EPiC Series in Computing
Volume 82, 2022, Pages 131–140
Proceedings of 37th International Conference on Computers and Their Applications

Hearthstone Battleground: An AI Assistant with Monte Carlo Tree Search

Namuunbadralt Zolboot[1], Quinn Johnson[1], Dakun Shen[1], and Alexander Redei[1]

Central Michigan University, Mount Pleasant, Michigan, U.S.A.
{zolbo1n, johns1qr, shen2d, redei1a}@cmich.edu

Abstract

We are in the golden age of AI. Developing AI software for computer games is one of the most exciting trends of today's day and age. Recently games like Hearthstone Battlegrounds have captivated millions of players due to it's sophistication, with an infinite number of unique interactions that can occur in the game. In this research, a Monte-Carlo simulation was built to help players achieve higher ranks. This was achieved through a learned simulation which was trained against a top Hearthstone Battleground player's historic win. In our experiment, we collected 3 data sets from strategic Hearthstone Battleground games. Each data set includes 6 turns of battle phases, 42 minions for battle boards, and 22 minions for Bob's tavern. The evaluation demonstrated that the AI assistant achieved better performance — loosing on average only 9.56% of turns vs 26.26% for the experienced Hearthstone Battleground players, and winning 56% vs 46.91%.

1 Introduction

Hearthstone is an online digital collectible card game (CCG) that has reached critical acclaim and commercial success. According to Blizzard's data, the game had 23.5 million players worldwide in 2020[12]. Hearthstone has a few variations of game modes that players can choose depending on how they want to play the game. Of the game modes, Battleground[10] has been the most popular with a large player base. A typical battleground game consists of 8 players at the time by using selections of 54 heroes with unique powers and 128 different minions that have various effects and attributes. Players compete against each other through matches in a turn-by-turn style, with randomly assigned opponents, where heroes and minions have unique strengths and weaknesses. The goal of the player is to survive until all other opponents are eliminated. Due to a vast number of possible combinations with the use of minions and heroes, the outcome of the game can be drastically different. Hearthstone Battleground is an example of a game where each action has a high value. Even top players must adapt and use their given opportunities wisely since any decision may not result in long-term success. During the game, the players are unaware of each other's battle board and what combinations they have built until the match begins. The only information given to the player is the most common type of minion present, which can appear as mixed minions. It is difficult to predict what each opponent will do as players will change their combinations rapidly as everyone advances into higher tiers. Furthermore, since a player can perform many actions, each turn such as buy, sell, upgrade, change positions, refresh freeze, hero power, end turn, and combinations of said actions, Battleground mode is exceptionally complex when building a model that could outperform a higher-ranking player.

A successful Hearthstone Battleground AI assistant makes high-value decisions quickly, adapts to the randomness presented in the game through the minions selected for the player. In this paper, we leveraged Monte Carlo Tree Search (MCTS) to narrow down which actions have high value by simulating all the possible actions and determining the values of each action. In doing so, we built an AI that could assist players with advancing into higher ranks through use of MCTS. A key aspect of this research was the development of our own simulator of the Hearthstone Battleground. The experiment results prove that the proposed AI assistant achieves a good performance, in terms of winning rate and damage done.

2 Related Works

In recent years, Hearthstone has become a popular game for building simulators and AI research [13]. Many developers and researchers came up with phenomenal tools such as Hearthstone Deck Tracker, HReplay[15], HearthSim[14], Fireplace[16], Sabberstone[17], Bob's Buddy[20], and many more[9]. HearthSim started with a few community developers who had a passion for building Hearthstone Simulation for the purpose of helping players predict the battle win-rate using given information. From there, they have built an overlay application called Hearthstone Deck Tracker(HDT), which is widely used by players today.

HDT helps players to predict the win percentage of the battle phase by simulating 10,000 instances. When it is finished, it displays the win, tie, and lose percentages on the player's screen. In addition, HDT uploads every match history to their database website called HReplay. HeartSim has also developed and partnered with few other simulation tools like Fireplace, Sabberstone, and, Hearthbreaker[14]. Fireplace is Hearthstone simulator written entirely on Python 3. The purpose of the project is to capitalize on the Hearthstone CardDefs XML files in order to collect accurate data of game's cards and have a default implementations of all the minions [16]. It allows other projects like HearthSim to simulate games efficiently without worrying about minion original effects with use of API. It also helps to keep track of the game state. Sabberstone is another simulator written on entirely on C#. Sabberstone's intention is to build fast AI that could play and perform better than an average player on Hearthstone's standard mode. Developers of Sabberstone have implemented methods that can clone any board state of any Hearthstone game and simulate on its own application with the use of SabberStoneGui [17]. This method allows applications like HDT to adapt their simulation on any given game state.

