Implementing an agent with reinforcement learning elements for Scripts of Tribute card game

(Implementacja agenta z elementami uczenia przez wzmacnianie do gry karcianej Scripts of Tribute)

Rafał Stochel

Praca inżynierska

Promotor: dr Jakub Kowalski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

2 września 2024

Abstract

The main idea for writing the thesis was to test machine learning methods in the card game environment. The "Scripts of Tribute" project was used for this purpose. This is an implementation of the "Tales of Tribute" mini game from "The Elder Scrolls Online" title that allows you to write your own agents. Using artificial intelligence solutions, the computer program should obtain the best possible results during the game.
The created code was submitted to the Tales of Tribute AI Competition (TOTAIC), held as part of the IEEE Conference on Games 2024 (IEEE CoG). The following text describes: rules of the game, TOTAIC competition, designed program and its results.

Głównym zamysłem na napisanie pracy było przetestowanie metod uczenia maszynowego w środowisku gier karcianych. Użyto do tego celu projektu "Scripts of Tribute". Jest to implementacja mini gry "Tales of Tribute" z produkcji "The Elder Scrolls Online" umożliwiająca pisanie własnych agentów. Przy użyciu rozwiązań z obszaru sztucznej inteligencji program komputerowy ma uzyskać jak najlepsze rezultaty podczas rozgrywki.
Stworzony kod został wysłany na zawody Tales of Tribute AI Competition (TOTAIC), odbywające się w ramach konferencji IEEE Conference on Games 2024 (IEEE CoG). Poniższy tekst opisuje: zasady gry, konkurs TOTAIC, wykonany program oraz uzyskane przez niego wyniki.

Contents

Chapter 1

Introduction

In recent years we have seen an increase in the achievements of artificial intelligence in playing games of all kinds. Starting with the important event of constructing a Deep Blue[1] computer system by IBM. It was a machine that could play chess at a master level. In 1996, a match was held between the reigning world champion Garry Kasparov and Deep Blue, which ended with a score of 4-2 for the human player. One year later, in 1997 there was a rematch, but this time computer player was the winner. In those days, it was a groundbreaking event that showed possibilities of artificial intelligence. Another significant program was AlphaGo [1] by DeepMind - board game Go agent. During Future of Go Summit event in 2017 it defeated Ke Jie, a number one ranked player in the world. Go is considered even more complicated for computers than chess, due to the greater number of possible games states and a more abstract evaluation of the board. The next complex problem approached was StarCraft II. This is a real-time strategy with a theoretically infinite set of options to make. Resource management, buildings placement, selecting the appropriate strategy to deal with the enemy, micro management of units, all of these aspects had to be done by the machine. AlphaStar (also by DeepMind) [2] was presented in 2019 and it was able to meet these requirements. What's more it managed to beat top tier professional players. Although the AI was doing in-game tasks with superhuman speed, the ability to understand such a complex game so well is still impressive.
Card games seem like a great area to verify the capabilities of artificial intelligence. A computer program faces problems such as reasonable deckbuilding, searching through multiple game states, evaluation of current situation and predicting the opponent's possible moves. To test these ideas Script of Tribute [3] project and related tournament - Tales of Tribute AI Competition [4] have been made. They allow to create own bots and then compete with other programs. This paper describes an agent, designed to check how reinforcement learning methods can help to obtain a better result in clashing with other programs.

[1] https://www.ibm.com/history/deep-blue

Chapter 2

Scripts of Tribute

Scripts of Tribute (SoT) is an implementation of card mini game Tales of Tribute from The Elder Scrolls Online MMORPG game. Project was done using C# programming language and .NET platform. Gameplay focuses on a duel between two players. The first person to score a certain number of in-game points wins.

2.1 Game rules

Game cards are splitted into eight decks, which are named after their Patrons. Before the game starts, players have to choose which decks will be available during the play (Treasury deck can't be selected and is always available in the game). These decks will create a Tavern. This is a set of possible cards to buy during the play. Then the turn-based duel starts.
In general turn looks like this: five cards are added to the hand of player from the Draw Pile. If Draw Pile is empty, then Cooldown Pile cards are shuffled and moved into Draw Pile. Player can play Action and Agent cards from hand. Alive Agents can be activated. Then player decides whether to spend resources on purchasing new cards (sometimes it is not worth to get anything) or on calling patrons (four selected before the game + Treasury). Tavern can only have 5 possible options at a time. Bought cards are replaced with new ones. Lastly Power points can be used to attack enemy Agents.
There are three resources in the game:

- Coins - basic currency. With Coins we can call some Patrons and buy cards from Tavern, which will extend our current deck. All gained Coins are cleared at the end of turn.

- Power - with Power we can call Patron or attack enemy agents, possibly remove them from the board and move to the Cooldown Pile. Remaining Power is converted to Prestige at the end of turn.

