

The Current State of StarCraft AI Competitions and Bots

Michal Čertický

Department of Computer Science
Czech Technical University in Prague
certicky@agents.fel.cvut.cz

David Churchill

Department of Computer Science
Memorial University of Newfoundland
dave.churchill@gmail.com

Abstract

Real-Time Strategy (RTS) games have become an increasingly popular test-bed for modern artificial intelligence techniques. With this rise in popularity has come the creation of several annual competitions, in which AI agents (bots) play the full game of StarCraft: Broodwar by Blizzard Entertainment. The three major annual StarCraft AI Competitions are the Student StarCraft AI Tournament (SSCAIT), the Computational Intelligence in Games (CIG) competition, and the Artificial Intelligence and Interactive Digital Entertainment (AI-IDE) competition. In this paper we will give an overview of the current state of these competitions, and the bots that compete in them.

Introduction

Real-time Strategy (RTS) games are a genre of video games in which players manage economic and strategic tasks by gathering resources, building bases, increase their military power by researching new technologies and training units, and lead them into battle against their opponent(s). They serve as an interesting domain for Artificial Intelligence (AI) research and education, since they represent a well-defined, complex adversarial systems (Buro 2004) which pose a number of interesting AI challenges in the areas of planning, dealing with uncertainty, domain knowledge exploitation, task decomposition, spatial reasoning, and machine learning (Ontanón et al. 2013).

Unlike turn-based abstract board games like chess and go, which can already be played by AI at

super-human skill levels, RTS games are played in *real-time*, meaning the state of the game will continue to progress even if the player takes no action, and so actions must be decided in fractions of a second. In addition to that, individual turns in RTS games (game frames) can consist of issuing simultaneous actions to hundreds of units at any given time (Buro and Churchill 2012). This, together with their partially observable and non-deterministic nature, makes RTS game genre one of the hardest game AI challenges today, attracting the attention of the academic research community, as well as commercial companies. For example, Facebook AI Research, Microsoft, and Google DeepMind have all recently expressed interest in using the most popular RTS game of all time: Starcraft as a test environment for their AI research (Gibney 2016).

Meanwhile, the academic community has been using StarCraft as a domain for AI research since the advent of the Brood War Application Programming Interface (BWAPI) in 2009 (Heinermann 2013). BWAPI allows programs to interact with the game engine directly to play autonomously against human players or against other programs (bots). The introduction of BWAPI gave rise to numerous scientific publications over last 8 years, dealing with all kinds of sub-problems inherent to RTS games. A comprehensive overview can be found in (Churchill et al. 2016), (Ontanón et al. 2015) or (Ontanón et al. 2013).

In addition to AI research, StarCraft and BWAPI are often used for educational purposes as part of AI-related courses at universities, including UC Berkeley (US), Washington State University (US),

University of Alberta (CA), Comenius University (SK), Czech Technical University (CZ), University of Žilina (SK) and most recently Technical University Delft (NL), where a new course entitled “Multi-agent systems in StarCraft” has been opened for over 200 students. The educational potential of StarCraft has recently been extended even further, when Blizzard Entertainment released the game entirely for free in April 2017.

Widespread use of StarCraft in research and education has led to a creation of three annual StarCraft AI competitions existing until today. The first competition was organized at the University of California, Santa Cruz in 2010 as part of the AAAI Artificial Intelligence and Interactive Digital Entertainment (AIIDE) conference program. The following year gave rise to other two annual competitions – Student StarCraft AI Tournament (SSCAIT), organized as a standalone long-term event at Comenius University in Bratislava and Czech Technical University in Prague, and CIG StarCraft AI competition collocated with IEEE Computational Intelligence in Games (CIG) conference.

In this paper, we will talk about these three major StarCraft AI competitions and provide the latest updates on each of them, with the following 3 sections detailing the SSCAIT, AIIDE, and CIG Starcraft AI Competitions. We will also take a closer look at the state of the research and briefly describe current participants (bots) and AI methods they use.

