TradingStone: Benchmark for Multi-Action and Adversarial Games

Manuel Del Verme, Simone Totaro,[1] Wang Ling[2]
[1]Universitat Pompeu Fabra, [2]DeepMind
manuel.delverme@gmail.com
simone.totaro@gmail.com
lingwang@google.com
Department of Information and Communication Technologies
Carrer de Tànger, 122-140
08018 Barcelona, Spain

Abstract

This paper introduces TradingStone, a simplification of the zero-sum game trading card game HearthStone that preserves the core elements of the game that challenge existing agents, namely: multi-action turns that require the management of a pool of resources; partially observable boards; Highly dimensional card spaces. However, it removes elements that are specific only to certain cards that render many existing approaches unusable, such as, effects that generate new arbitrary cards that lead to very sparse search spaces. We show that the core elements that can be studied with TradingStone are still unsolved by existing search and RL approaches and as such, it is a milestone that must precede solving Hearthstone.

Introduction

Hearthstone: Heroes of Warcraft is a zero-sum game that represents a multifaceted challenge to RL. Imperfect information, sthochasticity and longer-term planning represent only a fraction of the elements that Hearthstone agents need to master in order to excel at the game. While existing approaches attempt to tackle the full complexity of the game (Zhang and Buro 2017; Santos, Santos, and Melo 2017; Swiechowski, Tajmajer, and Janusz 2018), the approaches that have been applied have been limited to small variations of Monte Carlo Tree Search (MCTS), as the complexity of full Hearthstone game makes the application of many other solutions prohibitive. Examples include Counterfactual Regret Minimization (Zinkevich et al. 2007) and Information Set MCTS (Cowling, Powley, and Whitehouse 2012), which establish higher speed and memory requirements than pure MCTS. Thus, we argue that more restricted versions of the game that focus on the study of certain elements of the game such as imperfect information or stochastic would be beneficial to the existing researchers as it would allow such approaches to be validated as an intermediate milestone, prior to scaling them to the full game. This is analogous to the research in Poker(Zinkevich et al. 2007), which was initially conducted in simpler versions of the same game before work on more complex variants were developed (Brown and Sandholm 2017b; Moravcík et al. 2017).

In this work, we propose TradingStone, a restricted counterpart of Hearthstone that eliminates the non-core elements of the game, while preserving the key research challenges in the game. The game remains a two-player adversarial game, where at each turn players manage a pool of resources that must be managed and strategically viable actions must be chosen in order to win the game. More importantly, we isolate the stochastic and imperfect information elements of the game. We believe that both issues constitute important general problems in the field of RL and our framework allows the testing of each of these problems in isolation or jointly by providing different versions of the game.

Under our restricted version of the game, we also test the application of IS-MCTS (Cowling, Powley, and Whitehouse 2012) validate its usefulness in modelling the imperfect information elements present in Hearthstone. We show that our implementation of IS-MCTS in this environment compares favourably to previously implemented approaches for HearthStone based on PI-MCTS.

Related Work

Hearthstone belongs to the category of sequential stochastic games with imperfect information. Most methodologies that have been employed to play full HearthStone games have relied on Monte Carlo Tree Search (Santos, Santos, and Melo 2017). Here, PIMC (Perfect Information Monte Carlo) Tree Search, where hidden information the each game state is filled randomly converting the problem into a perfect information search problem. While variants of MCTS exist for imperfect information games exist, such as Information Set MCTS (Cowling, Powley, and Whitehouse 2012), only partial implementations have been proposed due to the complexity of the game (Swiechowski, Tajmajer, and Janusz 2018). Another relevant body of research has been conducted in solving imperfect information games by finding their Nash Equilibrium through counterfactual regret minimization (Zinkevich et al. 2007). Such approaches have shown their utility by achieving competitive levels of gameplay in imperfect information games, mainly Poker (Brown and Sandholm 2017a). Yet, this algorithm struggles to scale

as the number of possible game states increases. This is because these methods require a complete traversal of the game tree, which is intractable in Hearthstone.