- Prestige - main resource. Winner is determined based on these points. Usually obtained from Power. Sometimes can be gained directly from played cards.

Each competitor starts with 10 cards in deck, which are called Starters. They are mainly used to gain Coins. Cards are divided into four groups:

- Action - after playing its effects are generated. Goes to the Played Pile after purchase. When turn is over, Played Pile is moved to Cooldown Pile.
- Agent - card that can be placed on the board. Each round, if agent is alive, player can activate it, to perform its actions. Goes to the Cooldown Pile after purchase and after losing health points.
- Contract action - is played instantly after buying. Goes to the Discard Pile after playing. Cards remain on Discard Pile to the end of game.
- Contract agent - is placed immediately on the board after buying. Goes to the Discard Pile after losing health points.

Patrons are a significant tactic of the game. Each patron can be in one of three states: favors player, neutral, favors enemy. By activating them, player can trigger various effects depending on current state of patron. Brief description of available patrons:

Patron	Cost	Effect
Ansei	2 Power	Gain 1 Coin and gain 1 coin at the start of your turn if favored by Ansei
Duke of Crows	All Coins	Gain Coins - 1 Prestige, can be activated if not favored
Hlaalu	Sacrifice 1 played card	Gain Cost of sacrificed card - 1, points of Prestige
Orgnum	3 Coins	Gain Power based on number of owned cards
Pelin	2 Power	Move 1 agent from your Cooldown Pile to Draw Pile
Rajhin	3 Coin	Add empty card to opponent's deck
Red Eagle	2 Power	Draw 1 card
Treasury	Sacrifice 1 played card	Gain Writ of Coin (+2 Coins effect)

Another important aspect of the game are Combos. When multiple cards from one deck are played during one turn, they can produce additional effects. Tracking your own and enemy's possible combinations is crucial to winning the duel.

Player can win the game in one of three possible ways:

- Get favor of four patrons.
- Gain 40 or more points of prestige. Now game enters "sudden death" mode. If any player has fewer prestige points than his opponent at the end of his turn, he loses the game.
- Gain 80 or more points of prestige. After that, game immediately ends.

More detailed description of rules can be found on fan-made sites[1].

2.2 SoT GUI examples

SoT project provides an application with graphical interface. It is incredibly helpful during fixing and improving the agent. Program can be also used to practice your own Tales of Tribute skills.

