How Effective can an Adaptive AI Built With Predefined Expert Strategies be Against Competition Grade AI?

James Hellman
Falmouth Games Academy
UK, Falmouth
Email: jh182233@falmouth.ac.uk

***Abstract*—Effective macro-management (the ability to create armies and expand bases), is essential to obtaining victory in Real-Time Strategy (RTS), in the research community many Artificial Intelligence's (AI's) have been created to handle this. One method is to use a design approach to create what is known as build orders, many of these build orders take from expert strategies used by real people in high ranking tournaments. Build orders can be ridged during games leaving little room for adaptation to the opponents strategy. In this paper a collection of build orders will be used to create an AI capable of interchanging these build orders to effectively counter several strategies. An assumption is made here that the AI will only be effective in the early stages of the game and will be outmanoeuvred in late-game stages. The effectiveness of this AI will be its time survived, and winning state.**

I. Introduction

ARTIFICIAL Intelligence can be defined as anything that gives the illusion of intelligence to an appropriate level [1]. In games AI has been used in both single and multi-player environments to help create a more immersive, challenging and fun experience. One such area which AI is prominent is in the RTS genre and since the call for more research to be made for AI in Real-Time Strategy (RTS) games by Michael Buro in 2004 [2], research in this area has exploded, with hundreds of papers being written [3]. This has given rise to the creation of many AI's in RTS games from AI's that are built with pre-defined build orders [4] to deep Neural Networks [5] that can learn from game-play, which will be covered in more detail later on in the paper.

RTS is a great test bed in AI research for its complex systems, involving many areas of interest from planning Imperfect Information. [6]

In RTS, strategy selection is perhaps the most important choice any player or AI can make, as this will dictate the actions and reactions which they take during playtime. Though a human player can be proficient at choosing their strategy, by simply scouting the map, finding the enemy and seeing what they are building. The human player can then counter accordingly, and if they countered incorrectly the human player can simply change their strategy to accommodate. Creating an AI to do the same though can be a huge and complex task [7][8][9], one way to achieve this result without a huge effort is to create a library of expert strategies, and having the AI select the appropriate one throughout the game. This can be achieved using tools such as Advanced Behaviour Oriented Design Environment (ABODE) and Parallel-rooted Ordered Slip-stack Hierarchical (POSH) reactive plans [10], which will also be covered later in the paper. These tools allow for an iterative design approach for AI's in games and in this paper will be focusing on the macro-management with a particular focus on build orders and the selection of strategies rather than micro-management.

The method many papers use to create their AI is to use replays and train their AI to be able to counter strategies, this can be a slow process, so why not just program expert strategies into the AI and then the only thing the AI has to do is select one and follow it through, with some ability to jump between strategies at key points in-case a counter is detected.

II. StarCraft

StarCraft the most popular RTS game of all time [11] is an RTS game developed by Blizzard Entertainment [12][13], and released in 1998. Later that year StarCraft: Brood War was released and took hold in the e-sports community and is still popular today. StarCraft 2: Wings of Liberty was released much later in 2010, with most of the game mechanics the same other than balance changes, and the user interface (UI) were kept the same just with a visual overhaul. The premise of StarCraft is to gather resources, build a base, and build an army to then use to destroy an enemies base and army, during playtime there are also many upgrades available for these units to give them the edge over an enemy who did not spend the time acquiring them. There are many ways of doing this each player with a different order of building their armies/bases commonly referred to as their "Build Order" [14]. Build orders refer to a players macro-management, whereas in StarCraft Micro-management is a huge part of the game, as those with greater control over individual units can better outmanoeuvre their opponent, and thus defeat them.

III. Related Work

In the StarCraft research community there are many different methods of AI creation. Some focus on micro-management like S. Liu et al [15] that uses a Genetic Algorithm (GA) and others that focus on macro-management looking at the

build order like N. Justesen et al [8]. Many of these research methods are cross depended and utilise more than one method for example, D. Churchill et al [16] created the UAlbertaBot, which was intended to automate both build order planning and unit control. Though this paper is focusing on the planning aspect with an implementation of "either a Bayesian or other prediction algorithm" there are many other ways of creating an effective AI.

A. Datasets

A Dataset can be a collection of any data, for game AI a dataset can consist of thousands of replays with millions of game frames, and player actions[17]. This information can then put together to create a full game-state which allows for machine learning tasks [18]. In AI research, datasets can be used in many approaches of development, one such use is to recreate game-states and evaluate them for prediction in realistic conditions [19].

