Genetic Algorithm for optimizing Game using users' adaptation

SANGWON UM, TAEYONG KIM, JONGSOO CHOI
Image Information Lab, GSAIM
Chung-Ang University
221, Heuksuk-Dong, Dongjak-Gu, Seoul
REPUBLIC OF KOREA

Abstract: - This paper describes how we use Genetic Algorithm to modify the level of difficulty in a game based on a user. We use Factor of Difficulty Control (next FDC) as Gene and Factor of Users' Adaptation to a game (next FUA) as Fitness. Then we implemented a game that matches gamers' adaptation in our previous research. Games are composed of occurring sequential events and playing time is limited. In a game, because group searching is impossible, we must find the optimum as quickly as possible, like Genetic Algorithm finding the optimum as a Generation search. So we propose Cyclic Search Algorithm (next CSA) to create Genes quickly and evaluate the Fitness of Genes faster. By using CSA, we compose a game with more appropriate Genes for a gamer within a playing-time, therefore reaching an improved result than our previous research.

Key-Words: Games, Applications, Difficulty control, Game artificial Intelligence, User adaptive programming.

1 Introduction & Overview

Recently, the game industry has come up to being a part of important business sector alongside Cinema or Animation. International corporations like Microsoft or Sony are one of the main businesses in the game industry and even academies are showing interest in the gaming sector[10].

Because we are also focused on the gaming sector ourselves, we propose the following algorithm for Game A.I. But we'd like to explain the features of Computer Games in order to our proposed algorithm. Traditionally games have these following features

1. Must update scene every 1/24 sec.
2. Proceeded using sequential events by user.
3. Playing-time is limited and very short on artificial intelligence's point of view
4. Excitement is a vital part in a game, thus it doesn't need global optimum

If you use A.I for game, you must concern these features Developer is not focused on Game A.I because of Game graphic, he has enthusiasm to make scenes of 24 per second.

Recent development in hardware technology has increased the amount of usage in CPU resources for game A.I. Currently many companies are focusing on using CPU resources on game A.I is a key to success in a competitive gaming industry[8][9]. But most game A.I techniques so far have been focused on the realistic and smart behavior of game units or game appearance[9]. Most computer games have been created for one person to play. In the case of Chess or Checkers, there has to be two players to start a game, but if a person owns a computer, he can play games by himself at any time, thus making computer games prosperous. But recent games return to finding a good opponents using computer networking. Eventually people became accustomed to computer games and lost interest due to the lack of game A.I.

We think that game A.I must associate with not games' appearance but gameplay or intelligence. Controlling the level of difficult on a game is one of the important features to developing a game. Players want more difficult opponents but it offers limited opponents. It causes players to feel disgust for the game. Generally the factor of difficulty can be adjusted by changing the pace of a game or to increase the amount of enemies. We are focusing on controlling the level of difficulty.

we proposed an User adaptive Difficulty Control Algorithm(next UDAC) that controls the difficulty of a game based on the player's propensity and proficiency fundamental using Genetic Algorithm[11][12]. It is suitable to apply Genetic Algorithm into a game, the reason why as shown below.

1. There are infinite variations of optimum according to gamer.
2. It can't generalize the formula from the level of difficult on a game.
3. It doesn't necessary to global optimum.

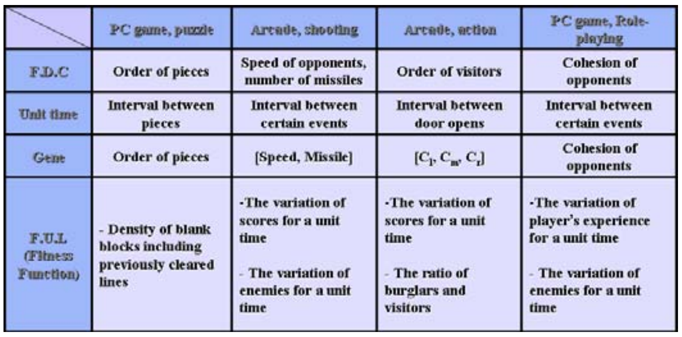
First means that searching space can be infinite, second means it is not continuous. During a game, simple or randomly appearing iterations of events occur and the harder game is not best one. Because of these reason, it is suitable to apply Genetic Algorithm into a game.

