Investigation of the behavior of centralized, decentralized and hybrid agent with different reward functions SMAC

Huy Huynh
Computer Science Department
Missouri State University
huy0312@live.missouristate.edu

Siming Liu
Computer Science Department
Missouri State University
simingliu@missouristate.edu

***Abstract*—SMAC is a co-operative multi-agent reinforcement learning environment for the Real-Time strategy game Starcraft2. It includes many maps with a variety of unit types that are both symmetrical and asymmetrical. In these experiments, we investigate the behavior of the agents under many different scenarios and parameters. Our goal was to develop a network that can achieve a stable high win rate, with a high number of units left alive. MAPPO was research on using Proximal Policy Optimization algorithm for multi-agent co-operative environment. In our investigation, we used MAPPO as a tool to construct our scenarios: a centralized network with a centralized replay buffer, a decentralized network with a decentralized replay buffer (multi-agent), and a centralized network and buffer based on the number of types of units (multi-type). It was observed that the centralized version achieves a very stable learning process, as all the agents have a shared knowledge space. However, the results have shown that the multi-agent and multi-type version often achieve a better win-rate towards the end of training. Even more surprising is the results of negative rewards versus positive rewards. With a correct balance between offensive and defensive rewards, one can completely outperform the other.**

I. Introduction

SMAC is an abbreviation for StarCraft 2 Multi-Agent Challenge, is a reinforcement learning environment [1].In this environment, the agent(s) will control a n-number of units to fight against the enemy units. The agent will perform action based on each agent's local observation for decentralized training or the global state for centralized training. The goal is to maximized the win rate in each scenario.

MAPPO will be the tool that we used in our investigation. This tool have provided us with base classes, that we can easily manipulate and configure to make our desire scenarios. Initially, this research perform centralised training for SMAC, which would be an ideal base metrics for us to make comparison.

In our investigations, we aimed to discover ways to achieve better win rate in the past methods by tuning the training scenarios. More specifically, we compared three different method of training using the based agent classes: Centralized, Decentralized and multi-type

In addition to these 3 scenarios, we also set out to investigate the scenarios with different rewards, namely between positive only reward and a combination of positive and negative rewards. In all of these environment, the networks will be updated using the Proximal Policy Optimization algorithm.

II. Background

A. Proximal Policy Optimization

Developed by OpenAI in 2017, Proximal Policy Optimization or PPO, is a fairly new on-policy learning algorithm to update a neural network, part of the policy gradient family in reinforcement learning. In other common Policy Gradient algorithm, they perform a single update with a single batch sample. Where as PPO will perform multiple updates with multiple minibatch. It also utilizes some of the advantages of the trust region policy optimization (TRPO)

B. MAPPO

MAPPO is a research in implementing PPO to multi-agent settings. It was because on-policy reinforcement learning algorithm are believed to be less sample efficient, comparing to other off-policy algorithm. Therefore, they have set out to implement PPO into multiple multi-agent environment such as: Hanabi, MPE, and SMAC

III. Training environment settings

A. Infrastructure

For this environment, we used the Actor-Critic method combined with the replay buffers. Depending on the scenarios, the replay buffer are configured differently as mentioned in the introduction.

For PPO, the number of epoch to update the network were set to 5 and the number of minibatch is set to 1. For the update schedule, the network will be updated on every iteration. The number of steps during one episode is set to 400.

For the SMAC environment, we mostly use the default configuration. In the training process, there is a total of 20 separated environment running at the same time to speed up the training process. This will help to provide some exploration in the training process. The actions the agent can perform include: moving(up to 5 directions), attack, stop, and do

nothing. Some other units such as the Medivacs will have a heal action instead of an attack action.

