StarCraft Bots and Competitions

David Churchill[1], Mike Preuss[2], Florian Richoux[3], Gabriel Synnaeve[4], Alberto Uriarte[5], Santiago Ontañón[5], and Michal Čertický[6]

[1] Computing Science Department of the University of Alberta, Edmonton, Canada.
cdavid@cs.ualberta.ca
[2] Information Systems and Statistics, Westf. Wilhelmsuniversität Münster, Germany.
mike.preuss@uni-muenster.de
[3] Nantes Atlantic Computer Science Laboratory (LINA) of the Université de Nantes, France.
florian.richoux@univ-nantes.fr
[4] Cognitive Science and Psycholinguistics (LSCP) of ENS Ulm, Paris, France.
gabriel.synnaeve@gmail.com
[5] Computer Science Department at Drexel University, Philadelphia, PA, USA.
{santi,albertouri}@cs.drexel.edu
[6] Agent Technology Center at Czech Technical University in Prague, Czech Republic.
certicky@agents.fel.cvut.cz

Synonyms

Real-Time Strategy games; RTS games; StarCraft; Artificial Intelligence; AI; Game AI; Competition

Definition

Real-Time Strategy (RTS) games is a sub-genre of strategy games where players need to build an economy (gathering resources and building a base) and military power (training units and researching technologies) in order to defeat their opponents (destroying their army and base). Artificial Intelligence (AI) problems related to RTS games deal with the behavior of an artificial player. Since 2010, many international competitions have been organized to match AIs, or bots, playing to the RTS game StarCraft. This chapter presents a review of all major international competitions from 2010 until 2015, and details some competing StarCraft bots.

State of the Art Bots for StarCraft

Thanks to the recent organization of international game AI competitions focused around the popular StarCraft game, several groups have been working on integrating many of the techniques developed for RTS game AI into complete "bots", capable of playing complete StarCraft games. In this chapter we will overview some of the currently available top bots, and their results of recent competitions.

Playing an RTS game involves dealing with a wide variety of problems, ranging from miro-management problems such as unit control, to macro-management problems such as resource allocation. A few approaches, like CAT [1], Darmok [7] or ALisp [5] try to deal with the problem in a monolithic manner, by using a single AI technique. However, none of those systems aims at achieving near human performance. In order to achieve human-level performance, RTS AI designers use a lot of domain knowledge in order to divide the task of playing the game into a collection of sub-problems, which can be dealt-with using individual AI techniques.

Figure 1 shows some representative examples of the architectures used by different bots in the AIIDE and CIG StarCraft AI competitions (see Section 1): BroodwarBotQ [8], Nova [9], UAlbertaBot [3], Skynet, SPAR, AIUR, and BTHAI [4]. Each box represents an individual module with a clearly defined task (only modules with a black background can send actions directly to StarCraft). Dashed arrows represent data flow, and solid arrows represent control (when a module can command another module to perform some task). For example, we can see how SPAR is divided in two sets of modules: *intelligence* and *decision making*. Intelligence in SPAR has three modules dedicated to analyze the current situation of the game. Decision making in SPAR is done through four hierarchically organized modules, with the higher-level module (*strategic decision*) issuing commands to the next module (*tactical decision*), which sends commands to the next module (*action implementation*), and so on. Only the two lower-level modules can send actions directly to StarCraft.

On the other hand, bots such as Nova or BroodwarBotQ (BBQ) only use a hierarchical organization for *combat* (controlling the attack units), but use a decentralized organization for the rest of the bot. In Nova and BBQ, there is a collection of modules that control different aspects of the game (workers, production, construction, etc.). These modules can all send actions directly to StarCraft. In Nova those modules coordinate mostly through writing data in a shared blackboard, and in BBQ they coordinate only when they have to use a shared resource (unit) by means of an arbitrator: a bidding market and broker for settling units control, military and civilian groups/task forces bid for units proportionally to their usefulness and the task importance.

By analyzing the structure of these bots, we can see that there are two main tools being used in these integration architectures:

- *Abstraction*: complex tasks can be formulated at different levels of abstraction. For example, playing an RTS game can be seen as issuing individual low-level actions to each of the units in the game, or at a higher level, it can be seen as deploying a specific strategy (e.g. a "BBS strategy", or a "Reaver Drop" strategy). Some bots, reason at multiple levels of abstraction at the same time, making the task of playing StarCraft simpler. Assuming that each module in the architecture of a bot has a goal and determines some actions to achieve that goal, the actions determined by higher-level modules are considered as the goals of the lower level modules. In this way,

