Beating BaghChal: A multi-agent competitive play approach to asymmetric board games

Ankit Shrestha
Department of Computer Science
Utah State University
Utah, USA
ankit.shrestha@usu.edu

***Abstract*—In real world, there are many cases where different agents need to collaborate or compete with each other and their objectives, actions and strategies can be different. We can see these differences in some board games where the method of play and the requirements to win are completely different for the players which we term as asymmetrical board games. In this paper, we have used a traditional Nepali board game called Baghchal to understand how multi agent competitive networks perform to address the different objectives and methods of play. This work implements a two-agent adversarial network with Monte Carlo Tree Search (MCTS) to simulate the instances of the game by competitive play between the agents. The adversarial networks are Proximal Policy Optimization (PPO) networks and the two networks are trained alternatively and are accepted only when they can consistently beat their previous adversary. This study presents that the use of a multi agent network to play the different players can improve their performance overall which indicates that an adversarial network can lead to efficient solutions for asymmetric board games. This also implies that the use of a multi agent network in a competitive setting to carry out asymmetrical tasks can be a viable solution that can be studied in future works for realistic environments.**

***Index Terms*—Adversarial Networks, Asymmetric Board Games, Monte Carlo Tree Search, Multi Agent Systems, Proximal Policy Optimization**

I. Introduction

Asymmetric board games are board games in which the players start with completely different states/pieces, have completely different methods of play and can also have different objectives or goals that the players need to complete to win the game. One of the asymmetrical board games that have different starting points, different methods of play and different objectives for the players is Baghchal. It is a strategic two-player board game that originated in Nepal. The game is asymmetric in that one player controls four tigers and the other player controls up to twenty goats. The tigers hunts the goats while the goats attempt to block the tigers' movements [1].

In this study, we have implemented a multi agent competitive play system to play the game of Baghchal and compare it with a baseline single agent self play system to understand its performance. We have multiple studies that have studied symmetric games using single agents and self-play. However, most of the real life problems are complex, have different starting points, require vast and different strategies and can work towards different goals. Asymmetric board games represent such scenarios better than the symmetric games and can be a great training group to experiment with multiple agent systems that can learn to do smaller tasks and act together as a better system collectively.

This study contributes to the ever growing literature on the reinforcement learning by providing better understanding of multi agent systems that use competitive play data for training. It aims to further evaluate the performance of such multi agent systems in playing asymmetric board games with different starting points, methods of play and objectives. This will help us understand its effectiveness in dealing with such environments. Such a system can be an effective solution in the future to solve complex real life problems by dividing the different tasks among the agents that can learn to work together to give the sense of a greater whole.

II. Related Work

There have been prior works that have been done in board games like Go [2] and Chess [3]. However, there have been very few works on asymmetric board games like Baghchal and the works that have been done on asymmetrical board games have rarely used reinforcement learning algorithms. A few research done on Baghchal specifically have not focused on mastering the game [4], [5]. We have further seen most of the agents implemented to play board games have used self-play and single agents to play the players which has with no doubt worked excellent for symmetrical board games [2], [3], [6]–[9]. However, in asymmetrical games, the agent needs to learn to play in different ways and the objectives of the games are completely different for the players as well. As such, a multi agent system with each agent simulating a player role could improve the system that plays the board game. We have used the game of baghchal which is very asymmetrical to understand and highlight the effectiveness of a multi agent system that makes use of competitive play data for training.

There have been few works that have worked on asymmetric board games using reinforcement learning. Some of the works have dealt with asymmetric information games where the information received by the agents are not equal or fair [10]–[12]. Recently, Shen et. al. [13] have worked using adversial reinforcement learning agents in an asymmetrical information game. The authors have worked on a stochastic optimization scheme to dynamically update the opponent ensemble to strike a balance between robustness and computational complexity and have shown that with the same computational resources, the agent can be made more robust using this method [13]. Similarly, Tampubolon et. al. [14] have worked using two non-cooperative Q learning agents to learn optimal strategies for asymmetrical information games. These works have dealt with asymmetric information games but they are not exactly asymmetrical like a board game as the agents are dealing with asymmetry of information provided to them but not the actual asymmetry of the game itself.

Moreover, after the success shown by Alpha Go, many works have worked on applying deep reinforcement learning in general game playing which can be considered asymmetric in some sense [15]–[17]. However, the agents are not dealing with completely different strategies, method of play and goals like in asymmetrical board games. Other works have focused on balancing the asymmetry of the game. Beau et. al. [18] have worked on using Monte Carlo simulation to analyse the relative impact of game actions to iteratively adjust the attributes of the game to make it balanced. However, in an asymmetric board game like Baghchal, the game is completely asymmetrical and the win probabilities cannot essentially be balanced without changing the rules of the games itself.

