Multi-Agent Systems CSCE-475/875

Final Project Proposal

Bala Subramanyam Duggirala (UNL School of Computing; bduggirala2@huskers.unl.edu)

Elio Saldañana (Durham School Arch Engr & Const., UNL; esaldana2@huskers.unl.edu)

Problem Statement:

We plan to design a system of n agents where each agent tries to survive and win a battle between all the agents by carefully choosing if it wants to attack an opponent, defend from attacks, recover, or even try to form an alliance with an opponent to become stronger and increase its chances for survival. The problem is interesting because each agent has to carefully choose its actions in order to win the battle, either independently or as an alliance. In order to win the battle, either the agent or the alliance of the agent should be the last one standing where every other agent/alliance had run out of health. Choosing only to defend, the agent would not contribute to killing other agents, and would eventually get attacked by multiple agents and get killed. Choosing only to attack, the agent would run out of its health very quickly and die. Even the alliance formation is challenging because each agent in the system has an animosity level towards each of its opponents, which would drive the probability of forming alliances. With this project, our aim is to analyze each agent's behavior and look into how it is responding to different situations throughout the game.

Agent Design Strategy:

There are n agents in the system, each of the same type. Initially, every agent is an opponent to every other agent. This changes with alliance formations. Alliances can only be formed between 2-agents.

The set of actions available to each agent are:

1. Attack_Agent_(1-n)
2. Defend (This applies to attack coming from any other agent)
3. Recover
4. Form_Alliance_with_Agent_i

The set of attributes each agent possesses are:

1. Health (values of 0, 1, 2)
2. Attack_Level: This is a function of the health of the agent
3. Defence_Level: Also, a function of health of the agent
4. Animosity_vs_Agent_i_(1-n): This depends on how frequently an opponent attacks the agent. This is mutual between two agents
5. Its own alliance details

Each agent senses the following things:

1. An agent can sense if an opponent is attacking it, taking defense or recover actions.
2. An agent has information about the health of only the opponents whom it is attacking or y whom it is being attacked. This health information is refreshed only after the agent attacks its opponent once again or when the agent is being attacked again by its opponent. This applies to allies too. By default, they assume a full health of their opponents at the beginning of the game, and after a certain time after their last attack.

The transitions for each attribute of an agent are:

Health:

- If the agent is attacking or defending, health transition is a function of attacker's attack level, attacker's alliance's strength (value being 1 or 2), defender's defense level, defender's alliance's strength (1 or 2). Attack and defense levels are functions of the respective agents' health.
- If the agent takes the action-Recover, its health increases with a probability dependent on whether an agent is being attacked, and how strong the attack is.

Animosity:

- Animosity is used to form alliances. It does not influence any other factors in the game.
- Animosity is a mutual attribute between any two agents. The animosity increases with certain probability whenever an agent attacks another agent.
- The animosity decreases with a certain probability when the agent doesn't receive an attack from the opponent.
- The animosity of an agent regarding every other agent, including the alliance member (just to prevent alliance formation) changes to a maximum value, once the alliance is formed, making it impossible for the agent to form any more alliances.

Alliance formation:

- Each agent can form an alliance with just 1 other agent in the game. The transition function for this depends on the animosity between the two agents. The higher the animosity, the lesser the chances of two agents getting into an alliance.
- Once the alliance is formed, we update the animosity of each agent on the team to a maximum value so that no other agent can form an alliance with the members of this alliance.

Design of rewards: The goal of each agent is to win the game by killing all its opponents (non-alliance members), and not die in the process. Hence,

- The rewards are a function of
    1. Agent's own health (To prevent the agent from reaching 0 health)
    2. Health of the agent's ally (To prevent the agent from attacking their ally)
    3. Health of the opponent the agent chooses to attack.
- Reward =
  + f(agent's own health)
  -f(agent's ally's health)
  +f(1/opponents' health the agent chooses to attack.