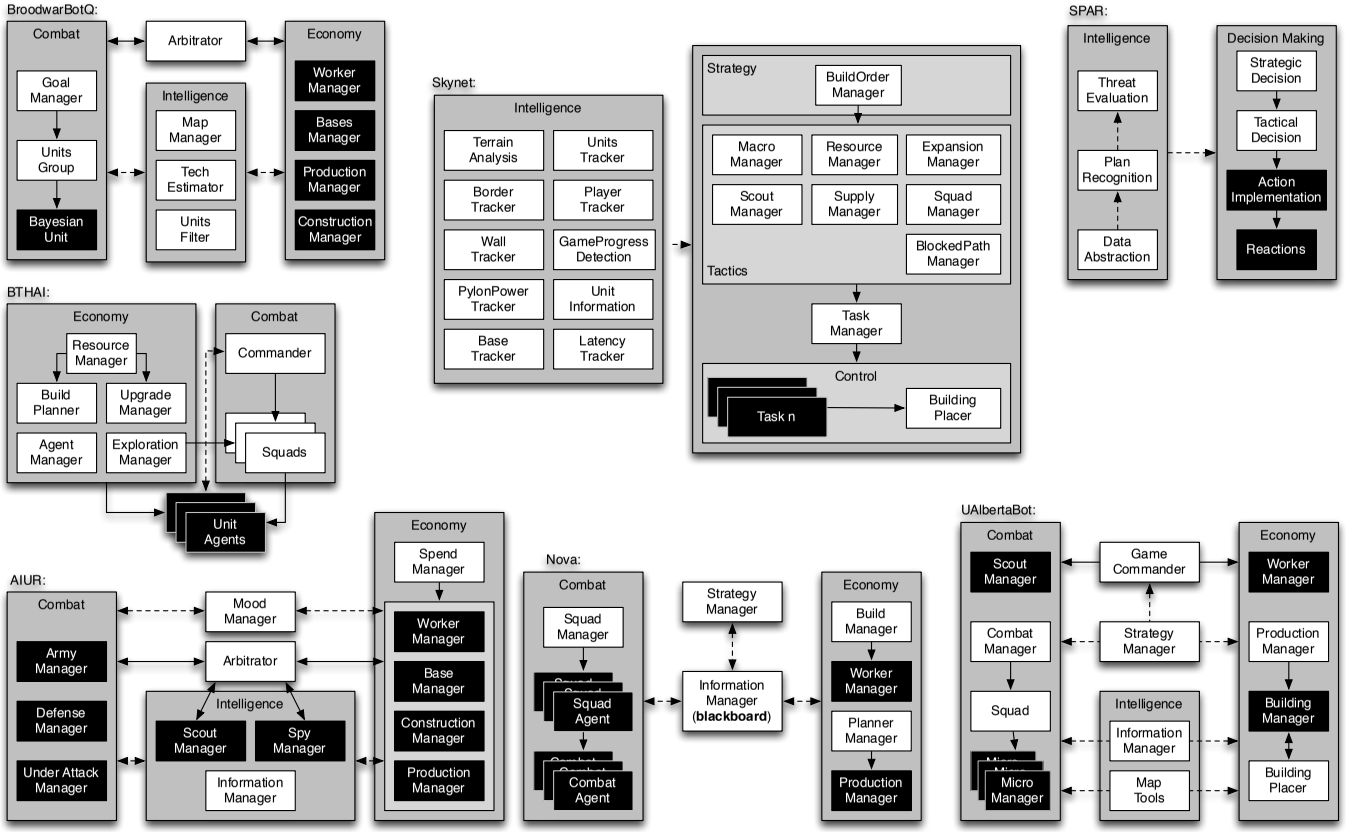

Fig. 1. Architecture of 7 StarCraft bots obtained by analyzing their source code. Modules with black background sent commands directly to StarCraft, dashed arrows represent data flow, and solid arrows represent control.

each module can focus on reasoning at only one level of abstraction, thus, making the problem easier.
- *Divide-and-conquer*: playing a complex RTS, such as StarCraft, requires performing many conceptually different tasks, such as gathering resources, attacking, placing buildings, etc. Assuming each of these tasks can be performed relatively independently and without interference, we can have one module focusing on each of the tasks independently, thus making the problem easier.

If we imagine the different tasks to perform in a complex RTS game in a two-dimensional plane, where the vertical axis represents abstraction, and the horizontal axis represents the different aspects of the game (combat, resource gathering, etc.), abstraction can be seen as dividing the space with horizontal lines, whereas divide-and-conquer divides the space using vertical lines.

Different bots use different combinations of these two tools. Looking back at Figure 1, we can see the following use of abstraction and divide-in-conquer in the bots:

- BroodwarBotQ[7]: uses abstraction for *combat*, and divide-and-conquer for *economy* and *intelligence gathering*. To avoid conflicts between modules (since the individual tasks of each of the modules are not completely independent), BBQ uses an arbitrator.
- Nova[8]: is similar in design as BroodwarBotQ, and uses abstraction for *combat*, and divide-and-conquer for *economy*. The differences are that Nova does not have an arbitrator to resolve conflicts, but has a higher-level module (*strategy manager*), which posts information to the blackboard that the rest of modules follow (thus, making use of abstraction).
- UAlbertaBot[9]: also uses abstraction in *combat* like the previous two bots. But it also uses it in *economy*: as can be seen, the production manager sends commands to the building manager, who is in charge of producing the buildings. This bot also uses divide-and-conquer, and tasks like scouting and resource gathering are managed by separate, independent modules.
- Skynet[10]: makes extensive use of both abstraction and divide-and-conquer. We can see a high level module that issues commands to a series of tactics modules. The collection of tactic modules queue *tasks* (that are analogous to the abstract actions used in SPAR). Each different task has a specific low level module that knows how to execute it. Thus, Skynet uses a 3 layered abstraction hierarchy, and uses divide-and-conquer in all levels except the highest.
- SPAR[11]: only uses abstraction. Its high-level module determines the strategy to use, and the tactical decision module divides it into a collection of *abstract actions*, that are executed by the lower-level modules.

[7] http://github.com/SnippyHolloW/BroodwarBotQ
[8] http://nova.wolfwork.com/
[9] http://github.com/davechurchill/ualbertabot/
[10] http://code.google.com/p/skynetbot/
[11] http://www.planiart.usherbrooke.ca/projects/spar/

– AIUR[12]: is mainly divide-and-conquer oriented, with a slight abstraction on *economy* due to a SpendManager deciding how to spend and share resources among Base, Production and Construction Managers. At the beginning of a game, the MoodManager initializes a "mood" which will influence both tactics and strategy. *Combat* is divided into three independent managers: the *Defense Manager*, controlling military units when there is nothing special, the *Under Attack Manager*, activated when the opponent is attacking our bases, and the *Army Manager*, taking control of units when it is time to attack, following a timing given by the current mood. This bot does not manage however any kind of reactive controls so far.
– BTHAI[13]: uses a two-tier abstraction hierarchy, where a collection of high-level modules command a collection of lower-level agents in charge of each of the units. At the high-level, BTHAI uses divide-and-conquer, having multiple high-level modules issuing commands to the lower-level units.

Additionally, except for BTHAI, all other agents use divide-and-conquer at a higher-level bot design and divide all the modules into two or three categories: *intelligence gathering* and *decision making* (sometimes divided into *combat* and *economy*).

Some bots using divide-and-conquer, assume that each of the modules can act independently and that their actions can be executed without interference. BBQ, UAlbertaBot and AIUR, however use an arbitrator (*Game Commander* in UAlbertaBot) that makes sure that modules do not send contradictory orders to the same unit. However, very little bots handle the problem of how to coordinate resource usage amongst modules, for instance BTHAI uses a first-come-first-serve policy for spending resources, the first module that requests resources is the one that gets them. Nova and Skynet are exceptions, and implement some rudimentary prioritization based on the high level strategy. Following available resources and timing, AIUR's *Spend Manager* orders Base, Production and Construction Managers what they have to build/produce. It also orders to start tech research and upgrades. The idea here is not to let the different managers allocate the resources they want, but to do the opposite, that is: finding how the AI can spend the available money.

One interesting aspect of the seven bots described above is that, while all of them (except AIUR) are reactive at the lower level (reactive control), most if not all of them, are scripted at the highest level of abstraction. BTHAI reads build and squad formations from a predefined script, Nova's *Strategy Manager* is a predefined finite-state machine, BBQ's construction manager reads the build order from a predefined script, and Skynet's *BuildOrder Manager* is basically a predefined script. Such scripts describe the strategy that the bots will use, however, such strategy is always fixed. One could see this pre-scripting as if each bot defined a "high-level programming language" to describe StarCraft strategies, and the bots themselves are just interpreters of such strategy. Compared to

[12] https://github.com/AIUR-group/AIUR
[13] https://github.com/jhagelback/opprimobot

current approaches for Chess or Go, this scripting seems a rigid and inflexible, but responds to the much higher complexity of the StarCraft game. An interesting exception to that is UAlbertaBot, which uses a search algorithm in the *Production Manager* to find near-optimal build orders. Another interesting case is AIUR, that uses a *Mood Manager* to randomly pick a mood among six (cheese, rush, aggressive, defensive, macro, fast expand), which will influence the build order, strategy and tactics.

