2012 IEEE 15th International Conference on Computational Science and Engineering

DCA:Dynamic Challenging Level Adapter for Real-time Strategy Games

Shin-Hung Chang
Dept. of Computer Science and Information Engineering,
Fu-Jen Catholic University,
Taipei, Taiwan.
shchang@csie.fju.edu.tw

Nai-Yan Yang
Dept. of Computer Science and Information Engineering,
Fu-Jen Catholic University,
Taipei, Taiwan.
400226014@mail.fju.edu.tw

***Abstract*—Recently, Real-Time Strategy (RTS) games, such as Star Craft and Age of Empire, become more and more popular. Reasons of these RTS games attracting many game players are not only the fancy game presentation but also challenging game AI of computer opponents. In order to match game challenging level to different game players, these RTS games always provide several default difficulty levels for game players' choosing. However, settings of these difficulty levels cannot always accommodate challenging level requirements of different level game players. Therefore, this paper proposes a Dynamic Challenging Level Adapter (DCA) mechanism to automatically adapt computer opponent's behaviors for different game players. Each game player doesn't need to choose a difficulty level in advance and the game AI of computer opponent controlled by the DCA mechanism can dynamically adapt for meeting the challenging level of each game player. This paper proposes that the warrior is the most important element to dominate the result of each match in RTS games. Therefore, the main idea underlying the DCA mechanism is based on analyzing warrior capabilities between two game players and real-time adjusting action strategies to fight with its opponent. In order to test the effectiveness of the DCA mechanism, this study applies the most popular RTS game, Star Craft II (SC II), as the experimental game platform. In the experiments, this study uses default difficulty level AI of SC II game to emulate game players and makes these emulated game players to fight with the opponents controlled by the DCA mechanism. Additionally, a challenging rate (CR) formula is proposed as a strength evaluation between two opponents. From the experimental results, the difference of CR value of computer opponent improved by the DCA mechanism can be reduced by more than 80%. Additionally, the duration of a game match is usually double. Furthermore, the winner is always decided in the rear of each game match.**

Keywords: Challenging Rate (CR), Challenging Level Adapter, Real-time Strategy (RTS) Game, and Artificial Intelligence (AI).

I. Introduction

Nowadays, competition is intense at various commercial games, especially for Real-time strategy (RTS) games. Many RTS games always provide difference level computer opponents, called "game AI" in this paper, to attract game players for increasing game playability. However, requirements of a good game AI in a RTS game are not only the competitive strength, but also trigging players' interests. Each RTS game is not a kind of easy-to-play games and game players need to spend much time on being familiar with the RTS game operations. The key of a good game AI is to attract game players in playing a RTS game before the game players lose their interests of this RTS game. Generally, a RTS game provides several default difficulty levels for game players' choosing. Take Age of Empire II (AOE II) for example [2]. This RTS game provides four difficulty levels, including standard, moderate, hard, and hardest, respectively. However, these default difficulty levels may not match the requirement of each game player. In other words, it is hard for a game player to choose a default challenging level which is exactly suitable for him/her. Therefore, this paper proposes a game AI control mechanism, called Dynamic Challenging Level Adapter (DCA), to automatically generate a good game AI of players' computer opponent. With the DCA mechanism, the challenging level of y computer opponent can dynamically match different level players.

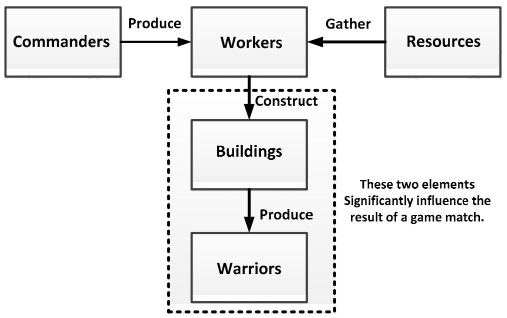


Figure 1. Relations among basic elements in a general RTS game.

Main idea underlying the proposed DCA mechanism is to analyze the basic elements, including commanders, workers, resources, building and warriors, in a general RTS game. Figure 1 presents the relations among these basic elements. At the beginning of each game, a commander is given. This commander will produce workers. The workers are used to gather resources for constructing many different functional buildings. These functional buildings can produce various kinds of warriors to attack player's opponent and win the game match. Form the analysis, this study proposes that building construction and warrior producing control are two magnificent elements that significantly influence the final result of each game match. Therefore, according to the RTS game strategy and characteristics of different warriors, this study classifies types of warriors into three categories, rush, supply, and subdued types. By integrating and controlling these three types of warriors, the proposed DCA mechanism automatically generate game strategy of a computer opponent.

