Issues in Information Systems
Volume 24, Issue 1, pp. 186-200, 2023

DOI: https://doi.org/10.48009/1_iis_2023_116

A study of emergent cooperative behaviors of multi-agent systems using reinforcement learning

Joshua Brown, *Carson-Newman University, jsbrown1061@cn.edu*
Seongyong Hong, *Carson-Newman University, shong@cn.edu*
Alden Starnes, *Carson-Newman University, astarnes@cn.edu*

Abstract

Multi-Agent systems consist of multiple interacting agents that solve complex problems too difficult for monolithic systems to solve. Each agent has a different role and is not aware of the state of other agents. Multi-Agent Reinforcement Learning is a common approach to training agents in these systems. Many researchers tend to focus on the implementation of reinforcement learning for training agents towards a particular goal. This paper, in contrast, implements reinforcement learning for a different purpose. This paper is about the analysis of emergent cooperative strategies between agents while achieving their goals. The research will be done using simulations of a cooperative game and documenting the strategies the agents create over a set number of episodes.

Keywords: multi-agent system, reinforcement learning, game theory, nash equilibrium

Introduction

The creation of intelligent and autonomous artificial agents to solve problems like humans have done for centuries is a current challenge in artificial intelligence research. This is especially apparent in the use of AI as an additional utility to modern computing systems. Some examples include upscaling images, recognizing faces and personal assistants like Siri. Siri, for instance, is a step towards human AI interaction towards accomplishing tasks like setting reminders and playing music. Multi-agent systems allow multiple agents to work together towards accomplishing tasks. They are particularly useful for traffic systems, manufacturing, logistics and games. The training of agents in multi agent systems is done via reinforcement learning within a closed environment. An explicit goal is given for the agents to complete, who then train until they acquire the skills and knowledge to accomplish the goal. However, the learned skills are inherently bounded by the given goal. After the goal is achieved there is little necessity for improvement; yet the goal requires individual agent skill and optimal cooperation amongst agents. However, there can be many ways to achieve a goal. Some ways are more optimal than others, but the agents ultimately achieve their goal.

This study hopes to answer these questions: How do multi-agent systems develop their emergent cooperative strategies for success? What changes in cooperation are made as the team and environment changes? Success is achievable without agent coordination. Even in those scenarios, “cooperation may improve the performance of the individual agents or the overall behavior of the system they form” (Kraus, 1997, p.80). Important concepts for the analysis of cooperative behavior within a limited environment are Game Theory and Nash Equilibrium. Game theory is a form of study concerned with the analysis of strategies in cooperative or competitive situations where the outcome of a participant's choice of action affects the actions of others. This is used not only in games but in Mathematics, Business, War and Biology. Game Theory is important for multi-agent systems since it allows one to determine how the actions of one agent affects the actions of the others. This, in turn, will influence how and when their collective goal is achieved. It is especially apparent when agents compete or form coalitions amongst each other. The application of game theory for multi agent systems is typically in a limited simulation environment. This allows researchers to make assumptions about the nature of their environment and consequently limits all possible actions of the agents. Given specific assumptions about the environments, game theory researchers identify strategies that are in equilibrium (Kraus, 1997, p.84).

Nash Equilibrium is the point in game theory where equilibrium is reached due to unchanging participant actions. For example, this can happen when two sports teams are evenly matched. Both sides are great offensively but are equally great defensively which results in no points for either team. This in turn causes a Nash equilibrium. Nash Equilibrium is also commonly found during two-player zero sum games. However, one can find it within cooperative settings when one considers all possible strategies within the environment. Although it can be exponentially more difficult depending on the number of cooperating agents. This is why much of the research for agent interaction is done through Markov Games. A Markov Game is a game that takes place in a sequence of stages. Whenever an agent performs an action, it creates another state in the game. Suppose there are multiple people playing a card game. After the cards are evenly distributed, the game is in a start state. When someone draws a card, the game moves to a different state where there are less cards in the deck. Agent interactions are analyzed in a similar manner within the Markov framework. This also applies when agents compete or cooperate to accomplish various tasks.

Although this study mainly focuses on cooperation amongst agents. Researchers like Angel Sánchez have explored human cooperation through social experiments and physics. Agents can cooperate like humans but with differing incentive structures. Agents have explicit incentives often of numerical value. Humans, in contrast, can cooperate given implicit and explicit rewards. They may work towards a goal without knowing if they will get a reward from it. Nor will they ever know the potential rewards of their neighbors (Sánchez, 2018, p.5). Agents having explicit rewards may limit the scope of their development but may not limit the cooperative strategies that they create. This study will explore these strategies and how many are created within a given environment.

Related Work

Previous studies on multi-agent systems mainly tend to focus on how agents achieve the optimal path towards achieving their objective. A dissertation by Minsuk Kim gives multiple case studies for intelligent multi-agent systems: agents with differing goals sharing information across a network, the improvement of multi-agent system performance by decreasing the amount of random exploration steps and implementing agent sub-goals and reinforcement learning reward schemes with a proposed environment-cluster-based-index (Kim 2016). This dissertation also outlines an automatic and asynchronous triggered multi-agent planning scheme. Kim's maze simulation shows promising results with the agents finding the optimal path after one episode of random exploration.

An alternative framework that optimizes agent progression towards an optimal result is the Cooperative Multi-Goal Multi-Stage Multi-Agent Reinforcement Learning (CM3) framework (Yang et al., 2018). The CM3 framework hopes to address the problems of current multi-goal multi-agent control scenarios: the inefficiency of random exploration in reinforcement learning and lack of proper credit assignment for single joint goals that reflect agent interactions. It is first done by training an actor-critic pair that achieve different goals in an induced single agent setting. This setting is formalized as an episodic multi-goal Markov game. CM3 achieves multi-agent credit assignment using a counterfactual baseline in an advantage function. A centralized critic then uses this function within its policy gradient for its multi-goal Multi-Agent

Reinforcement Learning (MARL). This is common within environments that contain cooperation and competition. A centralized critic will make leading a team towards more optimal decision making (Lowe et al. 2017). The curriculum for CM3 involves solving the single agent Markov decision process to then speed up future random explorations. However, there are issues that these studies and others in this area of research fail to address. Many do not properly explain the meaning of "exploratory phase". It can mean the first few time steps, or all the episodes up to achieving their first successful result. They also fail to determine how agents receiving good or bad information influence the results of the simulations.