Weighs and Bias will be the main tool that we used for logging and plotting the results

B. Methodology

Like any other reinforcement learning projects framework, the code for this project is very modular. For simplicity, only the relevant classes and functions will be mentioned. To start off with, you will have the main class, the main class will contain the following classes:

- Runner class: the runner class it self also inherit the base runner, which is that actual class that contains all the networks and the replay buffers class. The upper runner class is just another layer of abstraction to interact with the base one. This is the class is used to interact with the environment (functions such as step, reset), the policy networks(functions such as feed forward, compute return, learn), and he replay buffer (functions such as insert, sample)
- The SMAC environment wrapper: in this project, multi-threading is used, in order to boost training speed and improve exploration (as each environment has different seeds). You can set any amount of environment in training, however it is not wise to go over 10, unless you have very hardware. In the end, there is no functions in this wrapper that you should be worry about, except for the number of training environment.
- Policy class: this is mainly a wrapper for the Actor-Critic class and the Replay Buffer class. The difference is in each scenario. For centralized, only one instance of Actor-Critic class and replay buffer is created. For multi-agent, a list of length n will contain the n instance of Actor-Critic class and replay buffer class, n being the number of units. For Multi-type, a list of length n will contain the n instance of Actor-Critic class and replay buffer class, n being the number of units types. In the Policy class, you can find functions that will calculate loss, to perform ppo update, or to train the networks.

IV. Experiments on multiple scenarios

For all of the scenarios, we will use the same maps. In the environment, the rewards are set to positive only. each step will return rewards based on the damage output on the enemy units, with the death value of 10. The terminal state where the match is won will return an additional reward of 200:

- 3 stalkers and 5 zealots (3s5z)
- 3 stalkers and 5 zealots vs 3 stalkers and 6 zealots (3s5z-vs-3s6z)
- 1 Medivac, 2 Marauders, 7 Marines (MMM)
- 1 Colossi, 3 stalkers and 5 zealots (1c3s5z)

The reason these maps were picked were because they contain more than one type of units, which allow us to use them in the multi-type scenario. In the following sections, we will talk about the win rate and the average reward per step of the evaluation runs. Each scenarios are ran for 20 million steps, for 10 times with different seeds, with the rewards function set to only positive

A. 3s5z-vs-3s6z

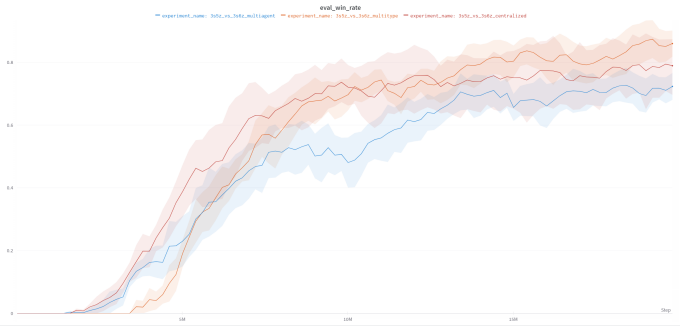

Fig. 1. Win rate of centralized, multi-agent and multi-type on 3s5z-vs-3s6z map.

In this map, there will be 3 stalkers, 5 zealots on our side and 3 stalkers, 6 zealots on the enemy side. Similar to the 3s5z map, this map will require micromanagement of the each individual units, especially the zealots, to maximize the amount of damage that they can take, while dealing the most amount of damage.

In the first 5 millions steps, it is quite clear that the multi-agent and the centralized were performing better than the multi-type. However, the multi-type scenario becomes better right after that step 5 million comparing to the multi-agent one, which remains the lowest until the end. From step 9 million to step 13 million, the centralized and the multi-type has roughly the same win rate; however the multi-type scenario surpassed the centralized scenario at step 13 million, ending with an average of 83.36% win rate. The centralized and the multi-agent scenario ended with 79.03% and 73.06% respectively

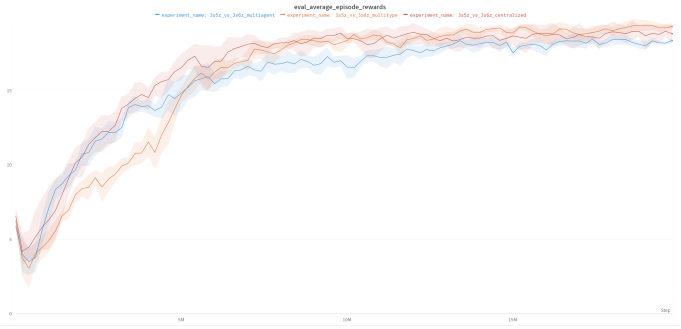

Fig. 2. Average reward per episode of centralized, multi-agent and multi-type on 3s5z-vs-3s6z map.

