Design Adaptive AI for RTS Game by Learning Player's Build Order

Guillaume Lorthioir^{1,2} and **Katsumi Inoue**^{1,2}
^{1}National Institute of Informatics
^{2}SOKENDAI (The Graduate University for Advanced Studies)
{lorthioir, inoue}@nii.ac.jp

Abstract

Digital games have proven to be valuable simulation environments for plan and goal recognition. Though, goal recognition is a hard problem, especially in the field of digital games where players unintentionally achieve goals through exploratory actions, abandon goals with little warning, or adopt new goals based upon recent or prior events. In this paper, a method using simulation and bayesian programming to infer the player's strategy in a Real-Time-Strategy game (RTS) is described, as well as how we could use it to make more adaptive AI for this kind of game and thus make more challenging and entertaining games for the players.

1 Introduction

Real-Time Strategy game (RTS), is a type of game where the players need to gather resources, build structures and units, and eventually defeat their opponents.*Workers* are able to gather resources that will be needed to build *buildings* and train *units*. The players usually have partial information about their opponent since they can only see the surroundings of their own units. In StarCraft[1] there are different types of military units that can be trained in different buildings. Players, therefore, are going to follow a certain pattern in the way they will manage their resources and buildings. This pattern is called "Build Order" and gives a good hint about the strategy that a player will use. In most of the current RTS games, the AI is often not challenging or interesting to play against. Because they usually do not adapt their strategy dynamically to the players and they often cheat and have bonuses that the players do not get. We believe that creating original adaptive AI would make RTS games much more entertaining and interesting for the players.

Our work is base on the game StarCraft because thousands of replays are available online and a lot of tools to simplify data-mining and replays analysis are available. A replay is a video of a game between several players. We use replays to extract players' actions and to infer their strategies with Bayesian programming. We are focus on the Build Order inference. Part of our method is based on the work of [Synnaeve and Bessière, 2011], they provide a method using a Bayesian model to give the probability of different build orders that a player is following along a game. However, they start from a set of possible build orders which is not very refined and could be improved along the game. We build a simulation of the player's economy during a game and thus generate the possible build orders that the player could have followed according to the time and the available resources. We then refine our hypothesis by using the in-game information about the player. After this, we use [Synnaeve and Bessière, 2011] to know which one is the more likely to be following by the player. Finally, we introduce a way to adapt the strategy of the AI to the player's strategy by using constrained optimization.

2 Related Works

This work is at the intersection between "Plan Recognition" and "Game AI". Plan recognition has been investigated in AI research and has many applications. [Schmidt *et al.*, 1978] were the first ones to introduce the problem and treat it from a psychological point of view. [Carberry, 2001] describes the plan recognition problem and surveyed different ways to tackle this problem, new approaches have since emerged and many can be found in [Sukthankar *et al.*, 2014]. The field of game AI though, has proven to be a very good test bed for plan recognition, as stated by [Kabanza *et al.*, 2010; Ha *et al.*, 2011]. Especially due to the huge amount of available data and the existing API in the fields. [Weber and Mateas, 2009] evaluated several machine learning algorithms on replays that were labeled with strategies, they also provide their dataset for further comparisons. Google did a great job with their AI AlphaStar beating top players in the game StarCraft 2 [Vinyals and others, 2019] but they use reinforcement learning which is not robust if the system is modified and they trained their AI on a supercomputer, which cost them several millions of euros. Only a few methods are taking into account uncertainty and they are not very accurate, we are trying to improve this by using simulation and Bayesian models.

3 Methodology

In StarCraft, there are some basic patterns to follow. We assume that the players are following this pattern. By assuming

[1]StarCraft and its expansion StarCraft: Brood War are trademarks of Blizzard Entertainment^{TM}

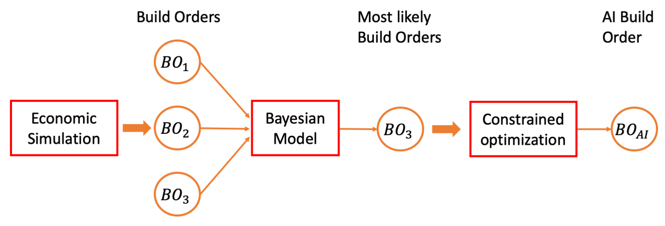

Figure 1: Illustration of the main method

this and knowing that there are only two types of resources in the game which are ”Gas” and ”Minerals” we can compute the resources available for the player at each time-step. We simulate the player’s economy by using the number of workers the player should have at a time t and using some equation we can compute the quantity of resources gathered by the workers at each time-steps. The productivity of the worker depends on the number of current workers gathering resources on the same spot. We then obtain the equation:

