MCTS with Influence Map for General Video Game Playing

Hyunsoo Park and Kyung-Joong Kim*
Department of Computer Science and Engineering, Sejong University, Seoul, Republic of Korea
hspark8312@gmail.com, kimkj@sejong.ac.kr

***Abstract*—In the General Video Game-AI competition in 2014 IEEE Computational Intelligence in Games, Monte Carlo Tree Search (MCTS) outperformed other alternatives. Interestingly, the sample MCTS ranked in the third place. However, MCTS was not always perfect in this problem. For example, it cannot explore enough search space of video games because of time constraints. As a result, if the AI player receives only limited rewards from game environments, it is likely to lose the way and moves almost randomly. In this paper, we propose to use influence map (IM), a numerical representation of influence on the game map, to find a road to rewards over the horizon. We reported average winning ratio improvement over alternatives and successful/unsuccessful cases of our algorithm.**

Keywords—general video game playing; influence map; Monte-Carlo tree search

I. Introduction

Recently, there has been much interest on the General Video Game Playing (GVGP), the video game version of General AI. Because ordinary game AI is based on domain knowledge, it is unlikely to be successful in different games without significant tuning or modification. The purpose of GVGP is to build a general game AI playing various video games. To realize this goal, AI system need not depend on domain knowledge of each game.

There have been a few AI research platforms for GVGP. For example, Arcade Learning Environments support libraries to use Atari emulators. GVG-AI platform (www.gvgai.net) has been used for GVGP research and GVGP competitions. It's based on VGDL (Video Game Description Language) to create video games. In the competition, participants submit their AI implementation to be tested on unseen games.

In the competition of 2014 IEEE CIG, organizers provided forward-model, Monte Carlo Tree Search (MCTS), and genetic algorithm samples. The entries were variants of the samples [1]. The final results showed that MCTS was successful to solve the GVGP problems. It can be designed without (or less) game domain knowledge and search game space efficiently. Because it's based on simulation of future states, it's important to test as many as states within limited time constraints. However, it's too short time to see the future in depth, it's likely to lose the way. It suffers from the *horizon effect*. Especially, this problem can be serious in video game environments with sparse rewards. If there are not enough rewards in the near future where MCTS searches, AI player doesn't know which action is the best. In this case, MCTS might choose a random action instead of the best action for potential rewards. D. Perez *et al*. proposed a new method to solve this problem [2]. They enhanced their previous fast evolutionary MCTS. In this work, they described how to choose the next action when there is insufficient rewards.

In this paper, we propose to use an influence map to solve the MCTS's horizon effect problem. Even if MCTS doesn't find rewards, the influence map guides the AI player to the directions of potential rewards. In other words, the MCTS searches local area (around the player) and the influence map supports a global view for potential rewards. At each game tick, an influence map is created to show which place is better than others. We assumed that the collision between the player and other objects is the most important event in this platform. Based on the collision data, we propose to guess objects' type whether it's good or bad. For example, items, coins and gems are good because they increase scores and collision to them wins the game. On the other hand, ghosts, aliens and bullets are bad objects for players.

II. MCTS with Influence Map

There are three steps in our proposed method: (1) determine goodness of objects, (2) create an influence map based on the information, and (3) use influence map when MCTS is running. Fig. 1 shows an overview of the proposed method.

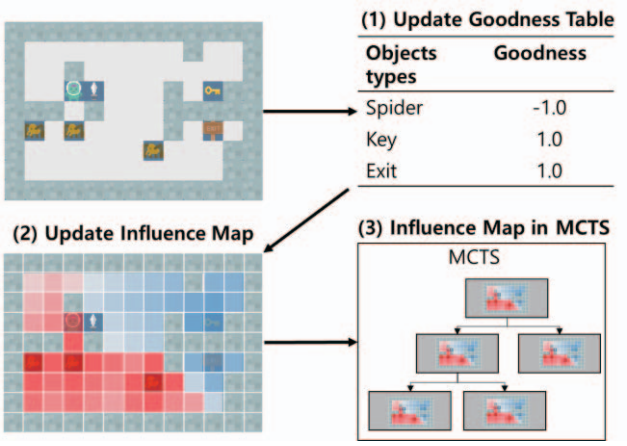

Fig. 1. Overview of proposed method (blue is good influence and red is bad influence)

At first, AI player should know which object is good or bad in order to create the influence map. Because the information is not available before the game play, AI player needs to obtain it from game playing or simulation. In this work, we used

* Corresponding author

simulation to determine objects' goodness. Initially, the value of unknown objects is set to 1.0. This is some kind of curiosity metaphor. The player don't know it is good or bad, but it has curiosity to know it. Therefore, we assign the smallest reward to the unknown objects. After starting the games, if the player gets scores when it collides to the object, it is identified as a good one. On the other hand, if the player loses scores or the game, the object's goodness is set to -1.0. In the same manner, the AI player updates the goodness table per every game tick. In some games, the player's sprite can be changed after some events. Because it's a big change for the character and there is chance that the old table is not reliable, we reset all the values in the goodness table.

