Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 315–320
DOI: 10.15439/2025F2657
ISSN 2300-5963 ACSIS, Vol. 43

StarCraft strategy learning refinement using replay snapshotting

Štefan Krištofík
0000-0002-9995-460X
Institute of Informatics, Slovak Academy of Sciences
Dúbravská cesta 9, 845 07 Bratislava, Slovakia
Email: stefan.kristofik@savba.sk

Hanková Michaela
Faculty of Informatics and Information Technologies
Slovak University of Technology
Ilkovičova 2, 842 16 Bratislava, Slovakia

***Abstract*—We propose a new replay snapshotting (RS) technique for strategy learning from past matches in real-time strategy game StarCraft: Brood War (SCBW). It allows for more precise understanding of particular strategy aspects by sampling the state of selected game features at important checkpoints during a match. We use RS to extract and refine a set of strategies from a large replay dataset STARDATA. To validate our approach in a competitive environment, we implement an AI agent for SCBW. It is able to perform the extracted strategy set against opponents in the BASIL Ladder competition. The agent consistently achieves rank C with 56 % win rate which is a significant improvement over our previous approaches.**

I. Introduction

REAL-TIME strategy (RTS) video games, especially the competitive 1v1 setting, are popular in AI research and education because of the inherent difficulties and challenges they present. During a match, only parts of the battlefield near their own structures and units are visible to players [18]. This partial observability of the game state forces players to make decisions under uncertainty based on available incomplete information. Players need to actively scout the map to gather information about the opponent's activities [11]. Moreover, the number of possible actions a player can execute at each time is overwhelming, making the decision process very complex [2].

The most popular and widely known RTS is Blizzard's StarCraft: Brood War (SCBW) released in 1998 (Fig. 1). One aspect which made the game successful and long-lasting is fair balance of 3 playable races: Protoss (P), Terran (T) and Zerg (Z), each offering unique aesthetics and play-styles. Also, no 'cookie cutter' strategy exists which would be able to defeat all opponents [11]. A strategy is a long-term plan for the current match. It typically involves plans on what structures to build in what order and what army composition to aim for. Strategies can be roughly divided into 3 categories capable of countering each other in a rock-paper-scissors manner (Fig. 2). Another aspect is easy accessibility for players and researchers. The game has low hardware requirements, is free-to-play since 2017 and plenty of tools are available for it to aid with

This work was supported by the Slovak Scientific Grand Agency VEGA under the contract No. 2/0135/23 *"Intelligent sensor systems and data processing"*. Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I05-03-V02-00055.

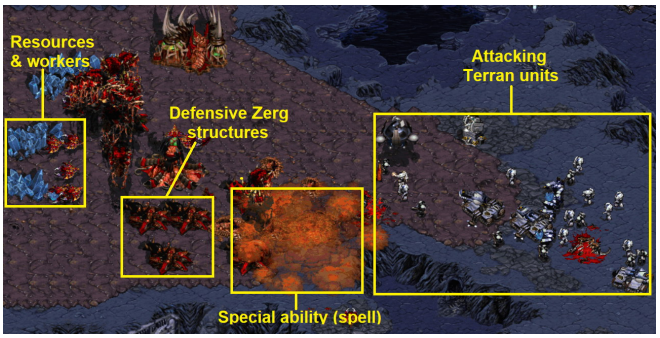

Fig. 1. StarCraft: Brood War

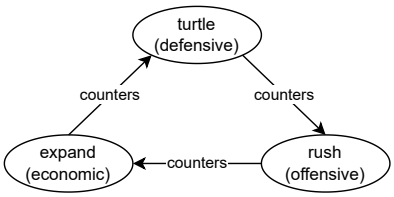

Fig. 2. Rock paper scissors nature of RTS game strategies

research and experimentation, e.g., BWAPI [1], *screp* [2]. Despite considerable progress in their quality over the years, RTS AI agents are not yet capable of beating human expert level players on a consistent basis [3], [9], [12] or are impractical for common video games due to high computational power demands [4], [17].