Orthogonal to the approaches that devise agents that fully play HearthStone are the tasks that model particular aspects of the game, such as, deck building (García-Sánchez et al. 2016) and predicting the winner from a given board state (Grad 2017). These can then be integrated to improve existing game agents (Swiechowski, Tajmajer, and Janusz 2018).

Environment

We introduce a simpler benchmark task that maintains the key features of HearthStone, which we name TradingStone. The aim of this benchmark environment is to understand the difference between search and reinforcement learning approaches and the effect of each different factor on the performance of each family of algorithm.

Hearthstone

The complexity of the original Hearthstone game(Zhang and Buro 2017) hinder the application of many algorithms. As the game elements have non-linear interactions, each card is a stateful element and several effects depend on latent variables which are not directly available to the player. The effect of each card is described in natural language and their descriptions are not necessarily coherent across different cards or across game versions. While playing the game, humans rarely play zero-sum games[1], and opponent distributions are conditioned on out of game trends and information. In this work, we identified key qualities of the original game and extracted them into a simple version, called TradingStone.

We remove the deck selection process with an automatically sampled deck generator. Players are given a established number of hand cards, which are sampled randomly. Decks are removed, thereby players only work with the initial set of hand cards in order to win the game. To remove the necessity to understand natural language, we removed card effects. Thereby, different cards are unique due to differences in their basic properties (e.g. Cost, Attack, Health, etc...). Hearthstone is an adversarial game, where the opponent is sampled from a distribution of possible opponents, hence players have to first understand what strategy the opponents are following by gathering information and then behaving accordingly to the estimated opponent strategy, considering non stationary adversaries introduces a non trivial level of complexity which is left for future work.

TradingStone

TradingStone is a two-players, adversarial, stochastic, imperfect information card game.

The game starts with an empty board, 7 cards in each player hand and both players start at 30 health. Each card is described by its *attack* and *defense* values and whether

[1]the goal of the competitive "Ranked" game is to ascend in the player ladder, players may also optimize for ascension speed and subjective fun instead of optimizing winning rate.

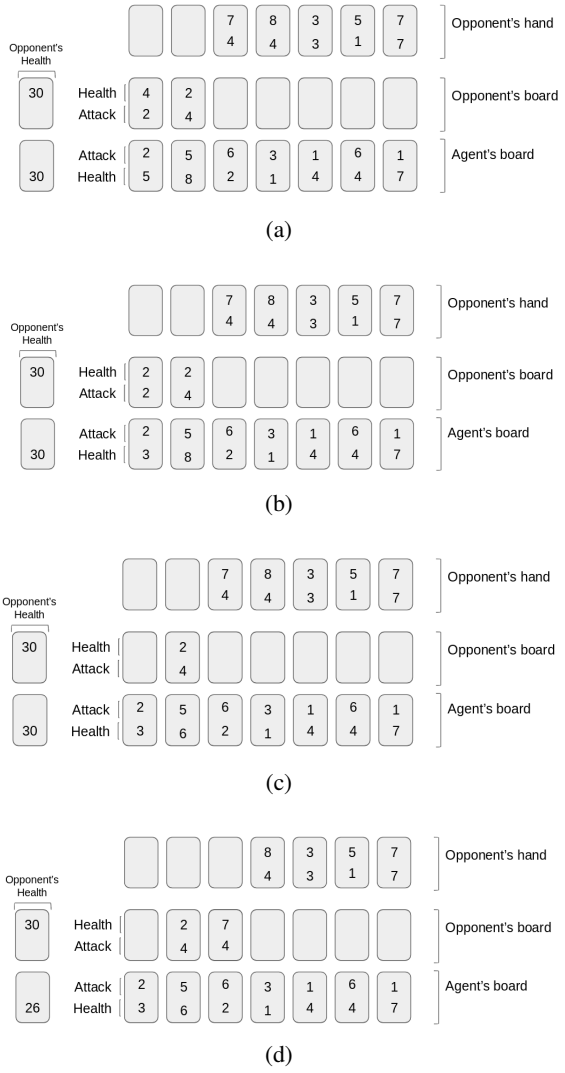

Figure 1: (a): A state of TradingStone, in the partially observable version the values in "opponent hand" are hidden from the Agent, (b) The resulting state after the left-most card (2 5) attacks the left-most opponent card (2 4), (c) the second left-most card (5 8) attacks the left-most opponent card, now (2 2) (d) The players passes the turn and the opponent directly attacks the player with one of the two cards in the board before passing the turn

they are *exhausted*, which determines whether they can attack this turn. The goal of the game is to reduce the health of the opponent player to 0.

