Rock, Paper, StarCraft: Strategy Selection in Real-Time Strategy Games

Anderson Tavares, Hector Azpúrua, Amanda Santos, Luiz Chaimowicz

Laboratory of Multidisciplinary Research in Games
Computer Science Department
Universidade Federal de Minas Gerais
Email: {anderson,hector.azpurua,amandasantos,chaimo}@dcc.ufmg.br

Abstract

The correct choice of strategy is crucial for a successful real-time strategy (RTS) game player. Generally speaking, a strategy determines the sequence of actions the player will take in order to defeat his/her opponents. In this paper we present a systematic study of strategy selection in the popular RTS game StarCraft. We treat the choice of strategy as a game itself and test several strategy selection techniques, including Nash Equilibrium and safe opponent exploitation. We adopt a subset of AIIDE 2015 StarCraft AI tournament bots as the available strategies and our results suggest that it is useful to deviate from Nash Equilibrium to exploit sub-optimal opponents on strategy selection, confirming insights from computer rock-paper-scissors tournaments.

1 Introduction

In games with large state spaces, players often resort to strategies, i.e., sequences of actions that guide their behavior, to help achieving their goals. For example, games like Chess, Go and StarCraft have known opening libraries, which are strategies that help players achieve favorable situations from initial game states. Strategies usually interact with each other. Dedicated players study and develop new strategies that counter older ones, and these new strategies will be studied in the future to be countered as well. Thus, the study and correct selection of strategies is crucial for a player to succeed in a game.

In real-time strategy games, several works deal with strategy construction, which involves developing a sequence of actions that would reach desired situations, as in (Stanescu, Barriga, and Buro 2014; Uriarte and Ontañón 2014), or strategy prediction, which involves comparing opponent behavior with known ones, as in (Weber and Mateas 2009; Synnaeve and Bessière 2011; Stanescu and Čertický 2016). Moreover, most state-of-the-art software-controlled StarCraft players (bots) employ simple mechanisms to strategy selection, ignoring the adversarial nature of this process, i.e. that the opponent is reasoning as well.

In the present work, we perform a systematic study on strategy selection in StarCraft. We model the process of strategy selection as a normal-form game. This adds a layer of abstraction upon StarCraft, which we call the strategy selection metagame. We fill the metagame's payoff matrix with a data-driven approach: strategies play a number of matches between each other and we register their relative performances. We proceed by estimating a Nash Equilibrium in the metagame which specifies a probability distribution over strategies to achieve a theoretically-guaranteed expected performance.

We observe that some strategies interact in a cyclical way in StarCraft, thus we draw some insights from computer rock-paper-scissors tournaments (Billlings 2001). In those, a player usually benefits by deviating from equilibrium to exploit sub-optimal opponents, but must guard itself against exploitation by resorting to equilibrium when its performance drops.

Experiments in this paper are performed with participant bots of AIIDE 2015 StarCraft AI tournament. Each bot is seen as a strategy, thus, in our experiments, strategy selection becomes the incorporation of which behavior (specified by the chosen bot) the player will adopt to play a match. Results show that it is indeed useful to deviate from equilibrium to exploit sub-optimal strategy selection opponents and that a player benefits from adopting safe exploitation techniques (McCracken and Bowling 2004) with guaranteed bounded losses.

This paper is organized as follows: Section 2 reviews related work. Section 3 presents the strategy selection metagame, whereas Section 4 instantiates it upon StarCraft. Section 5 presents and discusses results of experiments consisting of a tournament between different strategy selection techniques. Section 6 presents concluding remarks and opportunities for future study.

2 Related Work

Works on strategic reasoning in real-time strategy games can be divided (non-exhaustively) in strategy prediction, strategy construction and strategy selection itself.

Strategy prediction is concerned with recognizing a player's strategy, classifying it into a set of known strategies or predicting next moves from a player, possibly from partial and noisy observations. This can be done by encoding replay data into feature vectors and applying classification algorithms (Weber and Mateas 2009), using bayesian reasoning (Synnaeve and Bessière 2011) or answer set pro-

gramming, a logic paradigm able to deal with uncertainty and incomplete knowledge (Stanescu and Čertickỳ 2016).