There is also a Hearthstone AI Competition where teams compete against each other through the use of AI developed with specific decks. This competition ran from 2018 through 2020 [4]. Dockhorn Competition provides its own AI platform for competitors to use. The platform makes it easier for competitors to implement their AI without building a base simulator. Although the goal of the competition is building an AI that can beat professional players, the agent only makes decisions with a custom designed deck. Therefore, it does not work well when making complex decision that involves randomness and restricted information, as would occur in a real Hearthstone Battleground game.

Dota 2 is one of the biggest multiplayer online battle arenas (MOBA) game that has about half a million active players daily since 2013. MOBA type games often are incredibly complex to learn due to many factors that involves multitasking, strategic planning and paying attention to game state. Dota 2 is team-based game that has numerous possible combinations of heroes that players can choose. Dota 2 consist of 119 different heroes that categorize into carry, support, nuker, disabler, jungler, durable, escaper, pusher, and initiator. Each role has important role for building strong team. Ideal team wants to have a carry, a durable hero, a support hero, a jungler, and other roles depending on how the team wants to progress throughout the game. The goal of the game is to destroy enemies main base which is guarded by 11 towers and enemy heroes and every 2 minutes with minion waves spawning throughout the map. Heroes generate gold by killing enemy minions, heroes, and destroying tower. When players have some gold, there are about 208 items to buy from main shop and secret shop. Items helps heroes to get stronger such with benefits such as raw stat increase, special effects like invisibility, movement speed etc.[21]. In addition to the AI having to learn the basic mechanics, the agent needs to learn how to make a team play with this environment. OpenAI Five have made it possible to teach OpenAI Five to learn and think like real player through reinforcement learning since 2017. Developers have started to train AI agent to play one on one game against its previous version repeatedly recursively. After years of training, OpenAI Five have successfully achieve its current state with team composition and strong ability to make good team play. Currently OpenAI has played against many world champion teams and won against them in 2018. It shows that the AI could potentially get better at complex games like Dota 2 with endless possibilities of game mechanics. OpenAI is an artificial intelligence organization that is founded in San Francisco in 2015 by Elon Musk. The goal of the company is to research and develop AI so it would benefit humanity. Originally OpenAI was a non-profit organization, but it has changed to for-profit when Elon Musk resigned as its board member. Since then, OpenAI has successfully release various projects and solutions to modern AI world such OpenAI Gym, Infrastructures for Deep Learning, and evolution to Reinforcement Learning. It has changed state of AI since 2017 when they have started their OpenAI Five self-playing Dota 2 AI project. In 2018, OpenAI has successfully competed in the world championship of Dota 2 with the grand prize of 25 million dollars. Although OpenAI did not win the prize, they have successfully defeat world champions by incredible score by 2-0 [18]. Since 2018, team of OpenAI Five has been actively competing against top players of Dota 2 in the world.

Fundamentals of OpenAI Five is based on Large Scale Deep Reinforcement Learning [2]. Deep Reinforcement Learning is to think and learn like human by solving small problems until it has ability to tackle bigger problems.[2] Like human, agents should able act humanly, act rationally, think human, and think rationally. These concepts are essential when developing AI that can play just like humans [19]. In addition, agents can learn from small obstacle and challenges, and it helps them to handle bigger issue later. Furthermore, the idea of reward or punishment is main concept of reinforcement learning. If agents can solve the problem, it goes to next one whereas if it could not solve then agent would go back to smaller problem. By having its experience and data, the AI can make better judgement on related matters. Large Scale Deep Reinforcement Learning means that it runs with uses a lot more actions and observations to help agent learn. By using high number of CPU's and Memories LSDR is done faster and efficiently to train. Due to less complex and smaller size whereas Dota 2 requires long time horizons, partially observed state, high-dimensional state, and observation spaces. All these states must meet to make AI better at playing Dota 2. Although players play Dota 2 using mouse, keyboard, and monitor, the OpenAI Five does not need them to train against itself. OpenAI Five team designed their training models to run on the cloud server machine, so AI could learn the game from scratch from playing against each other millions of million iterations in short amount of time.

