A Benchmark for StarCraft Intelligent Agents

Alberto Uriarte and **Santiago Ontañón**
Computer Science Department
Drexel University
{albertouri,santi}@cs.drexel.edu

Abstract

The problem of comparing the performance of different Real-Time Strategy (RTS) Intelligent Agents (IA) is non-trivial. And often different research groups employ different testing methodologies designed to test specific aspects of the agents. However, the lack of a standard process to evaluate and compare different methods in the same context makes progress assessment difficult. In order to address this problem, this paper presents a set of benchmark scenarios and metrics aimed at evaluating the performance of different techniques or agents for the RTS game StarCraft. We used these scenarios to compare the performance of a collection of bots participating in recent StarCraft AI (Artificial Intelligence) competitions to illustrate the usefulness of our proposed benchmarks.

Introduction

Real-Time Strategy (RTS) is a genre of strategy games that has been recognized as being very challenging from an Artificial Intelligence (AI) standpoint. In a RTS game, each player must control a set of units in order to build an economy (gathering resources), produce an army (building training facilities) and destroy her enemy (commanding her army). This is challenging, since the state space of RTS games is typically very large given the enormous number of potential actions that can be taken at any time and that the players are able to make decisions in real-time rather than iteratively. Given these unique properties of RTS games, traditional AI techniques for game playing are not applicable to RTS games.

Since the first call for research in AI for RTS (Buro 2003), significant progress has occurred in many different subproblems on RTS AI (Ontañón et al. 2013). In order to assess progress and compare the strengths and weaknesses of different techniques, several competitions (beginning with the ORTS competition[1] and culminating with the current StarCraft AI competition[2]) were created in order to provide a common testbed. However, due to the multifaceted nature of RTS games, the results coming out of such competitions are insufficient for understanding the strengths and weaknesses of specific techniques or agents. For example, a StarCraft bot could potentially win a competition by employing a hard-coded strategy that no other bot had a counter-for, while it is clear that finding a good hard-coded strategy for a single specific game (StarCraft) does not contribute toward solving the general problem of AI for RTS games.

To address this problem, this paper proposes a new method of evaluation and comparison of the performance of StarCraft playing agents. We present a benchmark composed of a series of scenarios, each of them capturing a different aspect of RTS games. Each scenario is defined by a starting situation in StarCraft where the agent needs to either defeat the opponent or survive for as long as possible. Additionally, we provide an evaluation metric for each scenario, in order to assess the performance of a given bot in the given scenario. All the scenarios were created for the RTS game StarCraft: BroodWar, and these scenarios as well as the necessary scripts to automatically run them and obtain the evaluation metrics are freely available online.

While we do not expect this approach to replace the StarCraft competition (since the competition still seems to be the best way to evaluate the performance of complete agents in a fair way), the proposed benchmark aims at providing a more fine-grained picture of the strengths and weaknesses of specific algorithms, techniques and complete agents. By defining scenarios that capture different aspects of RTS gameplay, and evaluating a given agent in all of those scenarios, we obtain a picture of the strengths and weaknesses of the agent. Moreover, combining the results obtained in these benchmarks with the final result of the StarCraft competition, we can also infer which aspects of gameplay have a stronger impact in final play strength. Additionally, by having a uniform benchmark that is shared by the community, we ensure a standard way to compare AI techniques for RTS games, helping us measure progress in the field. Moreover, the set of scenarios presented in this paper only constitutes a starting point, which we expect to grow over time.

The remainder of this paper is organized as follows: First, we provide some background on previous benchmarks used by the RTS community. Then, we present our proposed benchmark by initially defining some evaluation metrics and then discussing an initial collection of scenarios. Finally, we present the results obtained from running a collection of

[1]https://skatgame.net/mburo/orts/#Competitions

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

current StarCraft bots on these scenarios evaluated by our defined metrics.

Background

Playing RTS games requires planning and reasoning at multiple time scales, from second-by-second reactive control of units to long-term planning. Usually, this continuum of time scales is divided into three levels (Ontañón et al. 2013):

Reactive Control: for short-term decision-making, such as unit movement, targeting or firing.

Tactics: for medium-term decision-making, like army control and building positioning.

