is occurring around them. Working together, these agents can build a global view of the environment.

For my PhD thesis, I worked on intent recognition in multi-agent systems[3]. Idle agents that were otherwise not doing anything would observe other agents in the system, try to recognize what they were trying to achieve, and then attempt to assist them with their tasks. These agents chose between set plans, or a list of instructions of how to behave. During many of my Udacity lectures, I would think about how the agents I created would be much more intelligent if they used machine learning concepts.

A classic way to see how well agents are working together is to have a team of agents attempt to herd other "animal" agents. This can be seen in the Microsoft Malmo Collaborative AI Challenge[4] and in several years of the annual Multi-Agent-Programming Contest[5].

2. Problem Statement

The animal herding domain is a well-known problem in multi-agent systems. It is a good test of the agents' abilities because, as the Multi Agent Programming Contest states: "The contest consisted of applying (or developing from scratch) a multi-agent system to solve a cooperative task in a highly dynamic environment." Animal herding simulations test agents' abilities to simultaneously work cooperatively with their teammates and competitively with the opposing team.

In the specific cow herding contest, two teams of agents face off in a grid-like world. Cow agents move around in a swarm-like way while each of the two teams try to corral the most cows on their side.

[1] Tsatsoulis, C., N. Ahmad, E. Komp, and C. Redford (2008). "Using a Contract Net to Dynamically Configure Sensor Webs," IEEE Aerospace Conference, Big Sky, MT, 1-6.
[2]Shoham Y. and K. Leyton-Brown (2009). "Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations." Cambridge University Press, Cambridge, UK.
[3] Intent recognition in Multi-agent Systems: Collective Box Pushing and Cow Herding. N Ahmad (2013)
[4] "The Malmo Collaborative AI Challenge" https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge/
[5] "Multi Agent Programming Contest" https://multiagentcontest.org/2010/

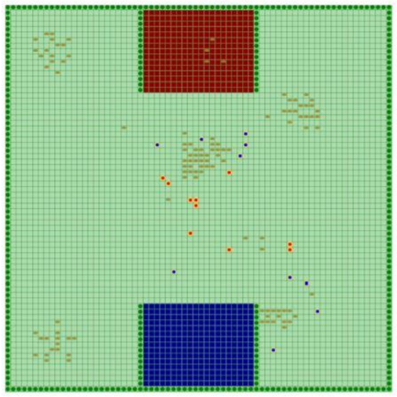


Figure 1 Example image from 2008 cow herding scenario document (Dix et al., 2008)[6]

In the first year that the cow herding scenario was used, the score was calculated by the number of cows that were brought into the corral. Once a cow entered the corral, it was not able to leave. After the first year, the score was calculated as the average number of cows in the corral instead. In this new version of the scenario, cows could repeatedly escape from the corral and be herded again until the simulation reached the maximum time limit. To calculate the score for this version, at every timestep, the cows in a team's corral are counted and added to the sum for all of the previous timesteps. The final sum is then divided by the number of steps, which yields the final score for that team. For my PhD work, I used the first version of the score calculation.

It is also easy to compare agent reasoning capabilities in this scenario by simply seeing which team wins most often. I want to implement four different teams. These are: two teams with different machine learning algorithms, one random or poor reasoning team, and one that follows a strategy followed by one of the winning teams from the multi-agent contest.

By adding machine learning agents to the cow herding scenario, I will be able to examine reinforcement learning in this dynamic and cooperative yet competitive environment.

3. Datasets and Inputs

For this project, I would build a version of the cow herding simulation. Inputs to the agent will be the environment state, which includes the location of all the cows, agents, and goals. The reward for each state will incorporate the score. The score will be calculated the same way as it is calculated in the Multi-Agent Programming Contest as noted above. The agent can use this state and reward to perform reinforcement learning. The project will be scaled down from the original cow herding simulation due to time and processing constraints.

4. Solution Statement

In order to study reinforcement learning in a dynamic multi-agent environment, I will create a cow herding environment in Python. It will be based off the environment detailed in the Multi Agent Programming contest and my PhD work. It however will be a scaled down version in terms of grid size and number of agents per team. This is because of time constraints for the capstone projects and processing power since machine learning agents will require more processing power than the agents I worked with previously.

It is easy to compare agent reasoning capabilities in this scenario by comparing team scores. Four different teams will be implemented: two teams with different machine learning algorithms, one

[6] Dix, J., M. Köster, and F. Schlesinger. (2013). "Multi-Agent Programming Contest." Multi Agent Programming Contest. Retrieved July 30, 2013, from http://www.multiagentcontest.org.

random or poor reasoning team, and one that follows a known and previously studied strategy followed by one of the winning teams from the multi-agent contest[7].

I will first implement a team using ε-Greedy Monte Carlo methods. If time allows I will also implement a team using temporal difference methods.

5. Benchmark Model

For the first part of the project, reinforcement learning agents will be an in environment with no opposing agents. There will be a group of cows and one corral. This will be to see what scores the agents are able to achieve without the added complications of opposition.

