ROB 538
Multiagent Systems
Homework 2: Autonomous Agents

Timothy Drury
Oregon State University
Corvallis, OR 97331
druryt@oredonstate.edu

***Abstract*—This report presents an analysis of reinforcement learning techniques applied to multi-agent and multi-target scenarios. We examine the behavior of agents both in individual policies and global reward structures. We utilize Monte Carlo (MC) policy evaluation with model-based policies. The MC evaluation allows agents to estimate value functions by averaging returns over complete episodes, while the model-based policies provide a structured approach for guiding agent decisions. By employing this approach, agents benefit from both detailed episodic feedback and structured decision-making. We observe the effectiveness of these methods through experiments with two autonomous agents interacting within a gridworld framework, assessing their learning dynamics and the number of steps required to capture targets.**

***Index Terms*—Reinforcement Learning, Monte Carlo, Epsilon-Greedy**

I. Introduction

A. Objective

The objective of this report is to desctribe the behaviors of agents within a 5×10 gridworld environment, as illustrated in *Figure 1*. In this gridworld, agents start at a fixed location and can perform four actions: moving up, down, left, or right. Agents receive a reward of 20 for capturing either target (T1 or T2). However, there is also a penalty of -1 for every time step an agent spends in the gridworld without capturing a target.

The report addresses the behavior in the following situations:

1) Single Agent, Single Target: In the initial scenario, only one agent is present to capture a single target (T1). The task is to devise a learning algorithm, utilizing either a neural network or reinforcement learning approach, that enables the agent to effectively reach T1. All relevant system parameters, including inputs, states, outputs, and training methods, must be clearly defined. We will look into reinforcement learning techniques later.

2) Two Agents, Two Targets with Individual Rewards: The second phase introduces a second agent (Agent 2) and a second target (T2). Both agents begin at the same location, and each agent receives its own reward based on the target(s) they capture individually. Agents are permitted to occupy the same square. Observations will focus on the resulting behaviors of the agents, examining whether they benefit from each other and determining whether any cooperation occurs.

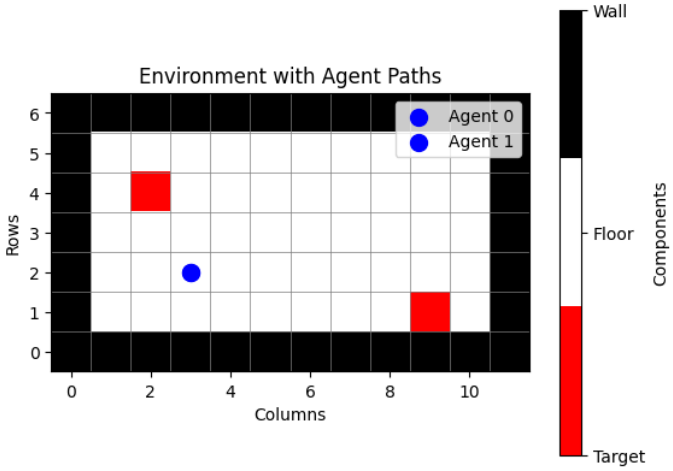

Fig. 1. A 5x10 gridworld, with agents in blue and two targets (T1 and T2) in red.

3) Two Agents with Global Reward: The final phase involves implementing a system-level reward (global reward) that reflects the total rewards collected from both targets. In this scenario, both agents receive the same reward. The objective is to evaluate how this global reward structure influences the agents' behavior, particularly in terms of whether they benefit from each other and whether any observed cooperation is intentional or incidental.

B. Background

In this section, we will explore reinforcement learning techniques and their application in multiagent systems.

1) Monte Carlo Evaluation: Monte Carlo (MC) methods are a class of techniques in reinforcement learning that estimate the value of states or state-action pairs by averaging the returns received following each visit to a state. Unlike Temporal Difference (TD) methods, which update value estimates after each time step, Monte Carlo methods wait until the end of an episode to update value estimates. This approach is particularly useful when dealing with episodic tasks where the environment naturally terminates after a sequence of steps.