2 U.A.D.C

The Factors of Difficulty Control(FDC) and Factors of Users' Adaptation to a game(FUA). Generally we have mentioned already that FDC is increasing game speed or amount of the enemies. There exist a level of difficulty according to the pattern of FDC and it is different from players' propensities. Generally FUL is game score, level of gamer and playing-time. We compose a game with appropriate FDC for a gamers' FUL then make a game based on gamers' propensity

2.1 U.A.D.C in a game

We analysis representative game genre and identify possibility that U.A.D.C should have been possible to be how application to different genre game. As Table1 shown, we define general FDC and FUL about each game genre[12].



	PC game, puzzle	Arcade, shooting	Arcade, action	PC game, Role-playing
F.D.C	Order of pieces	Speed of opponents, number of missiles	Order of visitors	Cohesion of opponents
Unit time	Interval between pieces	Interval between certain events	Interval between door opens	Interval between certain events
Gene	Order of pieces	[Speed, Missile]	$[C_1, C_m, C_r]$	Cohesion of opponents
F.U.L (Fitness Function)	- Density of blank blocks including previously cleared lines	-The variation of scores for a unit time - The variation of enemies for a unit time	-The variation of scores for a unit time - The ratio of burglars and visitors	-The variation of player's experience for a unit time - The variation of enemies for a unit time

Table 1, FDC and FUL about each Game Genre

We use FDC as Gene and FUA as Fitness in the Table1. And also to prove this, we used the game Tetris as our research.

In the case of Tetris, there may be easy sequence of pieces or hard one for the users to play according to the sequence of pieces. If four I-shape pieces and a square piece are coming out, you lay down parallel four I-shape pieces and the rest of blank area will be occupied by square piece, then you can see removing 2 lines in your visual. This is an example of easy patterns. The next Fig. shows the example of hard sequence of pieces (Fig.1 and Fig.2 below).

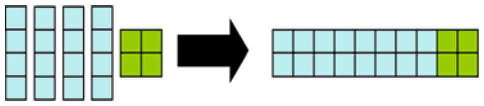


Fig.1, Easy sequence of pieces

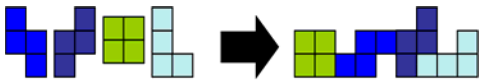
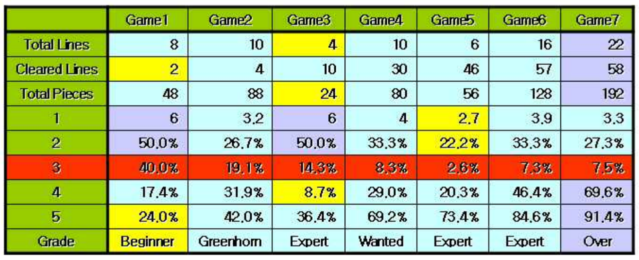


Fig.2, Hard sequence of pieces

And as a factor to measure the user adaptation we used density of blank block including clear lines as shown in Table2.



	Game1	Game2	Game3	Game4	Game5	Game6	Game7
Total Lines	8	10	4	10	6	16	22
Cleared Lines	2	4	10	30	46	57	58
Total Pieces	48	88	24	80	56	128	192
1	6	3.2	6	4	2.7	3.9	3.3
2	50.0%	26.7%	50.0%	33.3%	22.2%	33.3%	27.3%
3	40.0%	19.1%	14.3%	8.3%	2.6%	7.3%	7.5%
4	17.4%	31.9%	8.7%	29.0%	20.3%	46.4%	69.6%
5	24.0%	42.0%	36.4%	69.2%	73.4%	84.6%	91.4%
Grade	Beginner	Greenhorn	Expert	Wanted	Expert	Expert	Over

Table 2, the analysis of fitness function

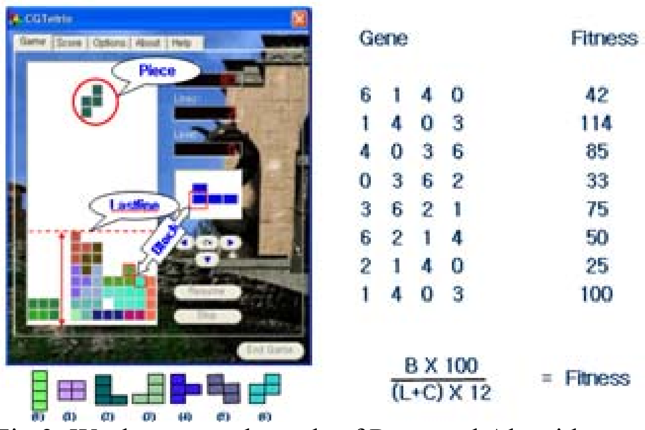
1. Blank blocks of each line:
   More than 25% - the beginner
   Less than 12% - the expert
2. The density of blank block except clear lines.
3. The density of blank block including clear lines.
4. The density of block per game board.
5. The density of block per game board including clear lines.