Strategy construction is concerned with constructing a sequence of actions from a given game state, which is related to search and planning. To deal with the huge search space of real-time strategy games, hierarchical or abstract representation of game states can be used in conjunction with adapted versions of classical search algorithms, as in (Stanescu, Barriga, and Buro 2014; Uriarte and Ontañón 2014). Also, techniques based in portfolio (a set of predefined strategies, or scripted behavior) are used either as components for playout simulation (Churchill and Buro 2013) or to generate possible actions for evaluation by higher-level game-tree search algorithms (Churchill and Buro 2015).

Strategy selection is concerned with the choice of a course of action to adopt from a set of predefined strategies. Marcolino et al. (2014) studies this in the context of team formation. Authors demonstrate that, from a pool of stochastic strategies, teams composed of varied strategies can outperform a uniform team made of copies of the best strategy as the size of the action space increases. In the proposed approach, a team of strategies votes for moves in the game of Go and their suggestions are combined. Go allows this consulting stage due to its discrete and synchronous-time nature. As real-time strategy games are dynamic and continuous in time, such approach would be difficult to evaluate.

Regarding strategy selection in real-time strategy games, Preuss et al. (2013) use fuzzy rules to determine strategies' usefulness according to the game state. The most useful strategy is activated and dictate the behavior of a StarCraft bot. A drawback of this approach is the need of expert knowledge to create and adjust the fuzzy rules.

A case-based strategy selection method is studied by Aha, Molineaux, and Ponsen (2005) in Wargus, an open-source Warcraft II clone. Their system learns which strategy to select according to the game state. However, the study assumes that opponent makes choices according to a uniform distribution over strategies, which is unrealistic.

State-of-the art StarCraft bots perform strategy selection, according to the survey of Ontañón et al. (2013): they choose a behavior according to past performance against their opponent. However, their mechanisms lack game-theoretic analysis or performance guarantees because they ignore the opponent's adaptation and strategic reasoning. The survey of Ontañón et al. (2013) also notes that bots interact in a rock-paper-scissors fashion, based on previous AI tournament analysis.

A game-theoretic approach to strategy selection is studied by Sailer, Buro, and Lanctot (2007), where authors also note the rock-paper-scissors interaction among strategies. At certain decision points, playouts among strategies are simulated to fill a normal-form game's payoff matrix. Nash equilibrium is calculated and a strategy is selected accordingly. This is performed online, during a match. In this sense, Sailer, Buro, and Lanctot (2007) go beyond the present work, because here the single decision point is at the game beginning. However, their online approach is possible because their simplified game considers only army de-

ployment[1]. Moreover, the game's forward model is available so that simulation of players' decisions is possible. In the present work we consider StarCraft, a complete real-time strategy game with unavailable forward model[2]. Besides, in the present work, we go beyond Nash Equilibrium by testing various strategy selection techniques, including safe opponent exploitation (see Section 5).

The major novelty of the present work is the focus on strategy selection instead of planning or prediction for real-time strategy games. Moreover, we analyse the interaction among sophisticated strategies (full game-playing agents) in a complex and increasingly popular AI benchmark, analysing the performance of various strategy selection techniques.

3 The Strategy Selection Metagame

We refer to the process of strategy selection as the *strategy selection metagame* because it adds an abstract layer of reasoning to a game. We refer to the game upon which the metagame is built as the *underlying* game. At principle, the strategy selection metagame can be built upon any game where it is possible to identify strategies.

In this paper, we define strategy as a (stochastic) policy, i.e. a mapping from underlying game states to (a probability distribution over) actions. The policy specified by a strategy must be a total function, that is, any valid underlying game state should be mapped to a valid action. This is important to guarantee that a strategy plays the entire game, thus we can abstract from the underlying game mechanics.

3.1 Definition

The strategy selection metagame can be seen as a normal-form game and formally represented by a payoff matrix P where component P_{ij} represents the expected value in the underlying game by adopting strategy i while the opponent adopts strategy j.