SSCAIT: Student StarCraft AI Tournament

The Student StarCraft AI Tournament (SSCAIT) is the StarCraft AI competition with the highest number of total participants. It started as a part of an AI course at Comenius University, and initial seasons included several dozen student submissions from this course, in addition to submissions from across the globe. Since then, SSCAIT started accepting non-student participants and team submissions. There are three fundamental differences between SSCAIT and the remaining two competitions:

1. SSCAIT is an online-only event. Unlike AIIDE or CIG, it is not co-located with a scientific conference or any other real-world event.
2. There are two phases of SSCAIT each year:

a competitive *tournament phase*, lasting for up to three weeks and a *ladder phase* which runs for approximately eleven months each year. In other words, SSCAIT is live at all times with only a few short interruptions for maintenance.

3. Games are played one at a time and are publicly streamed live on Twitch.tv¹ and SmashCast.tv 24 hours a day. The AIIDE and CIG competitions instead play as many games as possible at maximum speed, with no public broadcast.

SSCAIT 2016-17 Updates & News

The activity of bot programmers and the general public surrounding SSCAIT has grown considerably over last several months – mainly thanks to a number of improvements of live stream and better community engagement during the Ladder phase, which was possible thanks to new members of the organizing team.

First, the ladder phase was updated, with SSCAIT introducing so-called “weekly reports”. Every weekend, there is a 1-2 hours long segment of curated AI vs. AI matches with insightful commentary on the live stream.

Second, the voting system was implemented, allowing bot programmers and viewers to select which bots will play the next ladder match on live stream (fig. 1). This not only supports viewer engagement, but also greatly simplifies bot debugging process. Bot programmers can now quickly test their newest updates against specific opponents. This change might have contributed to the significant increase in bot update frequency. Approximately 5-6 bots are updated every day, in contrast to 0-2 updates per week in 2015.

Another update was the introduction of “mini-tournaments” to SSCAIT. These are easily configurable, irregular and unofficial short competitions, taking up to one day. The format of these minitournaments and the selection of participants is usually up to the stream viewers and moderators.

Visual quality of the stream itself was improved by perfecting the custom observer script (Mattsson, Vajda, and Čertický 2015) which now moves the camera fluently to most interesting parts of the game in real time and displays SSCAIT-related information on top of the game. The stream was also

¹<http://www.twitch.tv/sscait>

Vote for a bot to play next:
Only 1 vote per bot and 2 votes per game are permitted. A bot may NOT play more than 3 times every 100 games










BOT NAME	VOTE	CURRENT VOTES	ALL TIME VOTES	ELO RATING*	SSCAIT RANK*	RACE
 Steamhammer		5	1208	2269	A-	
 Bereaver		3	868	2212	B	
 UPStarCraftAI 2016		0	103	1899	D	

Figure 1: A web-based interface allowing SSCAIT viewers and bot programmers to vote for the next ladder match.



Figure 2: SSCAIT live stream running in HD resolution and controlled by the custom observer script (Mattsson, Vajda, and Čertický 2015).

upgraded to run in HD resolution using a “resolution hack” (fig. 2).

Furthermore, two additional metrics were added to the ladder ranking system due to popular demand: ELO rating (Elo 1978), which is commonly used in adversarial games like chess, and “SSCAIT rank”, based on so-called “ICCUP ranking system”, typical for competitive StarCraft.

The overall number of stream views has increased to 376,920 views on Twitch.tv and additional 434,216 views on SmashCast.tv over the past 12 months. The current number of active bots on the ladder is 70.

Tournament Phase Updates

The 2016/17 installment of SSCAIT’s tournament

phase took place during three weeks at the end of December 2016 and beginning of January 2017 and sported 45 participants. The tournament was divided into two divisions:

Student Division: Round Robin tournament of 1980 games, where every bot played two games against every opponent. Only the bots created by a single participant, who was a student at that time, were considered “student” bots and were eligible for victory in this division. Other bots were tagged as “mixed-division” bots (they played the games, but could not win the student division title). Winners of the student division in 2016/17 were:

1. LetaBot (Martin Rooijackers), University of Maastricht (Netherlands) with 82 wins
2. Wulibot, University of Southern California (USA) with 63 wins
3. Zia Bot, Ulsan National Institute of Science and Technology (South Korea) with 54 wins

The student division of SSCAIT exists so that the students stand a chance of winning in the presence of more experienced, non-student participants and team-created bots.

