Computer Science Department
CSCI 491 Undergraduate Research
Technical Report – Fall 2019

StarCraft AI

Dylan Atteberry, Tyler Barger, Matthew Carter, Chris Lokken, Daniel Peterson

Motivation

The abundance of Starcraft AI that already exist use many varying strategies to control gameplay. Currently the best bot, produced by the AlphaStar team, can even beat professional players. Regardless these AI still have trouble reacting to unobserved situations, and much of AlphaStar's success is attributed to their massive computational resources. Our implementation aims to improve the AI's ability to reason about and react to situations that it has never seen before. We hope to create an AI that can play at a high level regardless of training limitations by improving its ability to compare unknown situations to those that are known and react appropriately. To accomplish this we will use a system comprised of both AI planning and machine learning components that work together to learn by training and react to situations that could not be observed in training.

Background

Key Terms

For these sources to be truly analyzed one must understand the key concepts presented in them. The first will be reinforcement learning, which is a type of AI training where the program, "gets either rewards or penalties for the actions it performs" (7). AI Planning, or Automated Planning, is the process of creating a plan that, when executed, is a means to accomplish an end goal. In other words, "An AI planning system is charged with generating a plan that is one possible solution to a specified problem" (4).

Additionally, Santiago et. al. is a great overview of real time strategy playing AI, including AI designs and approaches written in 2013 (8). The section about the state of starcraft bots was particularly influential when designing our own approach and representing it (8). Also, liquipedia, a fan written encyclopedia is an excellent source of information about Starcraft: Brood War both for mechanics and learning about the game, as well as details like unit stats.

Related Works

The AlphaStar AI is one of the most efficient Starcraft II AI and was able to defeat a professional player 5-0 in a match. To accomplish this, they took a unique approach to training that allowed their AI to learn incredibly fast. First, they pitted the initial model against the Blizzard AI. After, the AI from the match created a child that used data from the match to learn. Multiple iterations went by until they started matches between the successor AI they had created. Every match yielded a new child created with the intention to learn from mistakes and information presented by its predecessors' game. Some of these children would simply iterate on past strategies, while others would create an entirely new build-order and playstyle (1). This training method sped up and improved the ability to create a model that could efficiently play Starcraft at a high level.

Replay-based strategy prediction and build adaptation focuses on analyzing publicly available replays in order to predict the player's strategy and adapt the bot's build order towards a favorable build matchup. Features are extracted from game replay data using custom software that considers the "Fog-of-War" seen when playing a live game. Once the relevant features are extracted from the replays, both machine learning and decision tree learning are implemented to predict the player's strategy. Using the findings of other research as the base, machine learning algorithms are implemented for early game strategy prediction. Using time dependent features, a "feature-expanded decision tree" (2) is used to easily interpret the choices of human experts into a build order that the system can follow. Then, based on the strategy prediction and win ratios of builds, a final decision is made to determine if the current build order should be changed or not.

The paper focuses on the performance from "connectionist reinforcement learning techniques" and if they can produce an AI for controlling units on a battlefield in StarCraft. They compare many different techniques for training, learning, and rewarding neural networks using reinforcement learning. The paper first compares two types of learning methods using the Sarsa method – Online Sarsa and Neural-Fitted Sarsa. Online Sarsa updates the neural networks after each round while Neural-Fitted Sarsa updates the neural networks every 50 rounds. The next comparison done was between two different learning techniques – incremental learning vs non-incremental learning. Incremental learning refers to training and learning on a smaller problem set to build a base of knowledge before training and learning on the full problem set. Non-incremental learning is the opposite where all training and learning is done on the full problem set. Their experiments found that both methods are efficient for learning small scale scenarios such as 3 vs 3 but "had significant problems" when trying to learn larger scenarios such as 6 vs 6 (12).

One system uses a 2-level automated planning approach to create and accomplish goals when playing StarCraft. The first level, the high level, focuses on macromanagement, searches, and creating soft-goals for the second level. The second level, the low level, focuses on micromanagement and plan generation for accomplishing the soft-goals. The high level uses temporal planning, "a richer version of PDDL" (7), where PDDL is the Planning Domain Descriptive Language. This allows for the output of parallel plans. The low level uses classical planning to achieve the soft-goals. Each goal must have a sequence of detailed actions for the low level to follow (e.g. every action necessary to build a new building). And the game is played through the analyzation of the map, creation of goals to "win" the game, and the processes used to complete the goals.