The world state is composed of (player health, player board, opponent health, opponent hand, opponent board), at each decision point each agent is able to observe the whole state except for the opponent's hand.

A turn is a set of action between two *pass* actions, at each decision point the player can choose between pass, play card, attack.

The pass action terminates the acting players' turn allowing the opponent to interact with the game. The play card actions allow one card to move from the agent's hand to the board. Attack actions are described by the tuple (attacker, defender), the attacker can be any card on the player's board without the *exhausted* status, while the defender can be any card in the opponent's board or the opponent himself. When card A *attacks* card B, B's health is reduced by A's attack and A's health is reduced by B's attack, when the process is completed, A's exhausted indicator is set to 1. When the pass action is taken the exhausted indicator for each card is set to 0.

In total, there are 56 attack actions (seven possible attacking cards times 7 possible defending cards and the opposing player) and a pass turn action.

For each game, we generate the seven hand cards for both players. Each card is generated by sampling a value between 1 to 9 uniformly for the attack and health attributes of the card.

When one of the players reaches a health value below 1, the game is terminal and that player receives the negative reward -1, while the other player who is declared the winner receives the reward +1.

Paritally Observable TradingStone

One of the complexities of HearthStone is the uncertainty introduced by the opponent's cards and the necessity to make decisions given their information, to test the influence of partial observability in the algorithm performance, the cards in the opponent's hand are hidden from the agent's observations in the partially observable variant (PO-TradingStone).

Stochastic TradingStone

Another important factor of Hearthstone is the effect of random effects in actions, we add noisy actions by making any attack from the player fail to attack the intended target with 10% probability, this is inspired by hearthstone cards such as "Ogre Brute" in the original game.

Formalism

This work sits at the intersection of game-tree search and reinforcement learning. Historically speaking the two fields developed separately but some elements are similar. Here we attempt to quickly clarify the terminology that will be using throughout the article.

The *Game State* is the set of all variables necessary to fully describe the game at some point in time. A *game* is *Fully Observable* (perfect information) if, at each interaction, the player can observe all variables in that state. A *game* is *Partially Observable* (imperfect information) if some information of the *State* is hidden to the agent at decision time (e.g. the card in opponent's hand).

While information sets and observations both deal with the information available to the agent's point of view, an information set is the set of game states sharing the same observation while an observation is a feedback received from the environment.

Methods

We are interested in understanding how multi-action sequences per turn and imperfect information affects reinforcement learning methods and how it does affect heuristic search such as MCTS. The methods of our choice are plain MCTS, PI-MCTS and IS-MCTS for imperfect information search and PPO for Reinforcement Learning. By construction all initial world state are solvable, the player must perform the best response action at every decision step, taking a wrong action at any decision point would always guarantee the game to be unsolvable. In the Reinforcement Learning framework, we say that the resulting Markov Decision Process is multi-chain (Puterman 2014). Exploration in multi-chain MDPs is much harder and the near-optimal regret known is linear in the diameter of the MDP (i.e., the length of the longest path among all shortest paths between any pair of states) (Fruit, Pirotta, and Lazaric 2018).

Proximal Policy Optimization (PPO)

Reinforcement Learning(Sutton and Barto 2018) is a framework for online learning in the sequential decision-making process, where the dynamics of the environments are unknown. Equipping Reinforcement Learning methods with powerful function approximators (e.g. Deep Reinforcement Learning) has been proven to achieve the state of the start results in many domains, including single-player games(Mnih et al. 2015), real-time multi-player strategy games(Jaderberg et al. 2018), (Bansal et al. 2017). For comparison purposes the method of our choice is Proximal Policy Optimization(Schulman et al. 2017) which is a state of the art on-policy, Policy Gradient method that computes a regularized policy update and its known to converge to the optimal policy(Neu, Jonsson, and Gómez 2017).