Mixed Division: After the student division ended, sixteen bots with the most wins among all the participants were selected for the additional mixed division elimination bracket. Since two bots (Steamhammer and Zia bot) were tied for 16th place, both with 54 wins, additional tie-breaker match was scheduled to decide who gets 16th position in the mixed division bracket.

The following 16 bots played in the mixed division elimination bracket: *LetaBot*, *Steamhammer*, *Overkill*, *WuliBot*, *ZZZbot*, *UAlbertaBot*, *IronBot*, *IceBot*, *Bereaver*, *BeeBot*, *XIMP*, *Skynet*, *KrasiObot*, *Flash.AI*, *KillerBot* and *tscmoo*. First round (Ro16) matches consisted of only a single game, but Ro8 and Ro4 were played as “best of 3” matches. Finals were played as “best of 5”.

Interestingly, *LetaBot* (created by Martin Rooijackers) managed to win the mixed division elimination bracket, in addition to winning the student division, after beating *KrasiObot* (created by Krasimir Krystev) 3 to 1 in the finals (fig. 3). *LetaBot* had hard-coded special strategies against specific opponents. It successfully executed an “SCV rush” in the final games – a strategy for

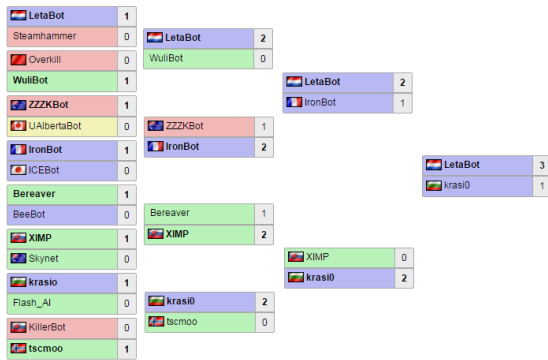


Figure 3: SSSCAIT 2016/17 mixed division elimination bracket.

which *Krasi0bot* was not prepared.

All the elimination bracket games were published as videos with commentary on SSSCAIT YouTube channel² and as replay files on SSSCAIT website.³ Tournament winner, Martin Rooijackers, has recently been invited to have a talk about this achievement at a business-oriented AI conference *World Summit AI*.⁴

AIIDE: Artificial Intelligence and Interactive Digital Entertainment

The AIIDE Starcraft AI Competition is the longest running annual Starcraft competition, and has been held every year since 2010 along with the AAI Artificial Intelligence and Interactive Digital Entertainment conference. Unlike the CIG and SSSCAIT competitions, the AIIDE competition requires that all bots be open source, and that their source code will be published for public download after the competition has finished. Running 24 hours a day for 2 weeks with games played at super-human speed, the competition is a single round-robin format with the winner being the bot with the highest win percentage when the time limit has been reached.

²<http://youtube.com/playlist?list=PLS6Qj916df6K9z3dOLQhCS0KatlvBqhbE>

³<http://sscaitournament.com/index.php?action=2016>

⁴<http://worldsummit.ai/>

AIIDE 2016-17 Updates & News

The 2016 AIIDE competition had a total of 21 competitors, and the round robin games ran on 12 machines for nearly two weeks. A total of 90 round robin rounds were completed, with each bot playing 1800 games. No new rules or maps were used for the 2016 tournament that were not in place for the 2015 tournament. As the AIIDE competition was held shortly after the CIG competition, many of the submissions were the same, which is reflected in the results of both competitions. The top four finishers can be seen in Figure 4, with Iron placing 1st, ZZZKbot placing 2nd, and tscmoo coming in 3rd place.

Bot	Games	Win	Loss	Win %
Iron	1800	1574	226	87.44
ZZZKBot	1799	1530	269	85.05
tscmoo	1799	1488	311	82.71
LetaBot	1796	1329	467	74

Figure 4: Results of the top 4 finishers in the 2016 AIIDE competition. Iron and LetaBot are Terran bots, while ZZZKbot and tscmoo played as Zerg.