OpenAI Five team can analyze the game from beginning to see where and which areas agents could improve on. OpenAI Five had gathered about 3 million batch of data after running 10 months to face against top players of Dota 2 mostly one on one. Although AI was able maneuver and play against some interesting plays against them. However it was not nearly enough to beat top teams due to lack of predicting what the enemy team is going to do [1]. Even though team of OpenAI Five is has accomplished to train the basics of the games and fundamentals though large-scale deep reinforcement, they still need to manually script what items are recommended to buy for each hero which is done by surgery method [2]. Because the iteration to calculate each item's value to too high, it is easier to maximize the optimization of items manually. OpenAI Five is consistently improving itself every day since they have more time to run more simulations and train the AI agent through LSDR. However, when Dota 2 gets new updates or new mechanics, it's difficult to re-run the some of the databases that previous mechanics are used.

3 Methodology

3.1 Hearthstone Battleground Game Simulation

Creating an effective AI agent requires a good model of the environment. Building our own simulator was necessary because the average Hearthstone Battleground game lasts between 20 to 40 minutes, which was impractical for training our agent. In later turns of the game, the MCTS grows so large that the computations take exponentially longer. We can reduce the time it takes to simulate each game result with the use of additional computation power and optimized set rules. Developers of Hearthstone Battlegrounds are regularly updating the game to make it more entertaining for the players every month. However, it was time consuming for us to update all the changes that they had made while building the game simulation. For the purposes of our project, we have decided to stick with Hearthstone Battleground v20.0.2 for our agent[11]. With updates, our method could be extended for use on the most recent version. Building a simulation of Hearthstone Battleground was a key contribution of this paper, allowing us to simulate game results efficiently.

In this project, we used Python 3 to build a simulator of the Hearthstone Battleground game. The most important parts of constructing a simulation were building an accurate game system that has the exact same abilities and effects of minions and heroes as in the original game. It includes accurate game system, details of heroes and minions, actions during buy and sell phase, and most importantly the battle phase. We have painstakingly implemented precise models of all the minions with exactly same attributes described in the Hearthstone Battleground's official website[10]. Furthermore, building an exact copy of Hearthstone Battleground's Battle Phase was important for gathering better results. Implementation of Bob's tavern was paramount for the simulation since players interact with it by buying minion, selling minion, freeze, refresh, and upgrade. Not only, Bob's tavern helped us to simulate multiple turn interaction, it also added more interaction for more complex decisions leading up to MCTS. The patch notes 20.0.2 consist of 54 heroes, each hero has their unique ability. It was challenging to apply all the unique effects of the heroes due to making custom changes that requires to restructure battle phase and bob's tavern phase.

Although Hearthstone Battleground requires 8 players in the lobby, only two players will play against each other in one turn. Therefore, we only implemented the battle phase for two players in each turn. This strategy reduces the computational power needed significantly, even though we still need to calculate all possibilities of battle in every turn. Battles with the same minions often have a completely different outcome because the game mechanics rely heavy on randomness. There are 128 minions with various effects, that have been categorized into 19 different variations, with each variation including different ranks of the minion within. For example, we have combined all the minions that have the "Deathrattle" effect into the same category. The term "Deathrattle" refers to a mechanism that triggers when a minion dies during battle phase. The mechanism that occurs can be a multitude of different actions: creating other minions on the board, increasing the health and attacking of other minions, or impacting the game in some other way. Due to the large scale of how many unique variations of minions there are and the exponential effect of how they interact with each other, it is critical to make sure they are sectioned by each variation and that they work properly. Thus, the solution created was to categorize down further than base mechanisms and be as specific as possible. Condensing into further subcategories also improved the simulation's ability to handle randomness. For example, there are "Deathrattle" mechanisms that can select and create a minion from a large pool of different minions. This large scale of randomness is also seen in other variations of minions. However, by condensing into very specific categories, we can predict what the variations might be doing and thus know if our results are accurate or not.

Finally, we built generic battle phase functions and battled one-on-one to compute the outcome of the battles. When we generate battle between players, we make a deep copy of their board states first. Then we determine which players have the highest number of minions on their board and player with higher number of minion attacks first. However, if both players have the same number minions on the board, the game will choose randomly. The result of the battle differs radically depending on who gets to attack or defend. Although there are advantages to attacking first, for some cases it can be quiet opposite. Our simulation tracks the full history of each player action including who wins or loses, and the resulting game data such as damage taken. We use this data to evaluate the outcome of each decision.

