Adaptive Strategy Decision Mechanism for StarCraft AI

Sangho Yi*

INRIA Grenoble Rhone-Alpes, France
sangho.yi@inria.fr

Abstract. Recent advancement of computer systems and their global networks have enabled us to enjoy many kinds of on-line games. StarCraft has been the one of the well-known on-line strategy simulation games, since it was made in 12 years ago. The game users have been extremely improved their playing level, but the AI (artificial intelligence) system of the game was rarely updated. At the consequence, current AI level is much poorer than that of the human players. In this paper, we present a noble AI mechanism for StarCraft based on the adaptive strategy decision mechanism. We implemented the proposed mechanism on the real StarCraft: BroodWar binary code. We compared the performance of the modified AI code, and the results show that the new AI code outperforms the existing artificial intelligence of the game.

1 Introduction

In these days, video game industry has been increased significantly, and now it has a major impact on the world economics. Especially, the computer games market has grown dramatically over the last decade by human demands. Current technologies enable the computer games to connect all the people at anytime, and anywhere in the world.

Based on the high-speed networks, distributed systems, and significant computing power, they become not only short-term enjoying entertainment, but also a kind of long-term sports whole the world. For example, the first e-Sports competition, *World Cyber Games* has been held since 2001 [1], and now it is the biggest annual Game-Olympiad in the world. Also, the *World e-Sports Games* [2] has been held annually after 2004. In addition, the words '*professional players*' are not exclusively possessed by the general sports. We already know the existence of the '*professional-gamers'* on many popular computer games [3, 4]. StarCraft [5] was made by Blizzard Entertainment in 1998, and it has been the most famous real-time strategy computer game in the world.

Even if above e-Sports have significant amount of players and fans, even if they are so popular in this time, they still have several challenges because of their characteristics. First, players may need to play with computer AI systems. Even if

* Corresponding author.

M.-W. Han and J. Lee (Eds.): EKC 2010, SPPHY 138, pp. 47–57.
springerlink.com

player vs. player games, the players may use some 'macro' commands which should be run with computer's AI. Second, someone (vendors, players or third-parties) should manage, update, and maintain the game core framework, because this is based on the large-scale computer software systems. In those points of view, we cannot assure the longevity and quality of the e-Sports, for example, StarCraft was made 12 years ago, and the main core was rarely improved after the year 1998. Just a few patches were distributed to fix bugs on it. At the consequence, level of the artificial intelligence becomes relatively much poorer than that of the human players. The players may win the game even if one human player fights with multiple computer AIs on the same custom game. Thus, StarCraft AI code has to be upgraded and patched to relieve huge gap between computers and players.

Considerable efforts [6-13] have been made to improve the poor AI of the StarCraft. However, some of them used only '*cheat codes*' to improve the power of AI in the order of magnitude [6]. That kind of work is just a piece of cheats, and they do not have any improvement in quality compared to the existing AI. In this paper, we present a noble AI mechanism for StarCraft based on the adaptive strategy decision mechanism. It was designed and developed to improve current level of AI. We implemented the proposed AI mechanism on the real StarCraft binary code, and made several experiments. The results show that the performance of the proposed AI mechanism outperforms the current AI on the StarCraft.

The rest of this paper is organized as follows. Section 2 presents work related to AI development on the video games. Section 3 presents the design and implementation of a noble AI mechanism on the StarCraft in detail. Section 4 evaluates the performance of the proposed AI mechanism compared with the current AI on the StarCraft. Finally some conclusions and possible future work are given in Section 6.

2 Related Work

In this section, we present a brief overview of previous research efforts on the AI of the video games and its implementation [6-15].

In 2001, IBM announced Robocode contest for having both educational effect and better AI for mobile tank robots [14]. This has been well-known contest in the world, and many algorithms have been made on top of the Robocode's framework.

In [15], Ian and James proposed a script language for programming artificial intelligence of games on Sony's PlayStation 2. This is a rule-based object-oriented programming language which called by *RC++*. It was designed to alleviate the complexity of programming codes in *C* language, and to give more flexible representative power. By using the *RC++*, the programmers may write out more efficient and well-structured codes of artificial intelligence for the games on PlayStation 2.

In addition, in [8], ScAIEdit (StarCraft Artificial Intelligence Editor) was proposed to represent and program the AI code of the StarCraft. ScAIEdit is a set of converter and editor of the artificial intelligence on this game. By using it,

programmers can easily retrieve and modify the original source code of the StarCraft. It represents the machine level code by their own language. In compiling phase, it transforms the source code to the binary code of the StarCraft.