There have also been some work on use of competitive play data with multiple agent system to create a combined greater system like with Open AI Five that beat the world champions of DOTA 2 which is a very complex strategy game [19]. Similarly, Barros et. al. has applied competitve play training in teaching reinforcement learning agents to play card games [20]. There have also been other video games where competitive play has been used to train the agent like in Starcraft II, ViZDoom, Quake II and many other games [6]. However, all of these games do not deal with the asymmetry of the games like Baghchal as the agents are learning to achieve the same objectives using similar method of play and strategies.

In this paper, we have used an approach of multiple agents with competitive play data to learn the different game play of the two players so that one agent only needs to learn to play as one type of player but the combination of agent creates an overall multi agent system that can perform better in the game than a single agent.

III. Model and the Environment

We used the environment provided in GitHub for Baghchal which also implemented an Alpha Zero type implementation to learn the game [21]. However, the code was an early prototype and needed lots of fixing to run successfully. Once, the environment was setup, we started experimenting with different agents and parameters. All the agents mentioned in this paper are PPO agents with MCTS. The number of simulations used to collect self-play or competitive play data is 20. For each iteration, 10 self-play or 10 competitive games are played. The exploration rate was high for generating the self play and competitive play data whereas exploitation was set to maximum when the agents were pitted with each other. The agents are trained for 20 epochs after each iteration. Extensive hyper parameter tuning of the neural network could not be done due to the hardware limitations. The state, action and rewards for the agents are described below.

The board game has 5x5 (25) places where the pieces can be placed or moved (See Fig. 1). The game starts with the 4 tigers in the 4 corners and the goats are placed and cannot be moved until all 20 goats have been used. The state space includes the 25 positions on the board. Each position can be represented by three numbers for the three states it can be in: the position has a goat, the position has a tiger and the position is empty. In addition to that, the number of goats eaten by the tiger and the number of tigers captured are also included in the state space.

Tigers can move along the lines in the board and the goats can also do the same movement once all the goats are placed. The tigers can also jump one goat along the line if there is empty space in the destination position to eat one goat. Further, at the start, the goats can be placed on any empty space in the board. The actions are denoted starting with G for goat moves and B for Tiger(Bagh) moves. Placing a goat action has only two numbers after G to represent the position (Example: G12). Moving the goat after placing them all and all the Tiger moves have four numbers after G and B respectively (Example: G1213 where the first two numbers represent the start position and the last two numbers represent the final position).

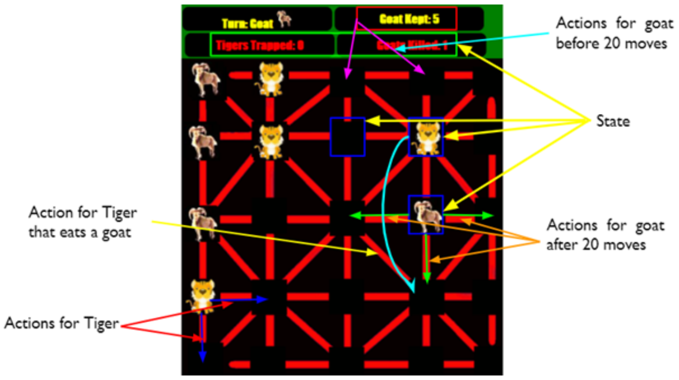
Turn: Goat
Goat Kept: 5
Tigers Trapped: 0
Goats Killed: 1
Actions for goat before 20 moves
State
Action for Tiger that eats a goat
Actions for goat after 20 moves
Actions for Tiger

Fig. 1. Explaining the game environment with actions and states

The reward is given like any board game which is 0 for draw, 1 for winning and -1 for losing. The board is also symmetrical which has been used to create additional data for training as the whole board can be rotated and flipped to create data for a symmetrical different game. The actions can also be mapped in similar way so that we can create symmetrical game data for training.

IV. Experiments and Results

To understand how well competitive play using multiple agent can work for asymmetric board games, we carried out a few experiments. The complete code that was used to run the experiments that follow can be found in the link provided in the bibliography [22].

A. Baseline/Self Play agents

We first implemented a baseline to compare the competitive agents and the results from different experiments. The baseline was implemented as a single Proximal Policy Optimization (PPO) agent with a Monte Carlo Tree Search (MCTS) which is an implementation similar to Alpha Zero but smaller in scale. We trained this single agent using self-play and only accepted the newer agent when it could beat the previous agent more than 50% of the times in the games that were not drawn. After running for around 80 iterations, the new agents were accepted 25 times. Since, the agents were rolled back when they were not accepted and the training started from the previous best agent, the last agent was trained 25 times in total. In all of the graphs in the result section, the accepted agent at 0 is the same first trained agent which is accepted by default here (referred to as base agent moving forward). We can see that in general, the later agents perform much better than previous agents as the losing rate decreases and the win rate increases (See Fig. 2).