In conclusion, we can see that there are two basic tools that can be used in an integration architecture: abstraction and divide-and-conquer, which are widely used by the existing StarCraft bots. For space reasons, we do not include an exhaustive comparison of the architectures of all the participating bots. Some more bots than the ones described here have been documented by their authors, such as SCAIL [11] or QUORUM [10]. Let us now focus on their performance in recent competitions.

Recent StarCraft AI Competitions

This section reviews the results of three recent international competitions on AI for StarCraft. These competitions have been possible thanks to the existence of the Brood War Application Programming Interface (BWAPI)[14], which enables replacing the human player interface with C++ code. The following subsections summarize the results of StarCraft AI competitions co-located with two scientific conferences – AIIDE (Artificial Intelligence for Interactive Digital Entertainment) and CIG (Computational Intelligence in Games) – and one standalone competition called SSCAIT (Student StarCraft AI Tournament), during the past years. Since 2011, the CIG and AIIDE competitions have typically been held in August / September in each year, as the conferences are scheduled quite close to each other, and SSCAIT have been held in December / January. As a consequence of this, AIIDE and CIG competitions share a big portion of the entrants.

AIIDE

AIIDE 2010. Started in 2010, the AIIDE StarCraft AI Competition[15] is the most well known and longest running StarCraft AI Competition in the world. Each year, AI bots are submitted by competitors to do battle within the retail version of StarCraft: Brood War, with prizes supplied by Blizzard Entertainment.

The first competition in 2010 was organized and run by Ben Weber in the Expressive Intelligence Studio at University of California, Santa Cruz[16]. 26 total submissions were received from around the world. As this was the first year of the competition, and little infrastructure had been created, each game of the tournament was run manually on two laptop computers and monitored by hand

[14] https://github.com/bwapi/bwapi
[15] http://www.StarCraftAICompetition.com
[16] http://eis.ucsc.edu/StarCraftAICompetition

Table 1. Ranking of the three best bots of the AIIDE 2010 competition

Rank	Bot
1	Overmind
2	Krasi0
3	Chronos

to record the results. Also, no persistent data was kept for bots to learn about opponents between matches.

The 2010 competition had 4 different tournament categories in which to compete. Tournament 1 was a flat-terrain unit micro-management battle consisting of four separate unit composition games. Of the six competitors, FreSCBot won the competition with Sherbrooke coming in 2nd place. Tournament 2 was another micro-focused game with non-trivial terrain. Two competitors submitted for this category, with FreSCBot once again coming in 1st by beating Sherbrooke.

Tournament 3 was a tech-limited StarCraft game on a single known map with no fog-of-war enforced. Players were only allowed to choose the Protoss race, with no late game units allowed. 8 bots faced off in this double-elimination tournament with MimicBot taking first place over Botnik in the final. As this was a perfect information variant of StarCraft, MimicBot adopted a strategy of "mimic its opponent's build order, gaining an economic advantage whenever possible" which worked quite well.

Tournament 4 was the complete game of *StarCraft: Brood War* with fog-of-war enforced. The tournament was run with a random pairing double-elimination format with each match being best of 5 games. Competitors could play as any of the three races, with the only limitations in gameplay being those that were considered "cheating" in the StarCraft community. A map pool of 5 well-known professional maps were announced to competitors in advance, with a random map being chosen for each game.

Results are shown in Table 1. The team that won was Overmind[17], from University of California, Berkeley. Using the Zerg race, their strategy was to defend early aggression with zergling units while amassing mutalisk units, which they used to contain and eventually defeat their opponents. The mutalisk is a very fast and agile flying unit which is able to attack while moving with no drawback, which makes them quite a powerful unit when controlled by a computer. Overmind used a potential-field based micro-management system to guide their mutalisks, which led them to victory. Krasi0 came in 2nd place with a standard defensive Terran opening strategy that transitioned into "mech" play in the late game.

AIIDE 2011. In 2011 the University of Alberta hosted the competition, with organization by Michael Buro and David Churchill[18]. Due to a lack of entrants in

[17] http://overmind.cs.berkeley.edu
[18] https://skatgame.net/mburo/sc2011/

tournament categories 1-3 in the 2010 competition, it was decided that only the full game category would be played in the 2011 competition, with 13 entrants. Another important change in the 2011 competition was the introduction of automated tournament-managing software running StarCraft games simultaneously on 20 computers, allowing a total of 2340 games to be played in the five days that the tournament ran. This increase in games played also allowed the tournament to switch to a round-robin format, eliminating the "luck" factor of the pairings inherent in bracket style tournaments. The bot that achieved the highest win percentage over the course of the competition would be determined the winner. Also, the competition became open-source, in an effort to prevent possible cheating and to promote healthy competition in future tournaments by giving newcomers and easier entry point by basing their design off of previous bots. All maps used by the competition were known long time by advance, so participants can test their bot on them. These 10 maps remains the same for all AIIDE competition, from 2011 until the last one at the time this chapter has been written in 2015. Maps are *(2)Benzene*, *(2)Destination*, *(2)HeartbreakRidge*, *(3)Aztec*, *(3)TauCross*, *(4)Andromeda*, *(4)CircuitBreaker*, *(4)EmpireoftheSun*, *(4)Fortress* and *(4)Python*. The number between parenthesis is the number of possible start points.

In the end, Skynet won the competition with its solid Protoss play (results are summarized in Table 2). The bot executed one of a small set of strategies randomly at the start of the match based on the map and the race of the opponent. Skynet would then amass a medium to large sized army and expand before moving out to attack. Good use of Dragoon (powerful ranged ground unit with clumsy movement) range and kiting micro-management allowed it to hold off the early aggression of other bots such as UAlbertaBot, which came in 2nd.

UAlbertaBot used an early zealot-rush strategy to take advantage of the power of early game Protoss units. It would send out the first zealots that were made and immediately attack the enemy base, using a unit counting heuristic to determine whether or retreat or keep pushing. Of note is that UAlbertaBot used an online planning algorithm to construct all of its economic build-orders[3], as no hard-coded build orders were used.

AIUR also chose Protoss, with a strategy that was in between Skynet and UAlbertaBot in terms of attack timings. At that time, AIUR chose one mood among five (leading to slightly different strategies and tactics) at the beginning of a game and kept it until the end. These five moods were: 1. Rush: the bot tries

Table 2. Results of the five best bots of the AIIDE 2011 competition

Rank	**Bot**	**Win %**
1	Skynet	88.9%
2	UAlbertaBot	79.4%
3	AIUR	70.3%
4	ItayUndermind	65.8%
5	EISBot	60.6%