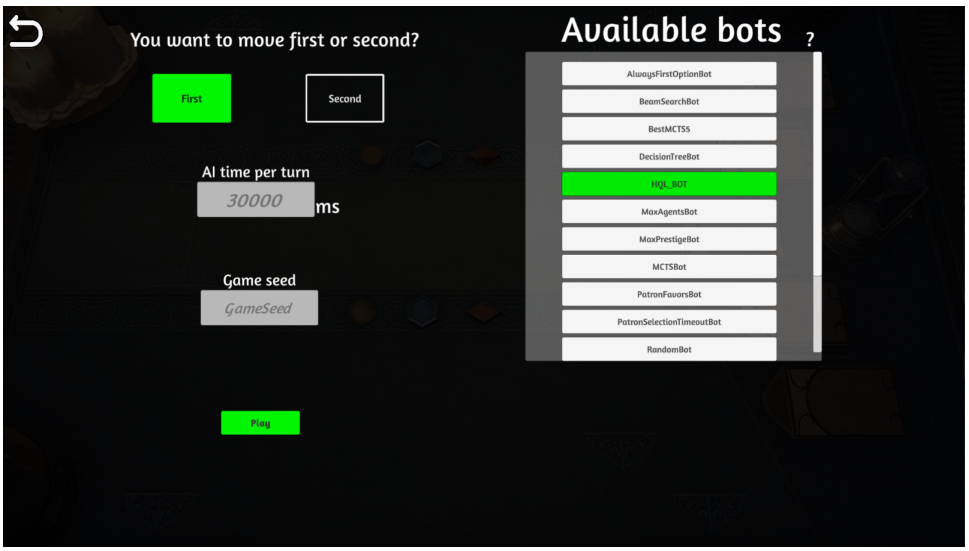

Figure 2.1: After choosing an option to start, we have to pick our enemy

[1] https://eso-hub.com/en/guides/tales-of-tribute-guide

Figure 2.2: Selecting patrons

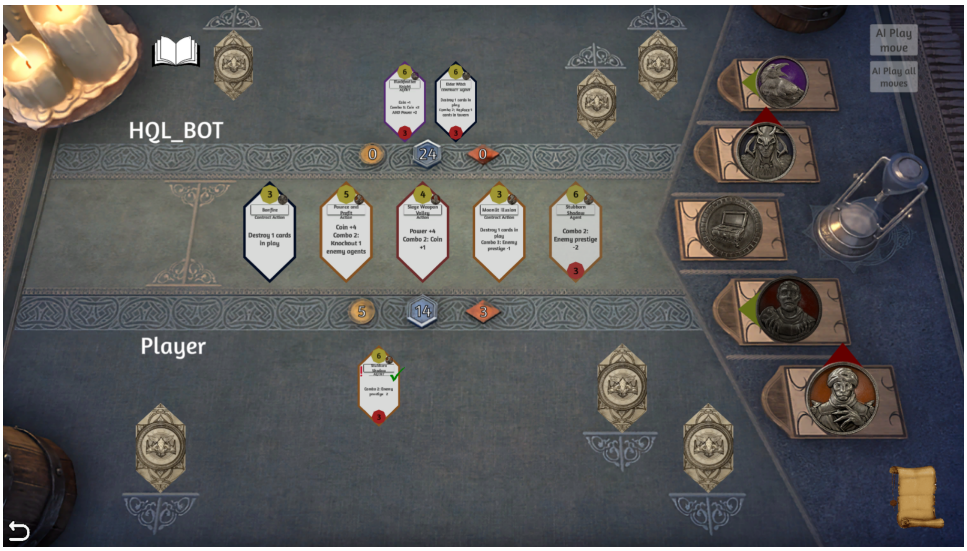

Figure 2.3: Example gameplay

Figure 2.4: Logs are a great way to debug written bot

2.3 How to create an agent

To write a bot user has to create a new class that inherits from provided "AI" class. Game engine initializes the instances of two bots that are going to clash. Then at specific moments it calls functions on both instances. Agent has to implement these three methods:

- **PatronId SelectPatron(List<PatronId> availablePatrons, int round)**
  Each game starts with this method. Agent has to select which Patron will be available during the game from list of available patrons. Round describes which time method was called (four selections overall).
- **Move Play(GameState gameState, List<Move> possibleMoves, TimeSpan remainingTime)**
  Essential point of every agent. From list of possible moves bot should return one, which will be played. GameState objects describes current state of board. Time spent on move selection should not exceed remainingTime value. Game is lost if illegal move is returned.
- **void GameEnd(EndGameState state, FullGameState? finalBoardState)**
  Auxiliary function used to debug bot's behavior based on EndGameState and FullGameState objects. Gets called after the game is over.

When program is done, we can test it in two ways: in GUI described earlier or using GameRunner command line interface. Basic command to run one game:
GameRunner.exe BotA BotB.
There are also additional flags to modify testing: **-n X** - run X games; **-s X** - run game with specified seed; **-t X** - run games on X threads.

Chapter 3

Agent implementation

Created bot consists of three main parts: searching through game states with Beam Search algorithm, usage of Q-learning method to determine the value of card, evaluating given state with heuristic function. The idea behind agent's implementation was to check whether machine learning can be helpful in rating cards.

3.1 Beam search algorithm

When agent is given multiple moves options in Play() method it must somehow decide, which one is best. Beam search procedure can be summarized using the following pseudocode:

1. Declare empty lists: *beamNodes*, *endNodes*.
2. Create nodes from given data and add them to *beamNodes*.
3. While *beamNodes* is not empty:
   (a) Declare empty list: *childrenNodes*.
   (b) Iterate over *beamNodes*: expand the given node and visit its children. If child is a final node (can't be expanded - has no child nodes) add it to the *endNodes* with calculated heuristic evaluation, otherwise add it to *childrenNodes*.
   (c) Remove k nodes from *childrenNodes* with worst heuristic score.
   (d) *beamNodes* = *childrenNodes*
4. Iterate over endNodes, return move from the element with the highest heuristic score.

3.2 Q-learning

Q-learning was selected from variuos reinforcement learning algorithms.

Main formula to calculate new q-values:

$$Q_{new}(S, A) = (1 - \alpha) \cdot Q(S, A) + \alpha \cdot [R(S, A) + \gamma \cdot \max_{a'} Q(S', a')]$$

Where variables are:

- S - state S of game
- A - action A that will be executed
- S' - new state of game, after action A is done
- $R(S, A)$ - reward function to check how good was taking action A in state S
- $\max_{a'} Q(S', a')$ - maximum reward that can be obtained from new state S', iterating over possible actions in S' - a'
- α - learning rate
- γ - discount factor

3.3 Heuristic

Chapter 4

Tales of Tribute AI Competition

Chapter 5

Agent testing

Chapter 6

Conclusions

Bibliography

[1] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016).

[2] Michaël Mathieu, Sherjil Ozair, Srivatsan Srinivasan et al. AlphaStar Unplugged: Large-Scale Offline Reinforcement Learning arXiv:2308.03526

[3] Dominik Budzki, Damian Kowalik, and Katarzyna Polak. Implementing Tales of Tribute as a Programming Game. Engineer's thesis, University of Wrocław, 2023.

[4] Jakub Kowalski, Radosław Miernik, Katarzyna Polak, Dominik Budzki, and Damian Kowalik. Introducing Tales of Tribute AI Competition. arXiv preprint arXiv:2305.08234, 2023