StarCraft:Broodwar AI Report

Sungguk Cha
navinad@naver.com

April 27, 2019

Abstract

Even though artificial intelligence and machine learning have proven their potential in Real-Time Strategy (RTS) games, it is not currently at the level to compete against human. I participated in StarCraft:Broodwar AI Competitions and NAVERxUNIST Undergraduate Poster Award. This report is about how I tried on the competitions and reviewing the results.

Keywords

Strategy search; Monte-Carlo tree search; genetic algorithm; Markov decision process; deep learning; competitions

1 Introduction

RTS games, especially *StarCraft*, have more restrictions and uncountable number of states than board games like Chess and Go. There have been lots of StarCraft AI competitions since 2010: *IEEE Computational Intelligence and Games (CIG)* since 2010[1], *AAAI Artificial Intelligence for Interactive Digital Entertainment (AIIDE)* since 2011[2], and *Student StarCraft AI Tournament(SSCAIT)* since 2011[3]. In 2016, when I took part in the leagues, state-of-the-art solutions consist of many hard-coded modules rather than machine learning. In 2018, today, some bots started having machine-learning-based decision makings. However, no AI bot is at the level of skillful human-players.([2, 3])

1.1 About the Competitions

In *AAAI AIIDE*, *IEEE CIG*, and *SSCAIT*, submitted bots coded in C++ or Java using BWAPI[4] play 1v1 StarCraft matches. Through these competitions, my goal was to practice implementing what I have learned, and apply machine-learnings to it.

2 Background

Restrictions on the game play

Because StarCraft is a real-time online game, latency matters a lot. Also, because the game is a video game, the game plays more than dozens of frames within a second. Therefore, at each frame, we have very little time to compute.

Managements

If we divide the game's strategies into two parts, one is micro-management, and the other one is macro-management. Micromanagement is about micro controls like controlling units, 'how to fight'. Macromanagement is about higher-level-strategies like build-order-planning; deciding 'when to fight' and 'where to fight'. In a sense that we do not have many time to simulate combats and

[1] http://cilab.sejong.ac.kr/sc competition/
[2] http://www.cs.mun.ca/∼dchurchill/starcraftaicomp/
[3] http://sscaitournament.com/
[4] BroodWar API : https://bwapi.github.io/

combat simulation is not accurate due to lots of variations such as terrain, and troop position; researching learnings in micromanagement is not practical at all. There were many trials in a small scale of combat using a reinforcement learning, or neural networks ([4, 5]). However, the bigger the scale, the weaker the results. So, I decided to concentrate on the macromanagement.

3 Competitions

3.1 2016 IEEE CIG

The first bot ranked (10th/16 (Winning rate:46.43%))[5]. It has a few number of heuristic strategies. It is forced to choose its strategy plans in code level. Because BWAPI is object-oriented, I could be familiar with OOP concept. Though its functions are modularized, codes are very messy. More than 6,000 lines are in one .cpp file.

3.2 2016 AAAI AIIDE

Ranked (11th/21 (Winning rate: 57.25%))[6]. As shown in Figure 1, this model has sets of early,

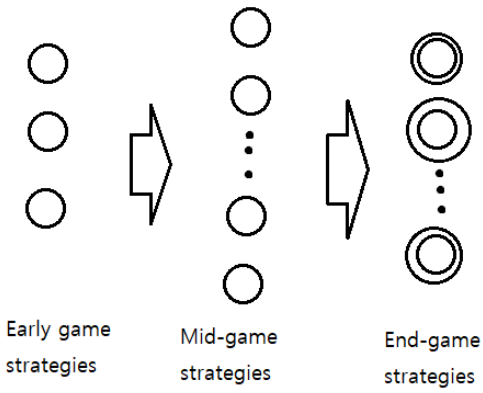

Figure 1: Strategy selection through Monte-Carlo tree search

mid, and end game strategies. Each state, circle, contains managements. At the beginning of each game, this version selects strategies (early-mid-end) empirically with Monte-Carlo tree search. More heuristics added. I could learn modern c++ styles like 'auto', and practice OOP style by modularizing functions.