One possible approach for SCBW agent improvement, especially the strategy selection aspect, is to learn from past matches [12], [14], [15]. SCBW allows recording and archiving of match replays. A vast amount of such data was accumulated throughout the years. For machine learning purposes, a dataset of replays should meet certain requirements to be considered viable and high quality. The largest of such datasets is STARDATA [1], containing 65646 files. Other smaller datasets include [5], [6], [7], [13].

One of the important tasks which can be addressed using

[1]https://github.com/bwapi/bwapi
[2]https://github.com/icza/screp

Topical area: Advanced Artificial Intelligence in Applications

STARDATA is strategy classification [1]. Knowledge about strategies extracted from replays of highly skilled players can benefit SCBW agents in multiple ways. It can be used during a match to predict opponent activities based on the observed information and help adjust further actions. As a part of the preparation process before and between matches, it can improve the ability to imitate and execute strategies that are statistically proven to be successful or pick from a pool of available strategies most suited for the opponent depending on their race or stored past results against them.

Strategy classification from SCBW replays was attempted [5], [6], [7], [1], but the results of each work were quite limited or very specific in some way with regards to dataset size, quality level of matches, number of strategies or races. Classification from STARDATA was also attempted. The work [8] extracted strategies only for one race (Terran) and [10] for all 3 races, but the selection of strategy defining features (SDFs) was not optimized and could be improved.

In this work, we build upon and improve the process of strategy extraction from [8] in multiple ways. We allow a broader range of strategies to be learned by extending the upper game length limit (Section II-A); refine the unit and structure production frequency calculation formula to provide more precise results (Section II-B); refine the SDF selection process for Terran by also considering the opponent's race (Section II-C); tweak strategy classification by adjusting clustering parameters (Section II-D); and finally, we propose a new technique called *replay snapshotting* (RS) to improve the analysis process of extracted data (Section II-E).

To validate our approach, we implement an AI agent nicknamed *MisHanBot* for SCBW playing Terran race and put it into the BASIL Ladder competition [3]. In this competitive environment, the agent is able to perform the set of strategies extracted by RS against opponents of varying quality. It consistently achieves rank C with current win rate of 56 % which is a significant improvement over our previous approaches.

II. Strategy learning

A. Dataset preparation

We use STARDATA to extract and classify Terran strategies for our SCBW agent from Terran matchups: versus Terran (TvT), versus Protoss (TvP) and versus Zerg (TvZ). *Extractor* proposed in [8] is used to collect raw data. Considered are only matches that meet the following criteria:

a) Valid 1v1, i.e., readable in BWAPI, featuring two opposing players using standard set of units.
b) At least one player is Terran.
c) No longer than 25 minutes.
d) Winner is agreed upon by both tools: *extractor* and *screp*.

Criterion a) filters out any custom or non-standard, non-competitive games. Criterion c) allows the agent to learn larger variety of middle and late-game strategies than 15 minute limit used in [8]. Criterion d) ensures that the quality of extracted strategies can be calculated precisely by inspecting their win rates. To help decide on the winning player, the *screp* utility is used to double-check the results provided by *extractor*. To identify a winner reliably in a non-labeled SCBW replay file is a non-trivial task and various replay analysis tools can produce different outcomes. Dataset cleaning results are shown in Table I. The total number of extracted Terran strategies is 28477; 2 from each TvT matchup (since both players are Terran) and 1 from other matchups (since one player is Terran). Next, raw data are extracted from the prepared dataset.