This study will mainly focus on how agents react within the environment on their path towards success. One example study by Ivan Kiseliou focuses on emergent forms of communication between agents and how it improves goal achievement (2020). Kiseliou also incorporates a linguistics perspective, factoring in how communication is shaped by interaction and one's immediate environment. This study uses a variation of the card game, The Mind, as a form of study. Agents generate singular symbol messages via multi-layer perception (MLP) layers to best describe the state of their hand of cards. The result of the game varies little depending on the length of the message. Communication did not only improve agent coordination but accelerates the learning process. OpenAI's competitive multi-agent hide and seek game using standard reinforcement algorithms outlines how even following a simple objective can cause agents to develop various strategies through their own trial and error (Baker et al., 2020). The agents implicitly create curriculum for themselves through reacting to other agents amidst competition. The learning of various actions, such as tool use, is not explicitly rewarded through its credit system like in other studies using intrinsic motivation. Instead, the agents implicitly develop these skills over time resulting in new strategies.

However, there are also weaknesses with the study of agent environmental interaction. An important one is that the environment must be randomized for the curriculum to be robust. This makes it difficult for skills to transfer to other environments, as seen in OpenAI's skills-based testing after the hide and seek game. Also, the number of possible actions exponentially increases as dimensionality increases. This exponentiality adds to the difficulty of visualizing the results for agent-based approaches. Visualization is a critical part of the methodology considering the complexity of agent actions and interactions (Elliott and Kiel, 2002, p.1). The easiest way to see the agents' progressive learning process is through diagrams or other visual aids. Overall, this study will incorporate the optimizations from goal achievement research while adding to agent environmental interaction research.

Methodology

The analysis of the agent movements is done using a Proximal Policy Optimization (PPO) algorithm with the default Multilayer Perceptron (MLP) Policy from the Python package stable_baselines3. The MLP Policy and PPO algorithm will be further explained in the following two sections. The model that will be created using these two components will gather data from 5000 timesteps. A timestep is a unit of time that signifies an action was taken by an agent. For example, if an agent moves left then right, it would be two timesteps. Afterward, there will be 100 test episodes for the model to see its success rate. The learning rate for the model will be 0.01. The learning rate will determine the rate at which a model learns an environment. If the learning rate is too high, the model will find an equilibrium value higher than the optimal value. If the learning rate is too low, then it will take much longer to maximize the rewards of an environment; thus, making the model take longer to reach Nash equilibrium.

Multilayer Perceptron Policy

A Multilayer Perceptron is a neural network architecture that uses perceptions to read inputs, gather output and find the optimal value based on the output. A perceptron is a single unit that takes input and predicts an outcome. This is a similar process used in the stock market. Wall Street often takes input from past years in the stock market to predict how the market develops in the future. Weight is given to different stocks based on perceived values. Perceptrons perform a similar calculation. Perceptrons predict their outcomes by determining input weights, performing a linear combination based on those weights and placing the output through some nonlinear function which returns a predicted value $\hat{y}$. Each action input vector x will have an associated weight vector w and the optimal value comes from a linear combination of those vectors. The bias vector b is then added to that linear combination. The value of a nonlinear function Θ will be between 0 and 1 with 1 being the optimal value. The output of a perceptron can be expressed as

$$\hat{y} = \Theta(w_1x_1 + w_2x_2 + \ldots + w_nx_n) + b$$

which can be rewritten as

$$\hat{y} = \Theta(w * x) + b$$

where

$$\Theta = \begin{cases} 1, & y = \hat{y} \\ 0, & othewise \end{cases}$$

Figure 1 provides a visual representation of how a neural network architecture like MLP operates. Neural networks in general are useful for analyzing and learning nonlinear behavior, capturing all variations of dynamic systems under all conditions (Al-Assadi, 2007, p.659). MLPs have been used for many nonlinear dynamic systems such as controls for electronic throttle systems. The MLP policy in stable_baselines3 uses two layers of sixty-four perceptrons and implements actor-critic. Actor – Critic is a temporal difference policy gradient method. The actor decides which actions should be taken with the critic telling the actor how good an action is and how to potentially adjust to perform more optimal actions.

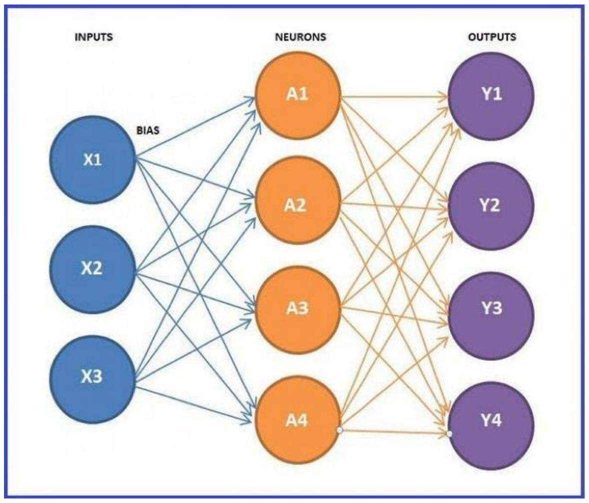

Figure 1: A diagram representing how perceptrons operate

Proximal Policy Optimization

Additionally, a PPO algorithm with the given MLP policy will allow the machine learning model to learn the environment at a faster and more effective pace. A PPO algorithm, unlike other methods, implements policy gradient methods by interacting with the environment and analyzing the rewards from all the previous timesteps after a given number of epochs. In essence, the algorithm gathers the data from the current action of an agent and all previous actions to best predict an outcome. The algorithm for PPO with given actions θ_t for a fixed amount of timesteps T (Schulman et al., 2007, p.5) can be seen below:

