Reinforcement Learning-Based Auto Chess Implementation

Kaya Daylor and Aleksei Glebov

*Problem Statement*

The goal of this project is to develop an Auto Chess agent using Reinforcement Learning (RL). Auto Chess is a strategy-based game that involves deploying units with different costs, abilities, and synergies, requiring the agent to make decisions on unit placement, itemization, and positioning. These decisions are based on game states which are constantly changing.

The problem we are working on is designing an RL model that can autonomously play Auto Chess while learning optimal strategies for the game. The challenge lies in the complex nature of the game, the large state and action space, and the need for the agent to adapt to its strategies based on both random and opponent-driven elements in the game. The RL agent must navigate these challenges without relying on predefined strategies but rather by learning directly from its interactions with the environment.

This project will explore the development of an RL agent capable of performing well in Auto Chess while addressing challenges like delayed rewards and decision making under uncertainty.

*Proposed Reinforcement Learning Techniques*

We propose using an RL approach tailored to a simplified Auto Chess environment with a defined economy, leveling system, and unit mechanics. A Deep Q-Network (DQN) would be able to handle the decision-making process, especially the shop decisions and money management. The agent needs to be able to evaluate the trade-offs between immediate rewards such as buying units and long-term rewards such as saving money. To address the multi-faceted nature of decisions, a DQN structure would separate state value estimation from specific action advantages. This would help distinguish between overall board states and individual moves. For handling the complex relationship between unit placement, synergies, and simplified abilities, we suggest using a neural network architecture that is able to recognize these patterns.

*Expected Challenges*

Due to the complex nature of Auto Chess, we expect this project to come with some significant challenges. It will be difficult to navigate a dynamic environment with large state and action spaces. The agent must learn to balance immediate rewards with long-term rewards while also adapting to both random and opponent-driven actions. Key challenges include effective synergy management, gold management, and unit placement.

To mitigate the complexity, we propose converting our implementation to a 2-player version instead of 8 players. Additionally, we can reduce synergies to the lowest amount possible and unit abilities to no more than 3 distinct categories for all the units. We will also save the bench and unit level mechanics, but delete the itemization factor.

Additionally, in original Auto Chess games it takes about 1 minute for a round to finish, which will cause us time consuming challenges, since one game takes about 30 minutes to complete.

*Datasets or Simulation Environments*

We will implement our custom simulation environment and collect the data along the way. Since this is going to be a new environment, there are no opponents yet, so we will introduce four different opponents to our RL agent:

1. Random Agent - this opponent will perform random actions the entire game. This can become either a VERY weak opponent due to its randomness, or some sort of a first-time player, who is not sure what to do, but tries its best.
2. EarlyDestroyer Agent - this opponent will perform a popular strategy among new players: it will try to maximize one synergy and its characters on very first levels. We expect this agent to be very strong right from the start, but it will become weaker as the game proceeds.
3. MidChillGuy Agent - this opponent will perform another basic game strategy, which requires choosing a figure of 2 or 3 cost and playing around it (i.e. maximizing the figure's level on a certain level, building the entire board to protect that figure, and providing this figure with some synergy). We expect this agent to be weaker in the early and later stages of the game, but it should win most of the fights in the mid game when it becomes much stronger because of its figures.
4. LateGameSpecialist Agent - this opponent will perform one of the best game strategies: it will collect gold and push levels, searching for high level figures. We expect this agent to perform poorly in early game, average in mid game, and very strong in later game states.

*Evaluation Metrics*

The primary evaluation metrics we will use are win rate, gold efficiency, board positioning effectiveness, and learning speed. The win rate measures the agent’s success for each game with the goal of increasing this metric overtime. Gold efficiency will evaluate the agent’s gold management, ensuring that it spends gold efficiently on resources. The board positioning effectiveness metric assesses the placement of units and their consequences. Lastly, we will keep track of learning speed to monitor how quickly the agent is able to adapt and improve.