Title of the Document

Author names, surnames and IDs

School ..., University ..., UK

email[@qmul.ac.uk](mailto:%7d@Springer.com)

**Abstract.** Summarize the paper in a paragraph of two. It should contain at least 70 and at most 150 words. You should motivate the research done, give some details about the experiments run and briefly mention the most important findings.

1. Introduction

Deep reinforcement learning (DRL) have increased flexibility and speculation to test large areas. Recently, DRL is an active research area with incredible results in Atari, Go and the latest multi-agent games (for example, StarCraft and DOTA 2). Probably the biggest test for an in-depth study of the reinforcement is the effectiveness of the test. However, when a DRL specialist is trained, he can be persuaded to act gradually, simply by drawing a conclusion from the prepared model. In addition, organizational methods, such as the Monte Carlo tree search (MCTS), do not have a preparatory stage, but perform reproductive implementations in anticipation of access to the test system to determine the best movement to carry. A serious problem with the vanilla MCTS is the adaptability to spaces with large proportions of elongation and long scenes, which require mallets of scenes based on free time to act in this direction, which makes the strategy inapplicable to applications that require leadership continuously [8].

There are several ways to overcome both DRL and research methods. For example, Alpha Go and Expert Iteration simultaneously proposed to combine DRL and MCTS in a performance training system in which the two parts improve each other. These works subsequently combine research and neural systems in a circle. To begin, the search is used to create a specialized movement data set that is used to prepare the arrangement of the agreement. Second, this system is used to improve the quality of the main research and is updated. However, the collection of information on the main movements through the calculation of vanilla investigations may be delayed in a consistent structure based on the competence of the test system. In this document, in relation to the existing work mentioned above, we show that it is also possible to combine research with DRL methods without common methods to the point that parts of the neuronal and research system can be performed simultaneously in a nearby project. The focus of this work is to show how we can use generally more fragile demonstrators (for example, lightweight MCTS with some launches or other organizers based on hunting) for safer models without RL, connecting the demonstrator and not connecting the RL model through reference activities. Here we will focus on catastrophic cases that occur as often as possible in the test proposed below to reinforce learning [10].

Pommerman's environs depend on excellent support for the Bomberman game. Pommerman's condition includes 4 aviation specialists, initially located in the four corners of the board, who perform synchronized actions. From the point of view, the best way to improve Pommerman’s condition (for example, kill a specialist) is to place the bomb (and the consequences of this activity are probably visible when the bomb explodes later, 10 temporary passages). In addition, this activity can generate a catastrophic opportunity for the operator to finish everything.

1. Literature Review

***Safe RL***

Safe reinforcement of learning is trying to ensure that the structure is reasonably implemented or potentially consider welfare requirements during training and additional forms of organization. In general, there are two different ways of guaranteeing the security of human resources: some methods regulate the measurement of optimization and others: the research system. Since old research processing methods, such as Boltzmann research, do not guarantee safety, the Research Territory of Safe Research governs research: how can we produce operators that consider welfare restrictions during normal activities, but also during the training below? A strategy-based model suggested that you familiarize yourself with the accuracy of this agreement and use a different safe approach only in dangerous circumstances. (Kartal et al., 2019) proposed to study a risk assessment model called a terrorist model, which provides unsafe opportunities for the formation of a training procedure. (Segal and Zilberman, n.d.) examined the question of whether mediation between people can avoid catastrophic opportunities; while in some Atari games the methodology was productive, the creators argue that it will not change in progressively difficult situations due to the required level of human work.

***Imitation Learning***

The areas where premiums are deferred and inadequate are research-related issues and are particularly problematic when studying a blank board. Representative training can be used to train operators much faster than training without training. For example, it approaches (Malysheva et al., 2018) or its global version, which presents presentation training as a manageable problem, in which the objective is to coordinate the demonstrator's presentation. In any case, the mapping of operators using these strategies is severely limited by the performance of the demonstrator. (Kartal and Taylor, 2018) proposed an RL strategy based on the characterization that uses Monte Carlo distributions for each activity to create a set of preparatory data to iteratively improve the approach. Subsequent work, for example, Expert Iteration extends the presentation training to the RL configuration, where the demonstrator continually improves during preparation. A wide range of representative training activities have been carried out, in which information on reproduced persons or demonstrators are used to accelerate the learning of RL strategies.