Next, AI player creates an influence map based on the objects' goodness at each game tick. Initially, at the place of each object, it assigns the goodness value from the table. After then, it spreads goodness near the object. Positive influence is originated from good or unknown objects and vice versa. Because this process spends some time, it can affect the number of simulations for MCTS.

$$UCB1_i = 0.8 \times \bar{x}_i + 0.2 \times \boldsymbol{IM(pos)} + \sqrt{\frac{2\ln n}{n_i}} \quad (1)$$

The influence map is combined with MCTS's UCB equation. *IM(pos)*, influence at the position, is normalized between 0.0 to 1.0. Even if the estimation of reward ($\bar{x}_i$) is 0.0 in UCB1, the *IM(pos)* can help to find a route.

III. Experimental Results

We tested the proposed method (IM) using 32 games for 50 runs. We used three sample AIs for comparison: MCTS, Online MCTS (OLMCTS), Genetic Algorithm (GA) embedded in GVG-AI platform. Our AI was implemented based on OLMCTS. Because each game has different scoring scale and competition is based on ranking, we selected winning ratio as a measure. Although it doesn't cover all aspects of performance, it can reflect the correctness of playing.

According to experimental results (Fig. 2), average winning ratio of the proposed method is 0.268. It is higher than others (MCTS: 0.230, OLMCTS: 0.257, and GA: 0.220). Among the 32 games, the proposed method was the best for 13 games. Especially, its winning ratio was 0.4-0.5 better than *Zelda* and *Egg Mania*. Also in *Overload*, *Aliens*, and *Boulder Dash*, its winning ratio was about 0.1-015 better than others. Those games can be beneficial from the IM because it can help to avoid the horizon effect.

However the proposed method (IM)'s winning ratio was the lowest for five games. And it showed relatively low winning ratio (below 0.3) for 20 games. Although there are many reasons, the biggest one is that leading the player to good objects is not successful or sometimes worse in some games. For example, in *Sokoban*, simple collides between good object and the player is not useful in this game, but we plays the game based on the assumption. In the worst case, the game can't be solved anymore. In *Sea Quest*, AI player should consider the remaining oxygen, the original MCTS can handle this quite well, but MCTS with our proposed method doesn't react well to the low oxygen level. After the influence map generation, it leaves less amount of time for MCTS. In addition, there are no good objects that make positive influence in the oxygen refill area.

Game \ AI	IM	MCTS	OLMCST	GA
Aliens	1.000	0.763	0.893	0.933
Butterflies	0.920	0.771	0.857	0.814
Infection	0.757	0.650	0.767	0.783
Zelda	0.643	0.154	0.108	0.046
Solarfox	0.594	0.175	0.175	0.650
Missile Command	0.573	0.538	0.446	0.446
Painter	0.514	0.625	0.675	0.550
Whackamole	0.400	0.600	0.889	0.222
Egg Mania	0.386	0.000	0.022	0.000
Survive Zombies	0.380	0.323	0.431	0.215
Overload	0.357	0.196	0.174	0.196
Frogs	0.333	0.200	0.231	0.154
Sea Quest	0.143	0.467	0.400	0.222
Chase	0.133	0.439	0.576	0.288
Average	0.268	0.230	0.257	0.220

Fig. 2. The comparison of winning ratio

IV. Conclusion

In the GVG-AI platform, MCTS shows good performance in many cases, but it suffers from horizon effect in some games. If there are sparse rewards in game environments like Zelda, it is hard to pursuit the rewards. In this paper, we propose to use influence map for MCTS to solve this problem. (1) Determine which object is good/bad for a player in simulation and record them in a table. (2) Create an influence map based on the table. Assign positive values around good objects and negative values around bad objects on the game map. (3) Integrate influence map into MCTS's selection. As a result, MCTS searches near future (local area) and influence map guides the player to find global path to rewards. We compared our approach with sample AIs in GVG-AI platform for 32 games. Experimental results show that our proposed method's average winning ratio is higher than others. Although it shows low performance for some games, it can boost scores in average.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (2013 R1A2A2A01016589)

References

[1] D. Perez, S. Samothrakis, J. Togelius, T. Schaul, S. Lucas, A. Couëtoux, J. Lee, C.-U. Lim, and T. Thompson, "The 2014 General Video Game Playing Competition," *IEEE Transactions on Computational Intelligence and AI in Games*, in press.

[2] D. Perez, S. Samothrakis, and S. Lucas, "Knowledge-based fast evolutionary MCTS for general video game playing," in *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, 2014, pp. 1–8.