Exercise 3
CodingGame Fall Challenge 2020
Artificial Intelligence for Games

Hubert Obrzut

November 26, 2020

1 Results

`Codingame nickname`: nan0S_

`Final league: Gold`

`Final contest ranking: 255th global / 175th league`

2 First approach

Although I managed to get to the gold league (which is not so bad, but certainly could be better), I am not satisfied at all from the performance of my agent. But let's start from the beginning. At first I have tried to think about different approaches to the problem but I could not think of anything other than search algorithm. So I started with BFS implementation, where state was player's inventory content. Basically what I did is the following - I have computed shortest path from the current state to every other state (which I could afford because we have no more than 11^4 states, as we have 4 ingredients and no more than 10 ingredients in total) and then I have taken the state with the highest $\frac{price}{distance}$ ratio, as in theory, we want to maximize rupees gained on average in each round (with a few exceptions, e.g. the endgame, but let's not think about it right now, it is just a simple solution). With this approach I have managed to get to the silver league (something around the upper/middle of the leaderboard, but of course we have to remember that at that time, leaderboard was still unstable, as the contest was ongoing). It also important to add that I did not consider resting while doing search - I still did them but only when resting was necessary (I couldn't cast the spell which I wanted to cast). When it comes to learning new spells I have just taken a few first spells at the beginning of the battle.

3 Final solution

Of course the next step to improve my agent was to consider more aspects of the game. When we do that we can no longer afford to do the complete search of the space, so I have chosen (as many people did) beam search. At first I have tried to write a working solution - not the fastest one but the one that worked properly. I have added simple evaluation of the state:

$$3 * potionPrice + 1 * tier0 + 2 * tier1 + 3 * tier2 + 4 * tier3 \tag{1}$$

With that I managed to get to the gold league. My beam search was still slow, but I have decided to first improve my evaluation function. I have added few parameters, like *decay* factor (multiplied *potionPrice* with $decay^{depth}$) which coerced my agent to value potions which are brewed faster or *ordersDone* which should value just the fact that I brewed potion - as it gets us closer to the end of the game. With these improvements I have managed to get to my final position - my position in gold league improved, but that is where I stayed at the end. Unfortunately I have not managed to improve my agent even further - I have tried to write beam search effectively which I believe I succeeded, but something was still off with my evaluation function - my agent was didn't brew potions as fast as gold league's boss, so most of the time I lost with 4 or 5 brewed potions. I have spent a lot of time to make my agent work but I have failed to do that effectively. Even though the beam search was pretty fast - I got around 20 beam depth with 2000 beam width, where I even considered actions like learning new recipes (I did it with decaying the value of learning next recipes based on the count of the recipes I have learned so far), something was off and I didn't locate that. I have tried a lot of different approaches and a lot of parameters.

4 Summary

My implementation language was C++ so I did not complain about the performance although I can see that it was a crucial thing in this contest (if you get the evaluation function right), so slower languages had major disadvantage. Moreover this caused practically all solutions to be fast beam search - as we could see in postmortems. One thing that really disturbed me was visualization of the battle - when it comes to step by step run, it was really hard to see what is going on in each step. For now I will take small break from this contest, as I am tired of debugging, but maybe I will go back to it later to fix all the problems I had.