Payoff matrix P can be filled according to domain-specific knowledge. In this case, an expert in the underlying game would fill the matrix according to strategies' relative performance. If domain-specific knowledge is not available, but strategies are known, data-driven approaches can be employed to populate the matrix. For instance, match records could be used to identify strategies and register their relative performances. Another data-driven approach would be to execute a number of matches to approximate the expected value of each strategy against each other. In any case, as a convention for zero-sum games, the matrix's diagonal can be filled with zeros, meaning that a strategy draws against itself, or that it wins half the matches if the underlying game cannot end in a draw.

---

[1]This is related to strategic combat decisions such as grouping forces, attacking a base or moving to specific locations.

[2]Although combat can be simulated via SparCraft (Churchill and Buro 2013), other important aspects do not have similar engines available.

3.2 Playing the Metagame

In several games with large state spaces, dominant strategies are usually not known, meaning that, in general, it is possible to defeat any sequence of actions. Moreover, known strategies can interact in a cyclical way, so that a given strategy defeats a second one and is defeated by a third one. This form of rock-paper-scissors interaction may suggest that insights from strategy selection in rock-paper-scissors would be useful for a strategy selection metagame.

The definition of the strategy selection metagame as a normal-form game (Section 3.1) allows us to employ game-theoretic reasoning. For instance, a metagame player can adopt a Nash Equilibrium strategy so that it has theoretical guarantees in its expected payoff over a sequence of matches. Nash Equilibrium can be determined by solving a linear program related to the game (Nisan et al. 2007, Section 1.4.2).

4 The Metagame in StarCraft

4.1 StarCraft

In this paper, the strategy selection metagame is played upon real-time strategy (RTS) game StarCraft, but the concepts are general enough for any adversarial game. StarCraft is increasingly being adopted as a benchmark for artificial intelligence techniques because of its challenging characteristics, which include imperfect information, dynamicity, and a huge state-action space. In RTS games, players usually perform hundreds of actions per minute. The actions are divided in several tasks involving resource gathering, creation of new units, construction of buildings, attacks to the enemy and technology advancements (Weber, Mateas, and Jhala 2011).

StarCraft has three playable races with different characteristics: *Protoss*, which has powerful and expensive units; *Zerg*, which has weak and cheap units and *Terran*, with units of intermediate power and cost. To win a match, a player must destroy all buildings of his opponent.

4.2 Metagame Instantiation

In StarCraft, bots that can play an entire game satisfy our definition of strategy (Section 3), because they depart from the game's initial state and are able to perform valid actions in any situation. Bots usually act differently from each other so that different bots can be considered distinct strategies within this given concept. Thus, in our instantiation of the strategy selection metagame, to choose a strategy means to play a StarCraft match following the policy dictated by the chosen bot. On the ongoing discussion, we use the terms bot and strategy interchangeably.

In our experiments, without loss of generality, the set of available StarCraft strategies is represented by bots that played with *Protoss* race in AIIDE 2015 StarCraft AI Tournament[3]. Thus, the strategy selection metagame's payoff matrix is estimated by counting the number of victories in matches among the tournament bots. AIIDE 2015 tournament data could be used to populate the matrix, but we ran

---

[3]http://webdocs.cs.ualberta.ca/~cdavid/starcraftaicomp/2015/

---

a new tournament with persistent knowledge disabled. In StarCraft AI tournaments, bots use persistent knowledge to accumulate experience and improve performance in future matches. When this happens, bots may change their behavior between matches, defining new policies. This non-stationarity in strategies is out of the scope of this paper. Nevertheless, bots' policies can be stochastic, i.e., they can perform different actions from a given state in different matches, as long as they are stationary, that is, for a given state, the probability distribution over actions remain unchanged.

Table 1 shows the percent of victories in matches among *Protoss* AIIDE 2015 tournament bots. Matches were executed with StarCraft AI Tournament Manager, modified to disable persistent knowledge[4]. Every bot played against every other for 100 matches in Fortress map. We ran a single map to reduce the influence of distinct maps in results. Match outcomes are either victory or defeat. If a match runs until timeout (one hour of gameplay), ties are broken by in-game score.