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search(Coulom 2006) is a method for decision making by taking random decisions and building a search tree, the tree is built incrementally using a *tree policy* to choose which new node to add, when a node is added a *simulation* is run to evaluate statistics about expected payoff from the selected node and the ancestors in a process called *back-propagation*.

In all the MCTS based experiments we use an Upper Confidence Bound (UCB) from UCT(Kocsis and Szepesvári 2006) with $C_p = \frac{1}{\sqrt{2}}$ based policy as *tree policy*, *simulation* is done by a random policy and back-propagation is unmodified.

MCTS with Perfect-Information Monte Carlo (PI-MCTS)

To adapt MCTS to imperfect information PI-MCTS(Long et al. 2010) uses the process of *determinization* to transform an imperfect information state into a perfect information one, this process is repeated and the statistics for the root nodes are averaged to take a final decision. This process samples the information hidden from the player according to the valid distributions of possible game states.

Information Set MCTS (IS-MCTS)

In IS-MCTS the nodes of the search tree represent information sets, an information set is the set of all the states that are indistinguishable from the agent's point of view differing only by hidden information, in this case the search tree is kept and updated, adding new edges when necessary, across determinizations allowing for reuse of previous statistics and decisions are taken only on available edges, in addition, since IS-MCTS continues using the same tree across determinizations, the hyperparameter of number of determinizations vs simulations per decision is removed.

Experiments

Setup

We compare experimentally the performance of the discussed algorithms. We run experiments for both perfect information and imperfect information. For PI-MCTS and IS-MCTS are implemented from baseline MCTS (Lanctot et al. 2019). PPO is made of two separated networks. For both the Actor and Critic Network we used non-linear function approximators and for the estimate of the advantage function, we use TD(0) (Sutton and Barto 2018).

A fair comparison between search and RL is not trivial for two reasons. Firstly RL doesn't use a simulator and it has to learn the dynamics of the environment from online samples, while search methods have to query the simulator to evaluate possible actions. Secondly RL can generalize at evaluation time without samples. Our approach is to compare the two approaches by fixing the number of samples available to 10000 and by using the same random seed. Therefore all methods are trained on the same sequence of possible games.

Results

In table 1 we report the winning ratio for fully observable TradingStone and in table 2 for partially observable TradingStone. The reported statistics are averaged over 100 randomly generated environments. Under perfect information MCTS achieves a higher winning ratio than PPO.

On Imperfect Information, IS-MCTS performs better than all other baselines.

We argue that if the optimal policy is deterministic and unique (as in TradingStone), the aggregation of all legal actions for each child from the root node combined with subset arm bandit algorithm at leaf node, delivers a more effective exploration strategy than UCT.

Experimental Results $(\mu \pm \sigma)$	
Algorithm	TradingStone
PPO	0.27 ± 0.10
MCTS	0.37 ± 0.13

Table 1: Algorithmic performance on the fully observable variant

Experimental Results $(\mu \pm \sigma)$	
Algorithm	PO-TradingStone
PPO	0.46 ± 0.12
PI-MCTS	0.47 ± 0.12
IS-MCTS	0.52 ± 0.10

Table 2: Algorithmic performance on the partially observable variant

The combination of multi-action turns with imperfect information poses a significant challenge for state-of-the-art-method in search and RL. In figure 2 we show that, interestingly enough, PPO under imperfect information performs better than PPO under perfect information. We argue that the function approximator overfit the specific game instance instead of learning to generalize over possible plans.

Conclusion

In this work, we introduced TradingStone, a simplification of the game HearthStone that reduces the complexity of the game while keeping the core aspects of the game that makes an interesting research problem. Namely, TradingStone remains a stochastic zero-sum with imperfect information, with characteristics that are reminiscent to trading card games, such as, the wide variety of game cards that need to be played optimally.

The goal of this simplification is to allow more RL methods to be easily integrated into this game. We show that IS-MCTS can be applied in this version of the game, which leads to improved results compared to previous approaches to solving HearthStone.