[3]https://www.basil-ladder.net

TABLE I
Extracted Terran strategies

Matchup	STARDATA replays	Selected replays	Extracted strategies
TvT	2550	1753	3506
TvP	17385	12916	12916
TvZ	14531	12055	12055
Total	34466	26724	28477

B. Data extraction and processing

For raw data extraction, we follow the process [8], obtaining detailed information about each game, including basic metadata (player and map names, match lengths, etc.), timestamps for structure, unit and upgrade creation and destruction, and more. For subsequent data processing and compaction, a few adjustments compared to [8] were made. Firstly, we modify the formula for unit frequency:

$$UnitFreq = \frac{1428.6 \cdot UnitCount}{MatchLength - FirstFrame} \quad (1)$$

where $UnitCount$ is the amount of a unit produced in the entire match, $MatchLength$ is given in game frames and $FirstFrame$ now denotes when the unit's first instance was produced. Previously it denoted when the requirements were first met for the unit's creation. Eq. (1) computes the average frequency of a unit's production per minute (1428.6 frames) since its first instance creation until the match's end. This is a more precise metric to measure unit usage than previously used. Secondly, we now calculate frequency for structures, too, using Eq. (1) but replacing $UnitCount$ by $StructCount$.

In this work, we define SCBW strategies by structure build order and construction frequency and unit composition, omitting technology advancements and upgrades. Put more simply, we are interested in structure and unit information. In the remainder of the text, structures and units together will be denoted as features. Next, a set of important features is selected which will be used to categorize strategies.

C. Feature selection

Terran can build 18 different structures and create 13 types of units directly and few more indirectly (see below). Not all features are equally popular and present in games. If a match is short, there is not enough time to reach advanced features. Additionally, players have discovered over the years which unit compositions are strong against certain opponents.

TABLE II
UNIT SELECTION EXAMPLE FOR TVT. PERCENTAGE OF STRATEGIES WITH INDICATED NUMBER OF UNITS

Unit	1+	5+	10+	Selected
SCV	100.0 %	99.9 %	99.3 %	No, essential
Vulture	80.6 %	56.0 %	44.0 %	Yes
Wraith	44.8 %	19.4 %	8.6 %	Yes
Valkyrie	1.9 %	0.3 %	0.0 %	No, low usage

TABLE III
STRUCTURE SELECTION EXAMPLE FOR TVT

Structure	Prerequisite for	Selected
Barracks	Factory, Academy	No, essential
Factory	Vulture, Tank, Goliath	Yes
Academy	Comsat Station	Yes
Comsat Station	Scan Sweep	Yes
Physics Lab	Battlecruiser	No, low usage

For example, Terran biological units are stronger and used more often against Zerg than other opponents. Consequently, some units are perceived as not effective and are rarely used. On the other hand, some features are essential for making progress and are present in vast majority of matches. If a feature belongs to one of these edge cases, it adds no value to strategy classification and can be disregarded.

Previous approach [8] used the same set of features in all 3 Terran matchups. In this work, we inspect each matchup more closely and consider the opponent's race, allowing us to fine-tune SDFs for each matchup. We select sets of SDFs based primarily on the statistical usage of units. Essential and rarely used units are disregarded. For TvT, units that appear at least one time in at least 40 % of strategies were selected. This bar was lowered to 8 % for TvP and 15 % for TvZ due to different usage distributions. An example of SDF selection for some units in TvT is shown in Table II.

Next, a set of structures is selected to complete SDFs. Again, essential and rarely used structures are disregarded. Terran race tech tree is inspected and for each matchup important structures are selected which are prerequisites for the selected units or other structures. An example of SDF selection for some structures in TvT is shown in Table III.

Selected SDFs are listed in Table IV. Note that although Spider Mines and Scan Sweeps can not be created directly, but are a result of unit ability or spell casting, they are still recognized by BWAPI as units, because they materialize on the battlefield and their statistics can be gathered by *extractor*. Next, compacted data about our selected SDFs from all 28477 Terran strategies are inputs to the unsupervised machine learning process of strategy classification.

D. Strategy classification

To classify SCBW Terran strategies into categories based on the compacted SDF data, we use the K-Means algorithm. It tries to find similarities in the SDF data and group similar strategies into clusters. Previous approach [8] used the same number of clusters (value $K = 10$) for all matchups. In this work, we inspect Terran matchups more closely and use the Elbow method [16] with inertia to find the optimal values of K for each, separately.