Algorithm 1 PPO, Actor-Critic Style

```
for iteration=1, 2,… do
    for actor=1, 2,… N do
        Run policy $\pi_{\theta_{old}}$ in environment for T timesteps
        Compute advantage estimates $\hat{A}_1, \dots, \hat{A}_T$
    end for
    Optimize surrogate L wrt $\theta$, with K epochs and minibatch size $M \leq NT$
    $\theta_{old} \leftarrow \theta$
end for
```

This algorithm will also be easier to implement within the model considering the number of agents and timesteps in the simulation program.

Maze Design

This study uses a maze environment to analyze cooperative behaviors in multi agent systems. A maze is defined as an N x N dimensional space (Kivelevitch & Cohen, 2010, p.392). It will consist of walls, traps and a goal at the bottom of the maze. A wall is defined as a 1x 1 dimensional block that an agent cannot pass through on any side. There will be walls around the maze area to keep agents within bounds. A trap is defined as a circle of radius 1 that an agent can step on that causes a failure state. An agent, in this case, is a 1x1 block with dynamic behavior that moves about the maze and finds the exit. An agent "continuously perceives [the] environment, makes decisions, and acts on the environment" (Zhao et al., 2012, p.64). The term emergent behavior will be used to describe the behavior of the group of agents. If an agent occupies a space, it is assumed that another agent cannot occupy that space. Each agent has a discrete action space of 5, consisting of these actions:

- 0 : The agent moves 1 space backward
- 1 : The agent moves 1 space to the right
- 2 : The agent moves 1 space forward
- 3 : The agent moves 1 space to the left
- 4 : The agent teleports the subsequent agent forward

The teleportation action is the cooperation choice, since the agent choosing it cannot move themselves but instead "trusts" the subsequent agent to move for them. They essentially sacrifice their turn for the possibility that the other agent gets closer to the goal. One may notice that the teleportation action has no definitive number of spaces the other agent moves. This is because the distance is variable based on the end destination. The other agent will move 3 spaces forward maximum. The number of spaces the teleporting agent moves to can decrease depending on if the end destination is out of bounds, a wall, or another agent.

There is a possibility that the teleporting agent does not move at all since there are other agents or obstacles in its path. Since teleportation is variable and somewhat unpredictable, it causes the agents to discover a flaw in the coordinate system during the model's learning process. It was possible for an agent to teleport another agent out of existence entirely if the right conditions are met; meaning, the other agent does not have a stored location to reference after teleportation, making them not appear in the maze. For instance, the seven-agent team on the second maze use this flaw to remove one agent to give the others room to explore the maze. It is an unforeseen strategy but a legitimate one nonetheless.

This maze can be represented as a Markov Game which requires agents to learn after each loss to gain success. There will be many winning scenarios but as the agents progressively learn many of the more selfish outcomes will wane. Since this maze is a joint-action space, then the maze can be defined as the cartesian product of the individual agents' spaces and respective actions spaces. If S is the set of maze states and $A_1 \ldots A_n$ represent the action sets of n agents, then the learning takes place in the following product space:

$$\boldsymbol{S\ x\ A1\ x \ldots An \rightarrow M}$$

where M is the product space of the maze (Tuyls et al., 2004, p.316). Since the agents share maze state information, then this equation will hold during the learning process. Assuming that S and A are nonempty with size B and C respectively, the number of possible action pairs will be $B * C^n$ for n agents. We know the set of maze state S will be {Success, Failure, Playable} and the action set A for any individual agent is {0,1,2,34}. This means the possible number of action pairs in space M will be $3*5^n$ for n agents. The exponential increase of actions in M per additional agent is why the incentive structure for the environment is important.

Product spaces like M within multi agent systems in general can be unpredictably large depending on the number of states, agents, and actions the agents can take. As a result, "small changes in learned behaviors can often result in unpredictable changes" (Panait and Luke, 2005, p.2). Since the agents all receive identical actions, the maze environment is using homogenous team learning. This is important since there is no need for task specialization between agents to solve the maze. Instead, it requires the agents to interact with each other with equal standing. This does not mean that the agents act as a single unit. The agents can act heterogeneously and can even sabotage each other. Throughout the training process for the agent teams there were many times where an agent teleports another one into a trap. Heterogenous behavior can exist for many other multi agent systems if the homogenous behaviors specify "sub behaviors that differ based on an agent's initial condition … or its relationship with other agents" (Panait and Luke, 2005, p.5). Suppose that there are two agents trying to exit the maze. Considering the maze as a Markov game, the game G will be defined as a tuple (T_1,T_2,V_1,V_2) where T_1 = Agent 1 mixed strategy set, T_2 = Agent 2 mixed strategy set, V_1 = payout function for Agent 1 and V_2 = payout function for Agent 2 (Tuyls et al., 2004, p.301). The payout functions take the cartesian product of the strategy sets (S_1 x S_2) and map them to the reward set R. Hence the payout function for an agent k is defined as

$$\boldsymbol{V_k : T_1\ x\ T_2\ \ x \ldots T_k \rightarrow R}$$

The expected payout for an agent k given M total actions $a_1 \ldots a_M$ in the agent action set A and their mixed strategy s in the set S_k will be

$$\boldsymbol{P_k(\mathrm{x}) = \sum_{a_M}^{M} P_k(a_1, a_2, \ldots., a_M) s_{1,a_1} \ldots . s_{M,a_M}}$$

where $P_k(a_1, a_2, \ldots., a_M)$ denotes the payoff of agent k for their action space (Mertikopolous & Sandholm, 2016, p.1299).