Research done by Ethan Dereszynski and his team focused on the different models used for learning in RTS games. The models discussed are handmade classifications of players, hierarchical, and a lattice structure. This team decided to use a lattice structure to train their model where nodes in the lattice, "Correspond to counts of different units, buildings, and research technologies" (3). They mentioned the main draw-back of previous iterations of this strategy is that time is not considered. Dereszynski and his team focused on adding time to the structure of the lattice. By the end of their research they were able to transition a player's state according to a set of transitional probabilities at regular time-intervals (3). This method of training was effective in making models better able to predict opponents, identify common strategies, inferring what state of a strategy a player would be in and identifying unusual strategies (3).

Charles Madeira and his team discuss the viability of machine learning for large-scale strategy games, whether it could lead to a higher performing AI, and the amount of time it might take to accomplish (6). They conclude that it is possible to create an AI that can successfully play and conquer a strategy game. However, their model learned in stages, as it would have been extremely complex to learn the best strategies for the entire hierarchy. Being able to make decisions in situations that have never been seen before relies on their use of parameterized function approximators to generalize between similar situations and actions (6). This allows their model to analyze unknown actions and respond in a way that would correspond with the most similar known action. The AI that this team made was found to outperform the commercial AI by far proving that it is superior to conventional models.

Perez-Liebana et. al. run a competition between AI designed for "general video game" playing (10). General video game playing is proposed as one way to evaluate general intelligence by Schaul (13). However, Perez-Liebana et. al. Focused on AI ability to succeed at playing games. In the conclusion they note that despite video game time constraints, approaches using something called a "Monte Carlo Tree Search" are still typically the leading contenders in the competitions for general video game playing like in general game playing competitions which may include physical games like chess (10).

Researches Robertson and Watson demonstrate a new process for extracting game data from Starcraft. Emphasizing the importance of a proper dataset in the process of machine learning and data mining in RTS games, the authors show the possibility of extracting data through not only the replay files generated by the game, but also through simulated replays generated in the game's engine. Their research also details an effective structure to store this data, making it accessible with a well-known query language (11).

Introducing and explaining the problem of unforeseen game situations in the context of game AI, authors Weber, Mateas, and Jhala detail their implementation of Goal-Driven Autonomy, intending to create agents capable of responding to unexpected circumstances. Beginning with a "Discrepancy Detector", which can deduce what kind of unexpected event has occurred and why using an "Explanation Generator", their AI can make an appropriate response. To do so, it takes whatever explanation it is given, and executes its "Goal Formulator" to make a plan, and the "Goal Manager" to handle decision making. While this is obviously a very high-level description of their project, this blueprint of an AI proves to be very effective in Starcraft, as EISBot has a 73% win rate versus built in AI, and ranks higher than 48% of competitive players on the ladder (14).

Research done by Young, Smith, Atkinson, Poyner, and Chothia shows everything that went into making SCAIL, an integrated AI system that plays full games of Starcraft. When thinking about an RTS game such as Starcraft, it logically follows that there are two lines of strategy in the game: Macro and Micro. With this in mind, the researchers moved forward by developing specific sections of SCAIL to deal with both macro and micro decisions in the game. Additionally, SCAIL attempts to make educated decisions on when to attack, retreat, and other imperative choices that rely on observation of game-state. In detailed quantitative analysis of their results, SCAIL has very high win rates against the default AI built into the game, but still struggles against third party AI (8).

"Measuring Intelligence through Games" was interested in how to measure and test general artificial intelligence. The paper argues that video games like Starcraft Broodwar, would be a good environment to test general artificial intelligence (13). Specifically, they suggested having the AI attempt to play games, that it had no prior knowledge of. Then to measure the AI's intelligence as defined in the paper, the AI's ability to achieve goals in that game would be quantified, weighted by the complexity of the game. This would allow researchers to compare AI intelligence and measure the success of an AI.

