Exploring the Impact of Reward Shaping on MicroRTS AI Agents: Milestone

Timo Loher **ID: 17-929-910**
Samuel Räber **ID: 19-924-588**

1 Introduction

1.1 MicroRTS

MicroRTS with its gym wrapper MicroRTS-py [1] is a RTS (real time strategy) game specifically developed for AI research. Compared to a traditional RTS game designed for human players there are a couple of advantages for AI:

- Simplicity: To make training RL agents as quickly as possible the game is kept at a minimum complexity. The game maps are kept very small and the actions that each unit can take is reduced to a minimum. The game has one type of resources, two types of building and four unit types. This is significantly simpler than commercial RTS games. Despite these simplifications, the core game play of RTS games and their challenges (gathering resources, micromanage units etc.) is still there.
- Interface: Usually, RTS games output an image and sound and take mouse clicks and keyboard keys as input. For an AI it is quite a challenging task to make sense of images and this is a field of research on its own. MicroRTS features a library for training AI agents called MicroRTS-py that removes this difficulty by giving the agents access to the game state directly. Agents can also issue inputs directly as opposed to a UI designed for humans.
- Performance: MicroRTS-py allows simulating games rather quickly due to the simplicity of the game itself and because it does not need to render an output image.

1.2 Difficulties

One main difficulty in training agents on MicroRTS is the sparsity of the rewards. An initially untrained agent is very unlikely to win a game of MicroRTS even if the enemy does literally nothing. Without any rewards the agent thus fails to learn. This is usually solved with shaped rewards, intermediate rewards for certain actions that we as experts think are good. These shaped rewards help the agent to make initial progress. However, they introduce a lot of bias and optimizing over these rewards instead of the true reward (winning) is often not optimal.

1.3 Action Guidance

Huang et al.[2] propose a technique called Action Guidance: The idea is that we train auxiliary agents on shaped rewards at the same time as the main agent, which only receives the sparse rewards we want to optimize over. The main agent then receives guidance from these auxiliary agents by sampling an using their actions with some probability. This probability decreases over time till it reaches zero. That way in the beginning the agent gets a lot of guidance from the auxiliary agents which learn a lot faster but with different shaped rewards. In the end the main agent only optimizes towards the sparse reward. This method aims to accelerate learning while still optimizing towards the real goal.
In the original paper they propose a set of auxiliary agents with different shaped rewards for action

Foundations of Reinforcement Learning, FS2024. Department of Computer Science, ETH Zürich.

guidance. However, they never tested more than one auxiliary agent at a time. It remains unclear if and what effect multiple auxiliary agents have. We hope that multiple auxiliary agents that are good in certain tasks (i.e. have shaped rewards tailored to specific parts of the RTS) could be beneficial over one single auxiliary agent.

2 Goals

In our work we are going to implement and evaluate action guidance on an agent for MicroRTS. We are going to experiment with auxiliary agents on the baseline implementation of Proximal policy optimization (PPO) provided by the MicroRTS-py library. We are interesting in a couple of things like:

- Does it help learning faster?
- Does it optimize true reward better?
- What impact has the number of auxiliary agents?
- How important is the tuning of shaped rewards in auxiliary agents?

In the Action Guidance paper they used easier tasks in addition to just winning the game. They introduced three tasks:
"Produce Combat Units" (PCU): The agent gets points for each combat unit created.
"Learn to Attack" (LTA): The agent gets points for each successful attack of any unit.
"Defeat Random Enemy" (DRE): The agent has to defeat a bot that performs biased random moves.
We are going to recreate these experiments and check if we can reproduce the paper's result and use these for comparison against further experiments. We are also going to make our own experiments to gather further insight into the impact of Action Guidance.