3.3 2016/2017 SSCAIT

Ranked (16th/45 (Winning rate: 61%)), and won 3rd place in student division [7]. Added more heuristics. Around the competition period, SK T-Brain offered research services for making StarCraft AI. Through some interviews and processes, I was accepted.

3.4 2017 IEEE CIG

Ranked (8th/20 (Winning rate: 61.75%)) and awarded for 3rd prize in student/young professor division [8]. It is the same version of 2016 AIIDE. Some bugs fixed.

3.5 2017 AAAI AIIDE

Ranked (28th/28th (Winning rate:17.21%)) [9]. I totally changed the bot. Modularizing every function into divided files, and built up a managing network. It is an improvement or optimization.

[5]https://sites.google.com/site/starcraftaic/result
[6]https://www.cs.mun.ca/ dchurchill/starcraftaicomp/2016/
[7]https://sscaitournament.com/index.php?action=2016
[8]https://cilab.sejong.ac.kr/sc_competition2017/?cat=17
[9]https://www.cs.mun.ca/ dchurchill/starcraftaicomp/2017/

I wanted to make evolving AI. I applied genetic algorithm that starts from nothing. The number of games through the competitions seems not enough for my genetic model to evolve to the level of itself.

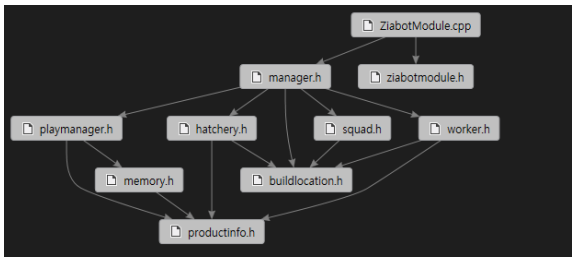

Figure 2: Managing network architecture - 2017 AIIDE version

Managing Network

Figure 2 shows the managing network architecture. Manager on the top is a kind of the bus, carrying and saving data for sub-classes. Playmanager manages high level strategies consisting of specific heuristic build-orders. It also determines what to produce(build) next. The information is carried to Manager. Hatchery produces or researches given build-orders or upgrades from Playmanager through Manager. Likewise, Worker builds buildings given build-orders from Playmanager. Squad launches attacks, holds, or defends based on the command from Playmanager; unit micromanagements are hard-coded with heuristics. Its advantages are cheap and easy to synchronize data. The architecture is the same as our computers' bus system. Likewise, the architecture is very cheap in data flow (less overhead) compared to other types of system. Modularization makes it harder to synchronize modules. It was the solution for the problem. Through developing the network, I could learn optimizations in memory usage, data structures, and time-complexities; practice OOP concept, and STL of C++ more.

Genetic Algorithm

Looking build-order planning (macromanagement) as a search problem that is with a number of unstructured environment data, I thought it could be solved by genetic algorithm with Monte Carlo tree search. Figure 3 is the core idea in Markov chain. Each state is a combination of

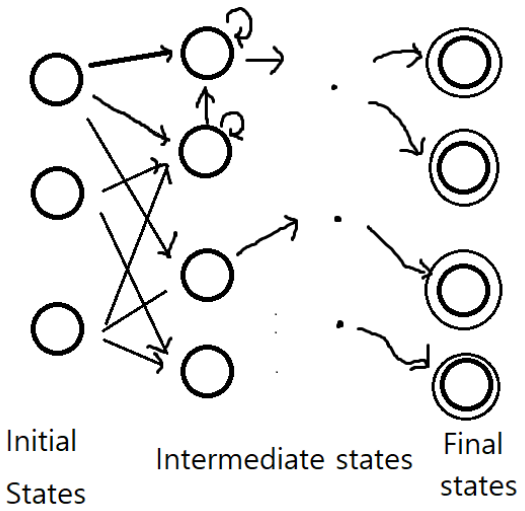

Figure 3: Strategy planning structure with Markov chain

macromanagement and micromanagement. Each state can be a final state or a non-final state. Non-final states have their own transition condition, so when if they reach the condition, they move to next connected state. If more than one state is possible, they will choose greedily. For

every non-final state, if the empirical winning rate is not enough, it can generate a new state either final or non-final, or make a new connection to existing states. Because in the process of generating a new state each refers itself and its neighbors, I named it genetic algorithm.