Usually the score of a game is an efficient way to appear users' adaptation, but density of blank block including clear lines provided better results and thus it is more useful in Tetris.

As formula below

$$Fitness = \frac{B \times 100}{(C+L) \times 12} (\%)$$

$$B: Blank\,Block \quad C: Clear\,line \quad L: Last\,line$$



Gene				Fitness
6	1	4	0	42
1	4	0	3	114
4	0	3	6	85
0	3	6	2	33
3	6	2	1	75
6	2	1	4	50
2	1	4	0	25
1	4	0	3	100

$$\frac{B \times 100}{(L+C) \times 12} = Fitness$$

Fig.3, Workspace and words of Proposed Algorithm

Fig.3 shows that Tetris is using the U.A.D.C and it description of the process. For the purpose of clear understanding, we defined some notations as 'Piece' or 'Block'. The Piece is composed of four blocks, and it is represented to the numbers 0 to 6. In the case of CG Tetris, the Game-board is composed of 12×23 blocks. The Fitness equation contains some elements like B, C, and L. B is the number of blank blocks in the game-board, C is the number of cleared lines, and L is the number of existing lines in the game-board.

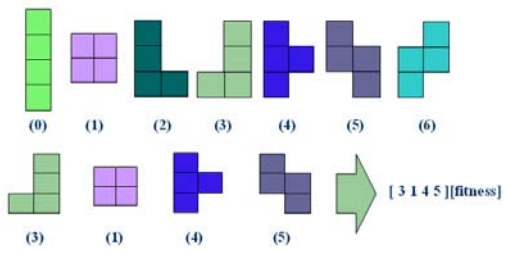


Fig.4, the pieces and encoding Gene

Fig.4 shows encoding gene, we use real encoding. If the time for one piece to fall to the bottom is T,

$$(L+C)\times 12(b) - B(b) = T\times 4(b)$$

$$(L+C)\times 12 = 4T + B \text{ (b means Block) - - - (1)}$$

then,

$$\frac{B\times 100}{4T+B} = F(\%)$$

$$100B = 4FT + FB$$

$$(100-F)B = 4FT$$

$$\frac{B}{T} = \frac{4F}{100-F}$$

$$\therefore \frac{B}{T} = C \qquad (\text{F : fitness, C : constant})$$

Therefore it makes constant number of blank blocks per T.

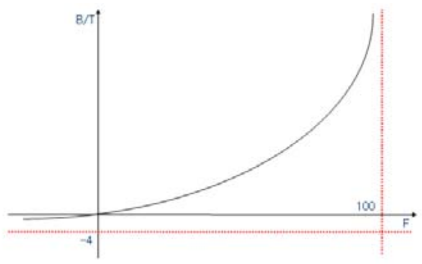


Fig.5. the graph of B/T follows Fitness

In our previous research, we control the difficulty of a game based on the player's propensity and proficiency fundamentally using Genetic Algorithm. As a result, we were able to see the clear difference between beginner and expert of this game in the number of clearlines[12].

2.2 The problems of Genetic Algorithm adapt

Searching time of algorithm is proportionally increased with searching a Gene in a game. Also we have mentioned already that searching space is limited as well, due to the fact that playing-time is limited. If a Generation is composed of 8 Genes, it spent 4T to evaluate a Gene and it spent 32T for a Generation. On the average, a game ends when it reaches 180T, thus it doesn't have enough time to find the optimum. Having concerns for the features of a game mentioned before, because games don't need a global optimum, it still shows a reliable result. But it is necessary to have quickly searching speed and wide searching space to complete a robust algorithm.

3 Cyclic Searching Algorithm

3.1 Making Gene

When we seek adaptive model for Sequential event driven environment, we attend to cyclic Crossover and sliding window method. We make Genes using sliding window memory in order to evaluate more Genes. Consequently it is possible to make an extension for searching space.