The environment and agents proposed in this work will be made available to the public.

Acknowledgments

Especially grateful to Marc Lanctot for the insightful discussions. Anders Jonsson and Gergely Neu for having hosted us for the duration of this work. This research is funded by Nexus Research.

References

Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; and Mordatch, I. 2017. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*.

Brown, N., and Sandholm, T. 2017a. Libratus: The superhuman ai for no-limit poker. In Sierra, C., ed., *IJCAI*, 5226–5228. ijcai.org.

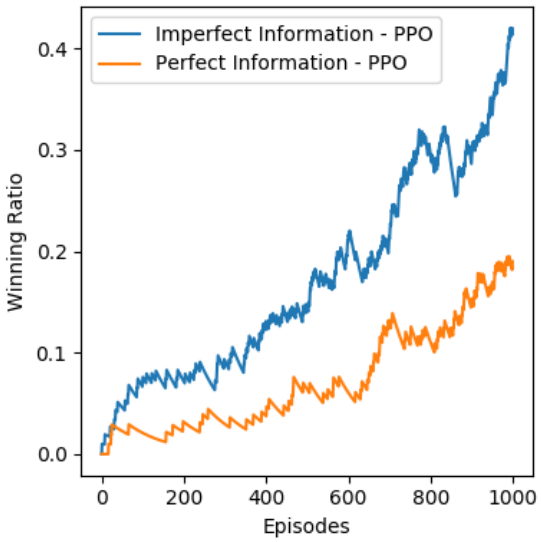

Figure 2: Evaluation of PPO at intervals of 10 training episodes for perfect and imperfect information environments.

Brown, N., and Sandholm, T. 2017b. Safe and nested subgame solving for imperfect-information games. *CoRR* abs/1705.02955.

Coulom, R. 2006. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In Ciancarini, P., and van den Herik, H. J., eds., *5th International Conference on Computer and Games*.

Cowling, P.; Powley, E.; and Whitehouse, D. 2012. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and Ai in Games* 4:120–143.

Fruit, R.; Pirotta, M.; and Lazaric, A. 2018. Near optimal exploration-exploitation in non-communicating markov decision processes. In *Advances in Neural Information Processing Systems*, 2994–3004.

García-Sánchez, P.; Tonda, A.; Squillero, G.; Mora, A.; and Merelo Guervós, J. 2016. Evolutionary deckbuilding in hearthstone.

Grad, 2017. Helping ai to play hearthstone using neural networks. 131–134.

Jaderberg, M.; Czarnecki, W. M.; Dunning, I.; Marris, L.; Lever, G.; Castaneda, A. G.; Beattie, C.; Rabinowitz, N. C.; Morcos, A. S.; Ruderman, A.; et al. 2018. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *In: ECML-06. Number 4212 in LNCS*, 282–293. Springer.

Lanctot, M.; Lockhart, E.; Lespiau, J.-B.; Zambaldi, V.; Upadhyay, S.; Pérolat, J.; Srinivasan, S.; Timbers, F.; Tuyls, K.; Omidshafiei, S.; et al. 2019. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*.

Long, J. R.; Sturtevant, N. R.; Buro, M.; and Furtak, T. 2010. Understanding the success of perfect information monte carlo sampling in game tree search. In *AAAI*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Moravcík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. H. 2017. Deepstack: Expert-level artificial intelligence in no-limit poker. *CoRR* abs/1701.01724.

Neu, G.; Jonsson, A.; and Gómez, V. 2017. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*.

Puterman, M. L. 2014. *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Santos, A.; Santos, P. A.; and Melo, F. S. 2017. Monte carlo tree search experiments in hearthstone. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 272–279.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Swiechowski, M.; Tajmajer, T.; and Janusz, A. 2018. Improving hearthstone AI by combining MCTS and supervised learning algorithms. *CoRR* abs/1808.04794.

Zhang, S., and Buro, M. 2017. Improving hearthstone ai by learning high-level rollout policies and bucketing chance node events. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 309–316.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2007. Regret minimization in games with incomplete information. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, 1729–1736. USA: Curran Associates Inc.