Additionally, the default difficulty level of a RTS game usually provides stereotyped computer opponent whose actions always follow the same pattern defined in advance. This paper calls this situation a problem of rigid opponent behaviors. The stereotyped computer opponents always build the same buildings in a fixed order, produce the same warriors, and start to attack at the same time point. In this situation, game players only observe or play several game matches and the rigid behaviors of their computer components will be saw through. Therefore, the DCA mechanism can keep competitive strength of each game match until the end of game match and narrow down the level difference between computer opponents and game players. Additionally, in order to solve this rigid computer opponent problem, the DCA mechanism improves game AI with a non-determined wheel spinning strategy. Except dynamically adapting challenging levels for different players, the DCA mechanism can run different strategies for increasing the game delight, and the match result can be different even if facing the same game player.

This paper uses the most popular game, Star Craft II (SC II), as the experimental platform [1]. This study selects several default AI levels in SC II game and implements DCA mechanism to produce the game AI of the computer opponent. This study uses a Challenge Rate (CR) to evaluate capability between game player and his computer opponent. The experimental results show that CR value of computer opponent improved by the DCA mechanism adjusts warriors training behavior to compete with game players, and the difference of CR value between default AI and AI improved by the DCA mechanism is reduced by more than 80%. The winner will always be decided in the rear of each game match. Therefore, the competitive challenge of a RTS game with the DCA mechanism is higher than that of traditional RTS game with fixed difficulty level defined in advance. Therefore, with DCA mechanism, game players can play with more challenging computer opponents in each game match of a RTS game and have much fun.

The rest of this paper is organized as follows. Section II presents related work of real-time strategy game. The proposed DCA mechanism will be presented in section III. The experimental result and analysis are presented in section IV. Finally, this paper states the conclusions in section V.

II. Related Work

A. *Previous research*

Several researchers have studied several issues in RTS games. In [3], J. K. Olesen et al. used NEAT (Neuroevolution of Augmenting Topologies) and rtNEAT (real-time NEAT) algorithm to produce the game AI of computer opponents on a RTS game platform, called "Globulation". However, this RTS game is too simple and has a large difference with the current popular RTS games. Therefore, in [5], K. Stanley et al. directly applied reNEAT algorithm to produce a game AI of computer opponents and focused on the issue of how to influence the match result through controlling behavior of each warriors. K. Stanley et al. took NERO [6] as the experimental RTS game platform and improved its AI to avoid eliminating enemies efficiently by its algorithm. In [7], J. Ludwing et al. proposed that game AI can changes different strategies through a weighted tree. By combining an AI tool and automatic state constructions, J. Ludwing et al. generate a script enhance learning algorithm. In [8], Marc Ponsen et al. handle a game platform to do experiment for unit moving and improve the efficacy of strategies and game balance mechanism. In [9], R. Hunicke et al. think that the fixed difficulty level options are few in most of current games. Most of game players will feel that the default RTS game setting is either too hard or too easy. This will make game players bored in playing most of the RTS games. In [9], R. Hunicke et al. used a First Person Shooter (FPS) game, called Half Life [10]", as platform. R. Hunicke et al. applied inventory theory to define an algorithm for adjusting level dynamically. However, the FPS game type is different from the most of popular RTS games. To summarize the previous research, algorithms proposed by the above-mentioned research have a common problem. Their proposed game AI algorithm always react the same strategies as their opponent's similar strategies. In order to solve this problem, the main idea underlying the proposed DCA is to analyze the warrior construction and then do the action instead of barely simulating the actions of its opponent. Therefore, with the DCA mechanism, the result of game match is often different even if its opponent is the same and the competition strength will increase.

B. *Experimental platform*

This paper uses the popular RTS game, Star Craft II (SC II), as the experimental platform. SC II, created by BLIZZARD corp., is one of competition events in WCG (World Cyber Games) competition. Two main reasons for using SC II in this study as experimental platform are: (1) SC II is an orthodox RTS game which contains complete elements of a RTS game, including acquisition, construction, attack and other factors. (2) SC II provides complete built-in tools, including functions and a variety of data output interfaces, to implementing a experimental game AI.

III. Proposed Mechanism

This paper proposes a Dynamic Challenging Level Adapter (DCA) mechanism to automatically adapt computer opponent's behaviors for different game players. Figure 2 presents the flowchart of the proposed DCA mechanism. The DCA mechanism controls a computer opponent AI and play with a human game player or another computer opponent. At the beginning of this DCA control scenario, the DCA mechanism will get its opponent player event. From this event, the DCA mechanism will decide whether the situation is suitable for starting rush attack. Additionally, the DCA mechanism will evaluate its opponent's strength extension because of its opponent's producing supply warriors. After considerations of rush attack and strength extension, the DCA mechanism will build weighted probability wheel which identifies the priority of producing each kind of warrior according to the mutual inhibition table and the game AI strategies, rush, supply, and subdued strategies. According to the weighted probability wheel, the DCA mechanism will decide whether it should produce warriors by rotating this wheel and handle attack. Following the attack decision, the DCA mechanism constructs the correlated buildings and produce specific warriors. If the DCA mechanism's decision is not handling attack, the DCA mechanism will handle construction. Finally, the DCA control will start again to get its opponent's event and follow this control flowchart. This DCA control mechanism will be triggered once every thirty seconds.