3 Current Results

We currently have established some baselines: We trained the "PPO Gridnet Large" agent, provided by the MicroRTS-py library, on the PCU and DRE tasks with both shaped and sparse rewards. For simplicity we did not change any hyper-parameters of the PPO implementation and kept them the same over all experiments. We decided to skip the "Learn to Attack" task because we lack information from the Action Guidance paper to recreate it.
We also re-implemented the Action Guidance for the latest version of MicroRTS-py. This was necessary since the code provided on the supplementary Action Guidance GitHub is not compatible with the available MicroRTS-py version and could not be run anymore.
We used this new implementation to recreate experiments from the action guidance paper. For our experiments the agents took guidance with probability 0.95 for the first 500,000 time-steps and then had an adaptation period of 1,000,000 time-steps where this probability linearly decreases to 0 and then finished training for in total 2,000,000 time-steps.
This worked well for PCU, the simpler task, but did not work for the harder to learn DRE. Figure 1 and 2 show the difference between action guidance and sparse rewards for PCU. Action guidance learns a lot quicker than the sparse reward and shows good results even though it was just trained for 1,100,000 time-steps.
For DRE action guidance performed a lot worse than standard PPO trained on shaped rewards and even after 2,000,000 time-steps did still not win games against the random biased bot as can be seen in Figure 3 and 4. We expected the action guidance agent to perform similarly (because in the first 500k time-steps it has a 95% chance of making the actions that the agent trained on shaped rewards would, so at least at the beginning we expected very similar performance. We will investigate further why this is the case or if this is due to a fault in our implementation.
However, PPO with sparse rewards did not learn anything as well in this time-frame. We then tried action guidance with a passive enemy which does not issue any actions at all. This which worked well, after 200,000 time-steps our agent started to win every time.

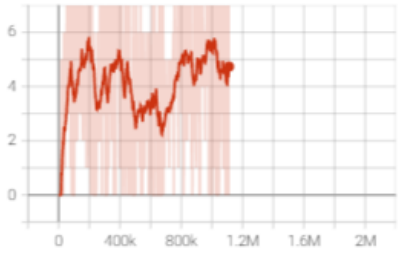

Figure 1: Produce Combat Unit reward when trained with action guidance

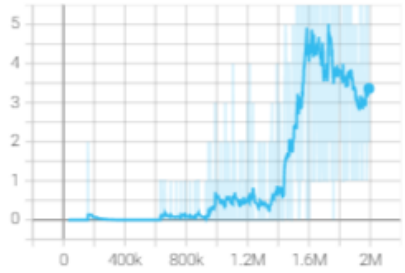

Figure 2: Produce Combat Unit reward when trained with sparse reward

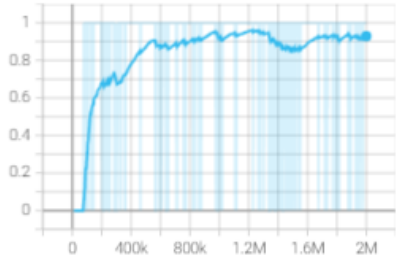

Figure 3: DRE win/lose reward when trained with shaped reward against a biased random bot

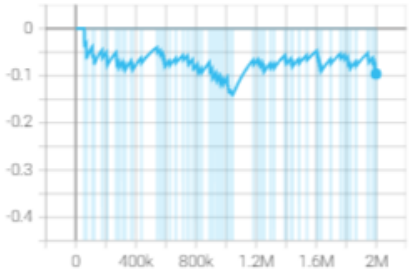

Figure 4: DRE win/lose reward when trained with action guidance against a biased random bot

4 Future Work

We have a couple of next steps we are going to work on:
We are going to expand our experiments to see if it also provides benefits on harder tasks like for example winning against more capable bots instead of only the biased random AI.
We are also very interested in the comparison between one auxiliary agent and many trained on different shaped rewards. That is, currently we trained the auxiliary agent on MicroRTS-py's default shaped rewards (+10 for winning, 0 for drawing, -10 for losing, +1 for harvesting one resource, +1 for producing one worker, +0.2 for constructing a building, +1 for each valid attack action it issues, +4 for each combat unit it produces). We want to investigate how the algorithm performs when multiple auxiliary agents are trained on only parts of the shaped rewards. For example, one auxiliary agent receives rewards only for harvesting resources, another for producing workers and so forth. The main agent can then learn to optimize the sparse reward, for example the reward for winning the game, while receiving guidance from a set of auxiliary agents which learn simpler tasks. We will experiment with different combinations of agents and rewards and compare them to shaped rewards and to each other.

References

[1] Shengyi Huang et al. *Gym-μRTS: Toward Affordable Full Game Real-time Strategy Games Research with Deep Reinforcement Learning*. 2021. arXiv: 2105.13807 [cs.LG].

[2] Shengyi Huang and Santiago Ontañón. *Action Guidance: Getting the Best of Sparse Rewards and Shaped Rewards for Real-time Strategy Games*. 2020. arXiv: 2010.03956 [cs.LG].