Strategy: for long-term decision, like army composition, build-order or determining counter-strategies against a given opponent.

This section briefly describes different testing scenarios that different research groups have employed to evaluate algorithms to solve problems in each of these three levels.

The problems most tackled by the research community are those related to *reactive control*. In this category, we can find problems such as unit formation (Young et al. 2012; Danielsiek et al. 2008), unit survivability (Uriarte and Ontañón 2012; Nguyen, Wang, and Thawonmas 2013) and target selection (Churchill and Buro 2013). Usually, different papers define different testing scenarios (usually not public) and sometimes they use the same scenarios as professional players use to train their skills[3] (Young and Hawes 2014).

In the case of *tactics*, spatial reasoning is of key importance. People have addressed problems such as how to solve qualitative navigation problems (Hagelbäck 2012), how intelligent agents use terrain analysis to exploit chokepoints or reasoning about the immediate enemy thread (Muñoz-Avila, Dannenhauer, and Cox 2015), optimizing resource gathering (Christensen et al. 2012; de Oliveira, Goldbarg, and Goldbarg 2014), or building placement (Certický 2013; Richoux, Uriarte, and Ontañón 2014). In these cases, the researchers use professional StarCraft maps as a benchmark, sometimes using the StarCraft internal score or just counting the amount of resources gathered in a certain amount of time.

Concrete experiments dealing with long-term strategy are rare in the literature, except for some isolated pieces of work focusing on base expansion importance or handling dynamic obstacles (Ontañón et al. 2010; Jaidee and Muñoz-Avila 2012). The common approach is to use a general map and the win ratio to test their methods.

Moreover, in this paper, we have not included scenarios to evaluate pathfinding algorithms, since well-known benchmarks exist for this task (Sturtevant 2012).

Metrics

This paper presents a benchmark for RTS IAs consisting of a collection of scenarios where the intelligent agents will be evaluated playing against a predefined opponent. In order to assess how well a given agent has addressed a given scenario, we need proper evaluation metrics. Moreover, different scenarios require different metrics. For example, while in some scenarios we are interested in measuring whether the agent was able to completely destroy the opponent, in other scenarios, we might be interested in measuring how long the agent is able to survive. This section presents a collection of metrics that we later use in the definition of the proposed scenarios. We will always assume that the metric is evaluated in a game state S, played by two players, A and B, where A is the player controlled by the agent and B is the opponent. Also, we will use t to denote the amount of time it took for agent A to complete the scenario (i.e., to win, lose, or reach a predefined timeout). All the metrics are designed to be normalized either in the interval [0,1] or [-1,1], with higher values representing better agent performance.

Survivor's life: To evaluate the performance of a *reactive control* technique, the typical scenario is a small map with two military forces fighting each other. In this case we want to win while losing as few units as possible and in the least time possible. For this reason we suggest a modified version of LTD2 (Life-Time Damage 2) proposed by Kovarsky and Buro (2005) where we only take into account the hit points of the surviving units and the time elapsed. Therefore, we compute the survivor's life metric (SL) as the sum of the square root of hit points remaining of each unit divided by amount of time it took to complete the scenario (win/defeat/timeout), measured in frames:

$$SL(S) = \frac{\sum_{a \in U_A} \sqrt{HP(a)} - \sum_{b \in U_B} \sqrt{HP(b)}}{t}$$

where $HP(u)$ is the Hit Points of a unit u, U_X is the set of units of the player X. Moreover, in order to be able to compare results obtained in different scenarios, we normalize the result to the interval $[-1, 1]$. To do this we need to compute the lower and upper bounds. The lower bound is when player A is defeated in the minimum time and without dealing any damage to player B:

$$timeToKill(A, B) = \frac{\sum_{a \in U_A} HP(a)}{\sum_{b \in U_B} DPF(b)}$$

$$lowerBound(S) = \frac{-\sum_{b \in U_B} \sqrt{HP(b)}}{timeToKill(A, B)}$$

where the DPF is the Damage Per Frame a unit can inflict. The upper bound can be computed analogously by considering the situation when player B is defeated in the minimum possible time. Once the bounds are computed, if the SL is negative, we normalize using the lower bound. Otherwise, we use the upper bound to normalize:

$$SL^{norm}(S) = \begin{cases} \frac{SL(S)}{|lowerBound(S)|} & \text{if } SL(S) \leq 0 \\ \frac{SL(S)}{|upperBound(S)|} & \text{otherwise} \end{cases}$$