B. Bayesian Approaches

Bayesian approaches are based on Bayes' Theorem, a calculation of probability or also known as a probabilistic model [20]. In papers by G. Synnaeve et al [21][19] they create an AI that controls units individually, they do this by using uncertainty which instead of asking where a unit might be, it makes rough estimations and acts upon that. Another use for the Bayesian approach is to predict strategies, by creating a probabilistic model that after learning from replays can predict an opponents strategy and adapt accordingly [22]. A major downfalls of Bayesian Approaches is that it can be computationally intense to calculate.

C. Micro-Management

Micro management is a fundamental side of StarCraft game-play and many papers have their own approach to this aspect of RTS [23][15][24][25][19][21]. Many of these approaches us either Genetic Algorithms (GA) or Evolutionary Algorithms (EA) [23][15][24], while others observe replays and apply a Monte-Carlo method to create data for practice use [25]. But most of these methods have one thing in common, they all use a version of machine learning [3].

D. Prediction

On a higher strategic level the prediction of the opponents strategy is an prominent approach used in research [26][22][27][28]. This type of research relies on the use of replays and machine learning to help the AI accurately predict a strategy, these do rely on the quantity and quality of replays used for the learning process[26][22][28]. Another method for prediction is scouting alongside machine learning, this eliminates the need for replay observation and allows for a more real-time prediction [27]. Though this method does still require several games to be played before the AI can begin to have an accurate prediction.

E. Full Game Play

Many papers try to create an AI capable of handling all aspects of an RTS [16][29][30][31]. These AI's tend to take several methods that have been created in other research and combining them to form a new AI [16]. Another use for the full game play AI is to try and create a "Human-Like" AI, which can mimic the play-style of an expert human player though the current AI's are limited in this as players reported that the AI's used unusual unit movements or building placement [32].

F. Neural Networks

Neural Networking are computational models loosely based on the functioning of biological brains [5]. Given an input it computes an output by using a large number of neural units, in StarCraft it can be used to predict strategies or in the case of StarCraft 2 with its new architecture it can be used for full game-play. Using a neural network would be impractical for the purpose of this paper as ti would take many months to train.

G. Planning

Planning in StarCraft usually deals with the build order that the AI will use usually only dealing with macro-management. There are several different ways to use a build order, some will use a static build order that will not change throughout the game [4], and the more popular route is to allow the AI to jump between build orders during play-time, another term is Reactive Planning [7][8][9]. there has been some work on creating the build orders on the fly by finding out that most optimal method of gathering resource and building units [14]. Planning is perhaps the most optimal approach to creating an AI as there is little real-time calculations to make. Through the use of POSH tools [10], you can iteratively design AI prototypes and deploy quickly [4]. Though many other papers focus on their micro-management abilities they lack the greater control of the game, the ability to macro-management. This lack of large scale control is usually due to the heavy computational requirements of controlling each individual component of the game. This is where

H. Competition Grade AI

In the StarCraft AI community there are many AI's that have been created to compete against each other, and in this paper a competition grade AI is defined as an AI that has been entered to the AI for Interactive Digital Entertainment conference (AAIDE) StarCraft AI Competition. A yearly competition hosted by David Churchill and sponsored by AIIDE. AI's of this grade include:

- ZZZKBot Winner of the 2017 AIIDE StarCraft AI Competition [33]
- Iron Winner of the 2016 AIIDE StarCraft AI Competition [34]
- UAlbertaBot Winner of the 2013/2011 AIIDE StarCraft AI Competition [35]

- Skynet Winner of the 2012 AIIDE StarCraft AI Competition [36]

These AI's use several strategies along with different factions, and were chosen as they have all previously won the AIIDE StarCraft AI competition [37].

IV. Method and Motivation

This paper will be focusing on the implementation of an AI with pre-built build orders and their counters taken from Liquipedia [38], a website dedicated to StarCraft, on the there they have a collection of strategies that are free to use in any capacity. Building upon these build orders the experiment will also include a method for swapping between the orders at any point, to know when to do this, the AI will scout the map in search of the opponent and compare their current building to its stored build orders and find an appropriate counter. The issue with this method is that in late game the AI will struggle to decide which build order to continue.

Research into AI in any form is valuable as it provides a greater understanding on how AI can work and could contribute to more complex systems by providing a foundation for others to build upon. The hope here is that this form of planning and strategy selection could be applied to future game AI's.

A. Tools

There are several tools that will be using in this experiment, these include The Brood War Application Programming Interface (BWAPI), POSH tools, specifically POSH Sharp which is an interface that uses cSharp instead of C++, and the ABODE editing software which uses POSH plans to create Behaviour Oriented Design's for AI's.