3.6 2017/2018 SSCAIT

Ranked (27th/78 (Winning rate: 63.64%) [10] and 4th in student division. The same version as AIIDE 2016.

3.7 Reviewing the competitions

Through these competitions, I felt that I need to learn more learning skills to implement better AI. When the first time I saw someone using LSTM in decision making, I was really shocked. In 2017, I concentrated on optimizing the data structure and the network; and tried to make evolving bot. Evolving means better in winning rate, and more variable in its strategies.

4 2017 Naver x UNIST Undergraduate Poster Award

(Bronze Prized) It was a poster award for UNIST undergraduate students researching or developing on computer topics. As an extension to my previous jobs on StarCraft, this time I applied machine learning onto it. I thought machine learning will perform better than the genetic algorithm I tried before in terms of convergence time and accuracy. I found a paper that DNNs can imitate build-order strategies given replays as data ([1]). Inspired by the paper, I tried 'Reinforcement Learning in StarCraft in Real-Time using Deep Learning'. The difference between my trial and the paper is the paper learns from replay data, and my trial learns from real-time data.

4.1 Introduction

The Real-Time Strategy game StarCraft has proven to be a challenging environment for artificial intelligence techniques, and as a result, current state-of-the-art solutions consist of numerous hand-crafted modules. One of the challenges is decision-making. It focuses on deciding build order, i.e., the order that the player produces materials such as units, buildings, and upgrades, which is a huge part of strategical decision over the game.

Goal

- Find best option in real-time: at which given backgrounds such as enemy units, and my units. It implies modeling opponent, resource management, and decision making under uncertainty.
- Build its strategy: In the end, starting from nothing, it learns how to win, and find its build order as a strategy by itself with reinforcement learning.

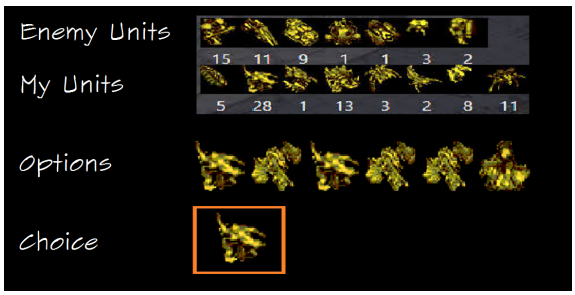

Figure 4: Goal: find out the best choice under the environment

[10] https://sscaitournament.com/index.php?action=2017

4.2 Approach

Dataset

During the games, all decision-makings regarding build order are recorded with back grounds: the numbers of my units, my upgrades, observed enemy units, and my resources.

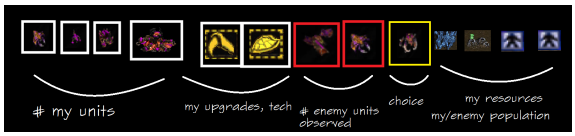

Figure 5: How the datasets are composed

Neural Network Architecture

Datasets become inputs (input layers). The input layers are connected to a hidden single layer with 100 RNN neurons. The hidden layer is connected to two output layers(win, and lose). The output layers are softmaxed.

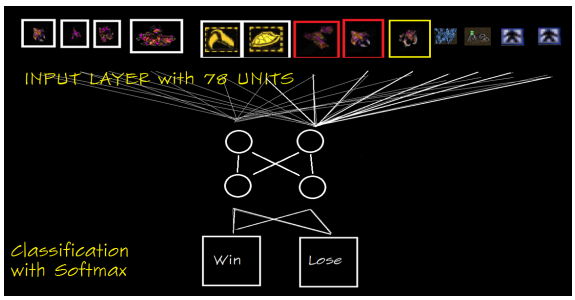

Figure 6: Neural network architecture

Reinforcement Learning

- Two AI modules fight against each other, which have the same structured network, but different initialization.
- After every game ends, it learns with the data of the game for 100 500 times.
- There are stalemates when their build orders converge to the same, even though they are initialized differently. If so, one is loaded with neuron data of 100 games before.