Time survived: In other scenarios we want to evaluate the amount of time the agent can survive in front of an invincible opponent. We can generate a score between $[0, 1]$ by normalizing the time the agent survived by a predefined timeout (TO): $TS = t/TO$

[3]http://wiki.teamliquid.net/starcraft/Micro_Training_Maps

Time needed: In other scenarios we want to evaluate the time to complete a certain task or to recover from a loss. In this case we start a timer when a certain event happens (e.g., a building is destroyed) and we stop it after a timeout (TO) or after a condition is triggered (e.g., the destroyed building is replaced). The score is computed as: $TN = (TO - t)/TO$.

Units lost: Sometimes, we want to evaluate how a bot can respond to a strategic attack. In these cases, we can count the difference in units lost by players A and B. We can normalize between $[0, 1]$ by dividing the number of units lost by the maximum units of the player:

$$UL = \frac{B_{lost}}{B_{max}} - \frac{A_{lost}}{A_{max}}$$

Where, A_{lost} and B_{lost} are the units that player A and player B lost during the scenario respectively, and A_{max} and B_{max} are predefined upper bounds on the expected number of units each player could lose in a given scenario (this is analogous to a time out in scenarios controlled by time, so if player A loses more than A_{max} units, the scenario ends).

Finally, some parts of StarCraft are stochastic and some AI techniques include stochastic elements. As such, each scenario should be repeated sufficiently large number of times in order to calculate an average value of these metrics that is representative of the bot's performance. All the experiments below show the average and standard deviation after running each scenario 10 times.

Benchmark Scenarios

In this section we present a wide range of scenarios for benchmark purposes. The list is not exhaustive, but is to be seen as a starting point. We expect to keep the number of tests growing. We choose StarCraft to implement the scenarios because of its recent popularity over other RTS games as a test bed for AI game research (Ontañón et al. 2013). All the scenarios described in this section for StarCraft are freely available online[4] to the research community.

In order to evaluate a StarCraft playing agent using these scenarios, it must support two types of behavior:

- In a micromanagement scenario (those where the player only controls military units) the goal for the intelligent agent should be to reach the starting position of the enemy. Those scenarios are designed in such a way that an enemy confrontation is unavoidable if the intelligent agent tries to reach the opponent's starting position.
- In full-game scenarios, the goal is just like a regular game: do whatever is necessary to win the game.

The following subsections present a detailed list of all the proposed scenarios. For each scenario, we provide the motivation, the map layout, the opponent behavior, termination condition and evaluation metric.

[4]Each scenario consists of a StarCraft map containing the initial situation plus a tournament module to run the benchmark. We provide versions for BWAPI 3.7 and 4.1 downloadable from: https://bitbucket.org/auriarte/starcraftbenchmarkai

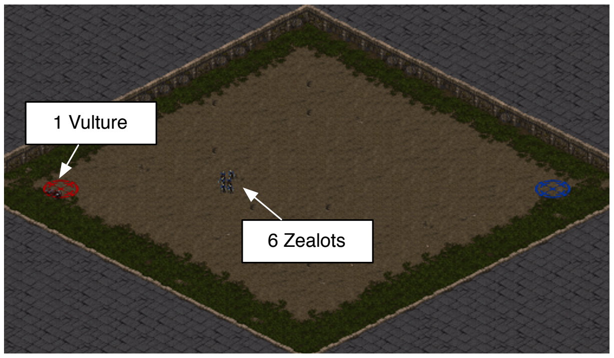

Figure 1: StarCraft scenario RC1-A-V6Z.

RC1: Reactive Control - Perfect Kiting

The purpose of this scenario is to test whether the intelligent agent is able to reason about the possibility of exploiting its mobility and range attack against a stronger but slower unit in order to win. In this scenario, a direct frontal attack will result in losing the combat, but via careful maneuvering, it is possible to win without taking any damage. Previous works have used this type of scenario to evaluate reactive control of their bots (Uriarte and Ontañón 2012; Wender and Watson 2012; Young and Hawes 2014; Liu, Louis, and Ballinger 2014). The different configurations are set to test the scalability and the awareness of the surroundings of the agent.