Also, there are considerable efforts [6, 16, 17] on developing and improving the artificial intelligence of the StarCraft.

First of all, an unofficial AI patches of the StarCraft were made by some anonymous programmers [6]. However, they used only '*cheat code*' to dramatically improve the existing artificial intelligence of the StarCraft. Thus, it is an unfair improvement to compare with the existing one.

In several players' forums, bunch of strategies have been developed based on the cost and time analysis under constraints of the required time and resources. However, no strategy did not be the greatest one, because StarCraft is a multi-player game, and there is no optimal solution. The strategies of each player may affect one another, and unlike to Chess or Game of Go, StarCraft players do not share their units, the amount of harvested resources, and etc. Therefore, to make an efficient and robust StarCraft AI, we have to adapt itself with dynamic conditions.

Recently (in 2009), Blizzard Entertainment opened the StarCraft:BroodWar API [9], which is C/C++ based programming interface to the computer AI robot. They also announced a AI programming contest [10] to get more volunteers on this opening.

Since the beginning of 2010, StarCraft II has been tested with volunteers, and the AI programming method [11] was discovered by anonymous users, and they developed several basic AI mechanisms to play with human players.

3 Design and Implementation of Adaptive Strategy Decision Mechanism on the Starcraft

In this section, we present design of a noble StarCraft AI mechanism which is based on the adaptive strategy decision mechanism. Also, we show programming method in detail to implement the AI mechanism on the StarCraft.

3.1 Starcraft in a Nutshell

In the world of the StarCraft, there are three kinds of races: Terran, Zerg, and Protoss. The three races have their own characteristics on buildings, units, and technical or magical skills. Buildings are used to train units and research some skills, and the units gather resources and construct some buildings, and fight with the opponents. Resources are statically deployed on the field, and the gathered resources are used to train units or construct some buildings with its production costs. Then, the trained units are organized into several troops, and it can attack to the opponents or defense them. The mission of the game is killing all other opponents, and thus, the efficiency of gathering resources, training units, constructing buildings, and using better military strategy are very important in the game.

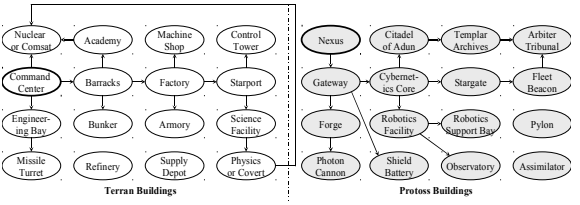

Fig. 1 Dependency relationships between buildings in Terran and Protoss races

In StarCraft, all buildings have their own dependency relationships. Fig. 1 shows the dependencies of the Terran and Protoss races [5]. For example, if we have no Barracks, we cannot build a Factory, and thus, we must build a Barracks first when we have to build a Factory. Also, we need to consider an appropriate number of peasants. Too many peasants are wasteful, but too small also be a cause of deficiency of the gathered resources.

3.2 Adaptive Strategy Decision on Starcraft

In real-time strategy games, strength of a player is determined by efficiency of the used strategies. The strategies on these kinds of games can be divided into following items: build-orders, combination of units, attack and defense timing, micro/macro unit control, and combat tactics. In those, current StarCraft AI is so weak in all the aspects compared with human players. That's because the computer's AI does not have the appropriate strategy decision method, and it was rarely improved since 1998. In the result, the current AI is not competitive compared with the players.

We have done related work on developing StarCraft AI mechanisms. In [12], we presented how to program StarCraft AI codes on the game, and in [13], we presented a knowledge-based strategy decision mechanism for improving StarCraft AI. In the previous work, we implemented the prototype version, and we shows significant performance improvement compared with the existing AI.

However, the modified AI was so predictable, and straight forward, because at that time we had only static decision mechanism which do not consider the dynamic situations. It did not work well with human players when the players understand their static decision functions.

Thus, in this paper, we propose an adaptive strategy decision mechanism to improve the existing intelligence of the StarCraft. The mechanism enables that each player adaptively decides an appropriate strategy by using the recent information from the opponents. The information consists of recent build-orders, combination of units, and timing of attack and defense. In addition, computer players does not need to scout the enemies because they already know what the enemies do. Thus, it easily enables the adaptive strategy decision mechanism on the computer's artificial intelligence.