1. Benchmark

Explain the problem that is being solved, including its main difficulties. Also give some hints on how your agent interacts with the framework, how the framework works, etc. Give references as needed.

1. Background
2. **Pommerman**

***Game Set-up:*** It is a randomly generated 11x11 board. With 4 agents, 1 in each corner with one bomb each, where each bomb has a life of 10 timer ticks. If teams, then teammate is in opposite quadrant diagonally. There are 9 kinds of tiles (boxes in the grid), those are, *passage* (players can walk through these), *rigid* (closed and indestructible), *wood* (can be destroyed by bombs; half of these walls spawn power-ups), *bomb*, *flames* (from bomb explosion), *fog* (player does not know what's there, could be any of the other tiles), *power-ups*, *dummyAgent* (used to signal no other opponent or teammate), agents. The game can be played in both, partial and full observability.

***Game Rules:*** The target to win the game is to stay till the end of the game as last standing individual or team (for team mode). The bomb can destroy every kind of tiles, whether be agents, wooden wall, power-ups and can chain explosions when exploded to a bomb. After the bomb is exploded it goes back to the agent’s bomb supply. Then there are earned *power-ups* which included *extra bombs* or *can kick* in which when encounters a bomb the agent can push it through tiles until gets hit by another bomb, agent, wooden wall. The bomb can explode during the time of travelling. For the matter of collisions, two agents or two bombs can stay on the same tile, then try to go to the same tile, bounces back.

***Game Actions:*** In some random turn, an agent can look over one of six actions which are; *Up* (move upwards), *Down*(moves downwards), *Left*(move left), *Right*(move right), *Stop* (this is just a pass action) and *Bomb* (lay a bomb).

***Game Modes:*** There are 3 types of modes which are *FFA (Free for All)*, *team* and *team radio*. In this paper, we did not work on team radio. While in FFA mode all four of the agents compete against each other and the one agent standing till the last wins the game. And in team mode, the agents in the opposite quadrant diagonals team up and plays against the other team and from whichever team the agent lasts till the end wins the game. The players who die loses the game.

1. **Monto Carlo Tree Search**

Monte Carlo Tree Search (MCTS) is an exceptionally particular best-first search technique. The Games like Tic-Tac-Toe, Rubik's Cube, Sudoku, Chess, Go and numerous others have a normal property that lead to an exponential increment in the number of potential activities that can be played. These potential advances increment exponentially as the game goes ahead. Preferably, if you can anticipate each conceivable move and its outcome that may happen later. You can build your opportunity of winning.

But as you move ahead the computation power needed to calculate the game instincts also increases exponentially. And many previously proposed algorithms fail to perform efficiently.

Monto Carlo is an algorithm precisely defined for the games in which, by following a policy one gets the winning path towards the final goal state.

Monto Carlo works well for the high branching factor and uses forward model (FM) or predictive model. Monto Carlo has 4 steps which works repeatedly until the end, those are;