- **Map description:** We propose two different maps and four initial army configurations. This gives rise to 8 different scenario configurations (**RC1-A-VZ**, **RC1-A-V6Z**, ..., **RC1-B-V6Zg**).

  Layout A: Small map of 64 × 64 tiles with no obstacles. Player A starts at 9 o'clock, player B at 3 o'clock.

  Layout B: A big region connected to one small region on each side. Player A starts at the west small region, player B at the east small region. In this layout, intuitively, the player should avoid conflict inside a small region where its troops could get stuck.

  VZ: Player A has 1 ranged fast weak unit (Vulture) and player B has 1 melee slow strong unit (Zealot).

  V6Z: Player A has 1 ranged fast weak unit (Vulture) and player B has 6 melee slow strong units (Zealot).

  3V6Z: Player A has 3 ranged fast weak units (Vulture) and player B has 6 melee slow strong units (Zealot). The purpose of this configuration is to test whether player A's units disturb each other.

  V9Zg: Player A has 1 range fast weak unit (Vulture) and player B has 9 melee fast weak units (Zergling).

  Figure 1 shows an example of scenario RC1-A-V6Z using the **Layout A** with 1 Vulture against 6 Zealots, and Figure 2 shows **Layout B** on the scenario RC1-B-V6Z.

- **Enemy behavior:** Player B will try to reach the player A's starting point. If it finds any player A's troops, it will chase it trying to kill it.

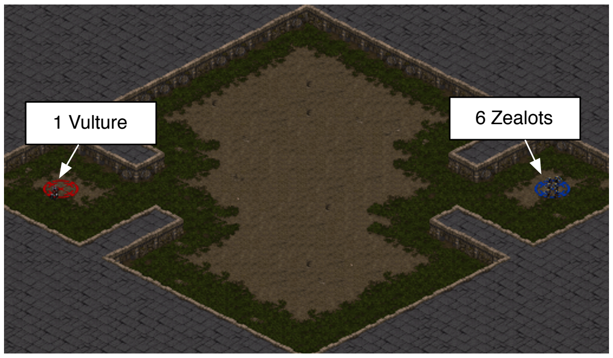

Figure 2: StarCraft scenario RC1-B-V6Z.

- **Termination condition:** One player's army is completely destroyed.
- **Evaluation:** *Survivor's life*.

RC2: Reactive Control - Kiting

In this scenario the intelligent agent is at a disadvantage, but using a hit-and-run behavior might suffice to win. The main difference with the previous case is that here, some damage is unavoidable. Some previous works done in this area in addition to the ones already presented in the previous scenario are (Young et al. 2012; Parra and Garrido 2013).

- **Map description:** Here we have two possible configurations:

  A-3D3Z: Using the same map **Layout A** as before, but where player A has 2 Dragoons in his starting locations and 1 Dragoon near player B starting locations, while player B has 3 Zealots. Intuitively, in this configuration player A should retreat his isolated Dragoon in order to wait for reinforcements.

  A-2D3H: Using map **Layout A** where player A has 2 Dragoons and player B has 3 Hydralisks.
- **Enemy behavior:** Player B will try to reach the player's starting point. If it finds any player troop, it will chase it trying to kill it.
- **Termination condition:** One player's army is completely destroyed.
- **Evaluation:** *Survivor's life*.

RC3: Reactive Control - Sustained Kiting

In this case there is no chance to win so we should try to stay alive as much time as possible. A typical example of this behavior is while we are scouting the enemy base. Previous work: (Nguyen, Wang, and Thawonmas 2013)

- **Map description:** This scenario uses a map **Layout C** consisting in two regions (R_1, R_2) connected by a chokepoint. Player A starts in region R_1, which is empty. Player B starts in region R_2 which it has 9 mineral patches and a Vespene gas geyser (like a regular StarCraft starting position); Player B has already a Nexus, 2 Pylons, 2 Zealots and 5 Probes; player A has 1 SCV.
- **Enemy behavior:** Player B will hold its position. If an enemy is within its range of vision, it will chase it with its two Zealots.
- **Termination condition:** One player's army is completely destroyed or timeout after 300 seconds.
- **Evaluation:** *Time survived* in frames since a Zealot starts chasing the SCV normalized by the timeout. Notice that this is a micromanagement map, and thus, the goal of the agent is to reach B's starting position.