Eight bots played with *Protoss* in AIIDE 2015 tournament: UAlbertaBot, Ximp, Xelnaga, CruzBot, NUSBot, Aiur, Skynet and SusanooTricks. Among these, UAlbertaBot, Ximp and SusanooTricks are not included in Table 1 because UAlbertaBot and Ximp "dominate" all others and SusanooTricks "is dominated" by all others. The term dominance here means that a bot wins more than 50% matches against another. Dominant bots were removed because otherwise a pure Nash Equilibrium strategy would exist (select the dominant bot in all matches), and dominated bots would never be chosen, which is not interesting.

Bot	Xelnaga	CruzBot	NUSBot	Aiur	Skynet
Xelnaga	-	**26%**	86%	73%	73%
CruzBot	74%	-	80%	67%	**16%**
NUSBot	**14%**	**20%**	-	74%	97%
Aiur	27%	33%	26%	-	79%
Skynet	27%	84%	**3%**	21%	-

Table 1: Win percentage of AIIDE 2015 *Protoss* bots in Fortress map. Cases where bots are dominated by others are highlighted in bold.

There is no dominant pure strategy from the metagame defined from Table 1, since any pure strategy has a best response, which is highlighted in bold. Moreover, strategies interact in a cyclical way. For example, Skynet dominates CruzBot, which dominates Xelnaga, which dominates Skynet.

Table 2 shows the calculated Nash Equilibrium, obtained with Game Theory Explorer (Savani and von Stengel 2015). Before entering Table 1 into Game Theory Explorer, we transformed each percentage of victories into an expected payoff (by adding the product of the victory percentage multiplied by 1 to the defeat percentage multiplied by -1) and filled the payoff's matrix diagonal with zeros.

---

[4]The modified manager is in http://github.com/andertavares/StarcraftAITournamentManager. It is a fork of Dave Churchill's

Strategy	Probability
Xelnaga	41.97%
CruzBot	28.40%
NUSBot	0%
Aiur	0%
Skynet	29.63%
Total	100%
Expected payoff	0

Table 2: Nash Equilibria among selected strategies.

In equilibrium, two strategies have zero probability: NUSBot and Aiur. This is the case because, although they dominate other bots, Xelnaga dominates them *and* their dominated bots. The expected payoff of zero means that the equilibrium probability distribution over strategies is expected to win half the matches.

4.3 Strategy Selection Techniques

The instantiated metagame from Section 4.2 has similarities with the classical game of rock-paper-scissors: strategies interact in a cyclical way and the expected payoff in equilibria is zero. A difference is that actual outcomes are stochastic, since a strategy does not lose every match against its best-response, given the imperfect information, map variations (e.g. starting locations) and the dynamic nature of StarCraft.

Computer rock-paper-scissors tournaments (Billlings 2001) have shown that it is useful to deviate from equilibrium to exploit sub-optimal opponents. In the first tournament, the participant playing the equilibrium strategy placed only 27th among 55 competitors. In general, strong competitors detect patterns in opponent actions and predict their next move, with several enhancements to anticipate second-guessing. Moreover, they adopt the equilibrium strategy as a failsafe, activated when their performance drops by failing to predict opponent moves. Strong competitors are differentiated by how well they exploit weaker ones.

In order to test insights from computer rock-paper-scissors tournaments, we evaluate the following strategy selection techniques in StarCraft (which could be referred to as metagame players):

1. Frequentist: attempts to exploit opponent by selecting the best-response of its most frequent strategy;

2. Reply-last: attempts to exploit opponent by selecting the best-response of its last strategy;

3. Single choice: selects a predefined single strategy, regardless of what the opponent does;

4. Nash: selects a strategy according to Nash Equilibrium, given in Table 2;

5. ϵ-Nash: attempts to exploit opponent with probability ϵ (by playing frequentist) and plays the safe strategy (Nash Equilibrium) with probability $1 - \epsilon$.