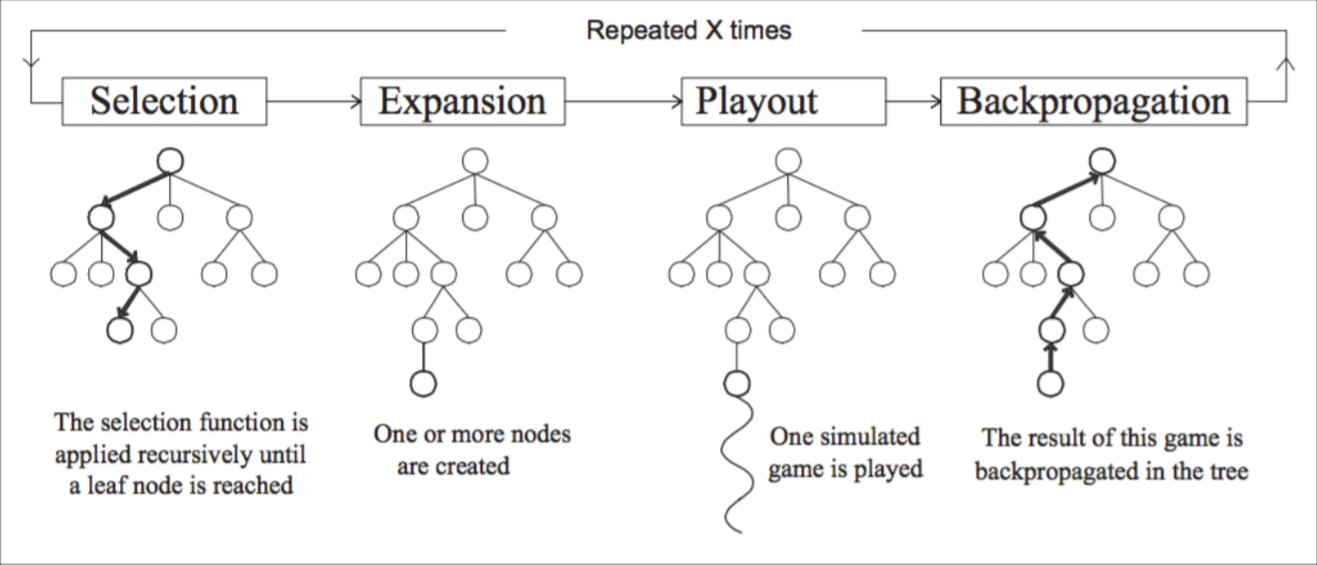


Fig. 1. Strategic steps of Monte-Carlo Tree Search [4]

***Selection:*** It is a process of selecting a child node for the expansion starting from the root node, this is done by a *tree policy,* and when a node with no children and is not a terminal state is found, it is chosen for the expansion.

***Expansion:*** The algorithm starts by building the tree from its root, which is the current state of play. The first step is to expand the root by adding children that represent all the states that can be reached in the next step. The expansion of the tree will continue any time a leaf node is selected. A leaf node is the node that does not have any children although it is not terminal [3].

***Playout (Simulation):*** When a leaf node is visited for the first time, a random agent will continue solving the problem. In a computer game, a random player will start from this point until the either the game has finished, or the playing budget has reached its limit (e.g., a time limit). This simulation will return a value that can be 0 for losing or 1 for winning, or a continuous number such as the score obtained. This value will be added to the node as its reward from the simulation. This step is generally known as a rollout or playout and the playing policy during the simulation is the default policy according to the terminology in [1].

***Backpropagation:*** The result of the visit counts of all traversed nodes in this iteration, as well as the final rewards obtained are updated [2].

1. **Decision Policy**

***UCT:***

***UCT Tuned:*** After UCT, UCT tuned was proposed by Auer et al. [11] as an improvement for UCT.

***Epsilon Greedy:*** In term of bandit problem [13] the best lever is selected for a proportion n of the trials, and a lever is selected at random (with uniform probability) for a proportion n.A typical parameter value might be*,* but this can vary widely depending on circumstances and predilections [14].

***Epsilon Decaying:*** It works same as epsilon greedy except that the value of decreases as the experiment progresses which results in highly explorative behaviour at the start and highly exploitative behaviour at the finish.

***UCT Bayesian:*** In Bayesian framework the number of node visits is made limited in pommerman framework which provides potentially better and more accurate estimations of nodes. It performs following UCT exploration-term modification;  
 *motivated by optimistic prior/ independence assumptions*

*motivated by CLT.*

***AMAF:*** The basic AMAF algorithm combines UCT with the AMAF update after each play-out. This algorithm rapidly grows the counts at the nodes in the UCT tree, and thus increases the algorithm’s confidence in the win rates. On the other hand, the counts at nodes are increased not because the move was made in the position represented by the parent, but because it was made in a (perhaps very) different context. This use of information from other contexts makes the counts from the AMAF update immediately suspect [1].