RC4: Reactive Control - Symmetric Armies

In equal conditions (symmetric armies), positioning and target selection are key aspects that can determine a player's success in a battle. This scenario presents a test with several configurations as a baseline to experiment against basic AI opponents. Some similar tests have been performed in the past (van der Heijden, Bakkes, and Spronck 2008; Blackadar and Denzinger 2011; Shantia, Begue, and Wiering 2011; Iuhasz, Negru, and Zaharie 2012; Churchill and Buro 2013; Bowen, Todd, and Sukthankar 2013; Zhen and Watson 2013; Liu, Louis, and Ballinger 2014; Justesen et al. 2014; Nguyen, Nguyen, and Thawonmas 2015).

- **Map description:** It combines **Layout A** with the following possible army configurations:

  5V: Each player has 5 Vultures. The goal is to test range units.

  9Z: Each player has 9 Zealots. The goal is to test melee units.

  12D: Each player has 12 Dragoons.

  12Mu: Each player has 12 Mutalisks. To test flying units with "splash" damage.

  20Ma8Me: Each player has 20 Marines and 8 Medics. To test mixed groups.

  5Z8D: Each player has 5 Zealots and 8 Dragoons.
- **Enemy behavior:** Player B will hold its position in order to maintain its formation.
- **Termination condition:** One player's army is completely destroyed.
- **Evaluation:** *Survivor's life*.

T1: Tactics - Dynamic obstacles

This scenario measures how well an agent can navigate when chokepoints are blocked by dynamic obstacles (e.g., neutral buildings). Notice that we are not aiming to benchmark pathfinding, but high-level navigation.

- **Map description:** Here we use a professional StarCraft map called Heartbreak Ridge[5]. This map has the particularity that the third closest base location from the starting point has two ways to access it. The first is through an open chokepoint. Alternatively there is a closer chokepoint blocked by an indestructible building and a stack of minerals. The default StarCraft pathfinding algorithm does not take into account dynamic objects, so it will

[5]http://wiki.teamliquid.net/starcraft/Heartbreak_Ridge

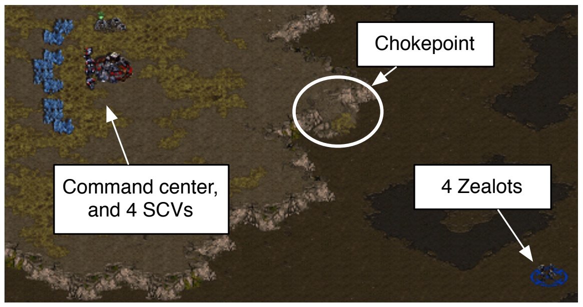

Figure 3: StarCraft scenario S1.

lead the unit through the shortest (blocked) path until the blocking objects are revealed form the fog of war, at which point it will recalculate the path with the new information. Player A starts with a worker in the regular starting position of the map, while the starting position for Player B (who has no units) is this third base expansion. Notice that since this is a micromanagement map, even if B does not have any units in this scenario, the goal of the agent is to reach B's position.

- **Enemy behavior:** None.
- **Termination condition:** Player A reaches the starting position of Player B or a timeout after 60 games seconds.
- **Evaluation:** *Time needed.*

S1: Strategy - Building placement

One of the possible strategies in StarCraft is to *rush* your enemy. *Rushing* consists of training a certain number of military units as fast as possible, then attacking your enemy while she is still developing her economy. A typical counter-strategy to a rush is to create a wall in the chokepoint of your initial region to prevent your opponent from reaching you (a.k.a. walling). This scenario simulates a Zealot rush and is designed to test whether the agent will be able to stop it (intuitively, it seems the only option is to build a wall). Some previous research has been done (Certický 2013; Richoux, Uriarte, and Ontañón 2014).