6. α-greedy: selects a random strategy (exploration) with probability α, and its most victorious strategy (exploitation) with probability $1 - \alpha$.

Frequentist, reply-last and single choice do not have theoretical guarantees on performance. Frequentist is based on the intuition that a player is likely to repeat its most frequent choice. Reply-last is based on the idea that a player can repeat its last choice, especially if it was victorious. Single choice is the most exploitable technique, since it does not react to opponent choices. Frequentist was also participant of the computer rock-paper-scissors tournament and reply-last had a similar counterpart (Billlings 2001). Single choice is a "dummy" technique put in the tournament to test other techniques' exploiting abilities.

Nash and ϵ-Nash have theoretical guarantees on performance. Nash is an equilibrium strategy for the metagame. It is expected to win 50% of matches regardless of its adversary. In ϵ-Nash, the exploitability (the maximum expected payoff that is lost by deviating from equilibrium) is theoretically bounded by ϵ. Thus it is an ϵ-safe strategy (McCracken and Bowling 2004). In the worst case, ϵ-Nash loses all matches where it tries to exploit its opponent, which is attempted only with probability ϵ.

Technique α-greedy is an action selection method designed to balance exploration and exploitation in multi-armed bandits, which is a problem of action selection with stochastic rewards (Sutton and Barto 1998, Chapter 2)[5]. α-greedy performs well when the process generating its rewards is well-behaved (e.g. stationary). In StarCraft, the reward generation process for α-greedy is an adversary, which, except for single choice, is not well-behaved.

Strategies available for the techniques to choose are the bots in Table 2. Techniques that play a best-response do so by querying Table 1. For example, if opponent selected Xelnaga in previous match, reply-last technique would choose CruzBot for the next match.

5 Experiments

5.1 Experimental Setup

Before evaluating the strategy selection techniques described in Section 4.3, we built a pool with records of 1000 StarCraft matches between each pair of AIIDE 2015 bots from Table 2, which are the available choices for the strategy selection techniques. When two techniques face each other and select their strategies, a previously recorded match result is selected from the pool, victory is awarded to the technique that has chosen the winning bot and the match is removed from the pool. If two techniques select the same bot, victory is randomly awarded to any technique. This process is repeated for the determined number of matches between the two techniques. For a new contest between strategy selection techniques, the pool is restored. This methodology was adopted to speed up contests between strategy selection techniques by avoiding the execution of a new StarCraft

software.

---

[5] α-greedy is usually referred to as ϵ-greedy in multi-armed bandit literature, but to avoid confusion with ϵ-Nash, we adopt α as the exploration parameter.

match every time techniques face each other. The pool is generated with bots' persistent knowledge disabled, so that bots' relative performance remain stationary, i.e., Table 1 would not change significantly if bots play more matches.

In order to evaluate the strategy selection techniques, we ran a round-robin tournament between them. In the tournament, every technique played 1000 matches against every other[6]. Before each match, techniques have access to previous match history and to the metagame's payoff matrix (constructed via Table 1) in order to select a strategy for the next match.

We configured single choice technique to pick Xelnaga every time, because it was the best-performing pure strategy from Table 1. Parameters α and ϵ were configured to 0.2 and 0.4 respectively, because, in prior experiments, they achieved a good trade-off between exploration vs. exploitation (for α) and exploitation vs. exploitability (for ϵ).

5.2 Results

Table 3 shows the results of contests between pairs of strategy selection techniques. Results are averaged over 30 repetitions. Figure 1 shows the average performance of techniques against all adversaries (last column of Table 3).
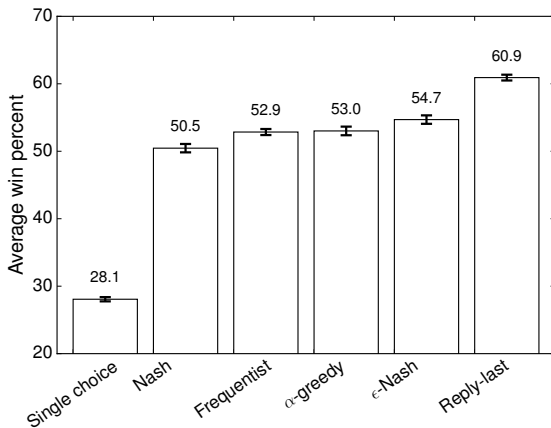

Figure 1: Average percent of victories of techniques against all adversaries. Error bars are the 95% confidence intervals.