In 2016, the AIIDE competition website was updated to include an archive⁵ of data and results of all of the annual AIIDE and CIG competitions, including final results, bot source code and binary download links, and information about each bot.

The 2017 AIIDE competition will have several updates:

- An updated map pool consisting of 10 new maps.
- Updated tournament managing software capable of playing more games in the same period of time.
- Support for BWAPI version 4.2.0
- Bots that achieved a 30% or higher win rate in the 2016 competition will be carried forward to 2017
- Plans to support GPU computation for bots

⁵<http://www.cs.mun.ca/~dchurchill/starcraftaicomp/archive.shtml>

CIG: Computational Intelligence in Games

The CIG StarCraft RTS AI Competition is a part of the program of the IEEE Computational Intelligence in Games (CIG) conference since August 2010. Similarly to AIIDE competition, all the bot games are run separately on several computers prior to the conference and the results are then announced during the event. Unlike in AIIDE and SSCAIT, the map pool of the CIG competition is not known in advance for the competitors. Also, the CIG competition no longer enforces an open source code requirement to compete.

CIG 2016-17 Updates & News

CIG 2017 is scheduled to happen in New York City, USA on late August 2017, which is just after the submission deadline of this paper. Therefore, we will report on CIG 2016, which happened on September 2016.

The 2016 installment of CIG competition hosted 16 participants, out of which 9 were new or updated bots and 7 were re-entries from previous year. CIG 2016 competition was divided into two stages: The *Qualifying* stage and the *Final* stage. The first qualifying stage consisted of Round Robin tournament between all 16 participating bots. In total, 11988 games were played in this stage. After the qualifying stage, best 8 bots were selected to proceed to the final stage: *tscmoo*, *IronBot*, *LetaBot*, *ZZZbot*, *Overkill*, *UAlbertaBot*, *MegaBot* and *Aiur*.

All the persistent files accumulated by the bots in the qualifying stage were deleted before entering the finals. This stage consisted of 2799 Round Robin games between the 8 bots. All the games ran on 17 computers for 8 days. The winner of the final stage, *tscmoo* bot created by Vegard Mella from Norway, was announced the overall winner of CIG 2016 tournament with 456 wins and 65.14% win rate in the final stage. Detailed results are depicted in Figure 5.

Current StarCraft Bots

Over the years, StarCraft AI competitions have motivated many individuals and groups to combine and integrate various AI techniques and methods into complete bots, capable of playing complete StarCraft 1v1 games.

Bot	Win %	tscm	Iron	Leta	ZZZK	Over	UAlb	Mega	Aiur
tscmoo	65.14	-	52/100	44/100	79/100	71/100	77/100	83/100	50/100
Iron	54.43	48/100	-	38/100	49/100	49/100	74/100	30/100	93/100
LetaBot	53.71	56/100	62/100	-	49/100	81/100	69/100	30/100	29/100
ZZZKBot	53.08	21/100	51/100	51/100	-	42/100	35/99	93/100	78/100
Overkill	51.43	29/100	51/100	19/100	58/100	-	43/100	81/100	79/100
UAlbertaBot	49.07	23/100	26/100	31/100	64/99	57/100	-	76/100	66/100
MegaBot	38	17/100	70/100	70/100	07/100	19/100	24/100	-	59/100
Aiur	35.14	50/100	07/100	71/100	22/100	21/100	34/100	41/100	-

Figure 5: Detailed results of the CIG 2016 competition final stage.

In this section, we provide an overview of a selection of bots (in alphabetical order) and discuss some of the AI approaches they implement. We only mention those bots that are currently active in one of the competitions, have recently been updated, and employ some more complex AI techniques (we do not mention simple hard-coded or rule-based bots).