- **Map description:** It uses **Layout C** where player A starts like a regular game (a Command Center and 4 SCVs); and player B starts with 4 Zealots. Figure 3 shows an example of this scenario using the **Layout C**.
- **Enemy behavior:** Controls player B and it will wait 250 seconds to launch an attack to player A.
- **Termination condition:** Player A loses 25 units (at this point, you are not able to recover from a rush) or player B loses all his 4 Zealots.
- **Evaluation:** In this case we use *units lost* metric. More specifically: (Units player B lost / 4) - (units player A lost / 25).

S2: Strategy - Plan Recovery

An agent should adapt on plan failures. This scenario tests if the AI is able to recover from the opponent disrupting its build order.

- **Map description:** It uses **Layout C** where player A starts like a regular game (a Command Center and 4 SCVs); and player B does not have units.
- **Enemy behavior:** Once player A's Refinery is finished it will be destroyed after 1 second by a scripted event.
- **Termination condition:** When the destroyed building is replaced or a timeout of 400 seconds.
- **Evaluation:** *Time spent* to replace a building normalized by the timeout.

Experiments

In order to illustrate the advantages of the proposed benchmark, we used it to evaluate the performance of some of the state-of-the-art bots available: **FreScBot**, the winner bot of micromanagement tournament at AIIDE 2010; **UAlbertaBot**, winner of AIIDE 2013; **Skynet**, winner of AIIDE 2011, 2012, and CIG 2011, 2012, 2013; and **Nova**, with a strong kiting behavior. Table 1 shows the average score and standard deviation of each bot on each scenario after 10 executions. Not all bots can play all 3 races, and some bots cannot play micromanagement maps. When a bot cannot play a given scenario, we mark it with the N/A (Not Applicable) label.

For scenarios whose metric is normalized in the interval [-1,1], negative values mean that the bots were not able to complete the scenario. At first glance, we can see that most scores are negative, meaning that bots fail to succeed in completing most of the scenarios. We would like to point out a few interesting trends from the table:

- **Nova**'s strong point is kiting, and it is no surprise that it is the only bot that scores well in the kiting scenarios (RC1, RC2, and RC3). However, since **Nova** can only control the Terran race, it cannot be tested in RC2.
- Newer bots (**UAlbertaBot**, **Skynet**, **Nova**) significantly outperform the older bot **FreScBot** in micromanagement, even though **FreScBot** was the winner of the micromanagement tournament in 2010. While **UAlbertaBot** and **Skynet** improved the micromanagement of Protoss' units, **Nova** outperformed **FreScBot**'s micromanagement with Terran's units.
- Despite improved performance in micromanagement, none of the bots tested were able to pass any of the tactics or strategic scenarios. This is consistent with previous observations indicating that the biggest weakness of state-of-the-art StarCraft bots is their high-level reasoning skills (tactic and macro), and not their micromanagement skills.

Additionally, we would like to point out that the proposed benchmark is not to be seen as a way to assess the strength of a bot overall (the StarCraft AI competition is a better way

Table 1: Score achieved by four different bots on each scenario. The right-most column shows the range of values achievable in each scenario (higher is always better).