The same behavior can be seen in this reward figure. The centralized and the multi-agent initially achieved a higher average reward than the multi-type. At step 2 million, the centralized scenario remain higher than the multi-agent, and ended with an average reward of 18.837. The multi-type scenario surpassed the multi-agent at around step 6 million and surpassed the centralized at around step 13 million, similar to the evaluation figures, and ended with an average of 19.265.

B. 3s5z

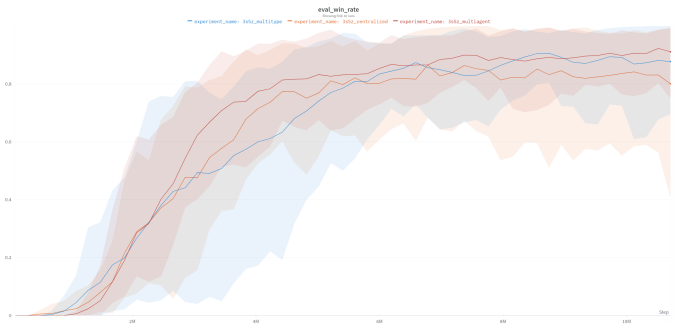

Fig. 3. Win rate of centralized, multi-agent and multi-type on 3s5z map.

In this map, there will be 3 stalkers, 5 zealots on our side and 3 stalkers, 5 zealots on the enemy side.Which mean that this map has a symmetrical scenario. Similar to the 3s5z-vs3s6z map, this map will also require micromanagement of units to maximize damage output and damage taken.

From the start, it seems that all three scenarios have relatively the same rate of winning, with the multi-type one learning a little bit better than the other two. However, at around step 2.2 million, the multi-agent scenario gain more win rate at a faster pace comparing to the other two. And it remains a better win rate until around step 6.5 million, the difference sometimes were as high as 12.98% when comparing to its nearest scenario. The remaining two scenarios only shows a noticeable difference at around step 3.2 million, with the centralized scenario being on top. On step 5.8 million, which is where the centralized scenario had caught up, and slowly gain a higher win rate than the multi-agent behavior. From here, the centralized scenario seems to be very close to the multi-agent scenario, at one time it manage to surpass it. However, this seems to not be consistent to the end of the running process, as the multi-agent scenario continues to remain the highest until the end, ending at 91.08%. The multi-type and the centralized scenario ended with 87.68% and 80.01% win rate respectively.

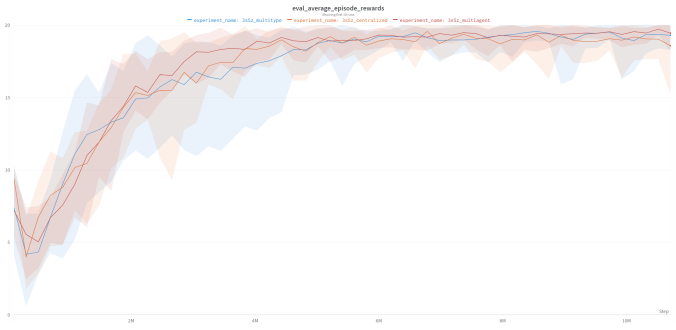

Fig. 4. Average reward per episode of centralized, multi-agent and multi-type on 3s5z map.