Some dividable stages exist on the StarCraft. Based on many experiences from users [16, 17], the whole game has been considered as the three stages as follows.

- *Opening*: This is the first few minutes of the game. Similar to the 'opening stage' in Chess and Game of Go, we have very well-known build-orders. In this stage, selecting an appropriate build-order based on the opponent's strategy is very important because it determines the successful defense from the early attacks from the opponents.
- *Middlegame*: This is started by the first massive attack. In this stage, players are preparing larger combat by expanding and increasing their own troops and production power. Thus, the utilization of the resources (gathering resources and producing units) is the most important in this phase.
- *Endgame (Last stage)*: This is after one become the dominant player in the game. Normally it comes with some massive attack. In this, a series of massive attack may occur, and the players may use almost all the field of the game to gather resources. Therefore, selecting an efficient massive attack and defense timing is the most important in this phase.

Based on the divided stages, we designed the adaptive strategy decision mechanism based on the basic principles listed below.

- *Use well-known strategies / build-orders.*
- *Utilize received information from opponents.*
- *Utilize known characteristics of units.*
- *Exploit computer's macro commands.*
- *Have various strategies to prevent prediction.*
- *Do not save resources for the future use.*

Table 1 Notations and functions used in this paper

Notation	Description
s_i	*i*-th possible strategy
t_c	Current time
t_e	Required execution time of the opponent's expected strategy
t_l	Laxity time from t_c to t_e
$t_r(s_i)$	Required execution time for s_i
$d(s_i)$	Degree of dangerousness of using s_i
d_{thres}	Predefined maximum threshold value of the degree of dangerousness

Table 1 shows the notations and functions used in this paper. The s_i represents *i*-th possible strategy of a computer player. Thus, the computer player has multiple strategies (i = 0, 1, … , n). The strategies include both attack and defense purposes. The t_e is the execution time of the opponent's expected strategy. The t_l is the remaining time to the expected execution time opponent's strategy. The $t_r(s_i)$ is the required time to execute the strategy s_i, and the $d(s_i)$ represents the degree of dangerousness of executing the strategy s_i. Finally the d_{thres} is the predefined maximum threshold value of the degree of dangerousness.

Algorithm 1. Adaptive Strategy Decision

- do the following when a computer player needs to select the next strategy:

1: $k := 1, x := 1$
2: **calculate** t_e
3: $t_l := t_e - t_c$
4: **for all** $s_{i \ (i=1,2,...,n)}$ **do**
5: **if** $t_r(s_i) < t_l$ **then**
6: $s_{ten-k} := s_i$
7: **increase** k
8: **end if**
9: **end for**
10: **for all** $s_{ten-i \ (i=1,2,...,k)}$ **do**
11: **if** $d(s_{ten-i}) < d_{thres}$ **then**
12: $d_{select-x} := d(s_{ten-i})$
13: $s_{select-x} := s_{ten-i}$
14: **increase** x
15: **end if**
16: **end for**
17: **randomly select** s_{final} **from** $s_{select-i}$ **where** $s_{select-i \ (i=1,2,...,x)}$

Algorithm 1 shows the adaptive strategy decision mechanism in detail. First of all, Algorithm 1 checks $t_r(s_i) < t_l$ for all possible strategies. Then, it saves the tentative strategies in $s_{ten-i \ (i=1,2,...,k)}$. Second, it checks the degree of

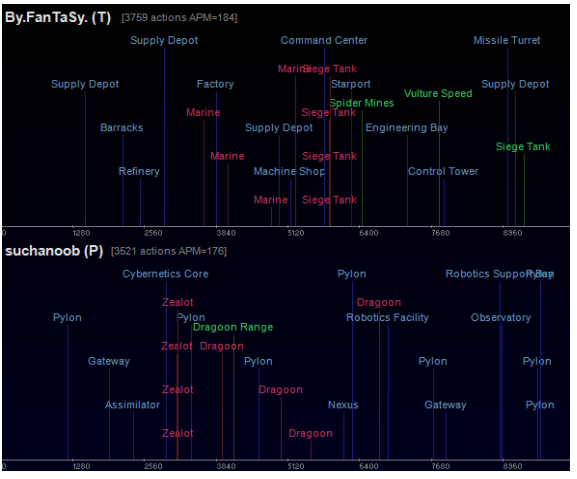

Fig. 2 An example of build-orders of a game

dangerousness of all possible s_{ten-i} . If the $d\left(s_{ten-i}\right)$ is smaller than d_{thres} , the strategy is saved on $s_{select-x}$. Finally, Algorithm 1 randomly selects one strategy s_{final} from the all possible strategies on $s_{select-i}$ $(i=1,2,...,x)$.