The key equation involved in MC evaluation: The value of state s is updated using the average of the returns:

$$V(s_t) \leftarrow V(s_t) + \alpha \left(r(s_t) - V(s_t)\right)$$

Here, α is the learning rate, which determines how much new information influences the value function.

Monte Carlo methods are beneficial for environments with sparse feedback since they leverage complete episodes to assess the total return. However, they can be slower to converge compared to TD methods because they only update at the end of episodes.

2) Epsilon-Greedy Policy: The epsilon-greedy strategy is a common approach to balance **exploration** (trying out new actions) and **exploitation** (choosing actions that are known to be effective). It helps the agent avoid getting stuck in suboptimal actions by occasionally exploring alternative actions.

- **Exploration** involves selecting a random action, which allows the agent to discover new states and possibly better long-term strategies. With probability ϵ, the agent selects a random action.
- **Exploitation** involves selecting the action that maximizes the current estimated value or reward, allowing the agent to leverage its existing knowledge. With probability $1-\epsilon$, the agent selects the action that has the highest estimated value.

The value of ϵ typically starts high to encourage exploration in the early stages of learning and is gradually decreased over time, allowing the agent to focus more on exploitation as it gains knowledge.

3) Separate Agent Policies: In multiagent reinforcement learning scenarios, each agent typically learns a separate policy that guides its actions based on its observations and experiences. The independence of policies allows agents to adapt to different roles or behaviors within a shared environment, facilitating diverse strategies for achieving a common objective. This setup is especially useful in environments where agents have different perspectives or roles that complement each other.

As agents independently update their policies through interactions with the environment, they learn to maximize their own expected rewards. Having separate policies for each agent allows the system to scale efficiently as the number of agents increases. It also enables flexibility, as agents can specialize in different roles without requiring a centralized control mechanism. However, making these agents act efficiently without a global perspective is an issue.

4) Global Reward Structure: In order to tackle the efficiency problem in individual multiagent systems, a global reward structure can be added. With this agents are rewarded based on the collective outcome of all agents' actions determines the reward received by each agent A global reward structure can encourage agents to work together more effectively by aligning their objectives with the overall goal of the system.

While global rewards can align agents' objectives, they can also introduce challenges, such as credit assignment. Determining how much each agent's actions contributed to the overall reward can be difficult, especially in complex environments. Moreover, it may slow down the learning process as agents try to understand their individual contributions to the team's success.

By using separate policies and a global reward structure, multiagent systems can balance individual learning with collaborative goals, leading to more effective and flexible solutions for complex tasks.

These concepts are the foundation for many reinforcement learning approaches in multiagent systems. By combining Monte Carlo evaluation with an epsilon-greedy policy and allowing each agent to maintain individual policies and a global reward structure, agents can deepen their understanding of the environment. This approach ensures exploration possibilities while agents converge toward optimal strategies. This will balance individual learning with the overall objective.

II. Methodology

A. Gridworld Environment

The gridworld environment is structured as a 7x12 grid, shown in *Figure 1*, where the outermost rows and columns are designated as walls. This creates a confined area for the agents to navigate. The walls are represented by a value of 1 (black cells), while the inner cells, which constitute the floor, are assigned a value of 0 (white cells). Within this environment, targets are strategically placed at predetermined grid locations, such as (4, 2) and (1, 9), and represented by -1 (red cells).

The environment supports multiple agents, each initialized at a fixed starting location denoted by the blue dot. Agents in this environment can perform one of four actions: move up (0), move down (1), move left (2), or move right (3). This functionality is implemented in the 'step' method, which updates the agent's position based on the selected action while ensuring the agent does not move outside the grid boundaries. If an agent successfully captures a target, the target is removed from the grid, and the agent receives a reward of 20. Conversely, if the agent does not capture a target during its turn, it incurs a penalty of -1.

Overall, this gridworld environment provides a structured framework and visualization for analyzing the dynamics of agent behavior and the debugging of different reinforcement learning techniques.