Table 1: The payout matrices X and Y for agent 1 and 2 respectively

$\mathrm{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$	$\mathrm{Y} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$

Table 1, as shown above, has the payout matrices for Agent 1 (X) and Agent 2 (Y) given they have two strategies. The element x_{ij} signifies the reward given to Agent 1 based on a strategy taken from S_1; the element y_{ij} signifies the reward given to Agent 2 based on a strategy taken from S_2. Since all the agents are a singular team, with agents not knowing the actions of other agents, this game is like the Prisoner's Dilemma. In the Prisoner's dilemma, both prisoners are interrogated in different rooms. If neither prisoner confesses to the crime, they both get an increased sentence. If one prisoner confesses but the other does not, the one that does not confess gets an increased sentence. If they both confess, they both get a decreased sentence. The dominant strategy that is the most beneficial for both prisoners will be to confess. In regards to the agents, Agent 1 or Agent 2 can step into a trap and cause failure for the whole team. Another scenario is that both agents move in different directions, and both hit empty spaces. Thus if $(x_{11} – x_{21})(x_{12}-x_{22}) > 0$ or $(y_{11} – y_{21})(y_{12}-y_{22}) > 0$ then at least one of the agents has a dominant strategy. This also means that there is one strict equilibrium (Tuyls et al., 2004, p.302). Since the mazes will be static, this equilibrium will not change. That does not mean that the agents will always find the optimal path through the maze. Instead, it means that once they find a path, they will choose to only go that path. Nash equilibrium occurs once the agents cannot increase their rewards by changing their strategy. In the given mazes, there is a given set of outcomes:

- Any of the agents fall into a trap
- Any of the agents find the exit
- The agents exhaust all their timesteps

The probability of those outcomes, assuming all are equal, will be $\frac{n}{2n+1}$, $\frac{n}{2n+1}$, and $\frac{1}{2n+1}$ respectively for n agents. This will not be the case in practice because it assumes that all agents will actively search the maze and not use teleportation. The way to determine the probabilities in practice will be through variations of the maze. There are multiple ways to alter a maze to analyze emergent behavior. They include"

- Changing the maze structure by changing wall and trap placement
- Changing the location of the exit
- Changes in behavior due to the actions of other agents in the multi agent group

This study uses the first and last methods since they are relevant to researching how the agents react with each other within a given environment. The agents interact with the environment and each other, which can be modeled as a Markov Decision Process (MDP) defined as a tuple (S, A_k, P, r^k) (Cassano et al., 2019, p.1). S maintains its definition as the set of states and A_k maintains its definition as the set of actions an agent k can take. $P(s' \mid s, a)$ is the probability of moving from state s′ to another state s having taken an action a from A_k. and r^k is the reward function of agent k. The goal of the agents will be to maximize their aggregated return, which is defined as

$$J(\pi) = \sum_{t=0}^{\infty} \frac{\gamma^t}{K} \left(\sum_{k=1}^{K} E_{\pi,P}[r^k(s_t, a_t, s_{t+1}] \right)$$

where s_t and a_t are the state and action at timestep t, $\pi(a|s)$ is the team's policy and $\gamma \in [0,1)$ is the discount factor (Cassano et al., 2019, p.2). The discount factor determines how the current factors contribute to the potential return. The scale ranges from unimportant (0) to very important (1). The reward structure will allow these equations to take shape within the model.

Reward Structure

The agent reward structure per timestep is as follows:

- -2 per timestep
- -200 when any agent steps on a trap
- $+200 * (1 + \frac{1}{current\ time\ step})$

The reward per each timestep is negative so then agents will have an incentive to find the exit in a lower number of timesteps. The trap penalty is large to show the severity of failure and make it a disincentive. The large reward is to substantially reinforce agent behaviors that result in success. The fractional increase in reward per timestep is also to incentivize the agents to find the exit faster. As seen in other studies, the addition of subgoals that lead to the main goal makes agents learn faster. In this case, the subgoal will be general exploration of new spaces on the maze. However, there is the issue of incentivizing exploration. There can be a point where the best strategy for the long term is to avoid traps and use all the timesteps since there is a comparatively low chance a given space is the goal. This chance decreases as the map size increases.

This reward structure also includes an exploration incentive, where there is a +1 reward for each new space explored regardless of the action taken to discover it. This makes the reward for the timestep -1 as new spaces are explored. Such a marginal change can induce cooperation "when the agent's marginal disutility with regard to helping effort is zero at zero help" (Itoh, 1991, p.613*)*. Since the agents do not have to cooperate to get the same reward, then the marginal disutility when they do not help is zero. However, the agents teleporting each other can be a better strategy in the long term since teleportation is more likely to move agents to new squares or even to the goal. After all, agents can move left/right against walls or even move backwards; both of which results in the default -2 reward. However, there can be disadvantages to a collective reward system for multi agent teams. Collective rewards "may cause 'inefficient over-cooperation'" (Itoh, 1991, p.612), which leads to agents overcompensating for maximum benefit. Another thing to consider is that this maze is fully cooperative, meaning there is a global reward scheme for all agents. As shown previously, the mazes would have a strict equilibrium. The convergence of the Nash equilibrium is straight forward due to having a global scheme.

Results

For all subsequent figures of the mazes, the turtle objects signify agents, the circles signify traps, the squares signify walls and the star signifies the exit of the maze.