- *Allien*: Allien is a relatively new Zerg bot. Various kinds of its decisions rely on “scoring” systems. Other than that, it employs state machines – especially for higher-level strategy and macro decisions.
- *GarmBot*: GarmBot is organized as a multi-agent system using the blackboard architecture. Every unit is controlled by a single agent, implemented as a state machine.
- *Ian Nicholas DaCosta*: This Protoss bot uses genetic algorithms for targeting, and detecting enemy army threat levels. Supervised learning was applied for the detection of opponent’s strategies and builds.
- *KaonBot*: For resource and unit allocation based on prioritized needs, KaonBot applies a competing priority algorithm. According to the author, the bot will learn these priorities from experience in future releases.
- *Krasi0bot*: Krasi0bot has been around for many years, but it is still being actively developed. Even though it originally started as a rule-based bot, it currently makes some use of genetic algorithms, neural networks and potential fields. The author also actively experiments with various other techniques at the moment.
- *LetaBot*: The most interesting technique used by Martin Rooijackers’ LetaBot is Monte Carlo

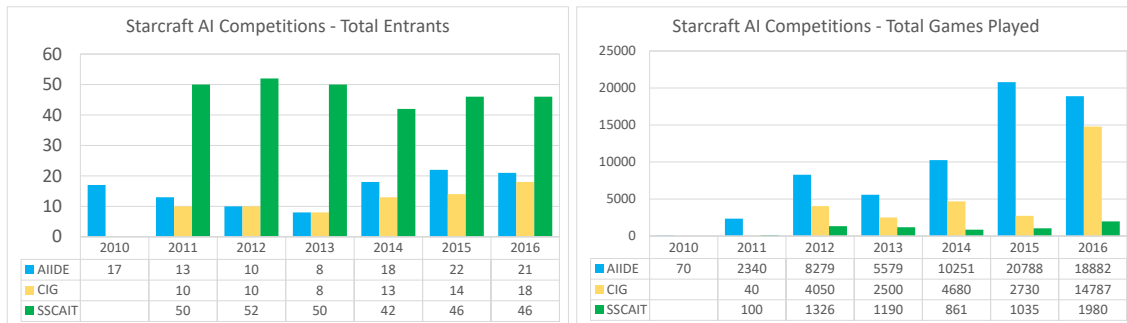


Figure 6: Statistics for each of the 3 major annual StarCraft AI Competitions: AIIDE, CIG, and SSCAIT, since the first competition in 2010. Shown on the left is the number of total entrants for each competition, and on the right are the total number of games played in each competition.

Tree Search (MCTS) algorithm, which is used to plan the movement of squads (groups of units) around the map. A similar approach has previously been used by the author of Nova bot, Alberto Uriarte (Uriarte and Ontanón 2014). An implementation MCTS algorithm for squad movement has also been very recently released in form of ready-to-use library StarAlgo.⁶ In addition to MCTS, LetaBot employs cooperative pathfinding for resource gathering and text mining to extract build orders directly from Liquipedia articles.⁷

- *MegaBot*: For every game, MegaBot (Tavares et al. 2016) chooses one of three approaches, each of which is implemented as a different bot (Skynet, Xelnaga or NUSBot). Algorithm selection is modeled as a multi-armed bandit. At the beginning of the game, an algorithm is selected using epsilon-greedy strategy. After the game, the reward is perceived (+1, 0, -1 for victory, draw and loss, respectively) and the value of the selected algorithm is updated via an incremental version of recency-weighted exponential average (Q-learning update rule).
- *Monica / Maria / Brenda*: Zerg, Protoss and Terran bots employing a game simulation inside the BEAM Erlang/OTP VM. It uses TorchCraft (Synnaeve et al. 2016) – a library for machine learning research on RTS games. The unit logic is written in Lua.