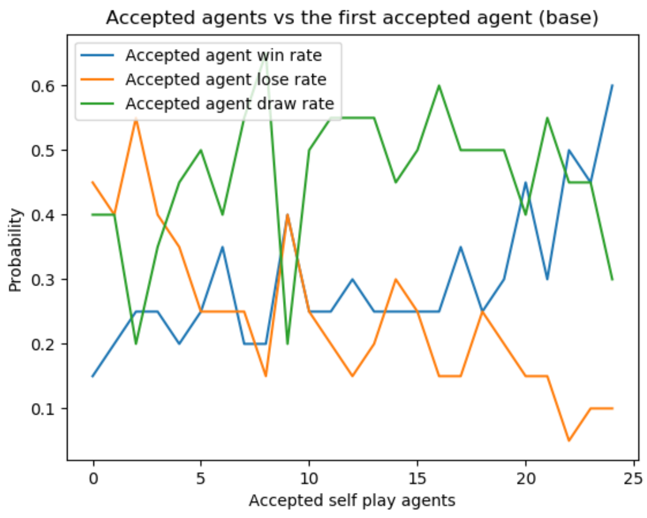

Fig. 2. Accepted self play agents vs the first default accepted agent

B. Competitive Play Agents

After implementing the baseline agent, we created two reinforcement learning agents and trained one of the agent to only play the goat (goat agent) and the another agent to only play the tiger (tiger agent). The data was collected for training by using competitive play between the current best goat agent and the tiger agent. We applied different strategies and ideas with competitive play agents that have been listed here.

1) Alternately Trained agents (AT agents): For this, we didn't change anything on the data and the implementation from the baseline. The only thing that was changed was that the goat and tiger agents were trained alternatively starting from the goat agent. Once the goat agent could beat the current best tiger agent in more than 50% of games that were not drawn, it was accepted as the best goat agent. After this, only the tiger agent would be trained going forward until it could beat the new accepted goat agent. After running for 80 iterations, only 4 goat agents and 3 tiger agents were accepted. This means, the goat agent was trained only for 4 times and the tiger agent was trained only for 3 times. All the agents were accepted early on before the 15 iterations were completed. This might be because it gets harder to beat the improved best competitive agent when trained alternatively as the iteration progresses. The performance of these agents against the base agent (at 0) (see §IV-A) followed by the five best self play agents from worst to best (at 1 to 5) is quite good (See Fig. 3 & 4) considering their early and lower acceptance.