3.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is an algorithm best suited for evaluating complex game simulations and predicting the best possible move [3]. This makes it ideal for our application. With the implementation of a MCTS, our model accurately discovers all the possible nodes and makes suitable moves depending on the information it's given. Our MCTS model follows the basic concepts of regular tree search, but with four core fundamentals that distinguish it from regular tree searches. MCTS finds the best move by selection, expansion, simulation, and back-propagation as explained below.

1. ***Selection.*** The Selection starts at the root node of the tree. It recursively determines the next node to be selected among its child nodes based on the node's Upper Confidence Bound (UCB) value. This step selects the child node with the highest UCB value, which represents the best action the player can make under a given circumstance.

   The UCB value is calculated based on the equation shown below:

   $$UCB(node_i) = \frac{w_i + d_i}{n_i} + c\sqrt{\frac{logN_i}{n_i}}$$

   where $node_i$ is the UCB value of $node_i$, w_i is the winning rate of $node_i$ after simulation, d_i is the average damage made of $node_i$ after simulation, n_i is the number of visits of $node_i$, c is a constant value, and N_i is the number of visits of the parent of $node_i$.

2. ***Expansion.*** After a child node is selected from the Selection step, all the possible actions for the node are discovered in the Expansion. To get the best action for the player,

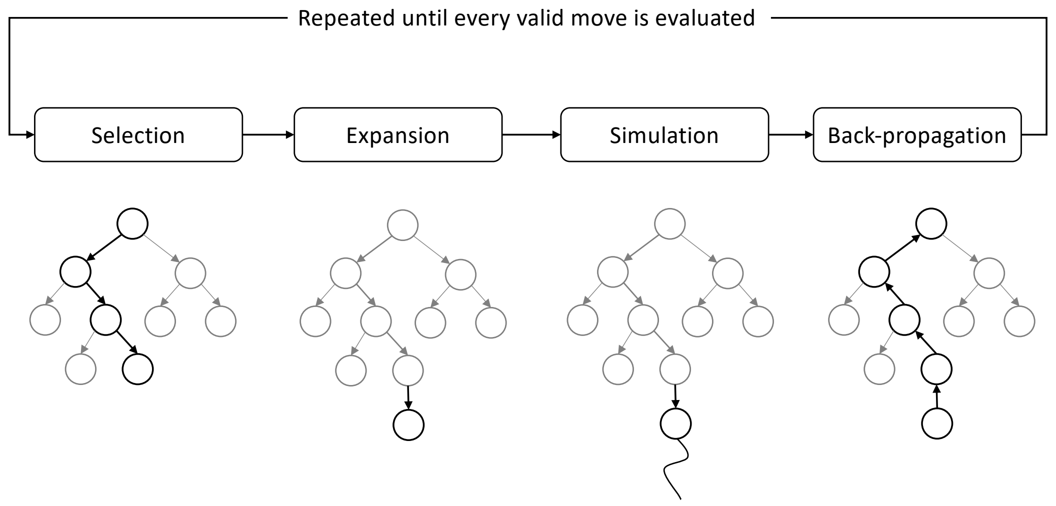


Figure 1: A diagram of the four core fundamentals in our MCTS implementation

it's crucial to check all the attainable actions that the player could make in each turn, including buying minion, selling minion, playing minion, and upgrading Bob's tavern. After the Expansion randomly selects an action among all the possible actions, it creates a node for the action and appends it as a child node to the tree. This step keeps selecting actions until there are no more actions that can be made.

3. ***Simulation.*** After the Expansion adds a child node, the Simulation begins to evaluate the performance of the node. Specifically, it simulates the battle between the player and a random opponent to determine the wining rate and the average damage taken for the selected action. Because of the randomness of the game, each simulation simulates the battle phase multiple times to get the most accurate outcomes. We simulated the battle phase 200 times for each node, which is further described in section 4.

4. ***Back-propagation.*** Once the simulation is finished, the Back-propagation returns to the root node. While traveling back to the root, all the child node values are updated, including the iteration time, the number of visits, and the UCB value. After the Back-propagation completes, the model goes back to the Selection to find the next action.