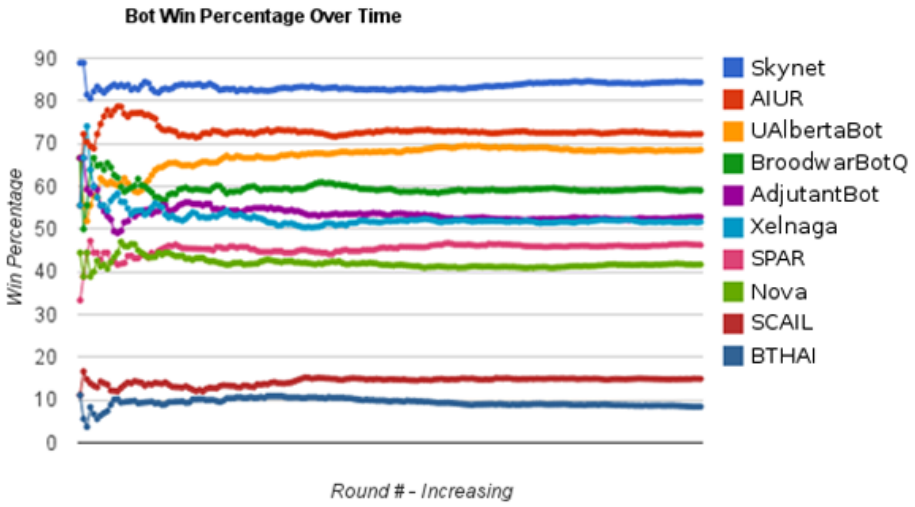

Fig. 2. Evolution of the win percentage of each bot participating in the AIIDE 2012 competition

early attacks, and have good probabilities to send the two or three first Zealots (basic contact attack ground unit) to harass the opponent. 2. Aggressive: it has less chance to perform harasses with the first Zealots, and the first attack is usually a bit delayed with regard to the Rush mood. 3. Macro: the AI do not try any early attacks and focus a bit more on its economy before attacking. 4. Defense: AIUR "turtles" and wait to have a consequent army before running an attack. 5. Fast expand: the first building constructed it a base expansion, for a very economical-oriented game. Notice that build orders are not fully hard-coded since they can be altered by AIUR's Spend Manager.

Of note in these results was that a rock-paper-scissors effect happened among the top 3 finishers. Of the 30 rounds, Skynet beat UAlbertaBot 26 times, UAlbertaBot beat AIUR 29 times, and AIUR beat Skynet 19 times. Another notable result is that Overmind did not choose to compete despite winning the 2010 competition. After the competition, many bot programmers (including the Overmind team) realized that their 2010 strategy was quite easily defeated by early game rushing strategies, and so they submitted a Terran bot instead, called Undermind, which finished in 7th.

AIIDE 2012. The University of Alberta also hosted the 2012 competition, with the major difference from the 2011 competition being the addition of file reading and writing for learning. Bots could now write information to disk during a match, and then read the information during other matches, allowing them to adjust strategies based on previous results. 6 of the 10 entrants used this feature to aid in strategy selection, including the top 4 finishers. More improvements to

Table 3. Results of the five best bots of the AIIDE 2012 competition

Rank	Bot	Win %
1	Skynet	84.4%
2	AIUR	72.2%
3	UAlbertaBot	68.6%
4	BroodwarBotQ	59.1%
5	AdjutantBot	52.8%

the tournament environment also meant that a total of 8279 games could now be played in the same time period. Results are shown in Table 3.

Skynet once again won the competition with its solid Protoss build orders and good Dragoon kiting. AIUR and UAlbertaBot switched positions from the previous year to come 2nd and 3rd respectively. Both AIUR and UAlbertaBot used data stored from the results of previous games to select a strategy for future matches. UAlbertaBot did this using the UCB [2] algorithm, while AIUR used a uniform distribution to choose its mood before altering this distribution after some games against the same opponent to favor efficient strategies, achieving similar results than UAlbertaBot. Notice that, compared to AIIDE 2011, AIUR proposes a new mood, *Cheese*, implementing a Photon Cannon rush strategy in order to surprise the opponent and to finish the game as soon as possible. The effect of this strategy selection process can be seen Figure 2 which shows bot win percentages over time. While the earlier rounds of the tournament fluctuated wildly in results, eventually the results converged to their final values. One of the main reasons for this is due to the bots learning which strategies to use as the tournament progressed.

AIIDE 2013. A total of 8 bots competed in the 2013 AIIDE competition with many of the same names from the 2012 competition, and some key updates were made to existing bots for 2013 which shook up the results from the previous years. A total of 5597 games were played during the 5 day tournament. UAlbertaBot took 1st place with a dominant 84.49% win rate, with Skynet in 2nd with 72.77%. The major addition to UAlbertaBot was a combat simulation package called SparCraft. UAlbertaBot was re-engineered so that it now always attacked from the first combat unit created, using the SparCraft system to simulate whether or not the bot could a fight against known enemy units, retreating automatically if it thought it could not. This addition, combined with some additional bug fixes led to the victory. Skynet and AIUR both implemented strategy learning in 2013, which is evident from 3 in which both Skynet and AIUR's win percentage over time climb dramatically from the first few rounds of the tournament to the later rounds. Ximp unfortunately crashed all games on the Fortress map and lost 10% of its games for free as a result, which meant it could have possibly came in 2nd place if not for all those crashes.

Table 4. Results of the five best bots of the AIIDE 2013 competition

Rank	Bot	Win %
1	UAlbertaBot	84.49%
2	Skynet	72.77%
3	AIUR	58.51%
4	Ximp	56.57%
5	ICEStarCraft	48.53%

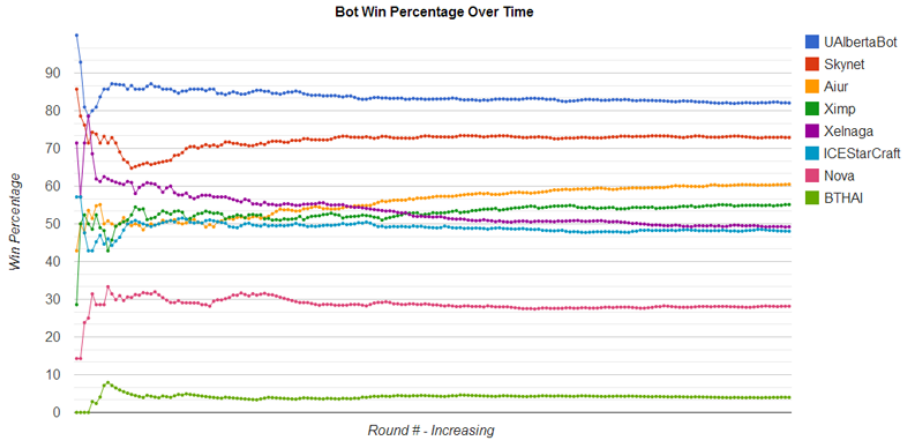

Fig. 3. Evolution of the win percentage of each bot participating in the AIIDE 2013 competition

AIIDE 2014. The 2014 AIIDE competition saw 18 total entrants, over double the number from the 2013 competition. This dramatic increase in numbers was partially because of a large advertising effort by competition organizers, combined with the inclusion of all the 2013 competition entrants (if they had not re-submitted new versions). UAlbertaBot and Skynet who finished 1st and 2nd the year before were not updated in 2014 and so their previous versions were re-submitted. Along with the larger registration numbers came a lot of new Terran bots, a welcome change from the Protoss-dominated fields of previous years. The tournament managing software also underwent some major updates in 2014, allowing for faster game scheduling and the easy pausing and resuming of tournaments, resulting in 10251 total games being played, which was almost double the amount of 5579 from the previous year.