TABLE IV
SELECTED STRATEGY DEFINING FEATURES

Structures	Matchups
Factory, Machine Shop, Mis. Turret, Academy, Comsat Station, Starport, Control Tower	TvT, TvP, TvZ
Science Facility, Bunker	TvP, TvZ
Units	**Matchups**
Marine, Siege Tank, Vulture, Goliath, Dropship, Wraith, Spider Mine (unit ability), Scan Sweep (spell)	TvT, TvP, TvZ
Science Vessel	TvP, TvZ
Firebat, Medic	TvZ

Fig. 3. Elbow method for TvT

The results of the Elbow method for TvT are shown in Fig. 3. Among considered values of K (4–6), $K = 5$ was chosen because of the best distribution of average game lengths between individual clusters and also fair distribution of number of matches per cluster. With $K = 5$, we identified 1 cluster containing short games, 2 clusters containing middle length games and 2 clusters containing long games. Detailed results of strategy classification for TvT are summarized in Fig 4. It displays amounts of strategies in clusters.

The resulting value for TvP was $K = 6$: 2 clusters with short, 2 with middle length and 2 with long games. For TvZ it was also $K = 6$: 2 clusters with short, 1 with middle length and 3 with long games.

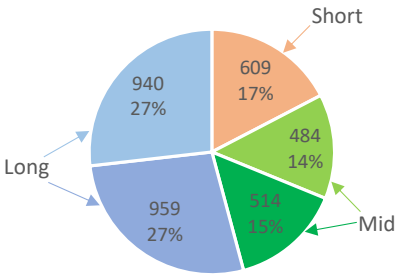

Fig. 4. Strategy classification into clusters for TvT

TABLE V
EXAMPLE OF 2 EXTRACTED STRATEGIES FOR TVT: CLUSTERS '1' AND '3'

Structure	Cl. '1' Freq.	Cl. '3' Freq.	Unit	Cl. '1' Freq.	Cl. '3' Freq.
M. Shop	4.3	3.3	Marine	6.9	3.7
Factory	2.6	3.5	Tank	2.4	2.4
Starport	5.7	3.0	Vulture	3.5	5.2
Academy	6.2	6.0	Goliath	3.8	8.2
Com. Stat.	3.2	5.7	Scan Sw.	4.3	7.2
C. Tower	5.3	4.7	Dropship	6.6	8.1
M. Turret	1.2	6.3	S. Mine	2.2	8.7
-	-	-	Wraith	7.5	2.4

Examples of 2 extracted strategies for TvT are shown in Table V. It shows average frequency rankings of selected SDFs. Lower numbers indicate more frequent appearance while higher numbers indicate less frequent appearance.

Cluster '1' represents a traditional long match strategy using mechanical units: Tanks, Vultures with their ability Spider Mines, and Goliaths. It utilizes structures Factory and Comsat Station often and will construct many Missile Turrets (static defensive). Cluster '3' represents a middle game strategy using a combination of ground (Marine, Tank) and air (Wraith) units. It utilizes structures Starport and Factory often.

The above examples show notable differences between clusters. To find such diversity was an important goal of strategy extraction. A total of 17 strategies were extracted and are available for our Terran agent.

E. Replay snapshotting

Extracted strategies as shown in Table V provide basic guidance on how to proceed in a game for our agent. It learns what types of features to create. We further refine strategies by a new technique called *replay snapshotting* (RS). The basic principle of RS is that a subset of strategies from each cluster is inspected and the amounts of currently existing instances of selected SDFs are snapshotted (sampled) at predetermined checkpoints. This enables our agent to imitate and implement strategies more precisely because it gives the agent more tangible objectives to reach in certain phases of a match. The agent now gains knowledge about how many instances of features are necessary to achieve those partial objectives.