The average reward per episode mirrors the pattern of the win rate figure, even though the difference seems to be much smaller when comparing the two figures. The multi-type scenario did also had a higher average reward comparing to the other two scenarios. Around step 2 million is where there are some difference. Similarly to the win rate figure, the multi-agent average reward mostly remain the highest, while the centralized is the second highest. From step 5 million on wards, the difference between the three scenarios is even smaller. However, most of the time, the centralized average reward mostly remain the lowest out of the three, ending with only 18.59, while the multi-type ended with 19.31 and the multi-agent ended with 19.453.

C. 1c3s5z

Fig. 5. Win rate of centralized, multi-agent and multi-type on 1c3s5z map.

In this map, there will be 3 stalkers, 5 zealots, 1 colossus on our side and 3 stalkers, 5 zealots, 1 colossus on the enemy side. This map is similar to the 3s5z map, with an additional unit, the colossus, which is a unit that can deal damage to multiple unit at the same time. Utilizing this unit is the key to winning this map.

Just from a quick look, it can clearly be seen that this map is easier to solve than the 3s5z and the 3s5z_vs_3s6z, as all three different scenarios achieved a 100% win rate. The difference lies in the first 7 million step. The multi-agent scenario quickly achieved the 100% win rate, just after 2 million steps. The centralized scenario and the muti-type scenario are less consistent, showing no real better one. At step 1.9 million, the multi-type finally be able to surpass the centralized one with consistency. It was able to achieved 100% win rate at step 4 million, which took more than twice as long comparing to the multi-agent scenario. The centralized scenario showed an even slower rate to achieved the same win rate, which occurred at step 7 million. From here, apart from someo outliers, all three scenarios remain consistent at 100% win rate.

The average reward is a mirror of the win rate figure, therefore there is not much to discuss on this figure that have not been discuss in the win rate figure. In the end, all three scenarios ended with an average of 20. The order in which they achieved this is also similar, where as the multi-agent achieved first, next is the multi-type, and the centralized achieved it last. Even though this map introduce an additional unit (the colossus) when comparing to the 3s5z and the 3s5z_vs_3s6z

Fig. 6. Average reward per episode of centralized, multi-agent and multi-type on 1c3s5z map.

map. The difficulty is not as high, because the additional is a very great tool that can be use very effectively. Since the colossus can deal damage to multiple units as once, and it is also a ranged unit, the agents in all three scenarios have learned that the best strategy is to focus fire the enemy colossus, while protecting its own unit. With this strategy, it can create a huge imbalance in the power of two sides, which favors the agent. Hence a faster learning process.

D. MMM

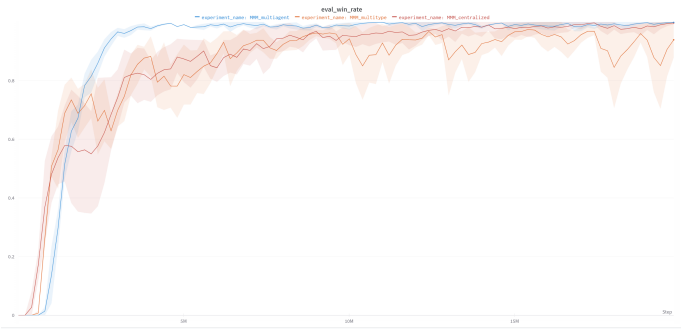

Fig. 7. Win rate of centralized, multi-agent and multi-type on MMM map.

This map is the most difference from the other three maps, where the units work entirely different. The marines and marauders are both ranged units, the medivac is a flying unit that heals it allies and can not attack. The marines can attack both ground and flying units, where as the marauders can not.