Wender, and Watson test four different reinforcement learning algorithms effectiveness at learning to micromanage units in combat. They note that the “Q-learning” has the smallest standard deviation by the end, and so is the closest of their tested algorithms to consistently performing optimally (15). Q-learning, and “one-step Sarsa” took longer to start improving making them worse if a constraining factor is time or learning speed (14). This research gives a good indication of what algorithms to use for parts of the AI in charge of managing combat units. Additionally, they provide useful information for when the AI may have a limited training data set, or time to practice.

In research by Justesen et. al, the authors discuss their attempts to use deep learning to learn macromanagement in StarCraft using real game data. Macromanagement in this case is learning general game patterns, including general production, movement, and scouting strategies based on opponent’s tactics. Doing this provides some level of guidance for smaller modules. This module can be installed in other StarCraft AI, enabling them to gain some general game strategy relieving some of the reliance on hard-coded modules (5). The authors did this, installing theirs in the UAlbertaBot.

Approach

Design Criteria

- Protoss and Zerg should both be playable on at least one map.
- Adapt gameplay strategy in response to opponent’s strategy.
- Beat the best player in our research team in a best of three.

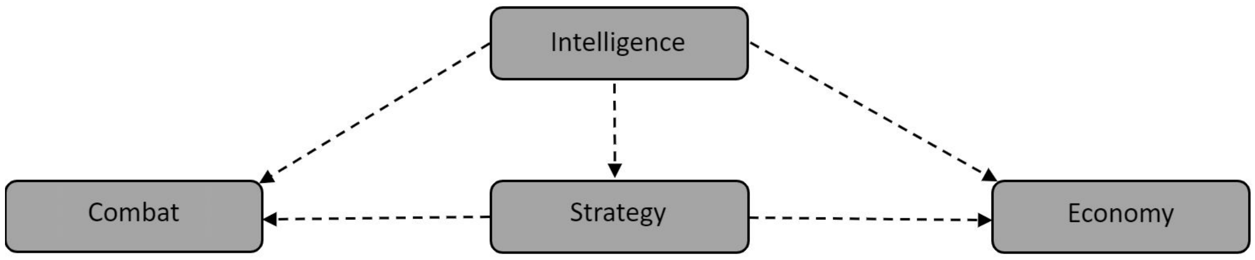

Fig. 1. Diagram depicting the interaction between systems.

Intelligence

One responsibility of the intelligence system is to calculate pertinent information that can be passed to the unit managers. This system stores information such as last known positions of enemy units, known locations of enemy buildings, whether the enemy has detectors, and whether the enemy has cloaked units. Each time this component is updated, every unit manager will also update with the new information.

Another responsibility is to perform path-finding and nearest neighbor calculations that can be used by other components. This involves factors such as enemy unit’s range and avoiding obstacles. This also involves storing information about when an area on the map was visited.

Economy

The responsibilities of the economic system include resource management, as well as resource deployment for non-combat units. Deploying workers to gather resources as well as build new sites. This module takes input from both the intelligence and strategy systems. The intelligence system will provide the necessary information, such as map-layout and location of enemy units, for the economic system to make micro-level decisions such as specific building placement. This allows the economic system to defensively build once sites are picked. The strategy manager will provide necessary information about goals and general strategy to inform the selection of macro-level resource gathering strategies. This allows for general locations to be determined as well as priority of resources.

The separate components of the economic system must be able to cooperatively perform resource gathering and production to provide the combat system with a solid foundation to build from. The economic system will also be a focus during early game, when build order is essential for the bot's long-term success.

Combat

The combat system will be responsible for scouting, controlling the army and those unit's abilities. It will use the combat units to accomplish the goals set by the strategy system, and gather information on the opponent's units to pass along to the intelligence part of the bot. It will also make decisions about how and when to use the combat units, based on the goals set by the strategy system.

The individual components of the combat manager will be in charge of making individual units work together, and scouting, and micromanagement of combat units.

Strategy

One of the roles of the strategy system will be monitoring information passed from the information system to manage discrepancies. These discrepancies are any action or unit that will require an adjustment in the current strategy of the bot. When a discrepancy is detected and a new strategy is formulated the active goals of every component will be updated to represent the actions needed to react to the discrepancy.