For snapshotting, we have selected strategies that are the best representatives from each cluster. More specifically, strategies used in matches with duration closest to the average cluster match lengths were selected. For example, Cluster '1' from Table V contains matches with an average duration of 18m:31s. Therefore, we select strategies close to this length for RS from this cluster. The number and spacing of checkpoints were set for each cluster individually, based on their average match lengths. The intent is to have checkpoints distributed across all game stages: early, middle and late game, to cover most situations and phases of the entire match. The first checkpoint is usually set at the 4 minute mark because the first important strategic decisions for Terran are made around this time. The last checkpoint is set near the average cluster duration, e.g., at the 18 minute mark for Cluster '1'. The number of strategies selected for RS was 50 per cluster.

TABLE VI
EXAMPLE OF REPLAY SNAPSHOTTING FOR CLUSTER '1' FOR TVT. m–MINUTES

Structure	4m	6m	9m	14m	18m
Factory	1.1	1.7	2.9	4.6	5.8
M. Shop	0.6	1.0	1.9	2.9	3.2
Starport	0.0	0.3	0.4	0.7	0.9
C. Tower	0.0	0.2	0.4	0.6	0.9
Academy	0.0	0.2	0.7	0.9	1.0
Com. Stat.	0.0	0.0	1.0	2.3	2.8
M. Turret	0.0	0.0	0.6	5.7	9.3
Unit	**4m**	**6m**	**9m**	**14m**	**18m**
Marine	2.3	2.1	1.3	0.6	0.3
Tank	0.0	1.7	4.5	11.5	15.8
Vulture	0.3	1.1	3.0	3.5	4.5
Goliath	0.0	0.8	1.1	6.0	9.8
Dropship	0.0	0.1	0.0	0.9	2.5
S. Mine	0.0	1.0	3.2	7.6	11.9
Wraith	0.0	0.1	0.1	0.4	0.4

An example of the RS technique for Cluster '1' for TvT is shown in Table VI. It shows average numbers of instances of selected SDFs snapshotted at exact checkpoints during matches selected for RS. For this cluster, 5 checkpoints were defined at 4, 6, 9, 14 and 18 minute marks. The 4 and 6 minute checkpoints suggest how to proceed in early game and prepare for the transition into middle game. Next, the 9 and 14 minute checkpoints suggest how to transition from early to middle game and prepare for late game. Finally, the last 18 minute checkpoint advises on what structures should be present in the player's base and what is the army composition to aim for in late game. Note that in Table VI RS is not able to track any instances of Scan Sweep spell (one of selected TvT SDFs) because it has only a very short duration and then it disappears from the battlefield.

To implement this strategy, the agent should start by constructing some Factories with Machine Shop, start training some Marines and Vultures, later adding Tanks and Goliaths, research the Vulture's ability to lay Spider Mines and start using them. In middle game, it should add more Factories with Machine Shops, construct an Academy and then Comsat Stations and start producing Missile Turrets. Army should now contain more Tanks and Goliaths while keeping adding Vultures in small numbers and increase the usage of Spider Mines and also slowing down Marine production. A late game player's base should have a healthy number of Factories with Machine Shops, an Academy, some Comsat Stations and many Missile Turrets should be scattered around the battlefield. Late game army composition should comprise mainly Tanks with Goliaths both in healthy numbers with the support of Vultures

and few Dropships and many Spider Mines should be laid on the battlefield. Starting from middle game, the agent can also consider occasionally constructing less used structures such as Starport or Control Tower and training some less used units such as Wraiths.

Based on values obtained by RS, we divide features into three categories:

- Mandatory: $Value >= 1$. Present in majority of matches. The agent should try to construct the indicated number or more of them.
- Optional: $Value \in (0; 1)$. Not present in each match. The agent can consider construction of one instance in case of structures and few instances in case of units.
- Not advisable: $Value = 0$. Not present in any match. It is not advised for the agent to construct these.