Experiment 1: The First Maze

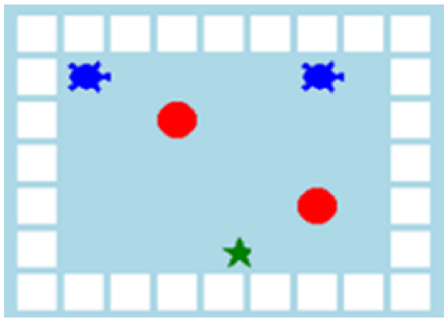

Figure 2: This is the layout of the first maze

The first maze is a custom 8x8 maze with a 7x7 playable area. There are fifty timesteps available regardless of the number of agents in the maze. There is minimal maze complexity due to its mostly open nature. However, agents do have to watch out for the two traps. The lack of complexity may make this maze seem simple to solve but can be more challenging for larger teams to navigate. The small size allows the smaller teams of agents to explore most if not all the squares before all the timesteps are used.

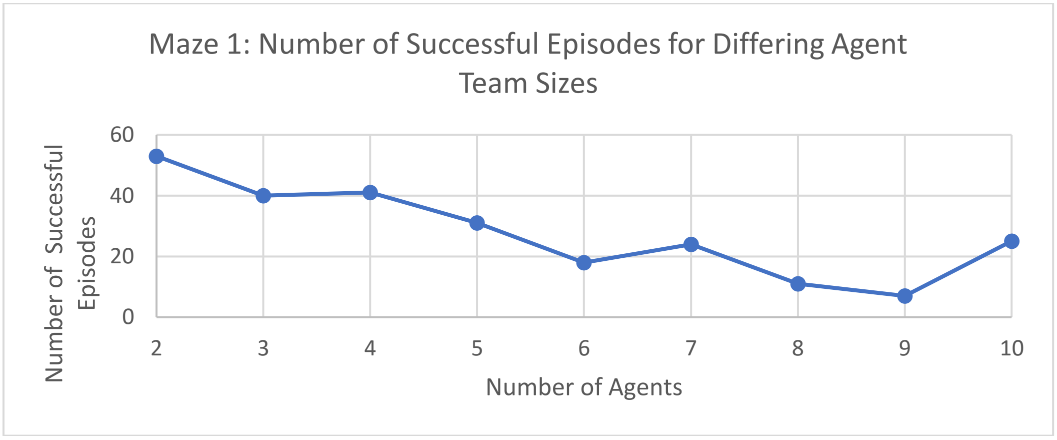

Figure 3: Number of Successful Episodes for Differing Agent Team Sizes

As one can see on Figure 3, the number of successful episodes decreases as the number of agents increases. One would think that as more agents coexist there will be advantages such as exploring the maze faster. However, there is a general decrease in success as the team increases. The percentage for the three agent teams is close to the set probability outcomes since $\frac{n}{2n+1}$ is 40% and the three-agent team had 40% success.

One thing to note is that the two-agent team was the most successful. Their strategy was Agent 1 moving forward, Agent 2 moving left, Agent 1 teleporting Agent 2, Agent 2 teleporting Agent 1 and finally Agent 1 teleporting Agent 2 towards the exit. Notice that once Agent 1 teleports Agent 2, Agent 2 reciprocates afterward. This is a trend in all the testing on the various mazes. This reciprocation can potentially be seen as a response rather than a strategy. It is as if an agent teleports first to test if it is a viable action. Then the second agent reciprocates. The two agents also explored more spaces in a shorter number of timesteps than the other teams, perhaps since they had the most timesteps per agent.

Table 2

Map 1 Episode Analysis			
Number of Agents	*Average Steps*	*Average Reward*	*Average Time(s)*
3	*8.98*	*-40.038689*	*1.09896571*
4	*14.25*	*-40.1170642*	*1.743904384*
5	*8.12*	*-83.6068796*	*0.993719551*
6	*9.63*	*-138.105001*	*1.178512226*
7	*9.34*	*-106.7756826*	*1.143022242*
8	*11.93*	*-80.0849176*	*1.459984512*
9	*13.32*	*-106.8099529*	*1.630091677*
10	*12.55*	*-108.634112*	*1.53585965*

However, as shown in Table 2 above, there is no row for the two-agent team. This is because the episode data for that team was erased by the software while collecting the data. In Figure 3, there are a couple of outliers in the general trend: the seven and ten agent teams. The following analysis will use strategies ***A*** and

B to signify the six agent team and seven team strategies repsectively. It will also use strategies ***C*** and ***D*** to signify the nine and ten agent team strategies respectively. The seven-agent team outperforms the six agent one despite having more agents. This is also interesting since they both have similar average steps per episode as seen in Table 2. This is also reflected in the similar number of steps for each strategy.

Table 3

Case A: Six Agent Team Strategy	Case B: Seven Agent Team Strategy
1. Stay Put	1. Stay Put
2. Agent 2 Teleports Agent 3	2. Agent 2 Teleports Agent 3
3. Agent 3 Teleports Agent 4	3. Agent 3 Teleports Agent 4
4. Agent 5 Teleports Agent 6	4. Agent 5 Teleports Agent 6
5. All Agents explore	5. Agent 1,2,3 Explore
6. Agent 3 Teleports Agent 4	6. Agent 4 Teleports Agent 5
7. Agent 4 Teleports Agent 5	7. Agent 5 Teleports Agent 6
8. Agent 5 Teleports Agent 6	8. Agent 6 moves left/right to find exit
9. Agents explore until exit is found	

Table 4

Case C: Nine Agent Strategy	Case D: Ten Agent Strategy
1. Almost all agents stay put	1. Agent 1 and 2 Stay put
2. Agent 2 Teleports Agent 3	2. Agent 3 Teleports Agent 4
3. All Agents explore	3. Agent 4 Teleports Agent 5
4. Agent 3 Teleports Agent 4	4. Agent 5 Teleports Agent 6
5. All Agents explore	5. Agent 6 Teleports Agent 7
6. Agent 4 Teleports Agent 5	6. Agents 8,9,1, and 2 explore top half
7. Agents 3,4,5 explore bottom half until exit is found	7. Agents 4,5,6,7 explore bottom half until exit is found

