Adrian Bremer

Report for microRTS

1. Basic Ideas

At the beginning of the project, for me it was about collecting ideas and inspecting the source code of microRTS to know how everything works and what is possible.

In the process, I came across that there are already implemented bots in the package "ai". Although my original plan was to apply a reinforcement learning algorithm, such as Monte Carlo Tree Search, we hadn't progressed that far in the lecture and it wasn't 100% clear to me how to put the theory into practice. For example, where do the values for the return come from? All this was not clear to me.

So the first step was to let some of the already implemented bots compete against each other and see which one performs best and at the same time is the easiest to understand. The result of this experiment was that WorkerRushPlusPlus, RangedDefense and CRush_V2 perform quite well and are also better than bots like MonteCarlo.

Since WorkerRushPlusPlus is only good on quite small maps, I didn't want to build on this bot, especially since then you don't have time to develop strategies. Much more interesting was RangedDefense.

2. Building up on RangedDefense

The problem with RangedDefense is that troops only attack when they get attacked. As a result, RangedDefense may win, but simply stop in the middle of the field, resulting in a draw.

So the first step in improving was to slowly increase the distance that the attacking units are allowed to be away from the base. This is simply multiplying the counter for the relative distance from the base by a small value plus 1 in each iteration, so that the bot becomes more and more aggressive as the game progresses.

The next thing I noticed is that Ranged units only have one hit point, which makes them very easy to destroy by Light and Heavy. On the other hand, what is very advantageous about them is that they don't have to move as much because of their range and other units can run in front of them. That's why I chose Heavy to put them in the front line to protect the Ranged units. To implement this "placing in front", a Heavy may stand one cell further away from the base than a Ranged.

3. More Improvements

But there were some problems: First of all, since the bot was becoming more and more aggressive, it's behavior evolved into that of a Rush bot, which wasn't good because then all the units got wasted. So, the next step was to reset the attacking behavior when the army died out. Later I changed it so that the distance troups are allowed to be away is measured from the nearest building, not only the base. Also, it would be very good if units returned to the front line. So they would not be such easy targets. However, this didn't work as easily as I thought, which is why I didn't implement it or fix it at first.

Also, once the base is destroyed, you have more or less lost, because then you can't save any resources and can only work with the units you currently have. That's why a base should be built when you only have one and there are enough resources for it.

Another point is that there is only one worker, which means that resources are collected very slowly and the infrastructure grows slowly. That's why I thought about that a base can be surrounded by 4 workers and so there can be a maximum of 4 workers per base. How many workers a base trains depends on the resources nearby. I defined this proximity so that several bases do not overlap. However, then there is only one worker, whereby the collecting becomes even slower. Later I changed this behavior, so that we have at least 2 workers (harvesting and building stuff) and there can be more, if the ressources are far away (distance/4 for the maximum amount of workers). If there are too many workers due to the mass of resources, then more than 2 bases can be built.

Otherwise barracks should be built as soon as enough resources are available for it. This way troops can be trained faster and the army grows faster, which makes it easier to take the enemy by surprise.

The building of the bases and barracks happens automatically, because the gathering speed of the workers is expressed in the number of resources in the base and the consumption is described by the barracks. So once we have enough resources, there is enough capacity to build more infrastructure.

Another extension is that troops and buildings block each other. To avoid this, an area is defined around each building where no other building may be built. This way there is always enough space for workers and units to move and build more infrastructure.

All this causes my "GrabAndShakeBot" to perform quite well, especially on large maps.

4. Building a Wrapper-Bot

But, there is a problem: Everything of the described takes really long to get started. So on small maps the bot is doomed. Instead of extending the bot, my idea was to implement some bot, that uses many other bots to choose it’s action.

For starters, when the enemy base is closer than 20 blocks, the bot will use the decisions made by WorkerRushPlusPlus. On larger distances there is enough time to build a barracks and train a heavy unit. So, currently the bot only differentiates between WorkerRushPlusPlus and the GrabAndShakeBot.

But there is one more thing that I wanted to address being that the terrain can split the map into multiple individual parts. The new wrapper bot will determine in the preGameAnalysis which these territories are and evaluates which bot it wants to use for it. The results of this analysis are saved to disk.

One more thing I did, was to use the gameOver() method, that tells the bot who is the winner. By saving which player index I am, the bot can decide if the strategy it played was good or not. If the bot has lost, it will change the GrabAndShakeBotSetting associated with this round by flipping around the used bots and save it to disk. Therefore, the bot has to save the current file. With this functionality in place, the bot can learn a little bit.

5. Still Open Improvements

It may not sound like much, but it took a very long time to get everything working and also to read up on everything and understand the game itself. There were a lot of problems and so despite working for more than a month, I couldn't complete everything I wanted to.

On the one hand there was the fact that due to the exercises I finally understood some of the reinforcement learning algorithms and would have liked to implement them, especially since then you only have to play around with parameters instead of thinking up a strategy yourself and implementing it correctly.

But even in my last version there are still some things missing: Troops should return to the front line and troops should attack as a company to be stronger.

Some of this, I already implemented, but it didn't work 100%, so I didn't include it in the final version.