The major challenge of using Algorithm 1 comes from measuring required time for each strategy. To do this, we used BWChart Replay Analyzer [18] to monitor the build-orders and the execution time for each strategy from the previous replay records. Figure 2 shows an example of build-orders of two players. By using the replay records of the top-class amateur players, we found the reasonable $t_r\left(s_i\right)$ for many possible strategies.

3.3 Implementation

To implement the adaptive strategy decision mechanism on the StarCraft, we used ScAIEdit [8] and WinMPQ [19] tools. In [12], we have summarized the overall programming methods of the StarCraft AI with many kinds of practical programming examples. In the system of the StarCraft, the AI code consists of two-level modules. The first one is on the core routines of the binary executable program, and modifying the core routines is relatively hard. Even we can use the recently opened BroodWar API, there is lack of documents and resources for doing this. This is not the main focus of this paper. The second one is separated from the binary program. They reside in the *Patch_rt.mpq* file of the StarCraft folder, and the code is easy to program in script language level by using the ScAIEdit and WinMPQ tools.

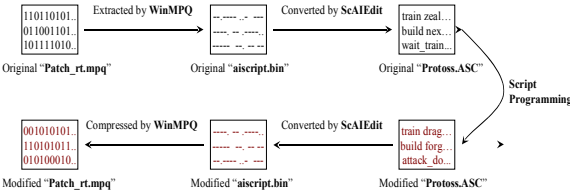

Fig. 3 Basic flow of developing StarCraft AI

Figure 3 briefly shows the overall flowchart of the AI programming method on the StarCraft. In this work, we extracted the AI source code from the original *Patch_rt.mpq* file, and then, we implemented the adaptive strategy decision mechanism on top of the script code[1].

4 Performance Evaluation

In this section, we evaluate the performance of the proposed artificial intelligence based on the adaptive strategy decision mechanism.

To fully evaluate the real performance of the modified artificial intelligence of the proposed artificial intelligence, we did experiments on the StarCraft. We used StarCraft: BroodWar ver. 1.15.2 [5] game package[2], and the series of official map files from [21].

We made experiments in both micro and macro scopes. Firstly we had micro benchmarks when a computer plays a Terran race while the opponent plays a Zerg race with fast 6 Zerglings rushes (the fastest attack strategy in StarCraft). We used the official map, 'The Hunters KPGA', and have tested if the computer Terran can defense the fast attack without significant sacrifice, and the Table 2 shows the results. In the results, the proposed AI outperforms the current AI on StarCraft, and the proposed AI can defense fast Zergling attacks for most cases except for the case of using only 4 Drones[3].

We have tested another popular initial attack situations, player's fast Zealot rushes to the computer AI player. We used the same condition for the rest, and Table 3 shows the results when computer plays a Terran race.

Secondly, we made macro benchmarks (whole game). We implemented the new AI mechanism on the Terran races, and we tested them with the unmodified AI. In the matches, we used the 'Python' map [21], and Tables 4 and 5 show the results of 1 vs 1 matches.

[1] The modified AI patch file is available at the website [20].

[2] The same AI patch can work with the recent version (1.16.1) of StarCraft: BroodWar.

[3] When a player uses a 4 Drones, 6 Zergling rush strategy, the opponent's troops can reach to the computer's base within 2 minutes and 10~40 seconds. If Terran builds a Bunker, it takes about 2 minutes and 30 seconds. Thus in some cases, it's almost impossible to defense the fastest Zergling rush.

Table 2 Experimental results of defensing fast Zergling rushes

Build-order	Proposed AI (Terran race)	Existing AI (Terran race)
4 Drones, 6 Zergling rush	3 / 10 (30% of defense)	0 / 10 (0% of defense)
5 Drones, 6 Zergling rush	5 / 10 (50% of defense)	0 / 10 (0% of defense)
7 Drones, 6 Zergling rush	9 / 10 (90% of defense)	2 / 10 (20% of defense)
9 Drones, 6 Zergling rush	10 / 10 (100% of defense)	4 / 10 (40% of defense)