⁶<http://github.com/vnikk/StarAlgo>

⁷<http://wiki.teamliquid.net/starcraft>

- *PurpleWave*: The decision making of PurpleWave bot is mainly based on hierarchical task networks. For the micromanagement, it uses a hybrid squad/multi-agent approach and nearest neighbors clustering. The bot then simulates the outcomes of battles and suggests tactics for the squads by min-maxing tactical approaches by each side (e.g. “charge in”, “run away”, or “fight with workers”). In the end, each unit takes the tactical suggestion under advisement, but behaves independently. The units choose between approximately two dozen simple, reusable stateless behaviors. Uses heuristics including potential fields for the movement.
- *StarcraftGP*: StarcraftGP is the first StarCraft meta-bot – a program that autonomously creates a program that autonomously plays StarCraft (García-Sánchez et al. 2015). Currently, StarcraftGP v0.1 is using (Linear) Genetic Programming and it is able to directly write C++ code. Its first creations, namely Salsa and Tequila, have been the first bots not written by a human to participate in international competitions.
- *Steamhammer / Randomhammer*: Zerg bot Steamhammer and its random-race version Randomhammer both employ sophisticated combat simulation with alpha-beta search and portfolio search to predict the outcome of battles. The bots also use hierarchical reactive control for the units. For Protoss and Terran production, Randomhammer uses branch-and-bound search, while Zerg production is currently rule-based.

- *tscmoo*: tscmoo uses no external libraries: it has its own combat simulation code to predict the outcome of battles (while others typically use the SparCraft combat simulation package⁸), it does not use BWTA⁹ to analyze the terrain and it even has its own threat-aware pathfinding for individual units. The bot can use many different strategies and selects among them based on their success in previous games. Recent versions of the bot experimented with recurrent neural networks for high-level strategy and build order decisions.
- *Václav Bayer*: Q-learning / Reinforcement Learning and Markov Decision Processes are used by this bot – mainly to select the best build order with respect to opponent’s strategy.
- *Zia Bot*: Despite the fact that most of Zia Bot’s functionality is heuristic and rule-based, the bot tries to recognize and remember all the strategies used by its opponents and by itself. This memory is used to select the better performing game plans in the following games.

Conclusion

In this paper we have given an overview of the 3 major annual StarCraft AI competitions, as well as some of the top performing bots that compete in them. As seen in Figure 6, each year, participation in these competitions has continued to rise, as well as the number of games played between bots in the competitions. In the past 2-3 years, the bots in these competitions have become more strategically complex and functionally robust, employing a range of state-of-the-art AI techniques from the fields of heuristic search, machine learning, neural networks, and reinforcement learning. While the bots currently play at an amateur human level, we hope that advancing RTS AI techniques coupled with the recent involvement of industry research, they will be able to compete with human experts in the next few years.

References

Buro, M., and Churchill, D. 2012. Real-time strategy game competitions. *AI Magazine* 33(3):106.

⁸<http://github.com/davechurchill/ualbertabot/wiki/SparCraft-Home>

⁹<http://bitbucket.org/auriarte/bwta2>

Buro, M. 2004. Call for AI research in RTS games. In *Proceedings of the 4th Workshop on Challenges in Game AI*, 139–142.

Churchill, D.; Preuss, M.; Richoux, F.; Synnaeve, G.; Uriarte, A.; Ontanón, S.; and Certický, M. 2016. Starcraft bots and competitions.

Elo, A. E. 1978. *The rating of chessplayers, past and present*. Arco Pub.

Garía-Sánchez, P.; Tonda, A.; Mora, A. M.; Squillero, G.; and Merelo, J. 2015. Towards automatic starcraft strategy generation using genetic programming. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, 284–291. IEEE.

Gibney, E. 2016. What google’s winning go algorithm will do next. *Nature* 531(7594):284–285.

Heinermann, A. 2013. Broodwar API. <https://github.com/bwapi/bwapi>.

Mattsson, B. P.; Vajda, T.; and Čertický, M. 2015. Automatic observer script for StarCraft: Brood War bot games (technical report). *arXiv preprint arXiv:1505.00278*.

Ontanón, 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* 5:293–311.

Ontanón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; and Preuss, M. 2015. RTS AI: Problems and Techniques.

Synnaeve, G.; Nardelli, N.; Auvolat, A.; Chintala, S.; Lacroix, T.; Lin, Z.; Richoux, F.; and Usunier, N. 2016. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*.

Tavares, A.; Azpúrua, H.; Santos, A.; and Chaimowicz, L. 2016. Rock, paper, starcraft: Strategy selection in real-time strategy games. In *12th Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 93–99.

Uriarte, A., and Ontanón, S. 2014. High-level representations for game-tree search in rts games. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.