The results of the 2014 competition were dramatically different from previous years. Since 2011 the top 3 bots in the AIIDE competition had been some permutation of Skynet, UAlbertaBot, and AIUR, and in 2014 the top 3 finishers were all completely different. In first place was IceBot with a win rate of 85.85%, which had a complete overhaul in terms of strategy and tactics from the previous year. IceBot had several strategies which it could implement, most starting with

Table 5. Results of the five best bots of the AIIDE 2014 competition

Rank	Bot	Win %
1	ICEBot	85.86%
2	Ximp	84.64%
3	LetaBot	82.09%
4	AIUR	70.94%
5	Skynet	68.74%

an early bunker defense and transitioning into attacking once a set of heuristics conditions had been met. IceBot's source code was originally based on AIUR's modular design, and changed to play the Terran race. In a close 2nd place finish was Ximp - which had fixed the crashes and logic errors which had plagued it the year before. Ximp implemented a solid forge-first fast expansion strategy which transitioned into late-game carrier play. By building a large amount of early photon cannons in its expansion, Ximp was easily able to stop most of the early rushing bots, and then holding them off until its carriers cleaned up in the late game. 3rd place was taken by LetaBot, a new Terran bot whose source code was written on top of the 2012 version of UAlbertaBot and changed to play the Terran race. LetaBot implemented a number of strategies including a 'cheesy' but strong bunker rush strategy which killed many bots very quickly, as well as a Wraith (mid-game flying unit) strategy.

AIIDE 2015. The 2015 AIIDE competition was the largest competition to date hosted by a conference, with 23 total bots competing. The competition environment was also changed so that the tournament was now run on virtual machines instead of physical machines. Doing this enabled the tournament to run for a full two weeks instead of the normal one week, resulting in a total of 20788 games being played - 90 full round robins between each bot pairing. 2015 saw the most even distribution of race selection ever in a Starcraft AI competition, with 5 new Zerg submissions along with 7 Protoss bots and 9 Terran bots. The first ever Random race entry was also submitted as UAlbertaBot had been updated to play all three of the available races. When playing Random race, Starcraft randomly assigns one of Protoss, Terran, or Zerg after the game has started, with your opponent not knowing which race you are until they have scouted you on the map. This provides a significant advantage as the enemy's initial build order must now have a period of uncertainty during which is has to guess what race it is playing against.

The results of the 2015 competition were extremely close, with 1st / 2nd place as well as 3rd / 4th place being separated by less than 1% win rate. 1st place was won by Tscmoo, a new Zerg bot which played nearly a dozen different strategies, learning which one to choose over time via the file I/O system. In a close second place was ZZZKbot, which implement a 4-pool Zergling rush strategy every game. Despite the relatively simple strategy, most bots did not have proper defense capabilities and lost in very short games. In 3rd place was

Table 6. Results of the five best bots of the AIIDE 2015 competition

Rank	Bot	Win %
1	Tscmoo	88.52%
2	ZZZKBot	87.84%
3	Overkill	80.69%
4	UAlbertaBot	80.20%
5	AIUR	73.02%

Table 7. Results of the first round at CIG 2011, held in two brackets. Qualified for the final round: UAlbertaBot and Skynet (from A), Xelnaga and BroodwarBotQ (from B, the latter by comparing direct encounters with BTHAI of which 6:4 were won)

Bracket A				
Rank	**Crashes**	**Games**	**Bot**	**Win %**
A1	0	40	UAlbertaBot	82.5%
A2	1	40	Skynet	77.5%
A3	2	40	AIUR	60.0%
A4	1	40	Nova	20.0%
A5	0	40	LSAI	10.0%

Bracket B				
Rank	**Crashes**	**Games**	**Bot**	**Win %**
B1	12	40	Xelnaga	62.5%
B2	3	40	BroodwarBotQ	57.5%
B3	0	40	BTHAI	57.5%
B4	17	40	Protoss Beast Jelly	42.5%
B5	0	40	EvoBot	30.0%

Overkill, another Zerg bot which had several different strategies including Mutalisks and Hydralisks, which learned over time as well. In a close 4th place was UAlbertaBot, which played Random and implemented three main rushing strategies - one for each race. An interesting result from this tournament was that despite UAlbertaBot coming 4th place, it finished with a winning rate greater than 50% vs each other bot. AIUR came in 5th place and was a clear demonstration of how learning over time can dramatically improve results in a tournament, going from 63% win rate early in the competition to a final win rate of over 73%.

CIG

An initial attempt to run a StarCraft tournament at the Computational Intelligence in Games conference (CIG 2010) suffered from technical problems. These mainly stemmed from the desire to use evolved, largely untested maps which initially looked interesting but made the submitted bots and the Brood War Terrain Analyzer (BWTA) provided with the BWAPI interface crash so frequently that it would have been unjustifiable to announce a winner.

CIG 2011. At CIG 2011, the tournament was therefore run with a (secret) selection of maps used in league play, which can be regarded as the most important difference to the AIIDE tournament that employed a known list of maps. The competition was organized by Tobias Mahlmann and Mike Preuss and attracted 10 bots. In addition to the ones discussed in previous sections (UAlbertaBot, Skynet, AIUR, Nova, BroodwarBotQ, BTHAI), the set also contained LSAI, Xelnaga, Protoss Beast Jelly, and EvoBot, these are shortly described in the following:

LSAI (Zerg) utilizes a heavily modified BWSAL[19] to divide management of the units to different modules that communicate via a centralized information module. It works using a simple reactive strategy to try and survive early game attacks and macro up to a larger attack force and maintain map control.

Xelnaga (Protoss) is a modification of the AIUR bot that chooses the Dark Templar Opening in order to destroy the enemy base before defenses against invisible units are available.

Protoss Beast Jelly (Protoss) always goes for a 5-gate Zealot rush, supported by an effective harvesting strategy named power-mining (2 probes are assigned to every mineral patch, thereby needing 18 probes for 100% saturation in a normal map, prior to expanding). Gas is not mined as it is not needed for constructing Zealots.

EvoBot (Terran) employs an evolutionary algorithm for obtaining rational unit combinations and influence map techniques for deciding the strategic locations. Note that this bot was submitted in a very early version, with many of its designed features not yet fully ready.

First Round As the CIG competition games were executed manually due to a lack of available software (the AIIDE program was not yet available at that time), the organizers separated the ten entries into two brackets. In each bracket of 5 bots, a round-robin tournament was held with 10 repetitions per pairing, resulting in 40 games per bot. The 5 maps chosen for the first round were selected from the pool of well-known league play maps found on the Internet: *(2)MatchPoint 1.3*, *(4)Fighting Spirit 1.3*, *iCCup Destination 1.1*, *iCCup Gaia*, and *iCCup Great Barrier Reef*. Each bot pairing played on every map twice, with switched starting positions.