Taking a look at the win rate figure, especially the multi-type and the centralized scenario. They seems to have the same learning rate, all throughout step 10 million, except for a small period from step 1.2 million to step 3 million, where the multi-type performed better. Though with same learning rate, the multi-type scenario had a sudden drop after step 10 million, this was because in some of the runs, there were some cases with worse performance, which drags down the overall performance. Therefore, the multi-type was only able to end with an average of 92.41%. The multi-agent scenario outperformed the other two tremendously, though only after step 2.2 million. It was close to achieving 100% win rate, however not as close as the 1c3s5z scenario. The centralized scenario also manage to achieve close to 100% win rate towards the end of the process

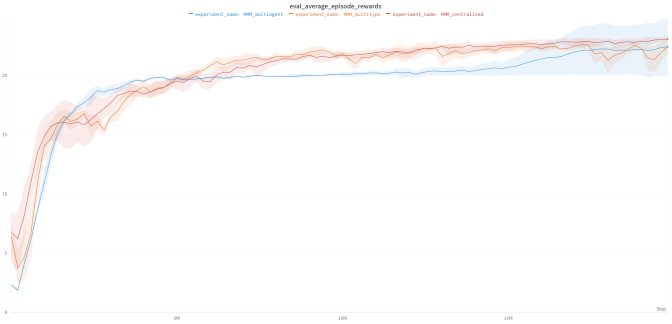

Fig. 8. Average reward per episode of centralized, multi-agent and multi-type on MMM map.

We can immediately see that this average reward figure is very different from the reward figure, where it was usually the same for the three previous map. Previously in the win rate figures, the multi-agent scenario showed the best out of the three, however it is not the case in this figure. Even though the pattern did match from step 2 million to step 6 million, which is where the average reward is 20. In every other maps and scenarios, they all have a ceiling of 20, but that seems to not be the case in this map, though it was able to break through the 20 ceiling at around step 13 million

E. Conclusion

In theory, in the centralized scenario, every agent has to make action in conjunction with each other, therefore, we predicted that it will not be very robust in term of strategy. In the multi-agent scenario, it's the opposite since each unit has their own policy and memory, they should be performing more robust in term of strategy. And lastly for the multi-type, it's a combination of the two, this will help balance out the amount of coordinated actions and independent actions. Therefore, the expected performance in the ascending order is: centralized, multi-agent, multi-type (centralized and multi-agent could have similar performance, multi-type should have a better performance than these two).

	centralized	multi-agent	multi-type
3s5z	80.01% (3)	**91.08%** (1)	87.68% (2)
3s5z_vs_3s6z	79.03% (2)	73.06% (3)	**83.36%** (1)
1c3s5z	100% (3)	**100%** (1)	100% (2)
MMM	99.28% (2)	**99.69%** (1)	92.41% (3)

TABLE I
ENDING WIN RATES

To summarize the results that we had above, Table 1 shows the summarized view of the final ending win rate of each scenario on each map. It is clear that using a the multi-agent method achieved the most positive results, being able to achieve it the fastest in 3 out of 4 maps, however it perform the worst on 3s5z_vs_3s6z. The centralized scenario and multi-type scenario both had 2 second place, but the multi-type

scenario had less a first place, comparing to the centralized 2 thirds, so it is also clear that multi-type is better, on average, performs better than the centralized scenario. Overall, the multi-agent have shown the best performance, with multi-type scenario coming in second and the centralized scenario performing the worst.

V. Positive vs Negative

In the positive and negative rewards experiments, we have decided to test it on 3s5z and 3s5z_vs_3s6z in the centralized scenario, because it can has the fastest training time.

A. 3s5z

In the positive only scenario, in each step, rewards are granted based on the damage the enemy takes, multiply by a scaling value of 10, winning a match will grant a massive reward (default is 200). While in the negative scenario, rewards are granted based on damage the ally and enemy units took, negative for ally unit and positive for enemy unit. In this scenario, we also want to steer the policy towards minimizing damage input and maximizing damage output. We did this by increasing the scale value for damage taken for both the ally and enemy units from 10 to 50, while decreasing the match win reward from 200 to 10. Each scenario is ran 3 times, at 20 millions steps

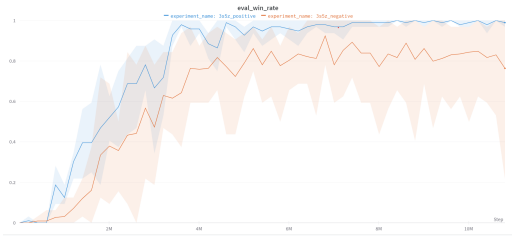

Fig. 9. Positive vs Negative rewards on 3s5z map.