To verify statistical significance of differences among averages, we performed one-way ANOVA and Tukey's HSD test with significance level of 0.05. These indicated that, except between α-greedy and frequentist, average performance is significantly different between all pairs of techniques.

Reply-last was the most successful technique in this tournament. It won most matches against all but ϵ-Nash and Nash, whose performances are theoretically guaranteed. Reply-last plays well against frequentist: it wins a sequence of matches until its current choice becomes the most frequent, then frequentist responds and wins one match. Reply-

---

[6]This is the same number of matches of computer rock-paper-scissor tournaments (Billlings 2001).

last responds right away and starts winning another sequence of matches and this cycle repeats. In fact, reply-last could be easily second-guessed by an opponent (by choosing in next match a response to the best-response of current choice), but no technique was programmed to do so.

On average (Fig. 1), Nash reached the expected payoff of zero by winning roughly 50% of its matches. Neither it exploits opponents nor it is exploited. Against specific opponents such as frequentist and single choice (Table 3), there were small deviations from its expected payoff. These differences can be explained because Nash Equilibrium is estimated from 100 previous matches between strategies (Table 1).

Reply-last and frequentist achieved identical performance against single choice, because its last choice is also the most frequent. In this case, deviating from equilibrium pays off. For example ϵ-Nash's performance against single choice is superior to Nash's. Besides, guarding itself against exploitation is useful. For example, reply-last consistently defeats frequentist, whereas it fails to do so against ϵ-Nash which can be seen as an enhanced version of frequentist, protected against exploitation. This illustrates that ϵ-Nash successfully performed safe opponent exploitation.

Technique α-greedy successfully learned how to exploit single choice, but has failed to do so against frequentist, reply-last and ϵ-Nash, because they aren't well-behaved reward generation process for a multi-armed bandit (as discussed in Section 4.3). Even so, it was not dominated by any adversary except reply-last. It performed similarly to frequentist (Tukey's HSD revealed no significant differences between their average performances), because their behavior is also similar: the most victorious choice can also be the one that counters opponent's most frequent choice.

5.3 Discussion

Insights from computer rock-paper-scissors tournaments were useful to strategy selection in StarCraft: a player benefits by exploiting sub-optimal opponents as well as by guarding itself against exploitation. This is remarkably done by ϵ-Nash. Moreover, for this specific tournament, if ϵ-Nash adopted reply-last as its exploitive strategy, its results could have improved, especially against frequentist.

Results in this paper are coherent with those of McCracken and Bowling (2004), where previously weak rock-paper-scissors bots performed better when enhanced with safe exploitation techniques. Here, ϵ-Nash which is an enhanced version of frequentist, performed better.

A limitation of our approach is that the game-theoretical guarantees (expected payoff and bounded exploitation) are valid only if players select strategies within the set used to calculate the equilibrium. In several games, including StarCraft, the number of possible strategies is infinite so that these guarantees seem little reassuring at first.

The mentioned limitation could be tackled by adopting a set of strategies that is general enough for the game. This way, opponent behavior can be observed and classified according to its similarity with known strategies. In StarCraft, this could be done by adapting some opening prediction or opponent modeling methods, such as (Weber and Mateas

Technique	Reply-last	ϵ-Nash	α-greedy	Frequentist	Nash	Single choice	**Average**
Reply-last	-	50.2	62.5	63.0	48.1	80.8	60.9
ϵ-Nash	49.8	-	49.8	53.6	51.3	69.1	54.7
α-greedy	37.5	50.2	-	52.5	51.3	73.5	53.0
Frequentist	37.0	46.4	47.5	-	52.5	80.8	52.9
Nash	51.9	48.7	48.7	47.5	-	55.5	50.5
Single choice	19.2	30.9	26.5	19.2	44.5	-	28.1

Table 3: Percent of victories between pairs of strategy selection techniques. Lines are sorted according to average performance against all opponents, shown in last column.