Table 3, as seen above, shows the two strategies ***A*** and ***B*** beside each other. One thing to note immediately is that they both have the same first step, but Strategy ***B*** has one less step than Strategy ***A***. The first step, "Stay Put", means that Agent 1 goes backwards; thus, remaining in the same position. The main difference between Strategy ***A*** and ***B*** starts at step five. Strategy ***B*** has Agents 1,2 and 3 explore the maze while Strategy ***A*** has all the agents explore the maze. Assuming none of the agents stumble upon the exit,

Strategy B uses three timesteps while Strategy ***A*** uses six. It is also significant that with Strategy ***B*** Agent 6 is able to exit by going right or left. In contrast, Strategy ***A*** does not have Agent 6 in an optimal position. Instead, all six agents must explore the maze until they find the exit. This may take a variable number of timesteps to exit, which makes Strategy ***A*** even less optimal. Ultimately, Strategy ***B*** manages the agents in a more optimal way in a similar number of steps due to spreading more agents within the bottom half of the maze around the location of the exit.

A final point of comparison for the agent team's performance on the first maze is the comparison between the nine and ten agent teams. Their strategies can be seen above in Table 4. It is significant that the ten-agent team achieves double the number of successful episodes as the nine-agent team. The ten-agent team takes one timestep less than the nine-agent team on average according to Table 2. This may not seem numerically significant, but it is significant in the context of the maze. One timestep means that an agent can move and potentially find the exit.

The differences between Strategy ***C*** and Strategy ***D*** may provide a more complete explanation of the disparity between Strategy ***C*** and Strategy ***D***'s success seen in Figure 2. One thing to notice immediately is that for Strategy ***C*** almost all agents stay put while Strategy ***D*** has only Agent 1 and 2 staying put. This means steps 2 through 5 in Strategy ***D*** will be done before Strategy ***C*** completes step 1. Strategy ***D***, at that point, will have its team split in half to explore both halves of the maze. Before Strategy ***C*** reaches the same status as Strategy ***D***, it repeats the cycle of the agents exploring for a timestep followed by an agent teleporting another agent twice. Strategy ***C*** almost evenly splitting its team in half to explore the maze is something the other teams besides the four, three, and two agents did not accomplish. It is easier to split the four-agent team and smaller since they have less agents overall. Regardless, the ten-agent team did develop a comparable strategy and made that team the fifth most successful team.

Experiment 2: The Second Maze

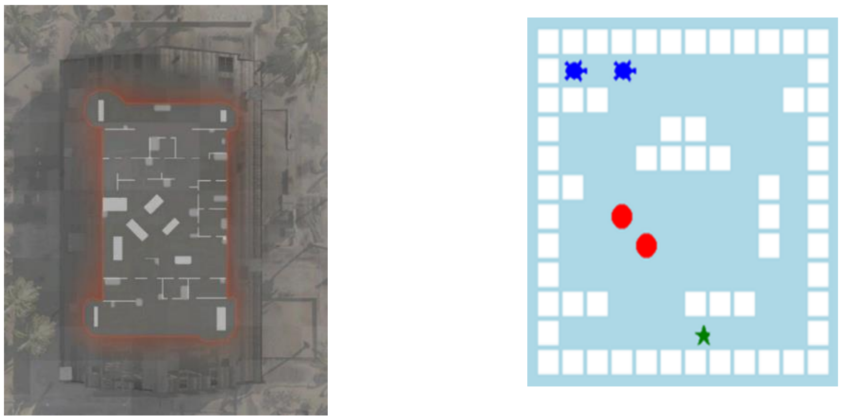

Figure 4: The left image is the layout of Killhouse and the right image is layout of the second maze

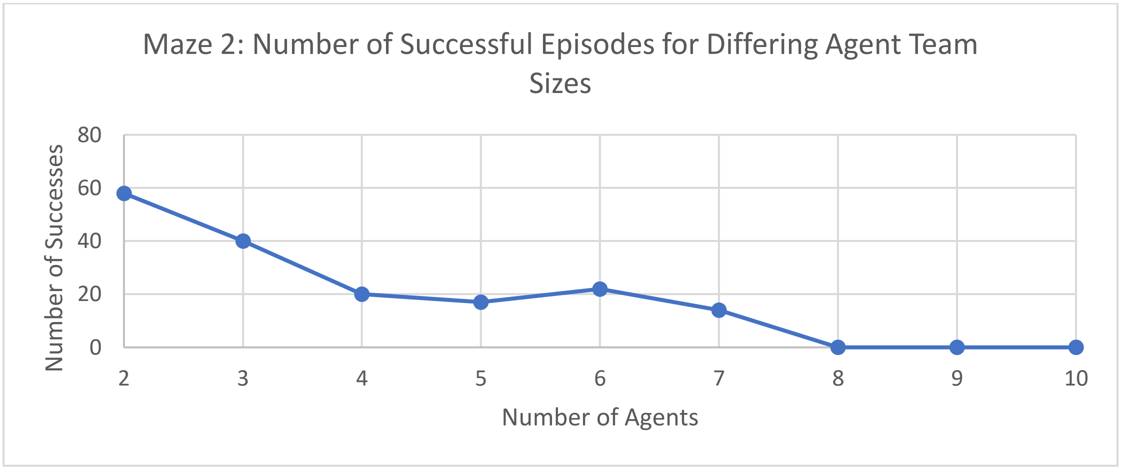

Figure 5: Number of Successful Episodes for Differing Agent Team Sizes

Table 5

Map 2 Episode Analysis			
Number of Agents	*Average Steps*	*Average Reward*	*Average Time (s)*
2	*34.32*	*51.9616001*	*4.200056032*
3	*35.02*	*-26.4649644*	*4.28572151*
4	*37.04*	*-89.5866235*	*4.532927605*
5	*36.24*	*-115.4588763*	*4.435024201*
6	*42.4*	*-80.4883032*	*5.188880412*
7	*40.87*	*-99.9708313*	*5.001640152*
8	*41.92*	*-154.24*	*5.13013837*
9	*49.56*	*-95.77*	*6.065115878*
10	*48.78*	*-98.05*	*5.969660059*