Because the most important condition which influences the result of a game match is warrior producing, the main idea underlying the DCA mechanism is to handle dynamic challenging level adaption only by deciding which type of warrior should be produced first at each time point. Furthermore, only controlling warrior producing is simple and easy to implement a good game AI for computer opponents of game players.

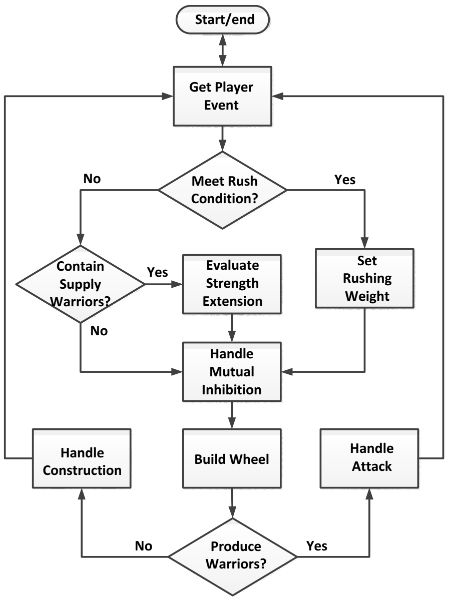


Figure 2. The flowchart of the proposed DCA mechanism.

TABLE I. A list of notations, abbreviations, and corresponding meanings.

Notations and Abbreviations	Meanings
$\mathbf{R_o}$	The race of player o, where o='p' means game player and o='D' means game player's computer opponent handled by DCA.
$\mathbf{\lvert R_o \rvert}$	Number of warrior types of player o using race R_o.
$\mathbf{W_o^i}$	Number of type i warrior of player o.
$\mathbf{W_o^r}$	Rush warrior of player o.
$\mathbf{W_p^*=\sum_{i=0}^{K(r)} W_P^i}$	Number of total game player's warriors.
$\mathbf{B_P}$	Number of total game player's Building.
$\mathbf{S_p^*=\sum_{i=0}^{k} W_P^i}$	Number of supply warrior of game player, where supply warriors are numbered from 0 to k.
$\mathbf{G_p}$	Number of the Worker(Gather) of game player.
$\boldsymbol{C_r^i}$	Cost of the type i warrior of race r.
M[i,j]	Value at row i, column j of mutual inhibition table.
H[i]	weighted probability wheel
α=2	2, number of commander.
β=3	3, number of total warriors.
γ=8	8, maximum allowed workers gather from a resource unit.
δ=0.1	0.1, 10% warrior strength extionsion with a supply warrior.
ρ=0.5	0.5, produce at least same level warrior with game player.