$$M_t = M_{t-1} + N_{exp}(N_{Wexp} * C_M * \delta) \tag{1}$$

Where equation 1 is used to compute the quantity of minerals available for the player with M_t the quantity of minerals at time t, N_{exp} the number of collecting spot for the player, N_{Wexp} the average number of workers on these spots, C_M the mineral collecting rate of these workers and δ the duration of a time-step. We have the same equation for gas with G instead of M. With these equations and the time elapsed since the beginning of the game, we can make a hypothesis about the Build Orders the player could have followed so far. The simulation allows us to generate a set of potential Build Orders that the player could have followed. Usually, in a game of StarCraft, the players are sending some units to the opponents’ bases to have an idea of what they are doing, we call this scouting and that provide partial information about the opponent’s buildings and units. This is how we get the in-game observations, by scouting. Using these observations we will keep only the Build Orders that are consistent with them. So we reduced the number of potential Build Orders by doing this, but we still need to make a choice about which one is the most likely to be followed. This is where we are going to use the Bayesian model of [Synnaeve and Bessière, 2011], we will use it on the set of Build Orders generated before to get the probability of each Build Order. To have a better understanding of this Bayesian model, please refer to the paper of [Synnaeve and Bessière, 2011]. We here focus on the ten first minutes of a game since they are decisive to understand which strategy a player will use. We call this early-game.

The final part of our method is how to adapt the strategy of the AI when the AI managed to infer the Build Order of the player using the previous methods. For this part, we plan to use constrained optimization to choose the correct strategy to use against the player. It is a bit similar to [Antuori and Richoux, 2019] but they use constrained optimization to manage their unit production. We will instead, fix a cost on each strategy to adopt and an optimization function to compute the efficiency of each strategy against the strategy of the player. We will also take into account the cost to change from the current strategy to a new one and see if it worth the cost.

4 Discussion

This work is an ongoing work, this is why the last part of the method is purely theoretical and not implemented yet. [Antuori and Richoux, 2019] shown that constrained optimization applied to RTS games is working quite well, especially they use this method during a competition of AI in RTS games and got a good score. Furthermore, we believe that our method has better scalability than methods using reinforcement learning or deep learning like [Vinyals and others, 2019] and could be applied to other RTS. We also thought about a heuristic search to find the Build Order followed by a player instead of a simulation. Because the search space is quite big and heuristic methods are very useful in these cases, although less precise. The problem is then to find the correct heuristic function, which is not trivial.

References

[Antuori and Richoux, 2019] Valentin Antuori and Florian Richoux. Constrained optimization under uncertainty for decision-making problems: Application to real-time strategy games. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 458–465. IEEE, 2019.

[Carberry, 2001] Sandra Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(5):31–48, 2001.

[Ha *et al.*, 2011] Eunyoung Ha, Jonathan P. Rowe, et al. Goal recognition with markov logic networks for player-adaptive games. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 2011*, 2011.

[Kabanza *et al.*, 2010] Froduald Kabanza, Philipe Bellefeuille, Francis Bisson, Abder R. Benaskeur, and Hengameh Irandoust. Opponent behaviour recognition for real-time strategy games. In *Proceedings of the 5th AAAI Conference on Plan, Activity, and Intent Recognition*, AAAIWS’10-05, pages 29–36. AAAI Press, 2010.

[Schmidt *et al.*, 1978] Charles F. Schmidt, Natesa S. Sridharan, and John L. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11(5):45–83, 1978.

[Sukthankar *et al.*, 2014] Gita Sukthankar, C. Geib, Hung H. Bui, David V. Pynadath, and Robert P. Goldman. *Plan, Activity, and Intent Recognition*. Elsevier, 2014.

[Synnaeve and Bessière, 2011] Gabriel Synnaeve and Pierre Bessière. A bayesian model for plan recognition in RTS games applied to starcraft. *CoRR*, abs/1111.3735, 2011.

[Vinyals and others, 2019] Oriol Vinyals et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[Weber and Mateas, 2009] Ben G. Weber and Michael Mateas. A data mining approach to strategy prediction. In *2009 IEEE Symposium on Computational Intelligence and Games*, pages 140–147, Sep. 2009.