Scenario	*FreScBot*	*UAlbertaBot*	*Skynet*	*Nova*	*Range*
RC1-A-VZ	-0.1603 ± 0.0242	-0.2216 ± 0.0265	-0.2429 ± 0.0431	0.3335 ± 0.0517	[-1,1]
RC1-A-V6Z	-0.0314 ± 0.0118	-0.0684 ± 0.0077	-0.0828 ± 0.0096	0.0115 ± 0.0271	[-1,1]
RC1-A-3V6Z	-0.0924 ± 0.0327	-0.1115 ± 0.0260	-0.1124 ± 0.0275	0.0944 ± 0.0598	[-1,1]
RC1-A-1V9Zg	-0.0183 ± 0.0060	-0.0438 ± 0.0093	-0.0361 ± 0.0042	0.0093 ± 0.0376	[-1,1]
RC1-B-VZ	-0.1785 ± 0.0163	-0.1406 ± 0.0162	-0.1865 ± 0.0138	0.2892 ± 0.0139	[-1,1]
RC1-B-V6Z	-0.0588 ± 0.0021	-0.0442 ± 0.0016	-0.0636 ± 0.0053	0.0159 ± 0.0254	[-1,1]
RC1-B-3V6Z	-0.1178 ± 0.0219	-0.0885 ± 0.0148	-0.1090 ± 0.0080	0.1074 ± 0.0371	[-1,1]
RC1-B-1V9Zg	-0.0457 ± 0.0035	-0.0283 ± 0.0047	-0.0363 ± 0.0063	0.0282 ± 0.0297	[-1,1]
RC2-A-2D3H	-0.2343 ± 0.0772	0.0127 ± 0.1261	0.0506 ± 0.1641	N/A	[-1,1]
RC2-A-3D3Z	0.0036 ± 0.0757	0.0718 ± 0.1453	0.2887 ± 0.0345	N/A	[-1,1]
RC3	N/A	N/A	N/A	0.0335 ± 0.0065	[0,1]
RC4-A-5V	-0.0370 ± 0.0129	-0.0081 ± 0.0187	-0.0591 ± 0.0290	-0.0298 ± 0.0672	[-1,1]
RC4-A-12D	-0.0027 ± 0.0614	0.0115 ± 0.0348	0.1031 ± 0.0239	N/A	[-1,1]
RC4-A-9Z	-0.0231 ± 0.0217	-0.0117 ± 0.0344	0.0239 ± 0.0427	N/A	[-1,1]
RC4-A-12Mu	0.1042 ± 0.0717	0.2335 ± 0.0601	0.3273 ± 0.0858	N/A	[-1,1]
RC4-A-20Ma8Me	-0.0145 ± 0.0306	0.0424 ± 0.0351	-0.0024 ± 0.0004	-0.1276 ± 0.0173	[-1,1]
RC4-A-5Z8D	-0.0402 ± 0.0170	-0.0462 ± 0.0249	0.0313 ± 0.0281	N/A	[-1,1]
T1	N/A	N/A	N/A	0.0000 ± 0.0000	[0,1]
S1	N/A	-1.0000 ± 0.0000	N/A	-0.7420 ± 0.3920	[-1,1]
S2	N/A	0.0000 ± 0.0000	N/A	0.0000 ± 0.0000	[0,1]

to assess that), but as a way to delve deeper into the specific strengths and weaknesses of different bots, and a way to guide future work in the area.

Conclusions

In this paper we presented a set of scenarios to benchmark the performance of RTS IAs on different tasks that humans are able to solve with relative easy effort. We believe that providing this common set of problems will improve the evaluation process of future RTS AI techniques and agents. We also presented some quantitative metrics that can be used to evaluate the performance of each scenario.

As part of our future work, we would like to expand the set of scenarios (which is currently heavily biased toward micromanagement), specially trying to identify interesting tactical and strategic scenarios. We plan to maintain a public repository of the project and provide basic scripted AIs other than the built-in AI in order to expand the set of opponent behaviors in each scenario.

References

Blackadar, M., and Denzinger, J. 2011. Behavior learning-based testing of StarCraft competition entries. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2011).* AAAI Press.

Bowen, N.; Todd, J.; and Sukthankar, G. 2013. Adjutant bot: An evaluation of unit micromanagement tactics. In *Conference on Computational Intelligence and Games (CIG 2013)*, 1–8. IEEE.

Buro, M. 2003. Real-time strategy games: a new AI research challenge. In *International Joint Conference of Artificial Intelligence (IJCAI 2003)*, 1534–1535. Morgan Kaufmann Publishers Inc.

Certický, M. 2013. Implementing a wall-in building placement in StarCraft with declarative programming. *CoRR* abs/1306.4460.

Christensen, D.; Hansen, H. O.; Hernandez, J. P. C.; Juul-Jensen, L.; Kastaniegaard, K.; and Zeng, Y. 2012. A data-driven approach for resource gathering in real-time strategy games. In *Agents and Data Mining Interaction*, Lecture Notes in Computer Science. Springer. 304–315.

Churchill, D., and Buro, M. 2013. Portfolio greedy search and simulation for large-scale combat in StarCraft. In *Conference on Computational Intelligence and Games (CIG 2013)*, 1–8. IEEE.