2009; Synnaeve and Bessière 2011; Stanescu and Čertický 2016), to predict a complete strategy.

Our model assumes that a strategy is followed until the end, but a player can naturally switch its strategy, responding to a real game situation. To tackle this, we could extend our approach by associating the metagame with a context related to the game state. This "contextual metagame" approach would be similar to (Sailer, Buro, and Lanctot 2007), where a metagame is solved online (during gameplay) to decide player's next strategy. Although such online approach is currently infeasible in StarCraft (see Section 2), offline simulations could be performed and state approximation techniques (Sutton and Barto 1998, Chapter 8) could be used to generalize from offline simulated states.

In our view of strategy selection as a multi-armed bandit problem via α-greedy we abstract the adversary, treating it as the bandit's reward generation process. This is not an issue when such a process is well-behaved, which is not the general case of an adversary in StarCraft. Thus, other sampling techniques based on the same assumptions as α-greedy are unlikely to perform significantly better. Instead, modeling strategy selection in StarCraft as an adversarial multi-armed bandit problem[7] (Auer et al. 1995) has more potential of success.

As we note on Section 2, StarCraft bots may change their behavior according to experience gathered against opponents (Ontañón et al. 2013). This could violate our assumption that bots are stationary strategies, because in this case they change with experience. However, without persistent knowledge, bots do not accumulate experience, thus they determine stationary strategies. Even if bots randomly select a behavior at match beginning, they would still be valid for the purposes of our experiments, because expected relative performance against opponents does not change with experience, that is, metagame's payoff matrix remain stationary.

To apply our findings in practice, that is, to employ a technique such as ϵ-Nash to choose strategies in situations such as a StarCraft AI tournament, all bots that had non-zero probability of choice in equilibrium (Table 2) should be merged into one, which we will refer to as *MegaBot*. At match beginning, MegaBot could apply ϵ-Nash to choose which bot it will enable to play that match. In tournaments, such an approach would have an advantage that does not exist in this paper: the adversary's bot is identified. In a

tournament, if MegaBot is facing a known adversary, it can select the best-response against it. The idea of merging several bots into one is being currently undertaken.

6 Conclusion

6.1 Overview

This work presented a systematic study of strategy selection in StarCraft, defining this process as a metagame, because it adds a layer of reasoning to the game. We modeled the strategy selection metagame as a normal-form game and discussed game-theoretical concepts such as Nash Equilibrium and safe opponent exploitation.

For experiments, we chose a subset of AIIDE 2015 StarCraft AI tournament Protoss bots as the set of strategies to be chosen, because each bot defines a complete mapping of states to actions, fitting our definition of strategy. We filled the strategy selection metagame's payoff matrix by running a prior tournament among selected bots and registering their relative performance. This allowed us to estimate the metagame's Nash Equilibrium and expected payoff. The metagame's equilibrium strategy wins half the matches in expectation.

In the metagame, we observed that strategies interact in cyclical ways and we tested insights from computer rock-paper-scissors tournaments. Our experiments suggest that, whereas equilibrium strategies indeed result in safe payoffs, it is useful to deviate from equilibrium to exploit suboptimal opponents and achieve superior payoffs, confirming insights from rock-paper-scissors. However, it is just as useful to guard itself against exploitation and this can be successfully done by adopting safe opponent exploitation techniques, as they have theoretical guarantees in the maximum loss attainable when deviating from equilibrium.

Metagame source code, including strategy selection techniques and tournament engine is available[8].

6.2 Future Work

Future work could address the issue that the expected payoff of the strategy selection metagame equilibrium is valid only if players select among the same strategies used to calculate the equilibrium. One possible approach to address this limitation is to adopt a wide set of strategies, so that arbitrary behavior can be classified into a known strategy through observation.