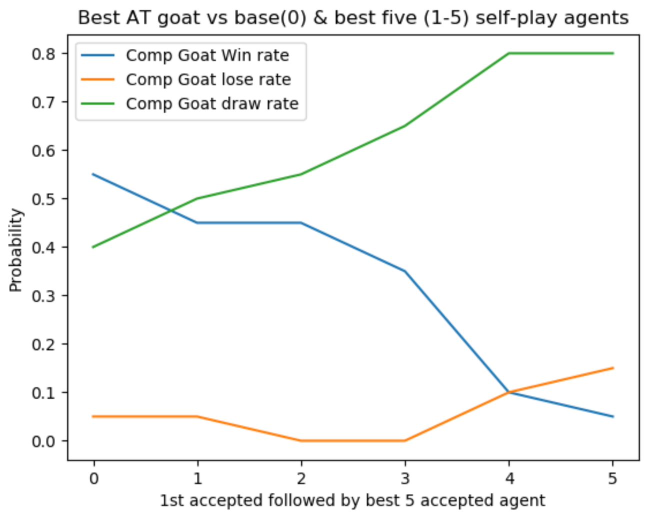

Fig. 3. Best AT goat agent vs base & five best self play agents

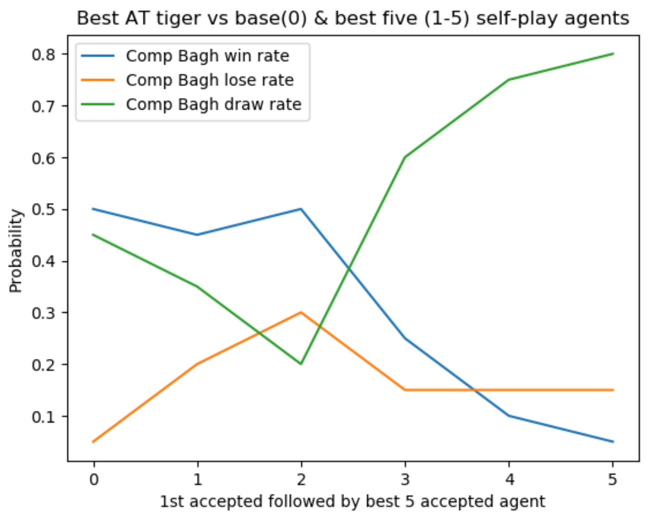

Fig. 4. Best AT tiger agent vs base & five best self play agents

2) Acceptance Condition Changed agents (ACC agents): From the above run, we realized that the agent acceptance was very low as the agents learned rapidly which is demonstrated by their good performance against the self-play agents considering only 4 and 3 iterations of training. Due to this, we opted to change the acceptance condition for the agents. When a new goat agent was accepted we set a variable to the value of -(new goat wins-new goat loss) which is the win rate of the current best tiger with the new accepted goat (See Fig. 5). Then, we trained the tiger agent to only have a win rate better than that instead of having to win against the goat agent which had improved a lot. This meant we wanted the new tiger agent to score better than what the best tiger agent did with the new accepted goat agent. We did the same when a new tiger agent was accepted and trained the goat agent to only beat its win rate against this new tiger agent. At the start of the training, we set the first win rate as a large negative value so the new win rate would be accepted by default. After doing this, 8 goat agents and 9 tiger agents were accepted in 80 iterations and the agents were also accepted in later iterations. The agents performed much better after acceptance condition was changed (See Fig. 6 & 7). This can be because in the previous condition we were also rejecting agents that had essentially improved but failed to beat its counterpart. This change in acceptance condition solved that resulting in higher number of accepted agents which performed better due to more training.

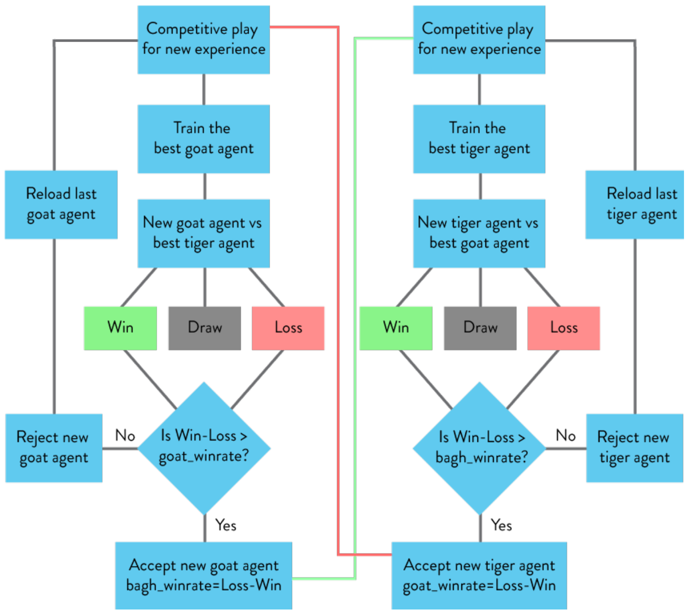
Competitive play for new experience
Train the best goat agent
New goat agent vs best tiger agent
Win
Draw
Loss
Is Win-Loss > goat_winrate?
No
Yes
Reject new goat agent
Reload last goat agent
Accept new goat agent bagh_winrate=Loss-Win
Competitive play for new experience
Train the best tiger agent
New tiger agent vs best goat agent
Win
Draw
Loss
Is Win-Loss > bagh_winrate?
No
Yes
Reject new tiger agent
Reload last tiger agent
Accept new tiger agent goat_winrate=Loss-Win

Fig. 5. Model of the multi-agent system with the new acceptance condition

3) Data Filtered agents (DF agents): After changing the acceptance condition, we filtered the data that was fed to each agents. We only used games where the goat won or the game was a draw to train the goat agent and only the games where the tiger won or the game was a draw to train the tiger agent. We wanted to see how this would impact the performance of the agents in a competitive play scenario.