B. Agent Design

The agents in this gridworld utilize a model-based policy structure encapsulated within the `Policy` class. Each agent begins with a foundational policy that defines its initial actions within the environment. This policy is represented as a 5x10 array, where each cell corresponds to an action the agent can take based on its current state.

The policy can be updated to reflect improved strategies for navigating toward targets. The `Policy` class manages this process by allowing changes to the actions within the array.

Key features of the `Policy` class include:

- **Initialization of the Policy**: Each agent starts with a default policy that guides its movements. This initial policy sets the framework for the agent's decision-making process. This just a general path that covers all cells shown in *Figure 2.*

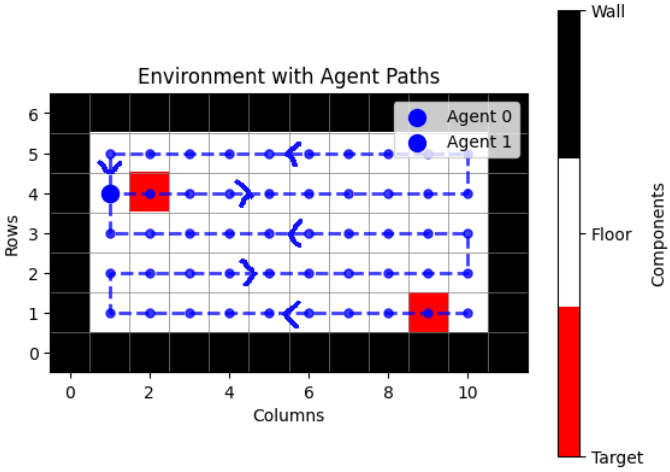

Fig. 2. Display of initial policy with the path and direction shown.

- **Target Capture Logic**: When an agent successfully reaches a target, the corresponding cell in the policy is modified to indicate that the target has been captured. This feedback mechanism reinforces successful behaviors and encourages further exploration.
- **State Transition Management**: The `Policy` class also includes methods to determine the next state based on the agent's current position and the action taken. This allows for a structured approach to movement within the gridworld, ensuring that the agents make informed decisions based on their surroundings.

C. Reinforcement Learning

The reinforcement learning process for the agents in the gridworld is encapsulated within the `Evaluation` class, which is responsible for assessing and improving the agents' policies through interaction with the environment.

- **Initialization:** The `Evaluation` class is initialized with the environment, two agents, and relevant parameters:
  - Learning rate (α): 0.7
  - Exploration rate (ϵ): 0.1
  - Threshold for value function improvement (*th*): 0
  - Max Steps for evaluation (*max_steps*): 100

  It maintains value functions (V and $V1$) for both agents, which are used to evaluate their performance and guide policy updates.
- **Evaluate Method:** The `evaluate` method runs a simulation for a specified number of steps, during which agents take actions based on their current policies. The method tracks the state history to penalize revisiting states by -3, thereby encouraging exploration of new paths. Rewards are accumulated based on the agents' actions and whether they successfully capture targets. The value functions are updated using a Monte Carlo equation given by:

Individual Policy

$$V[i\_new-1][j\_new-1] = V[i\_new-1][j\_new-1] \tag{1}$$

$$+\alpha\,(1 \cdot \text{reward} \; -V[i\_new-1][j\_new-1]) \tag{2}$$

- **Policy Update Mechanism:** The `iterate` method is responsible for refining the agents' policies through a series of learning iterations. Each iteration involves:
  - Storing the current policy and evaluating it to obtain the value functions.
  - Making random choice for actions, promoting exploration while allowing exploitation of known successful behaviors. The action selection for each agent is performed using:

$$a = \text{random.choices}([0, 1, 2, 3]$$

  - Evaluating the updated policies and comparing the new value functions against previous ones to determine if a policy improvement has occurred by finding difference:

$$th = (V1 - V0) > 0$$

    For example, the condition `if th1.any() or random.random() < self.epsilon:` checks if there are any positive changes in the value function for agent 0 (indicating potential improvement) or randomly decides to explore new actions with a probability defined by epsilon, thereby allowing for exploration in the reinforcement learning process.
- **Convergence Criteria:** The iteration process continues until a predetermined number of iterations is reached. During each iteration, agents dynamically adjust their policies based on the rewards received, leading to a convergence of policies toward optimal strategies for capturing targets.