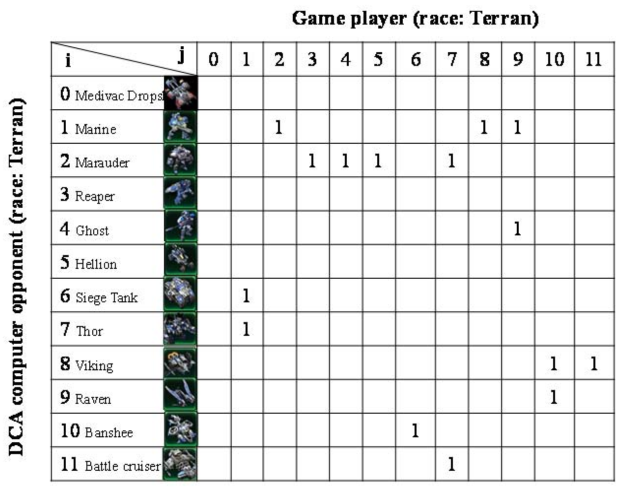
In order to clearly describe details of the DCA mechanism, TABLE I presents a list of notations, abbreviations, and corresponding meanings. The DCA mechanism receives two parameters, called game player, **P**, and its computer opponent, **D**, controlled by the DCA mechanism. The detail algorithm of the DCA mechanism is shown as follows:

```
DCA Mechanism (game player (P), computer opponent by DCA(D))
(1)  { //The DCA mechanism will be triggered once every 30 seconds.
(2)     DCA_event = getEvent( );
(3)     DCA_Build(DCA_event);
(4)     DCA_Attack( );
(5)     DCA_Produce(worker);
(6)     for (i=0; i < |$\mathbf{R_D}$|; i++)
(7)     {  $\mathbf{W_D^i}$=0; }
(8)     for (i=0; i < |$\mathbf{R_P}$|; i++)
(9)     {  $\mathbf{W_P^i}$=0; }
(10)    if (( $\sum \mathbf{B_P} \geqq \alpha$ ) and ( $\sum \mathbf{W_P} \leqq \beta$ ))
(11)    {  $\mathbf{W_D^r = G_p} / \gamma$; }
(12)    else
(13)    {   if($S_p^* > 0$)
(14)        {  for( i = 0; i<k ; i++)
(15)           {  $\mathbf{W_p^i = W_p^i} \times (1 + (\mathbf{S_p^*} / \mathbf{W_p^*}) \times \delta)$; } }
(16)        for( i = 0; i< |$\mathbf{R_P}$|; i++)
(17)        {  for (j = 0; j <|$\mathbf{R_D}$| ; j++)
(18)           {  if ( M[i,j] )
(19)              {  $\mathbf{W_D^j = C_{R_P}^i / C_{R_D}^j \times W_p^i}$; } }}
(20)     }
(21)     for (i = 0; i < |$\mathbf{R_D}$|; i++)
(22)    {  m = 0;  $\mathbf{W_D^*}$= 0;
(23)       if( $\mathbf{W_D^i}$ == 0)
(24)       {  $\mathbf{W_D^i = W_p^i} \times \rho$; }
(25)       else
(26)       {  Army[m] = i;
(27)          $\mathbf{W_D^*}$ += $\mathbf{W_D^i}$;
(28)          m ++; }
(29)    }
(30)     for (i = 0; i < m ; i++)  //H[-1]=0
(31)    {  H[i] = $\mathbf{W_D^{Army[i]} / W_D^* + H[i-1]}$; }
(32)     if ( $\sum \mathbf{W_D} < \sum \mathbf{W_p} * (1 + \mu)$)
(33)    {    r = random(0,1);
(34)         for (i = 1; i <m; i++)
(35)            if(r < H[i])
(36)            {  DCA_Produce(Army[i]); }
(37)     }
(38)  }// end of the DCA Mechanism
```

(2)~(5) indicates that the DCA mechanism calls getEvent( ) provided by SCII and gathers all game events into the DCA_event. Through DCA_build(DCA_event), the DCA mechanism handles building construction by DCA_event. DCA_Attack( ) handles attack strategies, including rush and normal attacks. DCA_Produce(worker) handles producing workers to develop economics. (6)~(9) indicates handling initializations. In (10)~(11), the DCA mechanism will decide whether its opponent focuses on developing economic by condition ($\sum \mathbf{B_P} \geqq \boldsymbol{\alpha}$) and ($\sum \mathbf{W_P} \leqq \boldsymbol{\beta}$). **α=2** and means that a game player begins to build second base (commander in SCII). **β=3** means that the number of warrior of a game player is less than 3. In most of RTS games, three warriors is capable of blocking its opponent's rush attack or doing rush attack. If these two conditions stands, the DCA mechanism increases the weight of rush warriors, $\mathbf{W_D^r}$. Take race "Terran" for example, $\mathbf{W_D^2}$ and $\mathbf{W_D^3}$ are used to handle rush attack. Increasing the weight of rush warrior, the DCA mechanism will produce rush warrior through DCA_Produce(Army[i]). Additionally, because rush warriors are good at attacking workers, weight of rush warrior is set as proportional to G_P/γ, where G_p is the number of workers and γ is the maximum allowed workers gather from a resource unit. In the SCII game, γ=8.

In (13)~(15), the DCA mechanism will evaluate the opponent's strength extension because of its opponent's producing supply warriors. Take race "Terran" for example, "Medivac" is a kind of supply warrior. Not all of warrior can be supplied by supply warrior, so the DCA mechanism only calculates warriors which can be supplied. With formula $\mathbf{W}_p^i = \mathbf{W}_p^i \times (1 + (\mathbf{S}_p^* / \mathbf{W}_p^*) \times \delta)$, the weight of type i warrior increases δ percentage. δ =0.1 means that warrior strength extension will increase 10% with a supply warrior. The number of supply warrior is arranged from 0 to k. In a RTS game, warriors alive still have complete attack strength even if the warriors only remain 1 Heal Point. Each supply warrior supplies only one warrior at the same time, so the DCA mechanism calculates the average number of warrior, $(\mathbf{S}_p^*/\mathbf{W}_p^*)$, that can be supplied.