Figure 1 shows the diagram of our MCTS algorithm. The four steps are repeatedly executed to test every possible action under a given state. Each iteration selects the node with the highest UCB value, which represents the best move under the current game state. In later phase of the game, players have access to more resources. Hence the number of action combinations grows at an exponential rate. As a result, the MCTS algorithm takes exponentially more time to evaluate all possible actions. To reduce the computing power, we added some limitations and optimizations to reduce the number of actions to simulation in each turn. More details can be found in Section 5.

4 Evaluation

4.1 Simulator Evaluation

Hearthstone Battleground Simulator includes 128 models of minions with special attributes and abilities. Minions with similar effects are categorized together. Categorization allowed us to implement minions efficiently and precisely. This includes minions that trigger its effect at the start of the game, end of the game, during the battle, during the buy sell phase, continuous, divine shield, overkill, deathrattle, and battlecry, as mentioned in Section 3.1. After the implementation of the minions, it was important to build an accurate game system that interacts with Bob's tavern with actions such as buy, play, sell, upgrade, freeze, refresh, change positions, and end turn. Therefore, we extensively tested Bob's tavern functions for correctness. Since Bob's tavern generates random minions based on the current player's tier, we simulated the Bob's tavern generation method more than 100 times to check if it accurately generates minions for each tier. In addition, we also examined all other Bob's tavern mechanics with implementation of basic battle phase. Initially we only tested minions with no special effects in battle phase to evaluate the calculations. Then we started to use minions that have less complicated abilities such as deathrattle and battlecry. When we deal with minions with the battlecry effect, we need to make sure the effect was correctly triggered. Minions with the deathrattle effect were by far the easiest minions to check since it was purely based on if the minion died or not.

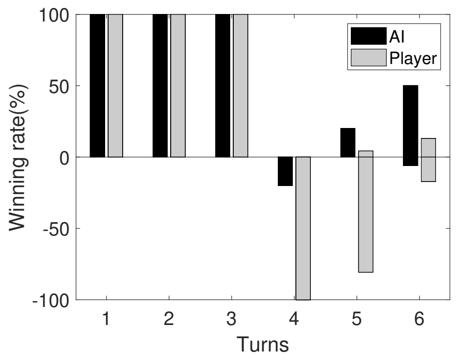
For testing purposes, we randomly generated minions from the database and used a single turn battle simulation to test the correctness of battle phase. After the battle simulation, we manually checked the result to ensure that more complex minions work as intended. As we were testing different scenarios with the simulation, we faced difficulties implementing the minions that trigger their effect twice: such as during the start of battle phase and during the buy phase. Some of the issues caused by trigger effects were fixed easily, however there are some cases we had to rewrite completely.

Furthermore, we have made sure all the players actions work as intended and corresponds to all other minion's abilities. Thus, we can accurately determine which actions have highest value with using of MCTS. It's important to calculate battle win rate for each player by running simulation at least 100 times to find the ratio between players for each action sequence. Evaluating the win-rate per-action helps our agent to learn the patterns. This was achieved by optimizing the UCB value.

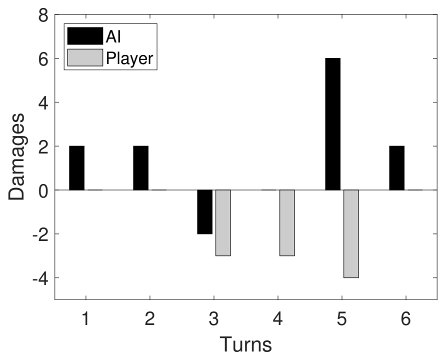
4.2 AI Assistant Evaluation

To evaluate the effectiveness of our AI assistant of the game, we collected 3 data sets from a rank 1 competitive Hearthstone Battleground player whose alias is Dogdog[7]. These data sets represent strategic wins for Dogdog which we aimed to emulate or beat with our AI assistant. Each data set includes 6 turns of battle phases, 42 minions for battle boards, and 22 minions for Bob's tavern. In this evaluation, we applied the data sets into the AI model and evaluated whether the model can achieve the same or better results as the rank 1 player. The win rates and player damage taken are demonstrated in Figures 2, 3, and 4.