The two top bots of every bracket qualified for the final round. Table 7 summarizes the results. Note that as BroodwarBotQ and BTHAI have the same number of wins, their direct encounter was evaluated which accounted 6:4 for the BroodwarBotQ. The bots going into the final were thus UAlbertaBot, Skynet (from bracket A) and Xelnaga and BroodwarBotQ (from bracket B). All qualified bots play the Protoss faction. Most bots proved fairly stable, only Xelnaga and

[19] https://code.google.com/p/bwsal/

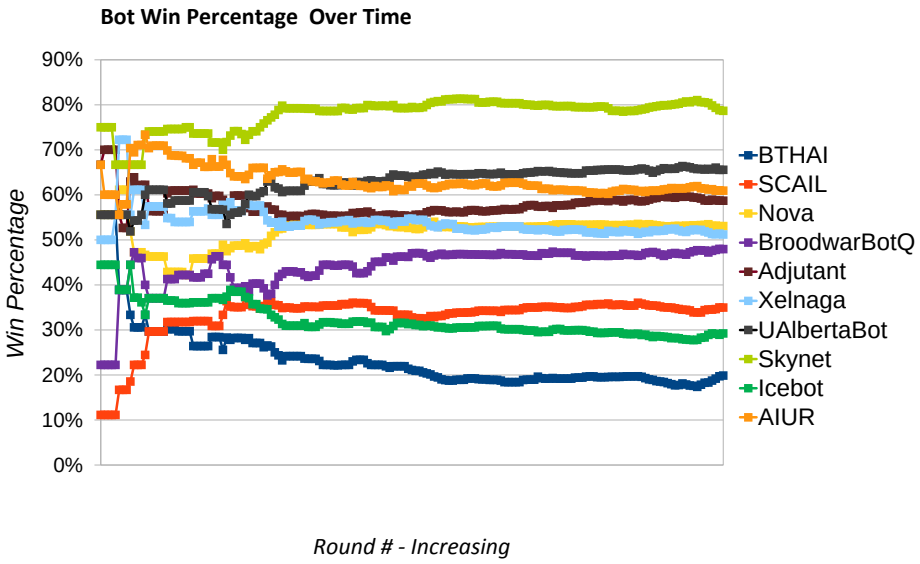

Fig. 4. Evolution of the win percentage of each bot participating in the CIG 2012 competition

Protoss Beast Jelly crashed relatively often (each in more than a quarter of the games). Crashing of course resulted in an instant win for the other bot. In some cases, neither bot was able to finish the other off completely, so that they went into a passive state. Such games were manually ended after around 15 minutes and victory were assigned to the bot that had obtained more points as indicated on the end game screen.

Final Round The final round was played in a similar mode as each of the first round brackets, using another set of 5 previously unknown maps: *iCCup lost temple 2.4*, *iCCup rush hour 3.1*, *iCCup swordinthemoon 2.1*, *iCCup yellow 1.1*, and *La_Mancha 1.1*. Letting each pairing play on each map twice again with switching starting positions resulted in 30 games per bot. The final results are displayed in table 8, indicating Skynet as winner and UAlbertaBot as runner-up, being almost equally strong, and the two other bots as clearly inferior. The com-

Table 8. Results of the CIG 2011 competition

Rank	Crashes	Games	Bot	Win %
1	0	30	Skynet	86.7%
2	0	30	UAlbertaBot	73.3%
3	3	30	Xelnaga	36.7%
4	2	30	BroodwarBotQ	3.3%

petition setup, documentation and results can be found in the 2011 competition web page[20].

CIG 2012. For CIG 2012, the AIIDE tournament software was employed, leading to a total of 4050 games played in 90 rounds of round robin. As 6 different maps were used, this means that each bot played every other on every map 15 times. As in the AIIDE competition, writing to and reading from a bot specific directory was enabled, however, due to technical reasons, this feature was constrained to the computer (of 6) the game was actually run on. We can therefore assume that this feature was of minor use for the CIG competition. The only other difference to the AIIDE competition was that the used maps were not made available to the competitors in advance.

These maps came in two flavors, namely three 3-player maps: *Athena-II*, *Neo Moon Glaive*, *Tears of the Moon*, and three 6-player maps: *Legacy*, *River of Light*, and *The Huntress 1.1*. We shall note that some bots consistently crashed on one of the originally considered maps which has thus been replaced. This is surprising as all maps are well known league play maps or have been provided with the StarCraft Brood War distribution itself. Setup, replays and results for the CIG 2012 competition can be found here[21].

The overall results are displayed in table 9, and the win rate evolution over time in figure 4. These are quite consistent with the results of the AIIDE 2012 competition, so that we can conclude that the best bots are not very dependent on knowing the maps beforehand. However, the bot vs. bot win rates as displayed in figure 5 show some interesting trends. On the maps with more possible start points, some bots do better than others, namely SCAIL, Adjutant, Nova, and UAlbertaBot, the latter probably due to its very efficient scouting routine. Some bots however suffer from the increased uncertainty about the enemies' position, namely Xelnaga and BroodwarBotQ.

[20] http://ls11-www.cs.tu-dortmund.de/rts-competition/StarCraft-cig2011
[21] http://ls11-www.cs.tu-dortmund.de/rts-competition/StarCraft-cig2012

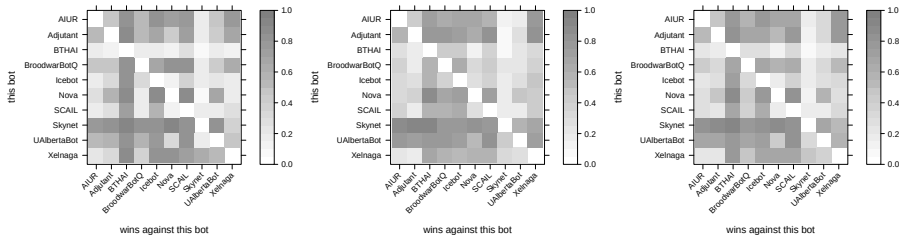

Fig. 5. Win percentages of CIG 2012 competition, from left to right: 3-player maps only, 6-player maps only, all maps. Read from line to column, bot in row wins given fraction of games against bot in column. For some bots, we find interesting differences, e.g. Xelnaga gets worse on 6-player maps, UAlbertaBot gets better. Only Xelnaga can reliably beat Skynet, but only on 3-player maps

Table 9. Results of the CIG 2012 competition.

Rank	Bot	Win %
1	Skynet	78.3%
2	UAlbertaBot	65.2%
3	AIUR	60.4%
4	Adjutant	58.6%
5	Nova	52.4%

As already observed before in the previously described competitions, there are also bots who consistently beat top ranked bots but have severe problems against lower ranked bots. E.g., Xelnaga is especially strong against Skynet on the 3-player maps (about 70% wins). Reviewing the replays led to the assumption that Xelnaga usually tries to attack Skynet's probes with a dark templar strategy, and often succeeds. Nova does very well against the UAlbertaBot, and the replays show that it sometimes succeeds to lure the probes into its own base, where they get killed, leading to severe resource problems. However, we cannot tell how often this happens as this would require to review every single replay between the 2 bots. Summarizing, most bots seem to have improved, which becomes clear if the nearly unchanged BTHAI bot is taken as a baseline. In 2011, it won more than half of its qualifying games, in 2012 it came out last with around 20% wins. However, designing a bot in order to beat a top bot (as for Xelnaga with Skynet) leads to a very restricted strategy that often leads to failure if playing against different bots. Note that in the direct encounter between Xelnaga and AIUR, its ancestor, Xelnaga looses consistently.