In (16)~(20), the DCA mechanism decides which warrior should have higher priority to be produce for handling game attack. A mutual inhibition (M) table provided by SCII official website will be applied by the DCA mechanism. Figure 3 presents a M table between two same races "Terran". The vertical axis of M table represents the type i warrior of the computer opponent controlled by the DCA and the horizontal axis of M table represents type j warrior of a game player. If the value of $\mathbf{M}$[i,j] is 1 (true), this indicates that type i warrior of the DCA computer opponent can inhibit type j warrior of a game player. According to this M table, the DCA mechanism will calculate the weight of $\mathbf{W}_D^j = \mathbf{C}_{R_P}^i / \mathbf{C}_{R_D}^j \times \mathbf{W}_p^i$, where $\mathbf{C}_{R_P}^i$ is the cost of producing type i warrior of race of a game player, $\mathbf{R_P}$, and $\mathbf{C}_{R_D}^j$ is the cost of producing type j warrior of race of a computer opponent controlled by the DCA, $\mathbf{C}_{R_D}^j$.


Game player (race: Terran)
DCA computer opponent (race: Terran)

i \ j	0	1	2	3	4	5	6	7	8	9	10	11
0 Medivac Drops												
1 Marine			1						1	1		
2 Marauder				1	1	1		1				
3 Reaper												
4 Ghost										1		
5 Hellion												
6 Siege Tank		1										
7 Thor		1										
8 Viking											1	1
9 Raven											1	
10 Banshee							1					
11 Battle cruiser								1				

Figure 3. Mutual Inhibition Table of two same "Terran" races.

In (21)~(24), if the resource is not enough for producing specific warriors decided by M table immediately, the DCA mechanism will produce some initial warrior for being destroyed. The DCA mechanism set the weight of type i warrior, $\mathbf{W}_D^i = \mathbf{W}_p^i \times \boldsymbol{\rho}$. Value of $\boldsymbol{\rho}$ is usually less than 1 and this will prevent from producing warriors specified by M table once the resource is enough. $\boldsymbol{\rho}$ is configured as 0.5 in this study and this represents the DCA mechanism will produce same level warrior of its opponent.

In (26)~(31), the DCA mechanism calculates the sum of weight of all producing in $\mathbf{W}_D^*$. **Army[ ]** records the all warriors, type **Army[0]**, type **Army[1]**,..., and type **Army[m-1]**, evaluated by the DCA mechanism. There are m kinds of warriors having higher priority to produce. With each weight calculated, the DCA mechanism calculates the percentage of each type warrior in the weighted probability wheel, **H[ ]**. Each section of the **H[ ]** is mapped to the warrior type recorded in an array **Army[ ]**. Figure 4 presents a description of the weighed probability wheel, **H[ ]**, calculated in the DCA mechanism and the mapped array **Army[ ]**.

In (32)~(36), the DCA mechanism will generate a random number r which value is between 0 to 1. If value of r is located in **H[2]**, and then the DCA mechanism will apply **DCA_Produce( )** procedure to produce type **Army[2]** warrior.

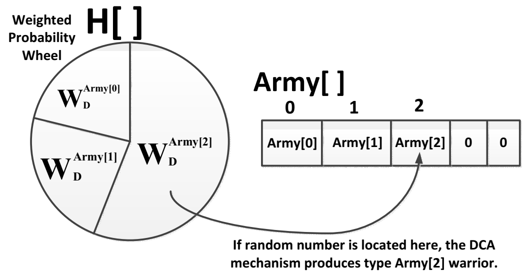

Weighted Probability Wheel
H[ ]
$W_D^{Army[0]}$
$W_D^{Army[1]}$
$W_D^{Army[2]}$
Army[ ]
0
1
2
Army[0]
Army[1]
Army[2]
0
0
If random number is located here, the DCA mechanism produces type Army[2] warrior.

Figure 4. Descriptions of the weighed probability wheel, H, calculated in the DCA mechanism.

IV. Experimental Results and Analysis

A. *Challenging Rate (CR) Formula*

In order to test the effectiveness of reducing strength between the DCA computer opponent and game player, an evaluation index should be fairly identified. This paper defines a challenge rate (CR) formula to represent the national power of a player. The player is one of human game player, fixed difficulty level player, and DCA controlled player. In this paper, the CR formula is defined as the following equation (1).

$$\mathrm{CR} = \frac{\mathrm{A}}{\mathrm{a}} + \frac{\mathrm{H}}{\mathrm{b}} + \frac{\mathrm{P}}{\mathrm{c}} + \frac{\mathrm{C}}{\mathrm{d}} \tag{1}$$

In the CR formula, parameter A(Attack) represents the sum of all warriors' attack power, parameter H(Health Point) represents the sum of all game units' health point, parameter P(Population) represents the sum of all game population, and parameter C(Cost) represents the total cost of constructing all game units. a, b, c, and d are four constants for handling normalization in this formula. a, b, c, and d are configured as 5, 300, 1, and 100, respectively. The reasons why select these parameters, A, H, P, and C, are described as follows:

- *Attack (A)*

In a RTS game, most of warriors and some functional buildings have contributed attack power. The attack power of a game unit indicates that this game unit can give harm to its opponent's units. The main influences of the value of attack power is significantly related to the capability of destroying its opponent, so the attack power is considered in the CR formula of this study.