---

[7]This variation considers that an adversary can change the rewards over player's choices every turn.

[8]https://github.com/h3ct0r/StarcraftNash

To implement the strategy selection metagame in an actual StarCraft bot, we need to address the techical challenge of merging distinct bots into one to allow the activation of a single bot when a match begins. This approach is currently being pursued. Moreover, future work could develop methods to track the non-stationarity in the metagame's payoff matrix that arises due to persistent knowledge that allows bots to evolve between matches (our experiments were performed without persistent knowledge).

Another useful extension of the present work is to deal with "contextual metagames", that is, metagames associated with game states. This would allow a player to switch strategies during gameplay: if the current situation is similar to a state associated with a metagame, the player can adopt the recommended strategy of that metagame.

Acknowledgments

References

Aha, D. W.; Molineaux, M.; and Ponsen, M. 2005. Learning to win: Case-based Plan Selection in a Real-Time Strategy Game. In *Case-based reasoning research and development*. Springer. 5–20.

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science. Proceedings, 36th Annual Symposium on*, 322–331. IEEE.

Billlings, D. 2001. RoShamBo programming competition. https://webdocs.cs.ualberta.ca/~darse/rsbpc.html.

Churchill, D., and Buro, M. 2013. Portfolio Greedy Search and Simulation for Large-Scale Combat in StarCraft. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, 1–8. IEEE.

Churchill, D., and Buro, M. 2015. Hierarchical Portfolio Search: Prismata's Robust AI Architecture for Games with Large Search Spaces. In *Proceedings of the Artificial Intelligence in Interactive Digital Entertainment Conference (AIIDE)*.

Marcolino, L. S.; Xu, H.; Jiang, A. X.; Tambe, M.; and Bowring, E. 2014. Give a Hard Problem to a Diverse Team: Exploring Large Action Spaces. In *The Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, 1485–1491.

McCracken, P., and Bowling, M. 2004. Safe Strategies for Agent Modelling in Games. *AAAI Fall Symposium on Artificial Multi-Agent Learning* 103–110.

Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. *Algorithmic Game Theory*, volume 1. Cambridge University Press Cambridge.

Ontañón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; and Preuss, M. 2013. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games* 5(4):293–311.

Preuss, M.; Kozakowski, D.; Hagelback, J.; and Trautmann, H. 2013. Reactive Strategy Choice in StarCraft by Means of Fuzzy Control. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, 1–8. IEEE.

Sailer, F.; Buro, M.; and Lanctot, M. 2007. Adversarial Planning Through Strategy Simulation. In *Computational Intelligence and Games (CIG). 2007 IEEE Symposium on*, 80–87. IEEE.

Savani, R., and von Stengel, B. 2015. Game Theory Explorer: software for the applied game theorist. *Computational Management Science* 12(1):5–33.

Stanescu, M., and Čerticky̆, M. 2016. Predicting Opponent's Production in Real-Time Strategy Games With Answer Set Programming. *IEEE Transactions on Computational Intelligence and AI in Games* 8(1):89–94.

Stanescu, M.; Barriga, N. A.; and Buro, M. 2014. Hierarchical Adversarial Search Applied to Real-Time Strategy Games. In *Proceedings of the Artificial Intelligence in Interactive Digital Entertainment Conference (AIIDE)*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge.

Synnaeve, G., and Bessière, P. 2011. A Bayesian Model for Opening Prediction in RTS Games with Application to StarCraft. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, 281–288. IEEE.

Uriarte, A., and Ontañón, S. 2014. Game-Tree Search Over High-Level Game States in RTS games. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*.

Weber, B. G., and Mateas, M. 2009. A Data Mining Approach to Strategy Prediction. In *Computational Intelligence and Games (CIG). 2009 IEEE Symposium on*, 140–147. IEEE.

Weber, B. G.; Mateas, M.; and Jhala, A. 2011. Building Human-Level AI for Real-Time Strategy Games. In *Proceedings of the AAAI Fall Symposium on Advances in Cognitive Systems*, 329–336.