The scores of the reinforcement learning agents will be compared to two other agent groups. The first group being a team of randomly acting agents. This will be the example of poorly functioning agents. The second group will be a team that is modeled after the JIAC V team by Heβler et al. (2010) whose work is referenced in a previous section. These agents won the cow herding competition in 2009. In this plan, agents herd individual cows into their team's corral. Agent's select cows to herd based on both the agent's distance to the goal and the cow's distance to the goal. This will be an example of high functioning agents.

To compare the learning agents with the other groups, the other two agent types will also be run in the environment with no competition to record their scores. The scores can then be compared to see if one team performed statistically better than the others.

If time allows, the agents will be put in competing teams in an environment with two corrals to see which team wins the most competitions.

6. Evaluation Metrics

The cow herding simulation will be run with the random agents, then the plan recognition agents, and finally the reinforcement learning agents. The score will be calculated over multiple episodes. In the end, the average scores along with the minimum and maximum score values will be used for evaluation. Also, the scores will be compared in earlier episodes versus later episodes to see if the reinforcement learning agents improved with experience. If all cows are herded before the time limit runs out, the time can be incorporated into the evaluation as well.

Statistical analysis will be done to determine whether the team scores are significantly different with an alpha level of 0.05.

7. Project Design

The cow herding simulation will be built in Python using Mesa[8]. According to the description, "Mesa is an Apache2 licensed agent-based modeling (or ABM) framework in Python. It allows users to quickly create agent-based models using built-in core components (such as spatial grids and agent schedulers) or customized implementations; visualize them using a browser-based interface; and analyze their results using Python's data analysis tools. Its goal is to be the Python 3-based alternative to NetLogo, Repast, or MASON." This is the Python alternative to the Java based Repast which was used for my PhD.

This project will be a pared down version of my PhD thesis work. A screenshot of that environment can be seen in the figure below.

[7] Heßler, A., Hirsch, B., and T. Küster. (2010). "Herding cows with JIAC V." Annals of Mathematics and Artificial Intelligence, 59(3-4): 335-349.

[8] "Project Mesa" https://github.com/projectmesa/mesa

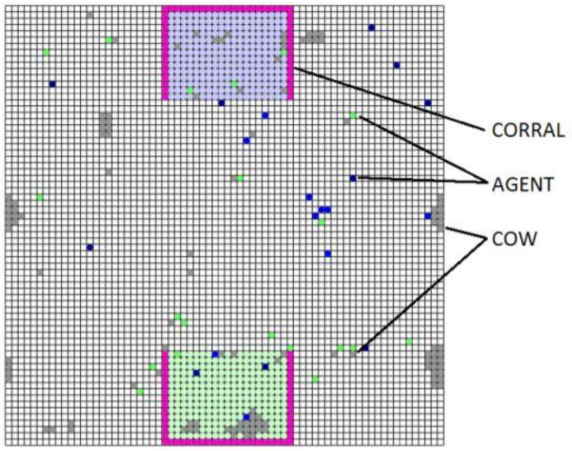


Figure 2 Cow Herding Simulation (Ahmad, 2013)

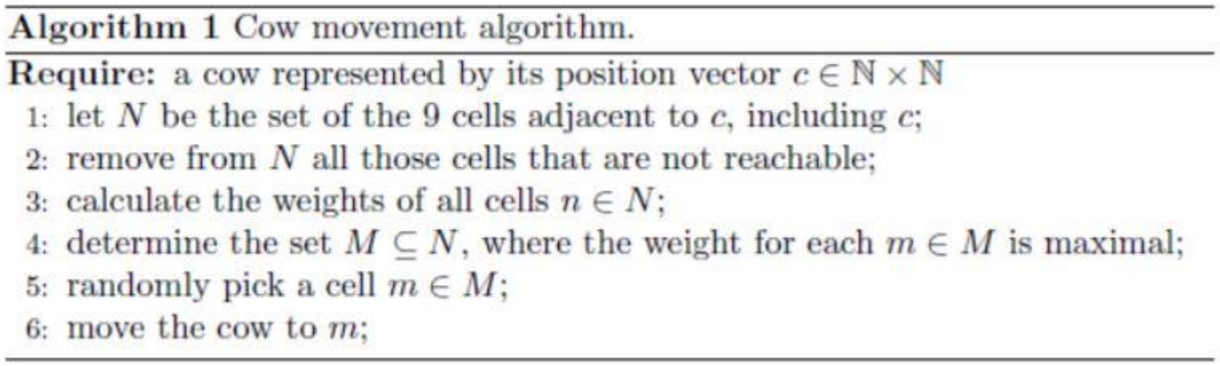
The agent environment consists of a grid world. Some grid areas are designated to be corrals. Two agents cannot occupy the same grid space at the same time.