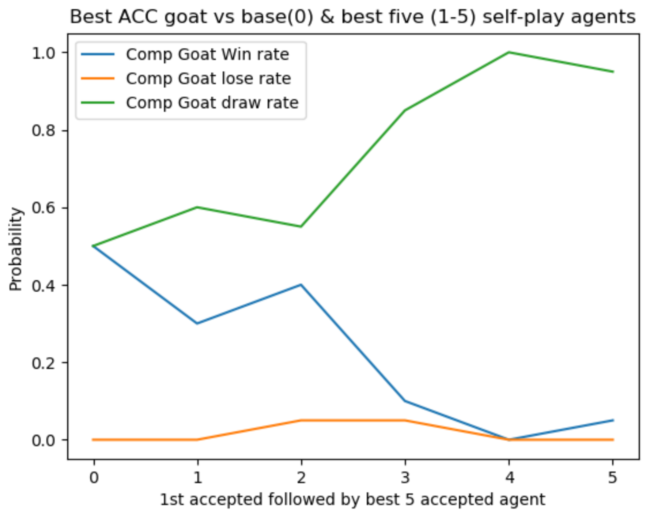

Fig. 6. Best ACC goat agent trained vs base & five best self play agents

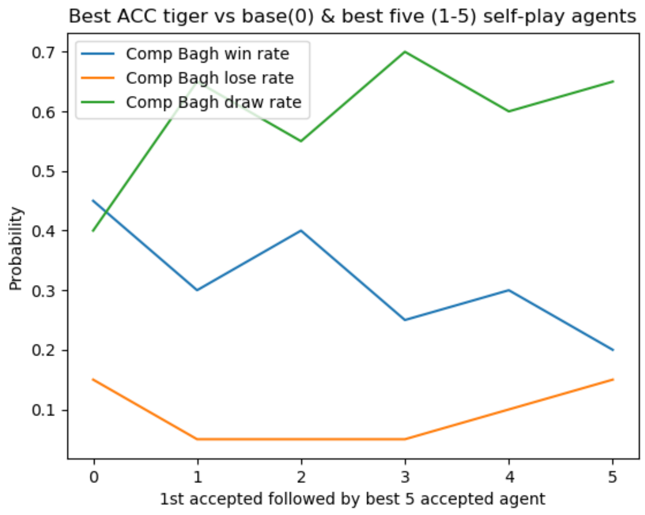

Fig. 7. Best ACC tiger agent vs base & five best self play agents

Further, it seemed interesting to find out how the lack of information on how the agents could lose would impact their behavior. The acceptance condition and the alternative training was the same as discussed before. In a total of 80 iterations, 8 goat and 8 tiger agents were accepted. The agents performed much worse due to their lack of understanding on how they could lose games (See Fig. 8 & 9).

4) Human Play Injected agents (HPI agents): For this experiment, we removed the data filter as we observed the decreased performance of the agent. After trying these different experiments, we played a few games with the reinforcement learning agents that performed best both as a goat and a tiger. As someone who can consistently beat the versions available on the internet more than 95% of the times, the games were extremely challenging. Out of the 10 games played (5 as goat,