- *Health Point (H)*

  In a RTS game, health points of all game units, including warriors, workers, and buildings, represent durability of the game units. When a game unit is attacked, its contained health point is definitely decreased. When the amount of health point decreases to zero, it represents that game units are totally eliminated. Therefore, health point is an evaluation index of defense power. Additionally, game units have shield value which is directly relevant to health point, so health point is considered in this study.

- *Population (P)*

  In a RTS game, the population of each player in each game match has a limitation. Each game unit will contribute one population value in each game match. When a player plans to produce any game unit, warriors, workers, and functional buildings, the player needs to use the limited resource to produce the most efficient game units. Value of population indicates the number of game unit which player can control and the power of nation, so population is considered in this study.

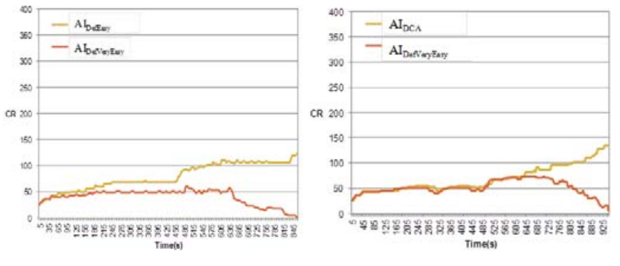
- *Cost (C)*

  In a RTS game, each game unit has its specific producing resource cost. The value of cost indicates how a game player can do transformation from resource to game units. Take SCII for example. Minerals and gas are two kinds of resources in SCII. A game player must consume resource to construct warriors, workers, and functional buildings. In order to clearly calculate the value of cost, this study combines two resources into one cost value. In SCII RTS game, a warrior, "Archon", can be fused with two warriors, "Dark Templar" or with two warriors, "High Templar". The cost of "Dark Templar" is 125× minerals+125×gas and the cost of "High Templar" is 50 ×minerals+150×gas. Therefore, 125x+125y=50x+150y and x:y=1:3. In this study, 1 cost = 3×minerals +1×gas.

B. The Effectiveness of Dynamic Game AI Controlled by the DCA Mechanism

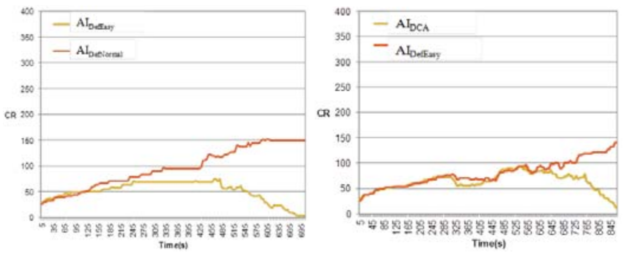
In order to clearly present the experimental results, AI_{DCA} represents the computer opponent AI controlled by the DCA mechanism and AI_{Def} presents the computer opponent AI controlled by the difficulty AI. Additionally, $AI_{DefVeryEasy}$, $AI_{DefEasy}$, $AI_{DefNormal}$, $AI_{DefHard}$, and $AI_{DefVeryHard}$, are five different difficulty levels of game AIs provided by SCII RTS game, respectively. In order to present the effectiveness of reducing the competition with using the DCA mechanism, the experiment results are divided into four battle combinations. (a) of each figure presents the battle combination between two different difficulty level AI and (b) of each figure presents the battle combination between the different difficulty level AI and DCA mechanism AI. The vertical axis of each figure indicates the variation of CR value and the horizontal axis of each figure indicates the playing time. Each figure contains two CR traces of each pair of battle combinations.

In these experiments, this study uses the difficulty level game AI as the emulated game players with different capability. The (a) of each figure simulates the battle combination of two different general game players and the (b) of each figure simulates the battle combination of general game player and game opponent controlled by the DCA mechanism. The playing time of each game match continues 6 to 10 minutes.