The strategy system will also contain agents in charge of overall strategy and gameplay management. These components will be tasked with passing along current goals for each system as well as issuing orders to the unit controllers. Avoiding conflicting orders and overwritten goals is the main purpose for these components.

Evaluation Plan

To test we will have our AI play matches against the built in AI as well as against multiple members of our team and players outside of our team. These tests will be able to observe the systems ability to adapt as well as its skill level.

Summary

By the end of this project we hope to submit our findings to the field of RTS AI research. We hope to find a working balance between AI planning and Machine Learning components that can successfully navigate games with less training time than other models. While working on this system we hope to learn how AI planning and Machine Learning methods function, particularly how they can function together to improve a model. We will also work to sufficiently document our implementation so that future groups can easily understand and expand on the data we gather. This could potentially lead to more efficient AI that would be able to accomplish satisfying results with less training.

References

1) AlphaStar Team. (2019, January 24). AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. Retrieved October 16, 2019, from https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii.
2) Cho, H. C., Kim, K. J., & Cho, S. B. (2013, August). Replay-based strategy prediction and build order adaptation for StarCraft AI bots. In 2013 IEEE Conference on Computational Intelligence in Games (CIG) (pp. 1-7). IEEE.
3) Dereszynski, E., Hostetler, J., Fern, A., Dietterich, T., Hoang, T., & Udarbe, M. (2011). . In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Retrieved from
4) Hendler, J. A., Tate, A., & Drummond, M. (1990). AI planning: Systems and techniques. AI magazine, 11(2), 61-61.
5) Justesen, Niels and Sebastian Risi. "Learning macromanagement in starcraft from replays using deep learning." 2017 IEEE Conference on Computational Intelligence and Games (CIG) (2017): 162-169.
6) Madeira, C. A., Corruble, V., & Ramalho, G. (2006, June). Designing a Reinforcement Learning-based Adaptive AI for Large-Scale Strategy Games. In *AIIDE* (pp. 121-123), from https://www.aaai.org/Papers/AIIDE/2006/AIIDE06-026.pdf.
7) Martınez, M., & Luis, N. Goal-Reasoning in StarCraft: Brood War through Multilevel Planning. Retrieved October 18, 2019, from http://hallsi.ugr.es/temp/CAEPIA2018/web/docs/CAEPIA2018_paper_171.pdf
8) Ontanon, Santiago, et al. "A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft." *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 4, 2013, pp. 293–311.
9) Osiński, B. (2019, July 5). What is reinforcement learning? The complete guide. Retrieved October 25, 2019, from https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/.
10) Perez-Liebana, D, et al. "The 2014 General Video Game Playing Competition." *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, 2016, pp. 229–243.
11) Robertson, G., and Watson, I. "An Improved Dataset and Extraction Process for Starcraft AI." *Florida Artificial Intelligence Research Society Conference,* pp. 255-260. Retrieved October 16, 2019 from
12) Shantia, A., Begue, E., & Wiering, M. (2011, July). Connectionist reinforcement learning for intelligent unit micro management in starcraft. In The 2011 International Joint Conference on Neural Networks (pp. 1794-1801). IEEE.
13) T. Schaul. "Measuring Intelligence through Games." *The Computing Research Repository.* (2011): 1-19. Web. https://arxiv.org/abs/1109.1314
14) Weber, B. G., Mateas, M., and Jhala, A. "Applying Goal-Driven Autonomy to StarCraft." *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment,* pp. 101-106. Retrieved October 16, 2019 from https://www.aaai.org/ocs/index.php/AIIDE/AIIDE10/paper/view/2142/2551

15) Wender, S, and I Watson. “Applying Reinforcement Learning to Small Scale Combat in the Real-Time Strategy Game StarCraft:Broodwar.” *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, 2012, pp. 402–408.
16) Young, J., Smith, F., Atkinson, C., Poyner, K., Chothia, T. “SCAIL: An integrated Starcraft AI System.” *2012 IEEE Conference on Computational Intelligence and Games (CIG).* Institute of Electrical and Electronics Engineers, December 6, 2012.