Looking at the results from the figure, it is very clear that the negative scenario (blue) give a significantly better results at every step compared to the positive results (orange). Because in this map, the key to winning is by micro-managing your units to minimize the damage taken and maximize the damage output. By introducing a negative reward into the equation, it helped the network learn how to minimize the damage taken, therefore, it was able to achieve great performance

B. 3s5z_vs_3s6z

Initially, we used the same parameters as the previous experiment in the 3s5z map,however, the result was not as expected. With an addition unit on the enemy team, the difficulty sky rocketed. The network was failing to discover a winning strategy, therefore we tried to change the match win reward to 100, to encourage the exploration of strategies that win the game, and increase the training to 100 million steps

From the result, if we compare the first 20 million steps of this scenario (orange) to match with the 3s5z scenario (blue), this scenario seems to not be performing to good.

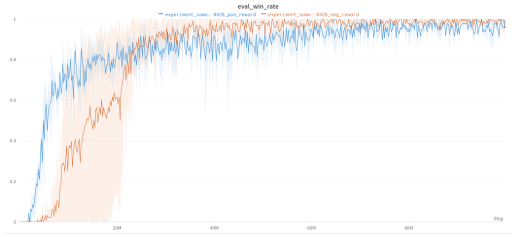

Fig. 10. Positive vs Negative rewards on 3s5z_vs_3s6z map.

The additional unit on the enemy side create too much of an imbalance that it is difficult for the network to explore the winning strategy. However right after this 20 million mark, the positive scenario stagnated and improve at a much slower rate, and in the end performed worse than the negative scenario. Further more, the area around each line is the max-min of all the runs. That means that in the first 20 million step, the values varied a lot, which might have been because of the additional unit on the enemy side. Therefore, the nework needed more effort to discover the right strategy.

C. Conclusion

Overall, taking negative rewards into considerations certainly helped improve the policy. In the default scenario where there is only positive result, even though to achieve a win, the policy has to learn to avoid damage, but it was still being rewarded solely on dealing damage any winning the game, or you can say that it was taught to only play offensively. With the introduction of negative rewards, it now has to learn to balance out the amount of defensive and offensive actions. Taking a defensive action might sound like it is not proactive, however saving the damage taken might give that unit a better position to output more damage compare to the case where they did not try to minimize the damage taken. Therefore, in strategy game where you have many trade offs and actions to take, it is important to take into consideration every variable (in this case to take into consideration the damage taken), as it could help the machine gradually learn the optimal policy to achieve the win.

VI. Future Work

There are still many work to be done, especially on the positive vs negative scenario, since it was only tested on two maps and one scenario. The three scenarios can also be further experimented on with different configuration, such as having a shared replay buffer for the multi-agent and the multi-type scenario. Furthermore, there is also room to introduce a better network architecture into the Actor-Critic network.

The scope of this project is also limited to only the SMAC environment,

References

[1] Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E.,

Dunning, I., Mourad, S., Larochelle, H., Bellemare, M. G., Bowling, M. (2019, February 1). The Hanabi Challenge: A new frontier for AI research. arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/1902.00506v1

[2] de Witt, C. S., Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P. H. S., Sun, M., Whiteson, S. (2020, November 18). Is independent learning all you need in the StarCraft multi-agent challenge? arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/2011.09533

[3] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I. (2020, March 14). Multi-agent actor-critic for mixed cooperative-competitive environments. arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/1706.02275

[4] Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., Whiteson, S. (2019, December 9). The starcraft multi-agent challenge. arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/1902.04043

[5] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017, August 28). Proximal policy optimization algorithms. arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/1707.06347

[6] Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y. (2021, July 5). The surprising effectiveness of PPO in Cooperative, multi-agent games. arXiv.org. Retrieved May 14, 2022, from https://arxiv.org/abs/2103.01955