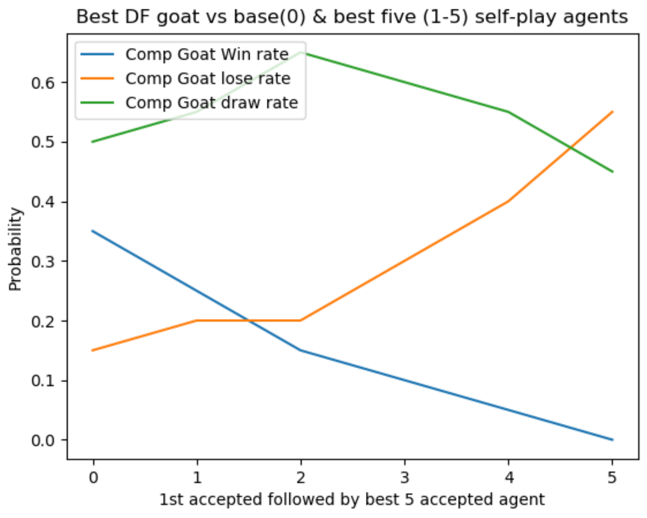

Fig. 8. Best DF goat agent vs base & five best self play agents

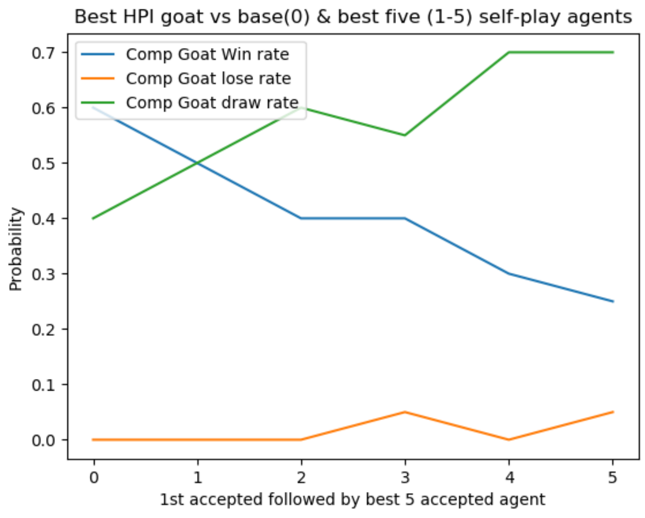

Fig. 10. Best HPI goat agent vs base & five best self play agents

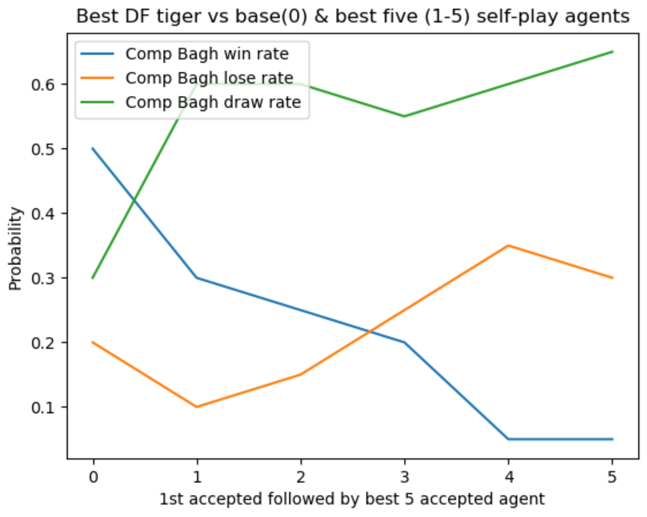

Fig. 9. Best DF tiger agent vs base & five best self play agents

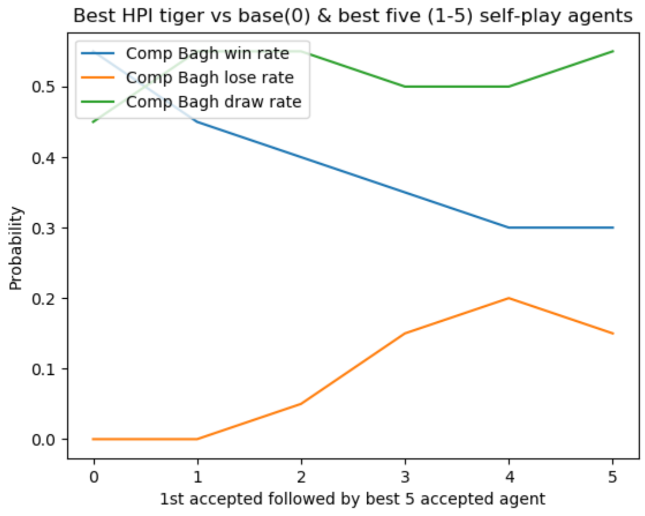

Fig. 11. Best HPI tiger agent vs base & five best self play agents

5 as tiger), 5 games were draw, 3 games were won by us and 2 games were won by the agent. We decided to add additional data to each training data which would be data from human play. After every 10 iterations, we played against the agent for 4 games (2 as goat and 2 as tiger) and used that data to train the agent along with competitive play data. The human play data was never removed from the memory and were aggregated over the iterations so they could have enough impact on the training of the agent. In 80 iterations, 14 goat agents and 15 tiger agents were accepted when training using a mixture of human play data and competitive play data. These agents performed the best and were better than the agents trained through self play data alone (See Fig. 10 & 11).

C. Comparison of Competitive Play Agents

When comparing the different competitive agents, we see that the the performance of the multi-agent system decreases considerable when the data is filtered whereas the performance increases considerably when human play data is injected along with competitive play simulation data (See Fig. 12, 13, 14 & 15). This is true for both the goat and tiger agents and could be because of the increased data in training as well as the higher quality of human play data. We can also observe that when we change the acceptance condition, the performance of the agent doesn't improve against the base agent but increases noticeably against the best self play agent.