In Table VI, an example of a mandatory feature is Factory at 9 minute mark ($Value = 2.9$). The agent should try to have at least 2 but optimally 3 or more of them at this checkpoint. An example of an optional feature is Machine Shop at 4 minute mark ($Value = 0.6$). The agent can consider construction of one at this checkpoint.

Strategies extracted and refined by RS (RS strategies) provide very detailed guidance for the entire match duration as can be seen in the above example. The next objective is to implement them into our agent.

III. Strategy implementation

Our agent *MisHanBot* is based on the public version of *Steamhammer* (SH) [4], a popular starter agent for development and experimentation. Its main race is Zerg, but can also play other races, however, on a lower level than the main one. It also supplies a set of starting strategies and provides many necessary functionalities of a SCBW agent already implemented on a basic level. While adopting existing functionalities from SH, like unit micromanagement, our agent also has many small improvements over baseline SH Terran, including better unit control and unit repair logic, or better ability and spell usage. Terran strategies are implemented in SH by:

- *Configuration file (config)*, an explicit list of items to create and other commands to perform from the start of the match. This list can contain all features, not only those selected as SDFs (Table IV). Depending on its length, it can be used for all game stages.
- *BOSS*, a module which takes over and continues to implement the strategy after all items in *config* have been created. It tries to keep the 'trend' set by *config* and continues with construction of items present in the list.

Strategy selection during a match is based on randomness with weighted probabilities. A pool of strategies is defined for various situations (matchups, opponent's plans, etc.), and each strategy in the pool is assigned a weight based on suitability for tackling that situation. When a situation occurs, SH selects one strategy from that situation's pool randomly.

[4]https://satirist.org/ai/starcraft/steamhammer/

TABLE VII
WIN RATES AND WEIGHTS FOR TVT STRATEGY POOL

Strategy (Cluster)	'0'	'1'	'2'	'3'	'4'
Win rate	52.8	55.5	28.7	50.3	53.6
Weight	22	23	12	21	22

We implement RS strategies into our agent as follows. We overwrite *config* so that it contains only our Terran RS strategies. We define 3 situations: TvT, TvP, TvZ. Note that in our agent, we do not create situations considering opponent's plans. Our strategy selection depends only on the opponent's race. Next, we create a strategy pool for each situation. To account for the randomness of RS strategies (see Section II-E), we round up the advised values from Table VI to the nearest integers. This means that values of all optional features will be rounded to 1, i.e., the agent can try to have one instance of them. Values of all mandatory features will be increased, too. Therefore, the agent is given the upper bounds of what numbers to aim for. After this adjustment, we put RS strategies into strategy pools in *config*; TvT strategy pool includes 5 strategies, TvP and TvZ pools both include 6 strategies.

The explicit lists in *config* for each RS strategy are set up using knowledge of the Terran tech tree while trying to follow RS advice (example in Table VI) as closely as possible.

Next, we set weights for strategy pools. Weights are assigned to extracted strategies in pools proportional to their win rate. Sum of weights in a pool is equal to 100. More successful strategies get higher weights and in turn will be picked more often in games than those with lower weights. TvT win rates were obtained as part of this work (see Section II-D). TvP and TvZ win rates were obtained from collaboration with the authors of two other agents (see Section IV) who did strategy classification for Protoss and Zerg races. Win rates and corresponding weights for the TvT strategy pool are shown in Table VII. It displays average win rates against all other Clusters in the pool. When our agent is in a TvT situation, Cluster '1' has the highest win rate and in turn the highest chance to be used: 23 %. On the other hand, Cluster '2' is the least successful and has the lowest chance to be used: 12 %.

During a match after processing *config*, strategy control of our agent is handed over to the *BOSS* module, similar to SH. It will follow a trend established by *config* and keep producing features as specified in the last RS checkpoint in the remainder of the match.