A highlight is the skill of our AI as shown in Figure 2. On average, the assistant wins 56% of turns vs 46.92% for the rank 1 player, and looses only 9.56% vs 26.26% of turns. Figures 2 (a), 3 (a) and 4 (a) highlight this effect and demonstrates that our AI assistant crushes the expert player. Specifically, in turn 6 of Figure 2 (a), the AI assistant achieves a winning rate of 50% and a losing rate of 6%, while the expert player had a winning rate of 13% and a losing rate of 17.1%. Top players like Dogdog gamble on high risk and high reward plays to make for

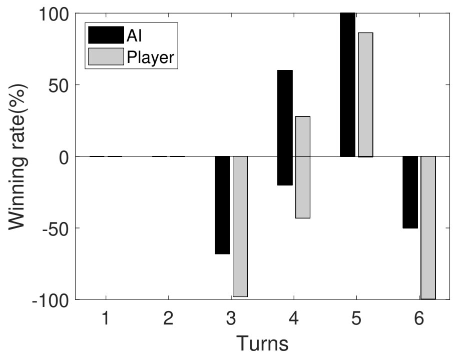


(a) Winning rates

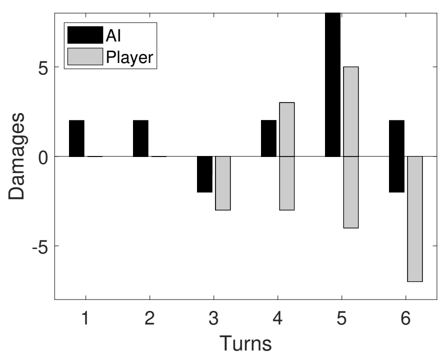


(b) Damages

Figure 2: Simulated results of Dogdog's battle with our AI assistant (data set 1) [5]

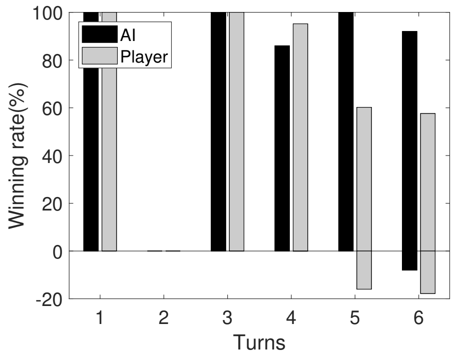


(a) Winning rates

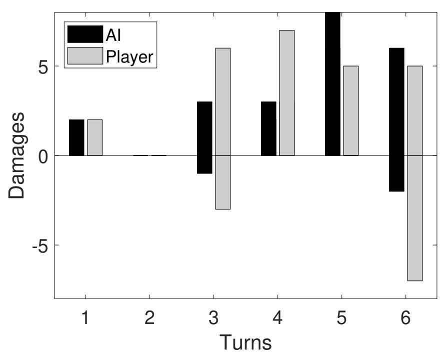


(b) Damages

Figure 3: Simulated results of Dogdog's battle with our AI assistant (data set 2) [6])



(a) Winning rates



(b) Damages

Figure 4: Simulated results of Dogdog's battle with our AI assistant (data set 3) [8])

an exciting game. The AI assistant, by contrast, chooses safer and stable plays, which higher likelyhood of success in preserving the player's Health Points (HP).

In addition, our model focused more on dealing higher damage in the early phase, as shown in 2 (b), and 3 (b). Specifically, in turn 5 of Figure 2 (b), the AI assistant made a damage of 6 to the opponent, while the expert player only made a damage of 4 to the same opponent. Making more damages to other players forces them to play defensively rather than rush through higher tiers quickly if players want to survive the early phases.

Throughout the extensive simulation, Hearthstone AI assistant came up with some fascinating decisions compared to a top player. It was done by optimizing the use of sell mechanics, board space and calculating every single UCB outcome with use of MCTS in later turns.

5 Discussion

5.1 Limitations

While MCTS's performance does not slow down much before turn 6, it takes a lot longer to calculate remaining turns. Initially, each player starts with 3 coins and 3 selections of minions from Bob's tavern. Not only does Bob's tavern offer additional unique minions as players upgrade their tavern tier, but it also adds an extra slot for a minion to appear each turn at tavern upgrades 2, 4, and 6. Therefore, the MCTS tree becomes massive requiring more computational resources by the third tavern tier upgrade. Currently, our model focuses on one-on-one battles up to turn 6 due to not having enough computational resources beyond that.