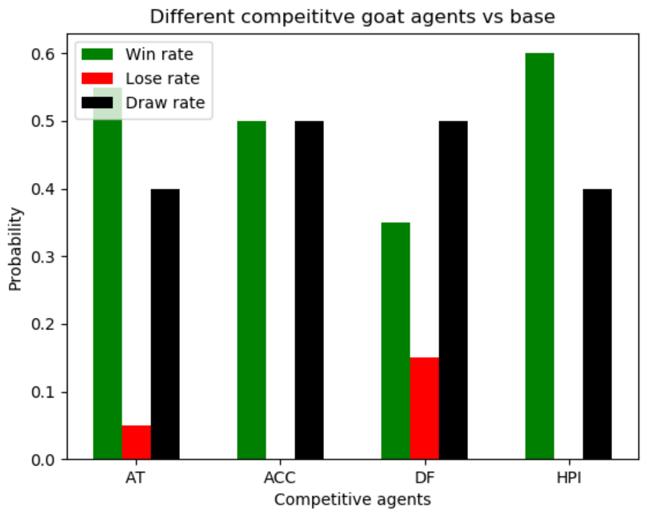

Fig. 12. Best competitive goat agents vs base

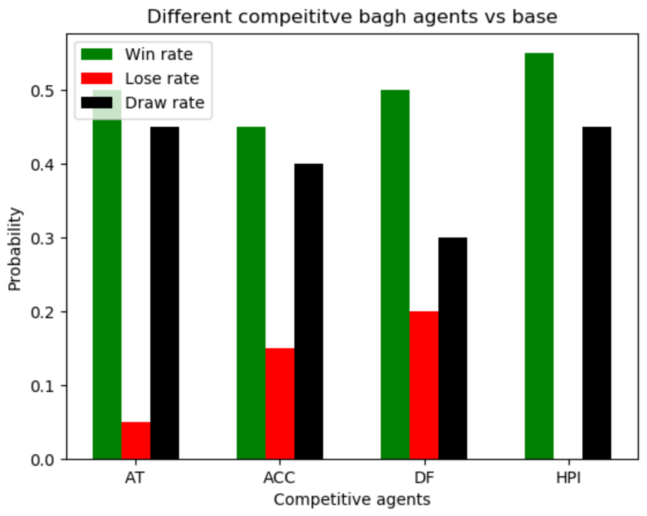

Fig. 14. Best competitive tiger agents vs base

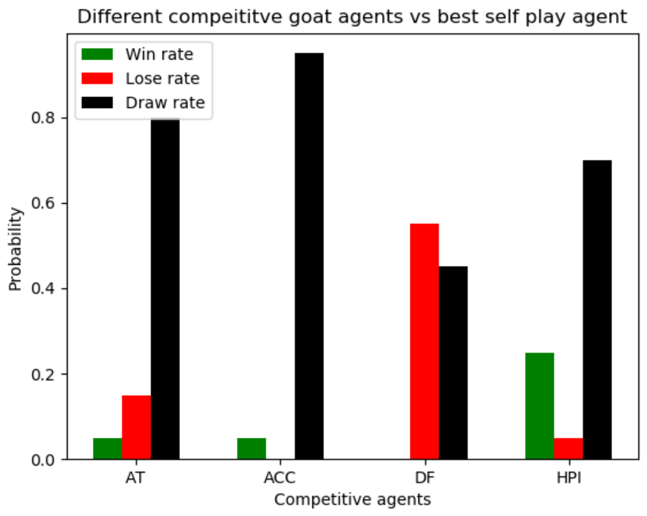

Fig. 13. Best competitive goat agents vs best self play agents

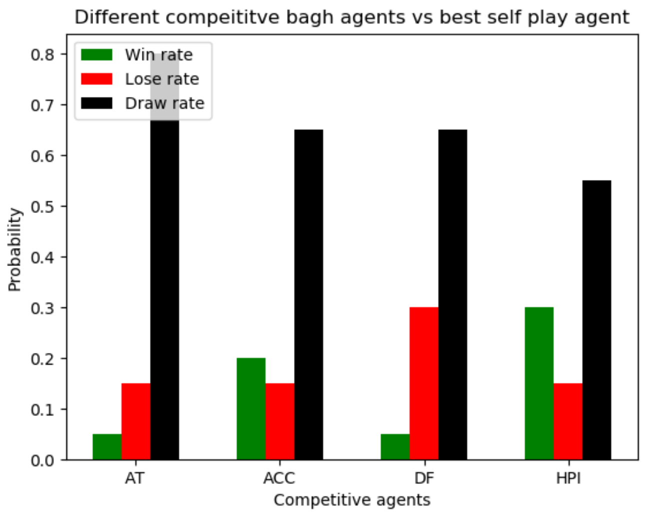

Fig. 15. Best competitive tiger agents vs best self play agents

V. Discussion

The multi agent systems that have been trained with competitive data to beat the other agent (adversarial) have performed quite good against the single agent systems using self-play when playing asymmetrical board games like Baghchal (at 0) (See §IV-C). This is even better considering the number of iterations the agents have been trained for when trained in competitive/alternative fashion. However, using such a system with multiple agents adds additional variable and constraints of the system which can make the implementation complex. There are many factors to consider like the collaboration and competitiveness of the agents, the relationship between the agents, the method of combining the agents to emulate a single better system and many more.

Further, we also see remarkable improvement in performance, when human data has been injected to the competitive play data used to train the agents (See §IV-B4) which implies that it is worthwhile to explore solutions that include human input from time to time. This results in a mixing of high quality data from human play against the agents with the auto generated data using simulations. From the findings of the study, we can see that such a mixing of data can improve the performance of the agent faster and beyond what is possible when only using auto generated data. However, generating data through human play can be very costly in due to many factors and can require very long duration of human involvement when dealing with even more complex tasks. This implies that there is a need for a development of simulation systems that themselves are intelligent to some aspect so that they can generate a higher quality of data. This has already been made possible in some cases by using already trained networks. However, there is still enough room for research in this area to create systems that can simulate a more human like behavior to generate quality data for faster training and improved performance.

Overall, the results from the study implies that the use of a multi agent system and dividing the tasks that needs to be mastered among the agents is a viable concept to deal with complex problems that have multiple dimensions of strategies, states and objectives. Using multiple agents can result in an overall combined system that can deal with a much more complex problem that would be otherwise very difficult to master for a single agent. This implies that there is a huge potential of such systems in solving complex real life problems with fewer resources in the future. Further, the results also imply that periodic human input to create rich quality data can be very valuable in training the agents.