Table 3 Experimental results of defensing fastest Zealot rushes

Build-order	Proposed AI (Terran race)	Existing AI (Terran race)
2 Gateways, 3 Zealots	8 / 10 (80% of defense)	2 / 10 (20% of defense)
3 Gateways, 6 Zealots	10 / 10 (100% of defense)	4 / 10 (40% of defense)
3 Gateways, 12 Zealots	10 / 10 (100% of defense)	7 / 10 (70% of defense)

Table 4 Experimental results of '1 vs. 1' matches against the unmodified Zerg computer AI

Build-order	Proposed AI (Terran race)	Existing AI (Terran race)
Average playing time	15m 21s	19m 41s
Wins / losses	20 / 0	11 / 9
Average resource use	14085.5	11499.2

Table 5 Experimental results of '1 vs. 1' matches against the unmodified Protoss computer AI

Build-order	Proposed AI (Terran race)	Existing AI (Terran race)
Average playing time	15m 17s	19m 51s
Wins / losses	20 / 0	8 / 12
Average resource use	14985.5	13903.5

Based on the results, we observe that the proposed mechanism outperforms the previous AI mechanism on StarCraft in terms of winning ratio, the amount of harvested resources, and the total playing time (less is better for the playing time).

5 Conclusions

In this paper, we proposed a noble AI mechanism for StarCraft based on the adaptive strategy decision mechanism. It was designed and developed to improve current artificial intelligence of the StarCraft. We implemented our mechanism on the real binary code, and we have tested it with the previous computer AI mechanism. Our micro and macro benchmarks showed that the performance of the proposed AI mechanism outperformed the existing AI on the recent version of tarCraft.

6 Future Work

We are currently extending our work to design and implement more efficient and adaptive strategy decision mechanism on the StarCraft, and we will draw the level of the artificial intelligence with user's level. In another study, we have implemented the proposed mechanism to play with with 2 previous AI machines (1 vs. 2), but then, the proposed AI worked in very defensive, and did not show aggressive attack patterns. We will extend our study by combining the current mechanism with BroodWar API [9] to have more possible way of implementing the adaptive strategy decision mechanism.

7 Availability

Data used in this study, the source code and binary and some results are available under the following web URL: http://aistarcraft.sourceforge.net

References

[1] World Cyber Games, http://www.worldcybergames.com (web)
[2] World e-Sports Games, http://www.worldesportsgames.com (web)
[3] KeSPA(Korean e-Sport Associaton) ProGamers, http://www.progamer.or.kr (web)
[4] Major League Gaming, http://www.mlgpro.com (web)
[5] StarCraft:BroodWar, http://www.battle.net/scc (web)
[6] BroodWar Artificial Intelligence Project, http://www.entropyzero.org/broodwarai.html (web)
[7] StarCraft Campaign Creations, http://www.campaigncreations.org (web)
[8] ScAIEdit Tool, http://www.camsys.org (web)
[9] BWAPI - An API for interacting with Starcraft: Broodwar, http://code.google.com/p/bwapi/ (web)
[10] StarCraft: BroodWar AI Competition, http://eis.ucsc.edu/StarCraftAICompetition (web)
[11] Writing your own StarCraft II AI, http://sc2.nibbits.com/articles/view/10/writing-your-own-starcraft-ii-ai (web)
[12] Yi, S., Heo, J., Cho, Y., Hong, J.: Programming method for improving performance of artificial intelligence on StarCraft. In: Proceedings of 2006 Winter Korea Game Society Conference, pp. 141–146 (2006)
[13] Yi, S., Heo, J., Cho, Y., Hong, J.: Adaptive Reasoning Mechanism with Uncertain Knowledge for Improving Performance of Artificial Intelligence in StarCraft. Journal of The Korean Society for Computer Game 7, 19–32 (2005)
[14] Hartness, K.: Robocode: Using Games to Teach Artificial Intelligence. Journal of Computing Sciences in Colleges 19(4), 287–291 (2004)
[15] Wright, I., Marshall, J.: RC++: a rule-based language for game AI. In: Proceedings of First International Conference on Intelligent Games and Simulation (2000)
[16] YGosu game forum, http://www.ygosu.com (web)
[17] Unofficial ProGamer Ranking, http://pgr21.com (web)
[18] BWChart Replay Analyzer, http://bwchart.teamliquid.net/us/bwchart.php (web)
[19] WinMPQ, http://shadowflare.gameproc.com (web)
[20] StarCraft:BroodWar Artificial Intelligence Patch, http://aistarcraft.sourceforge.net (web)
[21] Official Competition Maps for StarCraft: BroodWar, http://www.mapdori.com (web)