The second maze is a rendition of the Killhouse map from Call of Duty Modern Warfare as seen above in Figure 4. This maze is 11x11 with a 10x10 playable area. This is a good environment for testing due to its small (comparatively speaking to other maps in the game) map size with complex design. The amount of timesteps given to complete the maze is the same as the previous maze. Unlike the first maze, there is a significant decrease in successful episodes as the number of agents increases as seen above in Figure 5. However, the outlier to this trend is the improvement of the six-agent team over the five-agent team. The disparity between the two teams is also surprising considering that in Table 5, as seen above, the five-agent team has six less timesteps per episode than the six-agent team. Six timesteps means that the whole six agent team can go another timestep. Yet the average reward per episode for the five-agent team is -115.49 while the six-agent team has an average reward of -80.4883. The average reward per episode for the five-agent team being below -100 signifies that there were more episodes that resulted in failure rather than not finding the exit after fifty timesteps.

The average reward per episode for the six-agent team being approximately -80 signifies that there were more episodes where most if not all available timesteps are used. When analyzing the tendencies of each team for the 100 episodes, the difference in success became clear. The disparity is due to the tendencies of the two teams rather than their respective strategies. The five-agent team takes on average twenty timesteps to exit the first row. This signifies that the agents decide to fan out the row by going left and right rather than teleporting each other out. Once most of the team leave the first row, the agents primarily move forward and teleport each other towards the exit. The six agent team teleports each other more often and earlier on. Unlike the five-agent team, there will be two agents at the bottom half of the maze with two more shortly behind them at the twenty timestep mark.

The seven-agent team on the graph is also an outlier as it is part of the outward shape between the five and eight agent teams. This is mainly because the seven-agent team acted similarly to the six-agent team. This was in the literal sense, as the seven-agent team found a flaw in the teleportation system. Suppose there are two agents x and y. If x decides to teleport y but there are three walls in front of y, then y should not be able to teleport. However, the maze does not keep the coordinates after rechecking the new position in this situation. This results in the new location for agent y to be empty, removing the agent entirely from the maze. The seven-agent team eliminating the seventh agent using this exploit gave the other agents more space and consequently a strategy like the six-agent team. Another significant part of Figure 5 is the fact that the eight, nine, and ten agent teams never exit the maze during the 100 episodes. In fact, they all failed using the same initial strategy. Each of those teams decided as their first task to get all the agents out of the first row by having the agents move left and right rather than teleporting each other out of that row. This results in none of the agents moving as there is little space in that row with two walls on the edges.

Experiment 2: The Third Maze

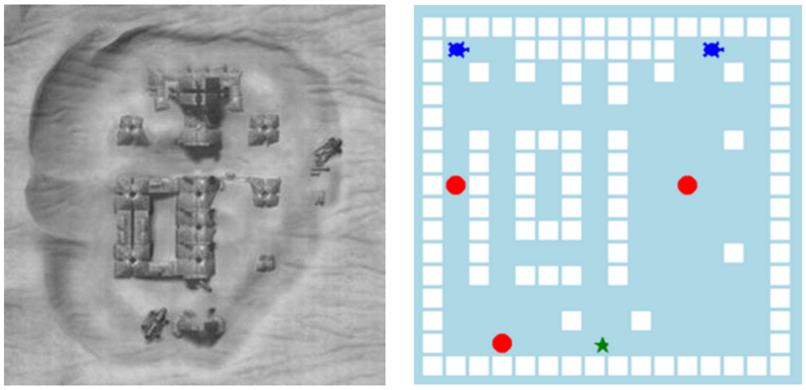

Figure 6: The left image is the layout of Sandtrap, the right image is the layout of the third maze

The third and final maze is a rendition of the Sandtrap map from Halo 3 as seen in Figure 6. This maze is 16 x 16 with a playable area of 15 x 15. There will be 60 timesteps available for solving the maze due to its size and the possibility of moving around its large structures. Additionally, this map is a good test case for how agents move around large structures in a bigger play area. This maze, unlike the previous two, had no teams that were successful in the 100 test episodes. In fact, all the teams have a similar issue to the eight, nine and ten agent teams issues in the previous maze. All the teams decide to explore the top half of the maze and never reach the bottom half. The two-agent team, for instance, explored their respective top corners of the maze. This meant that the existing reward scheme was not enough to get the agents to the exit. The first remedy to this issue was to increase the amount of available timesteps to 130. This did not solve the issue but rather confirmed the problems. Even with the extra timesteps, the agents spend them searching the top half of the maze.

An additional reward was added based on distance with the additional timesteps for an increased chance of success. The distance between a given agent during their turn and the goal is added as a negative reward. Hence, the agents will have an incentive to get closer to the goal to decrease their penalty. Unfortunately, this incentivized the agents to move horizontally rather than vertically. This is possibly because the distance between agents and the goal varies depending on the location of each agent. The best way to minimize the penalty across all agents is to move left or right to decrease the average difference in distance. This maze exemplifies the main hinderance for team success: the size of the maze. This maze does have nearly straight path to the exit like the first maze. It also has less complexity than the second maze, consolidating the walls into larger structures that can be avoided by the agents. On the other hand, this maze has the largest number of explorable spaces. It may be more advantageous in the long term for agents to explore the maze to get the exploration incentive rather than finding the exit at the opposite end of the maze.