The environment will originally be smaller than the original environment guidelines and will be scaled up to see the effect on episode time. In this way, the environment size will be set through trial and error. I will begin with small teams of two agents per team and increase this value slowly to see the maximum number of agents that are feasible to run at one time.

In the initial phase, there will be one team in the environment at a time and one corral. This will compare how well the agent teams perform the cow herding task. The team scores will be statistically compared with each other. If time allows, there will be a second phase with two teams and two corrals to study how the teams perform in a competitive setting. In this phase, we can see which team wins the most number of times.

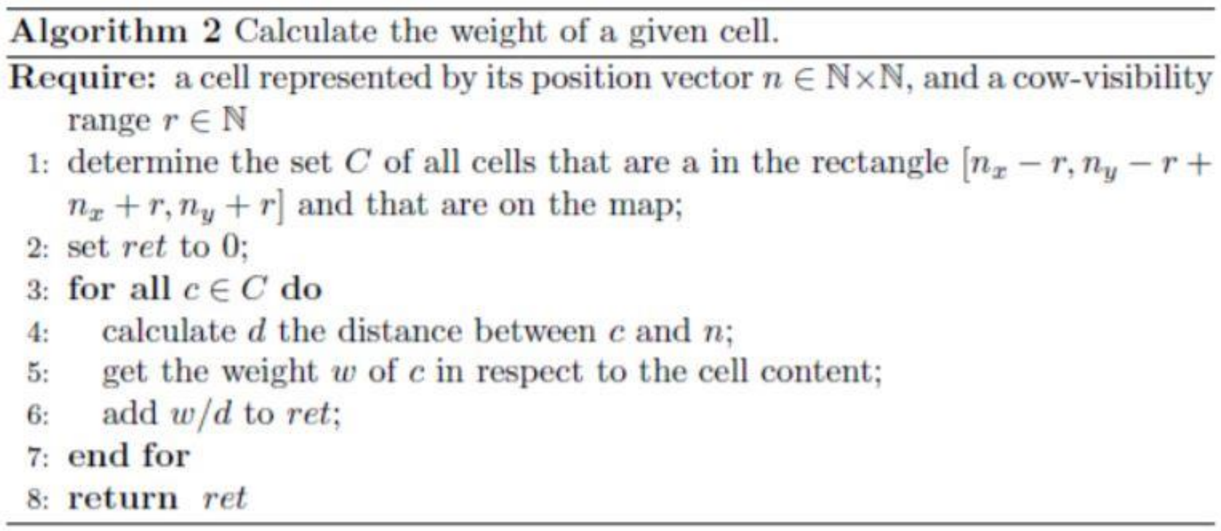
The reinforcement learning agents will use ε-Greedy Monte Carlo methods for learning since the episodes are finite. If time allows, I will also implement a team which use Temporal Difference (TD) methods.

The cow agents behave the same way as the cow agents in the Multi-Agent Programming contest. The algorithm is detailed in the figures below. This algorithm leads to a weak swarming behavior.



```
Algorithm 1 Cow movement algorithm.
Require: a cow represented by its position vector $c \in \mathbb{N} \times \mathbb{N}$
  1: let $N$ be the set of the 9 cells adjacent to $c$, including $c$;
  2: remove from $N$ all those cells that are not reachable;
  3: calculate the weights of all cells $n \in N$;
  4: determine the set $M \subseteq N$, where the weight for each $m \in M$ is maximal;
  5: randomly pick a cell $m \in M$;
  6: move the cow to $m$;
```

Figure 3 Cow Algorithm 1 located in 2010 MAPC scenario document (Dix et al., 2010).



Algorithm 2 Calculate the weight of a given cell.

```
Require: a cell represented by its position vector $n \in \mathbb{N} \times \mathbb{N}$, and a cow-visibility
    range $r \in \mathbb{N}$
1: determine the set $C$ of all cells that are a in the rectangle $[n_x - r, n_y - r + n_x + r, n_y + r]$ and that are on the map;
2: set $ret$ to 0;
3: for all $c \in C$ do
4:    calculate $d$ the distance between $c$ and $n$;
5:    get the weight $w$ of $c$ in respect to the cell content;
6:    add $w/d$ to $ret$;
7: end for
8: return $ret$
```

Figure 4 Cow Algorithm 2 located in 2010 MAPC scenario document (Dix et al., 2010).

The poorly functioning agents will be following a random plan. At each timestep, they will move to an empty grid location adjacent to the current location. This will continue until all cows are herded or until the maximum number of timesteps is reached.

The benchmark agents will be using a plan based on a plan which won the Multi-Agent Programming contest in 2009 and referenced above. These agents scan the environment to find the closest cow to their goal. They then herd the cow by first moving so that the cow is between the agent and the goal, then moving towards the cow. This part of the plan repeats until the cow enters the corral. The agent then looks for the next closest cow and repeats the process. This will continue until all cows are herded or until the maximum number of timesteps is reached.

This set of experiments will evaluate machine learning in a dynamic multi-agent system. If time allows, the agents will be compared in both a cooperative and competitive environment.