IV. Results and conclusion

At the time of writing this paper, our agent *MisHanBot* is active in the BASIL Ladder competition. It has rank C with ELO rating 2586, overall win rate 49.4 %, recent win rate 56.3 % and is situated at the 46th place out of 101 currently active agents. The overall number of agents in the ladder is 159. Agents ranked 102 or below are temporarily disabled to keep the competition's level high. They can be re-enabled again under certain conditions.

TABLE VIII
OUR STARDATA-TRAINED AGENT RESULTS IN BASIL LADDER AS OF MAY, 15TH 2025

Agent	ELO rating	BASIL rank	Games played	Win rate	Win rate 7 days
KasoBot (T) [8]	2390	E	48821	38.9 %	41.7 %
Azergo (Z)	2206	E	9034	28.9 %	26.1 %
TommyBot (P)	2203	F	9057	19.6 %	25.7 %
MisHanBot (T) [This work]	2586	C	10253	49.4 %	56.3 %

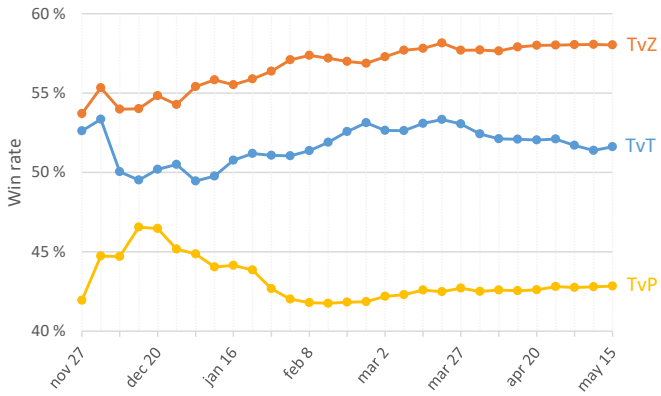

Fig. 5. Win rates of our agent against each opponent race in the last 6 months (Nov 2024–May 2025). Source: BASIL Ladder https://www.basil-ladder.net/bot.html?bot=MisHanBot

The comparison with our other currently active agents is shown in Table VIII. Results obtained by our new agent *MisHanBot* are a significant improvement over our other approaches. Note that both agents *Azergo* and *TommyBot* extract detailed data and learn strategies from STARDATA for their particular races, but use different approaches for their implementation. Strategies obtained by *Azergo* are not as effective and could be improved. *TommyBot* is an experimental agent built from scratch, i.e., it does not use any of available popular starter agents, such as SH.

Fig. 5 displays win rate trends of our agent in the last 6 months against all three opponent races. Results indicate that TvP win rate (currently at 42.8 %) is not satisfactory as it consistently trails behind other two matchups and brings the agent's overall win rate down considerably. On the other hand, TvZ win rate (currently at 58.0 %) is satisfactory while in TvT it is slightly above the average level (currently at 51.6 %).

Our results indicate that the proposed *replay snapshotting* approach for strategy refinement is beneficial for overall SCBW AI agent improvement. Our agent is able to imitate extracted strategies more accurately than our previous approaches and it performs better in a competitive environment. As future work, it could be worth looking more closely into TvP particularly, identify and remedy the weaknesses of our agent in this matchup. Other improvements could include: further SDF selection fine-tuning by adding or removing items; RS adjustments by considering the total amount of produced features at checkpoints instead of counting currently existing instances; including more matches per cluster into RS; adding checkpoints beyond the average cluster time; fine-tuning checkpoint distribution; setting probabilities of strategies within pools with low win rates to 0 to avoid their usage during matches; and more.