Future Study

There are many areas of this research that warrant further study. Firstly, there should be more research on cooperation in multi-agent systems in three-dimensional spaces. This was the original intention of this study, but hardware and time constraints made a two-dimensional environment more manageable. Some behaviors may remain the same in the context of maze exploration. Three dimensional agents will still largely move on the x-y plane with the z axis being stagnant. However, three dimensional environments do provide changes in elevation that cannot be done in two dimensional spaces. Secondly, there should be more research pertaining to how agent cooperation changes with scale.

This study has some insight into how maze scale affects agent behavior but there is more to be done. An additional method for testing the effects of scale is taking a singular map and multiplying each dimension by some scalar value for each test. Afterwards one can measure the number of timesteps needed to find the exit. Thirdly, there should be more research using larger scale maps in both two and three dimensions. Some video game related examples include the maps for popular battle royale games such as Apex Legends and Fortnite. In general, there should be more research using the maps from various video games. Random maze generation is useful to discover general trends, but many random maze generation algorithms cannot create a video game-like environment that can balance complexity and scale.

Conclusion

In this study, multi-agent systems have many forms of cooperation depending on the task in addition to the number of agents performing it. The data for maze 1 and maze 2 shows that on average the number of successful episodes decreases as the number of agents increases. This mostly is a result of a lack of available space per agent as the team size increases. Even though there are more agents in the maze, the amount of turns an agent can have also decreases. Maze complexity can be a hinderance for exploration, but maze size may be the more determinant factor of success as seen with maze 3. The two-agent team was the best performing team not only due to having more available spaces but more timesteps per agent to refine their strategy. Regardless of team size, there have been some common initial steps for each simulation. Firstly, the first half the team teleports the other agents forward. This can possibly be explained by the fact that those agents go first and thus can initiate. Once an agent teleports another one, other agents reciprocate this same action on the next agent. In essence, an agent must teleport another agent “out of faith”; if it succeeds, then others follow that course of action. Secondly, the agents create a hierarchal system that leads to their success. Some agents stay in the top half of the maze teleporting other agents to the bottom half of the maze. They are essentially a support group, helping other agents get to the exit faster. The agents in the bottom half of the maze lead the other agents, exploring until the exit is found. Another observation is that the average amount of timesteps and reward per episode does not fully explain the success or failure of a team. Instead, the differing strategies between teams provide a more complete explanation for how one team is successful over another.

The significance of this research is outlined in how these strategies develop and change. Each team had a differing emergent strategy even though the optimal solution was the same for each maze. The cooperation can drastically change from one team to another team with one more agent in it. Ultimately, this study shows how cooperation is not only mutual amongst agents but a differentiator for the effectiveness of various team strategies.

References

Al-Assadi, S. (2007). Neural Network-Based Model Reference Adaptive Control for Electronic Throttle Systems. *SAE Transactions*, *116*, 657–664. http://www.jstor.org/stable/44719937

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., & Mordatch, I. (2020). Emergent Tool Use From Multi-Agent Autocurricula. *arXiv [cs.LG]*. http://arxiv.org/abs/1909.07528

Cassano, L., Alghunaim, S. A., & Sayed, A. H. (2019). Learning for Multi-agent Reinforcement Learning. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3062–3066. doi:10.1109/ICASSP.2019.8683168

Elliott, E., & Kiel, L. D. (2002). Exploring Cooperation and Competition Using Agent-Based Modeling. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(10), 7193–7194. http://www.jstor.org/stable/3057838

Itoh, H. (1991). Incentives to Help in Multi-Agent Situations. *Econometrica*, *59*(3), 611–636. https://doi.org/10.2307/2938221

Kim, M. (2016). A study of collaborative distributed intelligent multi-agent reinforcement learning via multi goals for dynamic agent shortest path-planning (Order No. 10143404). Available from ProQuest One Academic. (1812542011). https://www.proquest.com/dissertations-theses/study-collaborative-distributed-intelligent-multi/docview/1812542011/se-2?accountid=9900

Kiseliou, I. (2020). Better cooperation through communication in multi-agent reinforcement learning (Dissertation). http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-412393

Kivelevitch, E., & Cohen, K. (2010). Multi-Agent Maze Exploration. *Journal of Aerospace Computing Information and Communication*, *7*, 391–405. doi:10.2514/1.46304

Kraus, S. (1997). Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, *94*(1), 79–97. doi:10.1016/S0004-3702(97)00025-8

Mertikopoulos, P., & Sandholm, W. H. (2016). Learning in games via reinforcement and regularization. *arXiv [math.OC]*. http://arxiv.org/abs/1407.6267

Östling, R., Wang, J. T. Y., Chou, E. Y., & Camerer, C. F. (2009). Online Appendix for "Testing Game Theory in the Field: Swedish LUPI Lottery Games".

Panait, L., & Luke, S. (2005). Cooperative Multi-Agent Learning: The State of the Art. *JAAMAS*. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.307.6671

Sánchez, A. (2018). Physics of human cooperation: experimental evidence and theoretical models. *Journal of Statistical Mechanics: Theory and Experiment*, *2018*(2), 024001. doi:10.1088/1742-5468/aaa388

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. doi:10.48550/ARXIV.1707.06347

Tuyls, K., Nowe, A., Lenaerts, T., & Manderick, B. (2004). An Evolutionary Game Theoretic Perspective on Learning in Multi-Agent Systems. *Synthese*, *139*(2), 297–330. http://www.jstor.org/stable/20118420

Yang, J., Nakhaei, A., Isele, D., Zha, H., & Fujimura, K. (2018). CM3: Cooperative Multi-goal Multi-stage Multi-agent Reinforcement Learning. *CoRR, abs/1809.05188*. http://arxiv.org/abs/1809.05188

Zhao G., Shen A., & Shen Z. (2012). Learning-by-Teaching: Designing Teachable Agents with Intrinsic Motivation. *Journal of Educational Technology & Society*, *15*(4), 62–74. http://www.jstor.org/stable/jeductechsoci.15.4.62