BWAPI [39] is a open source software that creates an interface for a custom AI to use to communicate with the game. BWAPI deliberately does not give access to all the games information [10], limiting the AI to only be able to have information on the enemy if there is no Fog-of-War currently covering them, as well as the size of the map and base locations. This prevents custom AI's from cheating and ensures a fair game, though this could be considered a plus as it means that the developers of these AI's do not need to worry about using information that could cause their AI to cheat. Though this does mean hat all the AI's must work in an imperfect environment which forces the AI to have to scout for information.

The POSH plan in the ABODE Environment as seen in Figure 1, is a visual planning tool that allows for hierarchy of actions with associated triggers.

```
1  [ExecutableAction("SelectExtractorLocation")]
2  public bool SelectExtractorLocation()
3  {
4      // enough resources available?
5      if (!CanMorphUnit(bwapi.
           UnitTypes_Zerg_Extractor) || !Interface
           ().baseLocations.ContainsKey((int)
           Interface().currentBuildSite))
6      return false;
7
```

Fig. 1: POSH plan for the Three Hatch Hydra build plan inside the ABODE editor.

```
8      TilePosition buildPosition = Interface().
           baseLocations[(int)Interface().
           currentBuildSite];
9      // are there any geysers available/visible?
10     IEnumerable<Unit> geysers = Interface()
11         .GetGeysers().Where(geyser => geyser.
               getResources() > 0);
12     if (geysers.Count() < 1)
13         return false;
14
15
16     // sort by closest path for ground units
           from selected build base
17     TilePosition closest = geysers
18         .OrderBy(geyser => geyser.getDistance(
               new Position(buildPosition)))
19         .First().getTilePosition();
20
21     // if there is a close geyers we are done
22     if (closest is TilePosition)
23     {
24         this.buildLocation = closest;
25         builder = Interface().GetBuilder(
               buildPosition);
26         //move(new Position(closest), builder);
27         // if (builder.getDistance(new Position(
               closest)) < DELTADISTANCE)
28         //     return true;
29         return true;
30     }
31     return false;
32 }
```

Fig. 2: cSharp behaviour code snippet for selecting an extractor location for the Zerg.

B. Hypothesis

To create a StarCraft AI you need to first have in mind what it needs to do, in the case of this paper the AI needs to

C. Metrics

In this paper the StarCraft AI will be measured on its success using two factors in order of importance:

Time Survived
End game Condition

Through these factors the effectiveness of the AI can be determined, as the average time of a StarCraft game can

last between 10-20 minutes if the AI survives pas the upper limit of this time or wins the game it will support a greater effectiveness. Though if the AI does not survive past the lower limit of that time or looses it will negate the effectiveness. To begin with the AI will be pitted against the inbuilt AI as a test-bed, if it should have a high effectiveness it will then be put against an open source competition grade AI, and measured from there.

REFERENCES

[1] D. M. Bourg and G. Seemann, *AI for Game Developers*. O'Reilly Media, Inc., 2004.

[2] M. Buro, "Call for ai research in rts games," in *In Proceedings of the AAAI Workshop on AI in Games*. AAAI Press, 2004, pp. 139–141.

[3] S. Ontan, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A survey of real-time strategy game ai research and competition in starcraft," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, pp. 293–311, Dec 2013.

[4] S. Gaudl, S. Davies, and J. Bryson, "Behaviour oriented design for real-time-strategy games: An approach on iterative development for sc starcraft ai," *Foundations of Digital Games (FDG)*, pp. 198–205, Jun 2013.

[5] N. Justesen and S. Risi, "Learning macromanagement in starcraft from replays using deep learning," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug 2017, pp. 162–169.

[6] D. Churchill, M. Preuss, F. Richoux, G. Synnaeve, A. Uriarte, S. Ontanon, and M. Certicky, "Starcraft bots and competitions," in *Encyclopedia of Computer Graphics and Games*, 01 2016, pp. 1–18.

[7] M. Preuss, D. Kozakowski, J. Hagelbck, and H. Trautmann, "Reactive strategy choice in starcraft by means of fuzzy control," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, Aug 2013, pp. 1–8.

[8] N. Justesen and S. Risi, "Continual online evolutionary planning for in-game build order adaptation in starcraft," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 187–194.

[9] B. G. Weber, M. Mateas, and A. Jhala, "Applying goal-driven autonomy to starcraft." in *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI, 2010.

[10] C. Brom, J. Gemrot, M. B ida, O. Burkert, S. J. Partington, and J. Bryson, "Posh tools for game agent development by students and non-programmers," in *9th International Conference on Computer Games: AI, Animation, Mobile, Education and Serious Games*, Jan 2006, pp. 126 – 135.

[11] D. Churchill and M. Certicky, "The current state of starcraft ai competitions and bots," in *AIIDE 2017 Workshop on Artificial Intelligence for Strategy Games*, 2017.