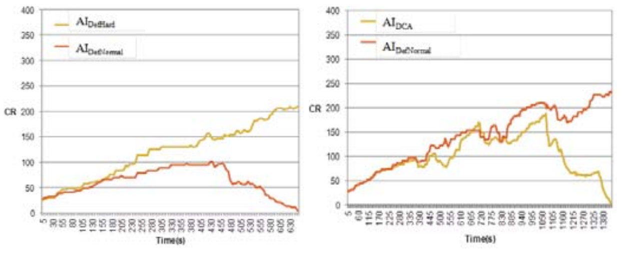
(a) **(b)**

Figure 5. (a) **$AI_{DefEasy}$ v.s. $AI_{DefVeryEasy}$.** (b) **AI_{DCA} v.s. $AI_{DefVeryEasy}$.**



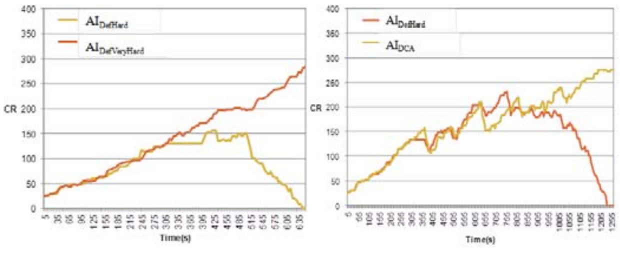
(a) **(b)**

Figure 6. (a) **$AI_{DefEasy}$ v.s. $AI_{DefNormal}$.** (b) **AI_{DCA} v.s. $AI_{DefEasy}$.**



(a) **(b)**

Figure 7. (a) **$AI_{DefHard}$ v.s. $AI_{DefNormal}$.** (b) **AI_{DCA} v.s. $AI_{DefNormal}$.**



(a) **(b)**

Figure 8. (a) **$AI_{DefHard}$ v.s. $AI_{DefVeryHard}$.** (b) **AI_{DCA} v.s. $AI_{DefHard}$.**

Additionally, the race setting of each pair of battle combination is the same. From the experimental results of Figure 5, Figure 6, Figure 7, and Figure 8, it is observed that the difference of two CR traces in (a) of each figure is more close to that of two CR traces in (b) of each figure. The experimental results indicate that the DCA mechanism effectively increases the competition strength. Furthermore, playing with the game AI controlled by the DCA mechanism, a game player doesn't need choose any difficulty level. The improvement of the DCA mechanism is significant.

TABLE II, TABLE III, and TABLE IV present more details in the experiments. The second column of TABLE II and TABLE III indicates average value of CR for each pair of battle combination. Value 1.88 of the first row in TABLE II is the average value of CR for the battle combination of the AI_{DCA} and $AI_{DefVeryEasy}$. Value 9.94 of the first row in TABLE II is the average value of CR for the battle combination of $AI_{DefEasy}$ and $AI_{DefVeryEasy}$. Column "AI_{DCA}/AI_{Def}" indicates the value of the second column divides the value of the forth column. The last column indicates the value of "1-(AI_{DCA}/AI_{Def})". In TABLE II, the value of "AI_{DCA}/AI_{Def}" in the first row is 18.92%. Therefore, the improvement of reducing CR difference between battle pair, AI_{DCA} and $AI_{DefVeryEasy}$, and battle pair, $AI_{DefEasy}$ and $AI_{DefVeryEasy}$, is 81.08%. From TABLE II and TABLE III, the experimental results show that competition strength of computer opponent controlled by the DCA mechanism can significantly close to that of each kind of difficulty level AI by more than 80%.

TABLE II. The average value of CR with playing time of 5 minutes.

AI_{DCA} VS. AI_{Def}	**CR**	**AI_{Def} VS. AI_{Def}**	**CR**	**AI_{DCA}/AI_{Def}**	**1 - AI_{DCA}/AI_{Def}**
AI_{DCA} VS. very easy	1.88	easy VS. very easy	9.94	18.92%	81.08%
AI_{DCA} VS. normal	1.92	easy VS. normal	8.32	23.09%	76.91%
AI_{DCA} VS. easy	1.76	normal VS. easy	8.32	21.11%	78.89%
AI_{DCA} VS. hard	1.72	normal VS. hard	14.15	12.17%	87.83%
AI_{DCA} VS. normal	1.92	hard VS. normal	14.15	13.57%	86.43%
AI_{DCA} VS. very hard	1.38	hard VS. very hard	3.59	38.48%	61.52%

TABLE III. The average value of CR with playing time of 10 minutes.

AI_{DCA} VS. AI_{Def}	**CR**	**AI_{Def} VS. AI_{Def}**	**CR**	**AI_{DCA}/AI_{Def}**	**1 - AI_{DCA}/AI_{Def}**
AI_{DCA} VS. very easy	2.59	easy VS. very easy	20.50	12.63%	87.37%
AI_{DCA} VS. normal	11.01	easy VS. normal	30.78	35.79%	64.21%
AI_{DCA} VS. easy	4.46	normal VS. easy	30.78	14.50%	85.50%
AI_{DCA} VS. hard	5.86	normal VS. hard	46.72	12.55%	87.45%
AI_{DCA} VS. normal	11.01	hard VS. normal	46.72	23.58%	76.42%
AI_{DCA} VS. very hard	8.28	hard VS. very hard	36.73	22.55%	77.45%

TABLE IV. The game duration of different game match.