Build Order Selection

At every decision making, it checks winning-rate-prediction of the network with possible choices, and choose one with the highest winning-rate(Greedy).

4.3 Results

Victory

As a result of choosing the best option in real-time, it achieved increasing winning rate against basic AI, and 100% winning rate in the end. For every 100 games of learning, it played against basic AI for 100 times without learning.

Strategy Found

At the beginning, it behaved very weird in point of view. It was inefficient, and unnatural. After some number of games, there is a highly likely used early game sharp strategy (a.k.a. 4drone rush).

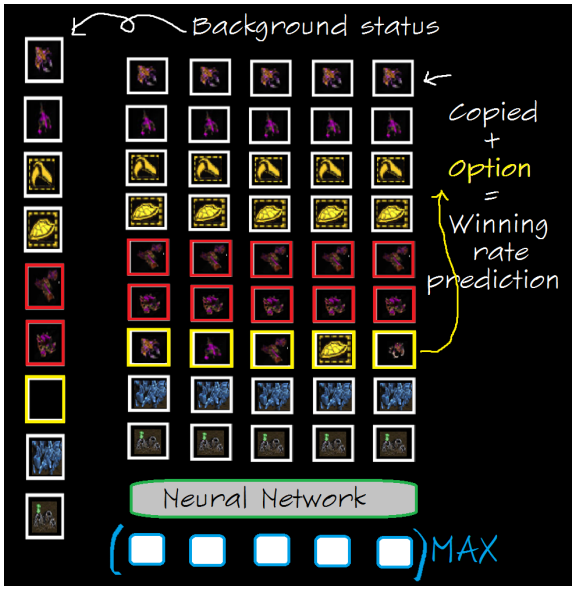

Figure 7: Build order selection through prediction

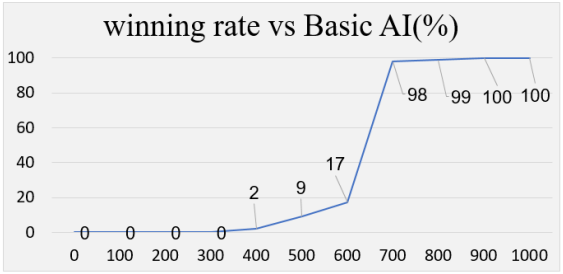

Figure 8: Winning rate change over the number of games

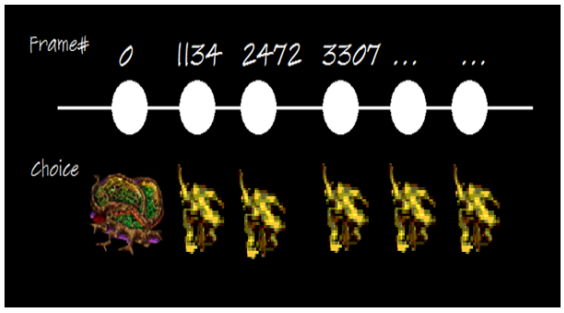

Figure 9: strategy found: 4drones strategy

References

[1] Justesen, N. and Risi, S., 2017, August. Learning macromanagement in starcraft from replays using deep learning. In Computational Intelligence and Games (CIG), 2017 IEEE Conference on (pp. 162-169). IEEE.

[2] Churchill, D., 2017. The Current State of StarCraft AI Competitions and Bots.

[3] Churchill, D., Preuss, M., Richoux, F., Synnaeve, G., Uriarte, A., Ontannón, S. and Čertický, M., 2016. Starcraft bots and competitions. Encyclopedia of Computer Graphics and Games, pp.1-18.

[4] Wender, S. and Watson, I., 2012, September. Applying reinforcement learning to small scale combat in the real-time strategy game StarCraft: Broodwar. In Computational Intelligence and Games (CIG), 2012 IEEE Conference on (pp. 402-408). IEEE

[5] Shantia, A., Begue, E. and Wiering, M., 2011, July. Connectionist reinforcement learning for intelligent unit micro management in starcraft. In Neural Networks (IJCNN), The 2011 International Joint Conference on (pp. 1794-1801). IEEE.