[12] Blizzard Entertainment. Starcraft. [Online]. Available: http://eu.blizzard.com/en-gb/games/sc/ [Accessed 03-11-2017]

[13] ——. 10 years of starcraft. [Online]. Available: https://web.archive.org/web/20080402134120/http://www.blizzard.com/us/press/10-years-starcraft.html [Accessed 16-11-2017]

[14] D. Churchill and M. Buro, "Build order optimization in starcraft," *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2011.

[15] S. Liu, S. J. Louis, and C. Ballinger, "Evolving effective micro behaviors in rts game," in *2014 IEEE Conference on Computational Intelligence and Games*, Aug 2014, pp. 1–8.

[16] D. Churchill and M. Buro, "Incorporating search algorithms into rts game agents," *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.

[17] G. Synnaeve and P. Bessire, "A dataset for starcraft ai and an example of armies clustering," *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2012.

[18] L. Zeming, G. Jonas, l. K. Vasi, and S. Gabriel, "Stardata: A starcraft ai research dataset," *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2017.

[19] G. Synnaeve and P. Bessire, "Special tactics: A bayesian approach to tactical decision-making," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, Sept 2012, pp. 409–416.

[20] K. B. Korb and A. E. Nicholson, *Bayesian Artificial Intelligence, Second Edition*, 2nd ed. CRC Press, Inc., 2010.

[21] G. Synnaeve and P. Bessire, "A bayesian model for rts units control applied to starcraft," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, Aug 2011, pp. 190–196.

[22] ——, "A bayesian model for opening prediction in rts games with application to starcraft," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, Aug 2011, pp. 281–288.

[23] I. Zelinka and L. Sikora, "Starcraft: Brood war - strategy powered by the soma swarm algorithm," in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug 2015, pp. 511–516.

[24] A. R. Tavares, H. Azprua, and L. Chaimowicz, "Evolving swarm intelligence for task allocation in a real time strategy game," in *2014 Brazilian Symposium on Computer Games and Digital Entertainment*, Nov 2014, pp. 99–108.

[25] J. Young and N. Hawes, "Learning micro-management skills in rts games by imitating experts," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI, 2014.

[26] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," in *2009 IEEE Symposium on Computational Intelligence and Games*, Sept 2009, pp. 140–147.

[27] H. Park, H.-C. Cho, K. Lee, and K.-J. Kim, "Prediction of early stage opponents strategy for starcraft ai using scouting and machine learning," in *Proceedings of the Workshop at SIGGRAPH Asia*. ACM, 2012, pp. 7–12.

[28] H. C. Cho, K. J. Kim, and S. B. Cho, "Replay-based strategy prediction and build order adaptation for starcraft ai bots," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, Aug 2013, pp. 1–7.

[29] M. Stanescu, N. Barriga, and M. Buro, "Hierarchical adversarial search applied to real-time strategy games," *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2014.

[30] B. Weber, M. Mateas, and A. Jhala, "Building human-level ai for real-time strategy games," *Advances in Cognitive Systems: Papers from the 2011 AAAI Fall Symposium*, 2011.

[31] J. Young, F. Smith, C. Atkinson, K. Poyner, and T. Chothia, "Scail: An integrated starcraft ai system," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, Sept 2012, pp. 438–445.

[32] M.-J. Kim, K.-J. Kim, S. Kim, and A. K. Dey, "Evaluation of starcraft artificial intelligence competition bots by experienced human players," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2016, pp. 1915–1921.

[33] C. Chris. Zzzkbot. [Online]. Available: https://github.com/chriscoxe/ZZZKBot [Accessed 16-11-2017]

[34] D. Igor. Iron bot. [Online]. Available: http://bwem.sourceforge.net/Iron.html [Accessed 16-11-2017]

[35] C. David. Ualbertabot. [Online]. Available: https://github.com/davechurchill/ualbertabot/wiki [Accessed 16-11-2017]

[36] S. Andrew. Skynet bot. [Online]. Available: https://github.com/tscmoo/skynet [Accessed 16-11-2017]

[37] C. David. Aiide starcraft ai competition official results. [Online]. Available: http://www.cs.mun.ca/ dchurchill/starcraftaicomp/results.shtml [Accessed 16-11-2017]

[38] Liquipedia. Protos strategy. [Online]. Available: http://wiki.teamliquid.net/starcraft/Protoss Strategy [Accessed 03-11-2017]

[39] BWAPI Development Team. bwapi - an api for interacting with StarCraft: Broodwar (1.16.1). [Online]. Available: https://github.com/bwapi/bwapi [Accessed 04-11-2017]