Danielsiek, H.; Stür, R.; Thom, A.; Beume, N.; Naujoks, B.; and Preuss, M. 2008. Intelligent moving of groups in real-time strategy games. In *Symposium on Computational Intelligence and Games (CIG 2008)*, 71–78.

de Oliveira, C. F.; Goldbarg, E. F. G.; and Goldbarg, M. C. 2014. Bi-objective worker assignment in the bases of StarCraft. In *Encontro Nacional de Inteligncia Artificial e Computacional (ENIAC 2014).*

Hagelbäck, J. 2012. Potential-field based navigation in StarCraft. In *Conference on Computational Intelligence and Games (CIG 2012)*, 388–393. IEEE.

Iuhasz, G.; Negru, V.; and Zaharie, D. 2012. Neuroevolution based multi-agent system for micromanagement in real-time strategy games. In *Balkan Conference in Informatics (BCI 2012)*, 32–39. ACM.

Jaidee, U., and Muñoz-Avila, H. 2012. CLASSQ-L: A q-learning algorithm for adversarial real-time strategy games.

In *Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2012).* AAAI Press.

Justesen, N.; Tillman, B.; Togelius, J.; and Risi, S. 2014. Script- and cluster-based UCT for StarCraft. In *Conference on Computational Intelligence and Games (CIG 2014).* IEEE.

Kovarsky, A., and Buro, M. 2005. Heuristic search applied to abstract combat games. In *Conference of the Canadian Society for Computational Studies of Intelligence (Canadian AI 2005)*, volume 3501, 66–78. Springer.

Liu, S.; Louis, S. J.; and Ballinger, C. A. 2014. Evolving effective micro behaviors in RTS game. In *Conference on Computational Intelligence and Games (CIG 2014)*, 1–8. IEEE.

Muñoz-Avila, H.; Dannenhauer, D.; and Cox, M. T. 2015. Towards cognition-level goal reasoning for playing real-time strategy games. In *Goal Reasoning: Papers from the ACS Workshop*.

Nguyen, T. D.; Nguyen, K. Q.; and Thawonmas, R. 2015. Heuristic search exploiting non-additive and unit properties for RTS-game unit micromanagement. *Journal of Information Processing (JIP 2015)* 23(1):2–8.

Nguyen, K. Q.; Wang, Z.; and Thawonmas, R. 2013. Potential flows for controlling scout units in StarCraft. In *Conference on Computational Intelligence and Games (CIG 2013)*, 1–7. IEEE.

Ontañón, S.; Mishra, K.; Sugandh, N.; and Ram, A. 2010. On-line case-based planning. *Computational Intelligence* 26(1):84–119.

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. *Transactions on Computational Intelligence and AI in Games (TCIAIG)* 5(4):293–311.

Parra, R., and Garrido, L. 2013. Bayesian networks for micromanagement decision imitation in the RTS game StarCraft. In *Advances in Computational Intelligence*, volume 7630 of *Lecture Notes in Computer Science*. Springer. 433–443.

Richoux, F.; Uriarte, A.; and Ontañón, S. 2014. Walling in strategy games via constraint optimization. In *Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2014).* AAAI Press.

Shantia, A.; Begue, E.; and Wiering, M. 2011. Connectionist reinforcement learning for intelligent unit micro management in StarCraft. In *International Joint Conference on Neural Networks (IJCNN 2011)*, 1794–1801. IEEE.

Sturtevant, N. R. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games (TCIAIG)* 4(2):144–148.

Uriarte, A., and Ontañón, S. 2012. Kiting in RTS games using influence maps. In *Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2012).* AAAI Press.

van der Heijden, M.; Bakkes, S.; and Spronck, P. 2008. Dynamic formations in real-time strategy games. In *Symposium on Computational Intelligence and Games (CIG 2008)*, 47–54.

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

Young, J., and Hawes, N. 2014. Learning micromanagement skills in RTS games by imitating experts. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2014).* AAAI Press.

Young, J.; Smith, F.; Atkinson, C.; Poyner, K.; and Chothia, T. 2012. SCAIL: an integrated StarCraft AI system. In *Conference on Computational Intelligence and Games (CIG 2012)*, 438–445. IEEE.

Zhen, J. S., and Watson, I. D. 2013. Neuroevolution for micromanagement in the real-time strategy game StarCraft: Brood War. In *AI 2013: Advances in Artificial Intelligence - 26th Australasian Joint Conference*, volume 8272 of *Lecture Notes in Computer Science*, 259–270. Springer.