Agent's Decision Making:

Each agent would pick the best action using the Markov Decision Process, by thoroughly updating the Q-values associated with each state, based on the above actions, transition probabilities, and the reward function. We plan to use the Q-learning algorithm for this purpose.

<u>Environment Design and Game Rules:</u>

The game starts with the n agents given a random initial animosity against each other. Every agent can attack every other agent as long as they are alive. An agent with their health reaching 0 is considered dead, and cannot be recuperated. Each agent takes part in the game until all their opponents (members of an alliance are not opponents among themselves) are dead. If there are only members of one alliance left alive, all of them are declared winners. Note that some of the alliance members could even have died during the game, and they are not considered to be winners.

Rules for alliance formations:

- The agents can only form alliances of two.
- One agent can decide to form an alliance with another agent, and this comes to fruition based on the probability associated with the animosity between them.
- An agent is allowed to betray its alliance member, and form alliance with another agent. This depends on how willing the other agent is to form an alliance. Also, an agent betraying another agent leads to increased animosity between them.

Simulation Logic:

1. An iteration starts for each agent in the system.
2. Each agent picks the best action according to the value-iteration of MDP.
3. The picked action is executed only at the end of each iteration and the next state is obtained according to the transition probabilities before the next iteration begins.
4. The iteration continues until the winners are declared.

Desired Emergent Behavior:

The desired emergent behavior is that each agent should try their best to survive the battle. They should attack the right opponent at the right time and defend whenever necessary. They also have to be cautious about choosing to form an alliance because they would lose a chance to not only attack in that time-step but also to defend themselves from attacks of other agents.

Hypotheses:

1. Decreased values of initial animosity should encourage agents to form alliances quickly in the game, despite the risk associated with being attacked without defense, and losing a chance to attack. We think the agents would do this because, as a team, they have a better chance of winning.
2. Even if we remove the penalty for attacking their allies, the agents would still learn to refrain from attacking them because, as a team, they have a better chance of inflicting damage to their opponents, and defending themselves.
3. Instead of each agent starting with full health, what can happen if some of the agents start with very low health? We think that we would see a pattern of healthy agents choosing to attack more, and agents with low health choosing to either recover or defend more, at least till their health reaches a certain level, after which they would begin to attack.
4. Since health of an agent does not change because of choosing to form an alliance in a particular time step, there would be no state transition. Since we are implementing Q-learning algorithm, the quality of choosing this action would be highly dependent on exploration, since the agent would know the impact of its decision only in subsequent state transitions. If not encouraged to explore more, the agents would not easily learn to form alliances.

Experiments: (corresponding to hypotheses index numbers)

1. We would give decreased animosity values in the initial distribution and check if there are any alliances forming quicker than otherwise.
   Experiment: 30 runs with high initial animosity levels vs. 30 runs with random initial animosity levels vs. 30 runs with low initial animosity levels.
2. We would tweak the reward system and remove the penalty for attacking an ally and then observe each agent's behavior regarding their allies.
   Experiment: 30 runs with NO penalty for attacking an ally = 0 vs 30 runs with some penalty for the same.
3. We would randomly distribute the agents' initial health, instead of each agent starting with full health.
   Experiment: 30 runs with low health for each agent vs. 30 Runs with full health for each agent vs. 30 runs with random health.
   We would then monitor the actions taken by healthy agents vs non-healthy agents.
4. We plan to experiment this hypothesis with multiple learning rates.
   Experiment: 15 runs with alpha = 0, independent
   15 runs with alpha = 1, time-independent
   15 runs with alpha = 0.5, time-independent
   15 runs each with alpha = 0.5, increases with time, and decreases with time

Work Allocation:

1. Elio will come up with the correct transition probabilities that would drive the game in a balanced way.
2. Bala will code the logic for transition probabilities.
3. Elio will implement the value-iteration algorithm for the MDP.
4. Elio will also work on the simulation logic for the MAS.
5. Bala and Elio will run and evaluate the experiments.