After the learning iterations, the learned policies can be tested in the environment. Agents navigate toward targets using the refined policies, and their performance can be monitored, allowing us to look into effectiveness and behavior of multiple agents in the learning process.

III. Results

A. Single Agent, Single Target

Objective: In this scenario, a single agent is tasked with reaching a target (T1) using a defined learning algorithm discussed previously.

Results:

- 22 Captures of T1 in 30 Trial

- 73% Success Rate

One example of these policy paths is illustrated in the figure below, showcasing the trajectory taken by the agent as it moves toward T1. This visualization highlights the agent's learned behavior, demonstrating its ability to balance exploration and exploitation to achieve the goal efficiently.

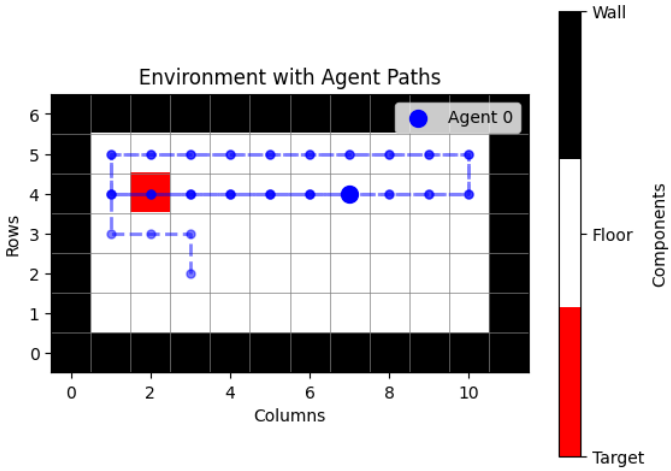

Fig. 3. Single Agent - Single Target Successful Policy

B. Two Agents, Two Targets with Individual Rewards

Objective: This scenario introduces a second agent (Agent 2) and a second target (T2). It examines how the dynamics change when both agents start from the same location and receive individual rewards based on their target captures.

IV. Results

- 18 Captures of T1 in 30 Trial for Agent 0
- 27 Captures of T1 in 30 Trial for Agent 1
- 60% Success Rate for T1 for Agent 0
- 90% Success Rate for T1 for Agent 1
- 3 Captures of T2 in 30 Trial for Agent 0
- 2 Captures of T2 in 30 Trial for Agent 1
- 10% Success Rate for T2 for Agent 0
- 6% Success Rate for T2 for Agent 1

One example of these policy paths is illustrated in the figure below, showcasing the trajectory taken by the agent as it moves toward T1. With these alpha, epsilon, and threshold values it was difficult to ever find T2.

Observations:

- Did agents benefit from each other's presence (e.g., faster target captures)?
- Evidence of cooperation or competition.
- Any conflicts or shared successes observed during trials.

A. Two Agents with Global Reward

Objective: The final scenario involves implementing a global reward structure that reflects the total rewards collected from both targets. Both agents receive the same reward.