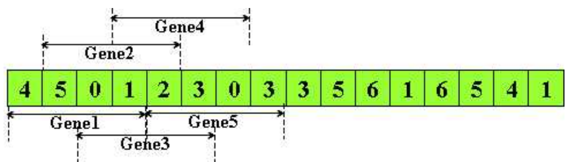


Fig.6, Cyclic Searching Algorithm

Through the changes we can see the algorithm's immediate response. If we call N the number of all the pieces that are played in a single game, then we search for the Genes of N/4 in U.A.D.C and then search Genes of N-G+1 in proposed algorithm..

3.2 Evaluation Gene

The evaluating time of a Generation is reduced as much as possible by using similarity of each Genes.

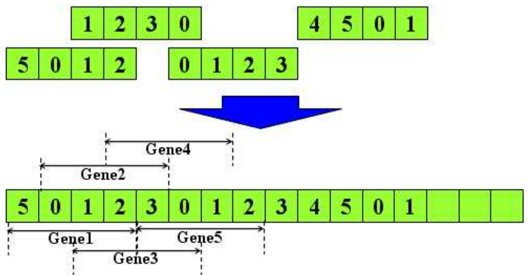


Fig.7, Evaluating Fitness of Gene

As Fig.7 shown, Gene1 is overlapping "12" with Gene2. We find similarity of permutation applying the head or tail of the Genes, and then search the Gene that is continuously overlapping the similar Gene. Even though each generation has its own Genes, if we call *k* the number of overlapped Chromosomes, then searching time of Generation is reduced by the amount of *k*T.

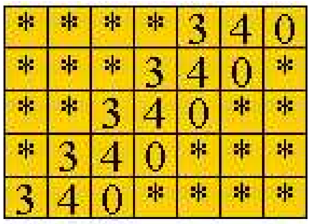
3.3 Schema on proposed algorithm and Game

First we focus on the pattern of events defined as FDC. There exist a level of difficulty according to the pattern of events and it is different from players' propensities Let's concern about the FDC in Tetris, even though a player has trouble with pieces "340" because it is too difficult for him to make a horizontal line, the position of "340" is irrelevant in the Gene. If we presented that using schema, the difference between schema "***340**" and "**340***" is insignificant.

4 Modified Schema Theory

4.1 Assumption

As an assumption, schema is the permutation of events (the pattern of integer that represents the piece in Tetris) and the position of schema is not affected. Exactly if the Gene contain schema "340," then the five genes shown on Fig.8 have same schema "340".



*	*	*	*	3	4	0
*	*	*	3	4	0	*
*	*	3	4	0	*	*
*	3	4	0	*	*	*
3	4	0	*	*	*	*

Fig.8, the schema for Cyclic Searching Algorithm

4.2 Influence of Crossover

If it performs Crossover which mate with neighboring Genes, then the portion of population is different from original Schema theory. We use O(H) as a substitute to δ(H), due to disregarding the location of *(don't care symbol) in proposed algorithm. Generally, O(H) is shorter than δ(H), and the Schema Theory of Cyclic Search Algorithm about Crossover is shown on formula below.

$$P_{dist}=P_c\frac{\delta(H)}{l-1},\quad P_{surv}=1-P_{dist}$$

$$\tilde{m}(H,k+1)=m(H,k)\frac{\bar{f}(H,k)}{\bar{f}(k)}\left[1-P_c\frac{\delta(H)}{l-1}\right]$$

P_c is percentage of Crossover and m(H,k) means the number of Schema H in the Generation of k, it changes formula as below

$$P_{dist}=P_c\frac{O(H)}{l-1}\times\frac{O(H)}{l-1}(\because O(H)=\delta(H)),$$

$$P_{surv}=1-P_{dist}=1-P_c\left(\frac{O(H)}{l-1}\right)^2,$$

$$\left[1-P_c\frac{\delta(H)}{l-1}\right]<<\left[1-P_c\left(\frac{O(H)}{l-1}\right)^2\right]$$

Therefore probability of existing genes is increased

4.3 Influence of Mutation

The schema theory of mutation is below,

$P_{surv}=(1-P_m)^{o(H)}\cong 1-P_m o(H)\quad(\because P_m<<1)$

If we apply the way of Sliding Window, the percentage regarding Selected Gene that includes Schema (H) is shown with formula below. If we call m that the number of cases has schema H in a Gene, E(H) is probability of Mutation containing schema H, E(H) follow as below formula.