Nevertheless, from the observations we made during the tournament, we can draw the conclusion that the available bots are still very constrained. No bot in the competition played the Zerg race, which is surprising as the AIIDE 2010 winner (Overmind) did so. Presumably, implementing a good Zerg strategy is more demanding than implementing one for the Protoss or Terran races. Many bots consistently crashed when playing against a random race built-in bot for testing, and also did so when the map size was changed from 128×128 to any other. Furthermore, every single bot sometimes failed to finish off an already beaten opponent, such that the game had to be stopped after a previously determined maximum time. It also seems that most of the current bots are not very good at adapting their strategy to the one of their opponent during a game, or at least (via the read/write procedure of game information) within a series of games.

CIG 2013. The CIG 2013 competition was once again organized by Mike Preuss (TU Dortmund) and Tobias Mahlmann (ITU Copenhagen), adding Antonio Mora García from the Universidad de Granada as third team member. Whereas the rules and even the participants to this competition were almost identical to the AIIDE 2013 competition setup, several technical changes were

Table 10. Results of the CIG 2013 competition.

Rank	Bot	Win %
1	Skynet	91.1%
2	UAlbertaBot	67.4%
3	AIUR	54.9%
4	Xelnaga	53.6%
5	Adjutant	42.4%

made in comparison to 2012[22]. A new competition software was implemented by Tobias but not completely finished in time, so that the read/write function that enables learning between games had to be disabled again. This is probably the most important difference to the AIIDE competition, however, as the result provided in table 10 shows, the effect is limited. Due to the late availability of the competition software, only 32 rounds of random robin were played, making 896 games altogether, and 224 games by each of the 8 submitted bots.

As the map set has also been changed to the 10 standard maps used by the AIIDE competition (without letting the competitors know about this in advance as usual for the CIG competitions), the competitions were otherwise very similar. We presume that disabling the learning was a disadvantage for the UAlbertaBot who won the AIIDE competition and was only runner-up to the Skynet bot here. The 3rd place went to AIUR as in 2012, followed by Xelnaga and Adjutant. A direct comparison to the results of CIG 2012 (table 9) shows that bot evolution between these two competitions was obviously limited. It also shows that some bots as the UAlbertaBot make good use of online learning, whereas others, as Skynet, do not profit from it that much.

CIG 2014. The CIG 2014 competition was organized by Kyung-Joong Kim, Ho-Chul Cho, and In-Seok Oh of Sejong University. For 2014, the CIG competition used a total of 20 different maps which were unknown to the competitors before the competition started, which was by far the most maps ever used in a Starcraft AI competition and presented a challenge to many of the bots who entered. A total of 4680 games were played which meant that each bot played each other bot 60 times, or 3 times per map. The CIG 2014 competition was held just a few weeks before the AIIDE 2014 competition and so many of the entrants to the competition were identical, and the results definitely showed this. The top 4 bots were all the same as the 2014 AIIDE competition with IceBot coming 1st, Ximp in 2nd, LetaBot in 3rd and AIUR in 4th place. For descriptions of these bots please refer to the AIIDE 2014 competition description as none of the top-finishing bots had any significant changes made.

CIG 2015. There were some significant rule changes to the CIG 2015 competition, which was once organized by members from Sejong University. The most

[22] http://ls11-www.cs.tu-dortmund.de/rts-competition/StarCraft-cig2013

Table 11. Results of the CIG 2014 competition.

Rank	Bot	Win %
1	IceBot	83.1%
2	Ximp	78.1%
3	LetaBot	68.5%
4	AIUR	66.1%
5	UAlbertaBot	60.0%

Table 12. Results of the CIG 2015 competition.

Rank	Bot	Win %
1	ZZZBot	81.03%
2	tscmoo-Z	73.59%
3	Overkill	62.05%
4	LetaBot	61.54%
5	Ximp	60.26%

significant rule change was that entrants no longer had to be open source, in an attempt to attract more competitors who may not want to open source their bots. The second rule change was that a single competitor could submit multiple entries to the competition - which caused some controversy among registrants since this introduces the possibility of collusion between entries. Thankfully no such collusion was detected during the competition. The 2015 competition was also not run quite as long as previous competitions with only 2730 games being played in total, or 30 between each bot pairing, 6 between each bot on each of the 5 chosen maps. There were several new Zerg entries to the CIG 2015 competition which ended up finishing in the top three positions, with results very similar to those of the AIIDE 2015 competition. ZZZBot took 1st place, tscmoo-Z (a Zerg bot written by tscmoo) came 2nd, and Overkill came 3rd. For descriptions of these bots and their strategies used please refer to the AIIDE 2015 competition section as the strategies remained largely unchanged between the CIG and AIIDE competitions.

SSCAIT

The Student StarCraft AI Tournament[23] (SSCAIT) started in 2011 at Comenius University in Bratislava, Slovakia and has been well known for being the Starcraft AI competition with the highest number of total participants. Started as a part of a course in artificial intellignece at Comenius University, initial SSCAIT seasons included several dozen student submissions from this course in addition to submissions from across the globe. There are three fundamental differences between SSCAIT and the remaining two conferences:

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.

[23] http://sscaitournament.com/

Table 13. Ranking of the best bots of the SSCAIT 2012 competition (52 participants)

Student division		Mixed division	
Rank	**Bot**	**Rank**	**Bot**
1	Matej Isteník's bot	1	ICEBot
2	Marcin Bartnicki's bot	2	Marcin Bartnicki's bot
3	UAlbertaBot		

2. There are two phases of SSCAIT each year: a *competitive* phase, lasting for up to three weeks and a *sandbox* phase which runs for approximately eleven months each year. In other words, SSCAIT is live at all times with only a few short interruptions.
3. All the games are publicly *streamed live* 24 hours a day. The stream uses a custom observer script [6] designed specifically to improve the viewer experience and creates a continuous feedback loop for the participants who can watch their bots play against the others.

SSCAIT 2011. The first SSCAIT tournament was organized by Michal Čertický as part of the *"Introduction to Artificial Intelligence"* course at the Department of Applied Informatics, Comenius University in Bratislava, Slovakia. It was inspired by the media coverage of AIIDE StarCraft AI competition held in 2010, but over the following years it diverged from AIIDE competition's format significantly.