***α-AMAF:*** It is an extension of simple AMAF and it keeps AMAF statistically separate in which the values of AMAF and standard UCT are used in;

The -AMAF algorithm blends the standard (UCT) score for each node with the AMAF score [1].

***Cut-Off AMAF:*** In cutoff AMAF, the AMAF algorithm is used to update statistics for the first simulations, after which only the standard UCT algorithm is used [1]. The purpose of cutoff AMAF is to warm up the tree with AMAF data, then use the more accurate UCT data later in the search [13].

***Grave:***

1. Techniques Implemented

I am writing in google doc will update when finished

1. Experimental Study

Detail the experimental setup used to test the different versions of the algorithm you have been working on.

Analysis of the results. Present the results in an understandable manner (graphics, tables, etc.). Draw conclusions about what things worked (and why) and which didn't (and why).

1. Discussion

Overall discussion of the results obtained in the experiments (What things worked? What things didn't work? Did something surprise you?).

1. Conclusions and Future Work

Explain the main contributions of this work: what are the most important findings. Finally, explain how this work could be extended. What would be the next steps?

Note that conclusions **must not** be a reflection of what did the assignment mean for you as a student, but a **critical summary** of the work and findings of the project

References

1. Browne, C., E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, A Survey of Monte Carlo Tree Search Methods. Computational Intelligence and AI in Games, IEEE Transactions on, 2012. 4(99): p. 1-43. 2. [[1]](#endnote-1)
2. Diego Perez-Liebana, Raluca D Gaina, Olve Drageset, Erc¨ument Ilhan, Martin Balla, and Simon M Lucas. Analysis of Statistical Forward Planning Methods in Pommerman. In Proceedings of Artificial Intelligence in Interactive Digital Entertainment, 2019.
3. Atif M. Alhejali, Simon M. Lucas, Using Genetic Programming to Evolve Heuristics for a Monte Carlo Tree Search Ms Pac-Man Agent, 978-1-4673-5311-3/13/$31.00 ©2013 IEEE
4. G. M. J.-B. Chaslot, M. H. M. Winands, H. J. van den Herik, J. W. H. M. Uiterwijk, and B. Bouzy, “Progressive strategies for Monte-Carlo tree search,” New Mathematics and Natural Computation, vol. 4, no. 3, pp. 343–357, 2008
5. Kartal, B., Hernandez-Leal, P. and Taylor, M.E., 2018. Using Monte Carlo Tree Search as a Demonstrator within Asynchronous Deep RL. arXiv preprint arXiv:1812.00045.
6. Kartal, B., Hernandez-Leal, P. and Taylor, M.E., 2019, October. Action Guidance with MCTS for Deep Reinforcement Learning. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (Vol. 15, No. 1, pp. 153-159).
7. Malysheva, A., Sung, T.T., Sohn, C.B., Kudenko, D. and Shpilman, A., 2018. Deep Multi-Agent Reinforcement Learning with Relevance Graphs. arXiv preprint arXiv:1811.12557.
8. Osogami, T. and Takahashi, T., 2019, October. Real-time tree search with pessimistic scenarios: Winning the NeurIPS 2018 Pommerman Competition. In Asian Conference on Machine Learning (pp. 583-598).
9. Segal, O.B.D.M. and Zilberman, G.K.V., Pommerman Agent.
10. Zhao, Y., Borovikov, I., Rupert, J., Somers, C. and Beirami, A., 2019. On Multi-Agent Learning in Team Sports Games. arXiv preprint arXiv:1906.10124.
11. D. P. Helmbold and A. Parker-Wood, “All-moves-as-first heuristics in Monte-Carlo Go,” in Proc. Int. Conf. Artif. Intell., Las Vegas, NV, 2009, pp. 605–610.
12. P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” Mach. Learn., vol. 47, no. 2, pp. 235–256, 2002.
13. Bubeck, S., Munos, R. and Stoltz, G., 2009, October. Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory* (pp. 23-37). Springer, Berlin, Heidelberg.
14. Sutton, R. S. & Barto, A. G. 1998 Reinforcement learning: an introduction. Cambridge, MA: MIT Press.

1. [↑](#endnote-ref-1)