REFERENCES

[1] Z. Lin, J. Gehring, V. Khalidov and G. Synnaeve, "STARDATA: A StarCraft AI Research Dataset," *13th AAAI Conf. Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2017,* pp. 50–56, https://dx.doi.org/10.48550/arXiv.1708.02139

[2] S. Ontañon, 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 Trans. Computational Intelligence and AI in games, IEEE Computational Intelligence Society, 2013,* 5(4), pp. 293–311, https://dx.doi.org/10.1109/TCIAIG.2013.2286295

[3] Mi. Čertický, D. Churchill, K.-J. Kim, Ma. Čertický and R. Kelly, "StarCraft AI Competitions, Bots and Tournament Manager Software," *IEEE Trans. Games, 2018,* 11(3), pp. 227–237, https://dx.doi.org/10.1109/TG.2018.2883499

[4] O. Vinyals, I. Babuschkin et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature, 2019,* 575, pp. 350–354, https://dx.doi.org/10.1038/s41586-019-1724-z

[5] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," *IEEE Symp. Computational Intelligence and Games, 2009,* pp. 140-147, https://dx.doi.org/10.1109/CIG.2009.5286483

[6] H. C. Cho, K. J. Kim and S. B. Cho, "Replay-based strategy prediction and build order adaptation for StarCraft AI bots," *IEEE Conf. Computational Intelligence in Games (CIG), 2013,* pp. 1-7, https://dx.doi.org/10.1109/CIG.2013.6633666

[7] G. Synnaeve and P. Bessière, "A Dataset for StarCraft AI & an Example of Armies Clustering," *Artificial Intelligence in Adversarial Real-Time Games, 2012,*, https://dx.doi.org/10.48550/arXiv.1211.4552

[8] Š. Krištofík, P. Malík, M. Kasáš, Š. Neupauer, "StarCraft agent strategic training on a large human versus human game replay dataset," *Federated Conf. Computer Science and Information Systems, FedCSIS 2020,* 21, ACSIS, pp. 391–399, https://dx.doi.org/10.15439/2020F178

[9] M. Świechowski, "Game AI Competitions: Motivation for the Imitation Game-Playing Competition," *Federated Conf. Computer Science and Information Systems, FedCSIS 2020,* 21, ACSIS, pp. 155–160, https://dx.doi.org/10.15439/2020F126

[10] Š. Krištofík, M. Kasáš, P. Malík, "StarCraft strategy classification of a large human versus human game replay dataset," *Federated Conf. Computer Science and Information Systems, FedCSIS 2021,* 25, ACSIS, pp. 137–140, https://dx.doi.org/10.15439/2021F48

[11] G. Robertson, I. Watson, "A Review of Real-Time Strategy Game AI," *AI Magazine, 2014,* 35(4), pp. 75–104, https://dx.doi.org/10.1609/aimag.v35i4.2478

[12] S. Xu, H. Kuang et al., "Macro action selection with deep reinforcement learning in StarCraft," *15th AAAI Conf. Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2019,* pp. 94–99, https://dx.doi.org/10.48550/ARXIV.1812.00336

[13] J. J. Merelo-Guervós, A. Fernández-Ares et al., "RedDwarfData: a simplified dataset of StarCraft matches," 2017, https://dx.doi.org/10.48550/arXiv.1712.10179

[14] F. Dai, J. Gong, J. Huang, J. Hao, "Macromanagement and Strategy Classification in Real-Time Strategy Games," *2nd China Symp. Cognitive Computing and Hybrid Intelligence (CCHI), 2019,* pp. 263–267, https://dx.doi.org/10.1109/CCHI.2019.8901957

[15] N. Justesen, S. Risi, "Learning Macromanagement in StarCraft from Replays using Deep Learning," 2017, https://dx.doi.org/10.48550/arXiv.1707.03743

[16] C. Shi, B. Wei et al., "A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm," *J. Wireless Comm. and Networking, 2021,* https://dx.doi.org/10.1186/s13638-021-01910-w

[17] N. Justensen, M. Kaselimi et at., "Human-like Bots for Tactical Shooters Using Compute-Efficient Sensors," 2024, https://dx.doi.org/10.48550/arXiv.2501.00078

[18] J. Gehring, D. Ju, V. Mella, D. Gant, N. Usunier, G. Synnaeve, "High-Level Strategy Selection under Partial Observability in StarCraft: Brood War," 2018, https://dx.doi.org/10.48550/arXiv.1811.08568