VI. Limitations and Future Work

Due to the lack of computational power, we could not carry out an extensive hyper parameter tuning. Further, the study could only implement PPO agents but there are other agents like Advantage Actor critic (A2C) agents that need to be studied in asymmetrical and complex environments. Future studies can focus on comparing multiple agents that use different models or a combination of such models to solve complex problems by dividing the tasks among the agents. Further, future studies can also focus on extensive tuning of the hyper parameters to add to the findings of this study. Other works can also focus on using larger group of individuals to generate data for training and mixing a more proportionate amount of human data with simulated data as this project only adds 4 games of human play data every 10 iterations. This is quite small amount of data even though the human data are not removed and only aggregated for 80 iterations.

This study also used only 20 simulations forward and trained the agents only for 80 iterations (each of 20 epochs) due to the limitations imposed from the computational power required to train the agents. Future work can focus on more extensive studies as real life complex problems require much larger number of simulations than 20 and the agents need to learn over many more iterations than just 80 to perform satisfactorily. There are also many challenges that needs to be studied in much more detail than in this study before such a system can be applied to solve complex problems. One of such problems is the way the agents can work together with each other using collaboration or competition. This was fairly simple in a two player board game but will become quite complex in real life problems. However, there can be huge benefit of solving these problems as the potential of a multi agent systems can be huge in tackling real life problems that are asymmetrical and extremely difficult to solve by using only a single agent.

Acknowledgment

I would like to express my gratitude to Dr. Nick Flann for teaching me to tackle problems related to reinforcement learning and guiding me throughout this project. I would also like to thank Chris Brown for his help and guidance throughout this project. Lastly, I would like to thank my colleagues for their valuable feedback in each stage of this project.

References

[1] Wikipedia, "Bagh-chal," https://en.wikipedia.org/wiki/Bagh-Chal, accessed: 2021-11-29.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[4] S. Agarwal, M. N. A. Khalid, and H. Iida, "Game refinement theory: Paradigm shift from performance optimization to comfort in mind," *Entertainment Computing*, vol. 32, p. 100314, 2019.

[5] A. K. Yadav and S. S. Oyelere, "Contextualized mobile game-based learning application for computing education," *Education and Information Technologies*, vol. 26, no. 3, pp. 2539–2562, 2021.

[6] Y. Bai and C. Jin, "Provable self-play algorithms for competitive reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 551–560.

[7] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," *arXiv preprint arXiv:1603.01121*, 2016.

[8] D. Zha, J. Xie, W. Ma, S. Zhang, X. Lian, X. Hu, and J. Liu, "Douzero: Mastering doudizhu with self-play deep reinforcement learning," *arXiv preprint arXiv:2106.06135*, 2021.

[9] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[10] V. Könönen, "Asymmetric multiagent reinforcement learning," *Web Intelligence and Agent Systems: An international journal*, vol. 2, no. 2, pp. 105–121, 2004.

[11] ——, "Dynamic pricing based on asymmetric multiagent reinforcement learning," *International journal of intelligent systems*, vol. 21, no. 1, pp. 73–98, 2006.

[12] S. Reis, L. P. Reis, and N. Lau, "Game adaptation by using reinforcement learning over meta games," *Group Decision and Negotiation*, vol. 30, no. 2, pp. 321–340, 2021.

[13] M. Shen and J. P. How, "Robust opponent modeling via adversarial ensemble reinforcement learning in asymmetric imperfect-information games," *arXiv preprint arXiv:1909.08735*, 2019.

[14] E. Tampubolon, H. Ceribasi, and H. Boche, "On information asymmetry in online reinforcement learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4955–4959.

[15] A. Goldwaser and M. Thielscher, "Deep reinforcement learning for general game playing," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 02, 2020, pp. 1701–1708.

[16] M. G. Reyes, "Reinforcement learning in a marketing game," in *Intelligent Computing-Proceedings of the Computing Conference*. Springer, 2019, pp. 705–724.

[17] V. Kononen and E. Oja, "Asymmetric multiagent reinforcement learning in pricing applications," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 2. IEEE, 2004, pp. 1097–1102.

[18] P. Beau and S. Bakkes, "Automated game balancing of asymmetric video games," in *2016 IEEE conference on computational intelligence and games (CIG)*. IEEE, 2016, pp. 1–8.

[19] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[20] P. Barros, A. Tanevska, and A. Sciutti, "Learning from learners: Adapting reinforcement learning agents to be competitive in a card game," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 2716–2723.

[21] S. J. Basnet, "Mastering the nepali board game of bagh chal with self-learning ai," https://www.programiz.com/blog/mastering-bagh-chal-with-self-learning-ai, accessed: 2021-11-29.

[22] A. Shrestha, "Competitiveplay_multiagent_baghchal_v4," https://colab.research.google.com/drive/1w6jCKoYvh_OBXlM4hZeaddViEa5yDCr0?usp=sharing, accessed: 2021-12-01.