5.2 Future Optimizations

Deciding when the agent will optimize sell mechanics can be crucial to reduce computational power that it takes to calculate MCTS. The sell action can be tricky since the agent needs to calculate each outcome after using sell action. Players often tend maximize their resources by predicting future outcomes, however, that isn't feasible for our model given the resource constraints at higher turns. Nevertheless, the calculation time can be reduced by optimized set rules. Currently, the model uses optimization rules such as do not sell if the player has less than 2 coins, but it can be further enhanced by a more better set of rules. For example, the agent should be able to calculate the best use of coins by calculating the upgrades and most valuable minions first, then figure out how many coins it would need to do those actions rather than trying to sell it first. In addition, current set rules can be upgraded more with use of fixed coin number like 2, 5, 8. It would help the agent maximize the impact of their coins and buy as many valuable minions as possible.

6 Conclusion

We modeled the 3 strategic wins of a rank 1 - expert level - Hearthstone Battleground player whose alias is Dogdog. We developed an AI assistant that outperforms the expert player, helping you achieve higher ranks and a better understanding of the game. This was a complex challenge involving the implementation of 128 minions, each with their own attributes and special abilities. In addition, we simulated 54 heroes and all outcomes up to turn 6. In the future, more computational power and a few optimizations would allow us to simulate the entire match (beyond turn 6). We custom built a simulator that allowed us to train efficiently and gather accurate results without having to wait 20-40 minutes for the game to play at human speed. In conclusion, our AI assistant achieves better performance in terms of win rate, damage dealt, and loose rate.

References

[1] Open AI. Open ai five. https://openai.com/projects/five/, Jul 2021.

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[3] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[4] Alexander Dockhorn and Sanaz Mostaghim. Introducing the hearthstone-ai competition, 2019.

[5] Dogdog. At the End of the Turn, Gain +21/+24 — Dogdog Hearthstone Battlegrounds. https://www.youtube.com/watch?v=IrLGJGikcos&ab_channel=dogdog, March 2021.

[6] Dogdog. George vs George: Bubble Battle — Dogdog Hearthstone Battlegrounds. https://www.youtube.com/watch?v=syrrusvSSNs&ab_channel=dogdog, March 2021.

[7] Dogdog. Hearthstone Battleground Player: Dogdog. https://www.youtube.com/channel/UCdKdlJV1DsRHtKMBPqEhJww, 2021.

[8] Dogdog. The Whole Game Comes Down to a 50/50 — Dogdog Hearthstone Battlegrounds. https://www.youtube.com/watch?v=hGE3jvCbGhs&ab_channel=dogdog, April 2021.

[9] Teng-Sheng Moh Dylan Wang. Hearthstone ai: Oops to well played. In *ACM SE '19: Proceedings of the 2019 ACM Southeast Conference*, 2019.

[10] Blizzard Entertainment. Battlegrounds. https://playhearthstone.com/en-us/battlegrounds, 2020.

[11] Blizzard Entertainment. 20.0.2 patch notes. https://playhearthstone.com/en-us/news/23658923/20-0-2-patch-notes, Apr 2021.

[12] Blizzard Entertainment. Year of the phoenix in review. https://playhearthstone.com/en-us/news/23625669, Nov 2021.

[13] Pablo García-Sánchez, Alberto Tonda, Giovanni Squillero, Antonio Mora, and Juan J. Merelo. Evolutionary Deckbuilding in Hearthstone. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, 2016.

[14] HearthSim. Hearthsim: Hearthstone Simulation & AI. https://hearthsim.info/simulators/, 2015.

[15] HSReplay. Battlegrounds status. https://hsreplay.net/battlegrounds/heroes/?hl=en, 2020.

[16] Jerome Leclanche and Benedict Etzel. Fireplace Hearthstone Simulator. https://github.com/jleclanche/fireplace/wiki, 2015.

[17] Milva and Darkfriend77. Sabberstone: Hearthstone Simulators. https://github.com/HearthSim/SabberStone, 2016.

[18] OpenAI. Open AI Five Defeats Dota 2 World Champions. https://openai.com/blog/openai-five-defeats-dota-2-world-champions/, Sep 2020.

[19] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach*. Pearson, 3 edition, 2009.

[20] Jonathan Segal. Introducing bob's buddy. https://articles.hsreplay.net/2020/04/24/introducing-bobs-buddy/, Oct 2020.

[21] Fandom Dota 2 Wiki. Fandom wiki: Dota 2 role. https://dota2.fandom.com/wiki/Role, 2017.