AI_{DCA} VS. AI_{Def}	**Time(s)**	**AI_{Def} VS. AI_{Def}**	**Time(s)**	**AI_{DCA}/AI_{Def}**	**1 - AI_{DCA}/AI_{Def}**
AI_{DCA} VS. very easy	950	easy VS. very easy	45	2111.11%	2011.11%
AI_{DCA} VS. normal	1425	easy VS. normal	110	1295.45%	1195.45%
AI_{DCA} VS. easy	1120	normal VS. easy	110	1018.18%	918.18%
AI_{DCA} VS. hard	1235	normal VS. hard	185	667.57%	567.57%
AI_{DCA} VS. normal	1425	hard VS. normal	185	770.27%	670.27%
AI_{DCA} VS. very hard	905	hard VS. very hard	320	282.81%	182.81%

TABLE IV presents the game duration of each game match of different battle combination. It is observed that the game duration of the game match with the DCA mechanism is two times longer than that of two difficulty level AI. With the DCA mechanism control, the winner will always be decided in the rear of each game match.

V. CONCLUSIONS

Reasons of RTS games attracting many game players are not only the fancy game presentation but also challenging AI of computer opponents. In order to match game challenging level to different game players, these RTS games always provide several default difficulty levels for game players' choosing. However, settings of these default difficulty levels cannot always accommodate challenging level requirements of different game players. In order to solve the problem of rigid difficulty level provided by RTS games, many game AI control algorithms proposed by previous researchers. Their proposed game AI algorithms always adopt simulating the same strategies as their opponent's. With these game AI algorithms, the scenario of each game match is usually similar and the result of each game match is rigid. In this paper, the DCA mechanism is proposed. Main idea underlying the proposed DCA mechanism is to analyze the warrior construction and then produce a smart strategy and do action instead of barely simulating the actions of its opponent. Therefore, with the DCA mechanism control, the result of each game match is often different even if its opponent is the same and the competition strength will significantly increase. Therefore, the proposed DCA mechanism can automatically generate an intelligent computer opponent for different game players. In order to test the effectiveness of dynamic game AI controlled by the DCA mechanism, this study handles many experiments on the most popular RTS game platform, Star Craft II (SC II). In these experiments, this study uses the difficulty level game AIs in SC II as the emulated game players with different capability. Furthermore, this study handles another computer opponent controlled by the DCA mechanism and makes this DCA controlled opponent fight with default difficulty AIs in SC II. Additionally, a challenging rate (CR) formula is proposed as the strength evaluation of two opponents. From the experimental results, the difference of CR value of computer opponent improved by the DCA mechanism can be reduced by more than 80%. Additionally, the duration of a game match is usually double. Furthermore, the winner is always decided in the rear of each game match. Therefore, with the DCA mechanism, each game player doesn't need to choose a default difficulty level in advance and the game challenging level of computer opponent can automatically adapt to match that of each game player.

REFERENCES

[1] Star Craft II (SC II), RTS game developed by Blizzard Entertainment, official website http://us.battle.net/sc2/en/.

[2] Age of Empire II (AOE II), a RTS games developed by microsoft. http://www.microsoft.com/games/age2.

[3] J. K. Olesen, G. N. Yannakakis, and J. Hallam, "Real-time challenge balance in an RTS game using rtNEAT," in Proceedings of the IEEE Symposium on Computational Intelligence and Games, Perth, December, 2008.

[4] Globulation 2, Free software RTS game with a new take on micro management, 2008. http://www.globulation2.org/.

[5] K. Stanley, B. Bryant, and R. Miikkulainen, "Real-time evolution in the NERO video game," in Proceedings of the IEEE Symposium on Computational Intelligence and Games, Colchester, UK, April 2005.

[6] NERO's official public website is http://nerogame.org and the project's official e-mail address is nero@cs.utexas.edu.

[7] J. Ludwig and A. Farley, "A learning infrastructure for improving agent performance and game balance," in Proceedings of the AIIDE Workshop on Optimizing Player Satisfaction, 2007.

[8] Marc Ponsen, Peter Spronck, Hector Mu Avila and David Aha, "Knowledge Acquisition for Adaptive Game AI," Science of Computer Programming, Springer, Netherlands 2007.

[9] R. Hunicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in Proceedings of 19th Nineteenth National Conference on Artificial Intelligence (AAAI'04) 2004.

[10] "Half-Life", A science fiction first-person shooter video game developed by Valve Corporation, wiki introduction. http://en.wikipedia.org/wiki/Half-Life_%28video_game%29.