- 24 Captures of T1 in 30 Trial for Agent 0

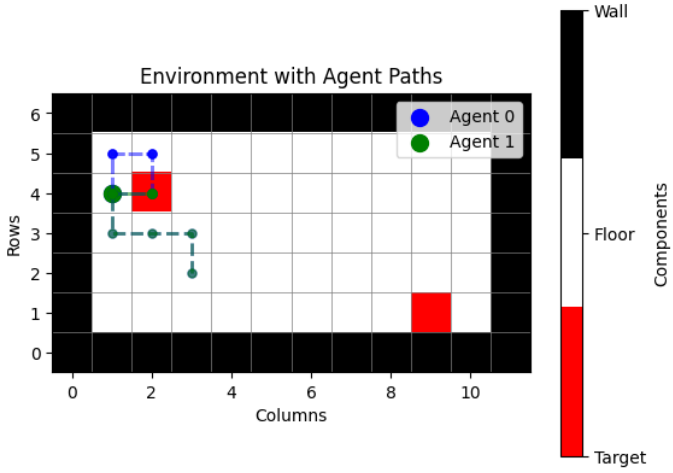

Fig. 4. Double Agent - Double Target Policy

- 12 Captures of T1 in 30 Trial for Agent 1
- 80% Success Rate for T1 for Agent 0
- 40% Success Rate for T1 for Agent 1
- 5 Captures of T2 in 30 Trial for Agent 0
- 7 Captures of T2 in 30 Trial for Agent 1
- 16% Success Rate for T1 for Agent 0
- 23% Success Rate for T1 for Agent 1

One example of these policy paths is illustrated in the figure below where found both targets.

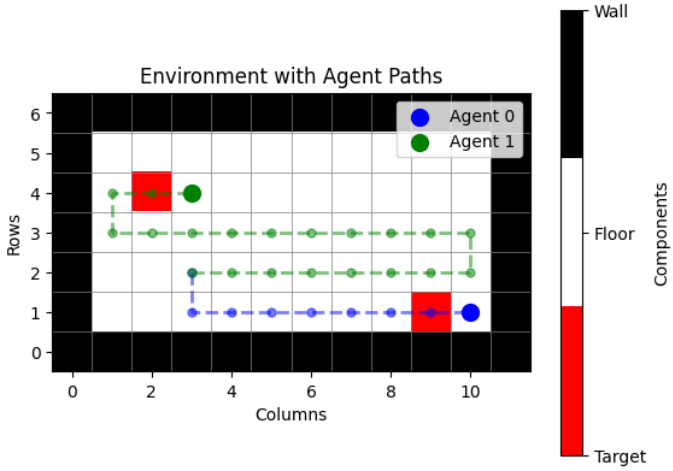

Fig. 5. Global Reward Structure Policy

V. Conclusion

In analyzing the behavior across the three scenarios (single agent with a single target, two agents with individual rewards, and two agents with a shared global reward) several key observations emerge, highlighting the impact of reward structures on agent behavior and the need for further tuning of key parameters such as α (learning rate), ϵ (exploration-exploitation balance), and initial policy settings.

In the first scenario, the single agent learns to navigate efficiently toward T1, but its performance is highly dependent on the proper tuning of learning parameters. Adjustments to these parameters could improve convergence speed and ensure the agent consistently finds the optimal path to the target.

The introduction of a second agent and target in the second scenario reveals that the agents, each motivated by their own rewards, primarily act independently. The benefits they derive from each other are largely incidental, as one agent may not reach its target before the other. While implicit cooperation could emerge from coincidental movements, further tuning of the exploration rate might foster more strategic interactions.

The third scenario, which employs a global reward structure, fundamentally shifts the nature of agent behavior. Here, the shared reward encourages both agents to locate their targets more frequently. However, observations indicate that while one agent might actively explore the environment, the other may simply circle around to minimize its movement penalty. This behavior reflects a lack of explicit communication and coordination between the agents, leading to varied strategies that may not always align. Fine-tuning the learning rate, exploration strategies, and adjusting the starting policies could help refine coordination further, encouraging more robust cooperation.

Overall, while the reward structure significantly influences agent behavior, refining the parameters and initial policies of the learning algorithm could enhance both individual and cooperative performance, leading to more efficient task completion in multi-agent systems.

VI. Acknowledgments

I would like to acknowledge the assistance of ChatGPT, an AI language model developed by OpenAI, for providing support in refining grammar and enhancing sentence structure throughout this document.

References

[1] OpenAI, “ChatGPT: A language model for conversational AI,” 2023. [Online]. Available: https://www.openai.com/chatgpt. [Accessed: 15-Oct-2024].