The first year of SSCAIT hosted 50 participants, all of whom were students of Comenius University at the time. The participants were allowed to select any race and use all the units and technologies in complete 1 vs. 1 games of StarCraft. Vast majority of the bots were implemented to execute hard-coded strategies and behavior. Participants were divided into 10 groups of 5 people, and 16 of them advanced into a double elimination bracket. Final match of the elimination bracket was won by R. Danielis with a 2-1 score against M. Piovarči (both playing as Zerg).

2011 was the only year when the games were not streamed live. However, the replays were published in real time and the games from the elimination bracket were recorded and uploaded to YouTube.

SSCAIT 2012-2015. In 2012, the SSCAIT became a truly international competition with 52 participants from numerous universities including UC Berkeley (USA), University of Alberta (CAN), Washington State University (USA), University of Nantes (FRA), University of Grenoble (FRA), New University of Lisbon (POT), Gdansk University of Technology (POL), Ritsumeikan University (JAP), University of Bielefeld (GER), Sofia University (BUL), Comenius University (SVK), University of Žilina (SVK) and Technical University in Košice (SVK), as well as a number of non-student participants.

The format of SSCAIT changed significantly in 2012. The tournament was divided into two phases:

Table 14. Ranking of the best bots of the SSCAIT 2013 competition (50 participants). The student division winner was determined by means of 1190 round-robin games

Student division		Mixed division	
Rank	**Bot**	**Rank**	**Bot**
1	Ximp	1	Krasi0
2	W.O.P.R.	2	ICEBot
3	UAlbertaBot		

Sandbox phase is played non-stop during first 11 months of the year. All the games are streamed via a video streaming service, such as Twitch.tv or Hitbox.tv. The stream is meant to attract new entrants and to help current participants improve their bots before the second phase begins. The participants can watch their bots play against randomly selected other bots. They are allowed to upload new versions of the bots via a web interface at any time during this phase. The sandbox phase has no winners. However, the bots may use this long phase to collect useful data about their opponent's behavior.

Competitive phase is played for up to three weeks (depending on the number of participants) at the end of the year. New bot versions cannot be uploaded during this phase and the matches are not scheduled randomly any more. This phase has two divisions with different rules:

- *Student division:* Only the bots created by a single participant, who is currently a student, are considered "student" bots. Other bots are tagged as "mixed-division" bots. Each student bot plays a single game against every other bot and collects points for the wins. Three student participants with the highest score are considered winners and receive certificates of achievement (and in years 2012 and 2013 also financial prizes collected via crowdfunding campaigns). Ties are broken using the results of the mutual game and (if necessary) additional games are scheduled. The student division was created so that the students stand a chance of winning the certificates in the presence of more experienced, non-student participants and team-created entries.
- *Mixed division:* Eight or sixteen bots with the highest win rate among all the participants (including student and non-student entries) are selected for the additional "mixed division" elimination bracket. This part of the tournament is typically recorded with a commentary and uploaded to YouTube, concluding each SSCAIT season since 2012.

The results of SSCAIT seasons 2012-2014 are provided in the tables 13 to 15. At the time of writing this, the SSCAIT 2015 is still in a sandbox phase. 45 bots have registered for the competitive phase so far, including two bots playing as "Random" race. The season will be concluded by an elimination bracket featuring 16 bots in 2015.

From a technical perspective, the tournament was run manually in 2011 and by a collection of custom automated scripts in 2012. Since the second half of 2013,

Table 15. Ranking of the best bots of the SSCAIT 2014 competition (42 participants). The student division winner was determined by means of 861 round-robin games

Student division		Mixed division	
Rank	**Bot**	**Rank**	**Bot**
1	LetaBot	1	LetaBot
2	W.O.P.R.	2	Ximp
3	UAlbertaBot		

SSCAIT has been using the automated tournament managing software developed at University of Alberta, which was modified to suit the needs of SSCAIT format and technical environment. In contrast to AIIDE and CIG competitions, there is only one game running in SSCAIT at any given time, so that each game can be streamed live (approximately 150 games are played every day). The live stream makes use of a custom made observer script by Mattsson et al. [6] which implements a smooth, prioritized camera movement during the game and a third-party libraries that allow the game to be displayed in high definition screen resolution.

Thanks to the tournament managing software, the game rules and I/O procedures have been synchronized with AIIDE and CIG competitions, so the participants are able to submit the same version of their bot to all three of them. In addition to bots written in C++, SSCAIT supports bots using Java interfaces to BWAPI: BWMirror and JNIBWAPI.

Conclusions

Competitions have historically been an important drive in artificial intelligence, with challenge tasks such as *Chess*, *Go*, autonomous vehicle driving, among others having resulted in many algorithmic and theoretical breakthroughs. For that reason, we believe RTS AI competitions are playing and will continue to play an important role in the field by both motivating new research and also being an incentive to attract new students and researchers to work on some of the key open challenges of RTS AI.

This chapter has provided a summary of the results of the most three most important StarCraft AI competitions, organized during the past six years, as well as a summary of the architectures used by some of the bots that regularly participate in these competitions. Moreover, as evidenced by the poor results that the top bots in these competitions achieve when playing against good human players, a significant amount of open questions remain about how to design AI systems that can handle real-time adversarial domains such as StarCraft. Hopefully, these competitions can play an important role in motivating research programs that can ultimately answer these questions.

Cross References

In the Springer Encyclopedia of Computer Graphics an Games volume 1:

- Santiago Ontan, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill and Mike Preuss, *RTS AI: Problems and Techniques*.
- Kazuki Yoshizoe and Martin Mueller, *Computer Go*.

References

1. Aha, D.W., Molineaux, M., Ponsen, M.J.V.: Learning to win: Case-based plan selection in a real-time strategy game. In: ICCBR. pp. 5–20 (2005)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning 47(2), 235–256 (2002)
3. Churchill, D., Buro, M.: Build order optimization in starcraft. Seventh Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2011) pp. 14–19 (2011)
4. Hagelbäck, J.: Potential-field based navigation in starcraft. In: CIG (IEEE) (2012)
5. Marthi, B., Russell, S., Latham, D., Guestrin, C.: Concurrent hierarchical reinforcement learning. In: International Joint Conference of Artificial Intelligence, IJCAI. pp. 779–785 (2005)
6. Mattsson, B.P., Vajda, T., Čertickỳ, M.: Automatic observer script for starcraft: Brood war bot games (technical report). arXiv preprint arXiv:1505.00278 (2015)
7. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: On-line case-based planning. Computational Intelligence 26(1), 84–119 (2010)
8. Synnaeve, G., Bessiere, P.: A Bayesian Model for RTS Units Control applied to StarCraft. In: Proceedings of IEEE CIG 2011. p. 000. Seoul, Corée, République De (Sep 2011)
9. Uriarte, A., Ontañón, S.: Kiting in rts games using influence maps. In: Eighth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2012) (2012)
10. Young, J., Hawes, N.: Evolutionary learning of goal priorities in a real-time strategy game. In: Eighth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2012) (2012)
11. Young, J., Smith, F., Atkinson, C., Poyner, K., Chothia, T.: Scail: An integrated starcraft ai system. In: CIG (IEEE) (2012)