$$k=l-O(H)+1,\ E(H)=\frac{\sum_{i=1}^{k}i}{k\cdot n}$$

$$P_{surv}=1-P_m\cdot E(H)$$

$$(1-P_m)^{o(H)}>>\left[1-P_m\cdot\frac{\sum_{i=1}^{k}i}{k\cdot n}\right]^{o(H)}$$

Therefore more schemas exist in the Generation of proposed algorithm. Modified schema theory follows bellow.

$$\tilde{m}(H,k+1)=m(H,k)\frac{\bar{f}(H,k)}{\bar{f}(k)}\left[1-P_c\left(\frac{O(H)}{l-1}\right)^2\right]\left[1-P_m\cdot\frac{\sum_{i=1}^{k}i}{k\cdot n}\right]$$

$$=m(H,k)\frac{\bar{f}(H,k)}{\bar{f}(k)}\left[1-P_c\left(\frac{O(H)}{l-1}\right)^2-P_m\cdot\frac{\sum_{i=1}^{k}i}{k\cdot n}\right]$$

$$\left(\because P_c\left(\frac{O(H)}{l-1}\right)^2 P_m\cdot\frac{\sum_{i=1}^{k}i}{k\cdot n}<<1\right)$$

5 Experiments

5.1 Searching Space :

If we count all the pieces that were played during one game of Tetris and call it N, then U.A.D.C evaluates Genes of N/4 but proposed algorithm evaluates Genes of N-L+1.

the Gene of UDAC	46	35	41	36	31	36	28	39	36
the Generation of UDAC	6	4	5	4	4	5	4	5	5
the Gene of Proposed Algorithm	181	135	162	139	119	142	110	153	143
the Generation of proposed algorithm	23	17	20	17	15	18	14	19	18

Table3. the number of Gene and Generation

5.2 Searching Time :

If there are similarities between Genes, the processing time of Proposed Algorithm is reduced. If the similarities are higher, it will quicken the processing time. As shown Fig.9, if the time for one piece to fall to the bottom is T, the graph means the searching time per T.

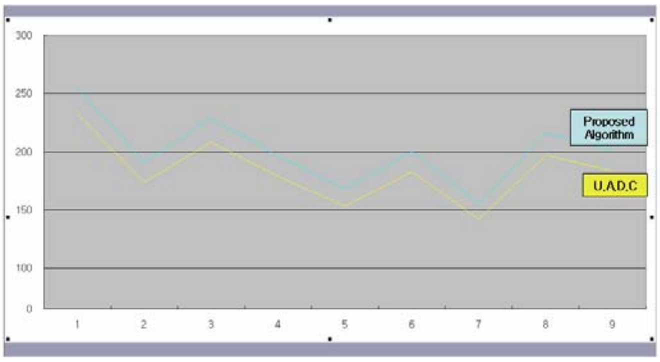
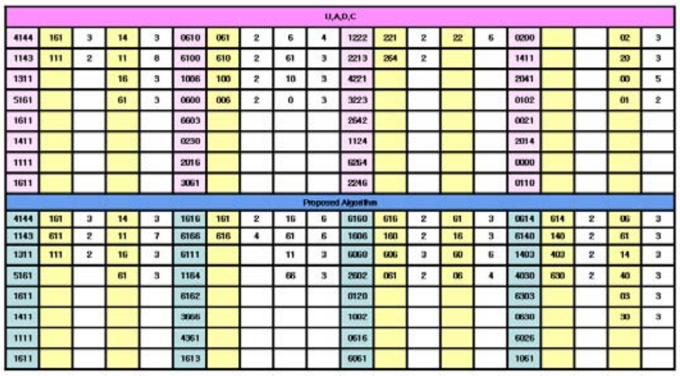


Fig.9. Searching time

5.3 Similarities between Genes :

If we use our proposed algorithm, between Genes increase. Since we use sliding window method, nearest Genes have the same chromosomes except for the head and tail.

Fig.10 shows similarities between U.A.D.C and proposed algorithm. We analysis the games of 8 which are played by beginner and expert. The top of Fig.10 is the results of 4 games and the bottom is a proposed algorithm. Yellow rows are overlapped permutation of 2 or 3 chromosomes and white rows are the number of overlapped permutations. Obviously, similarities between Genes are increased and therefore, CSA is a proper fit for a game.



U,A,D,C																			
4144	161	3	14	3	0610	061	2	6	4	1222	221	2	22	6	0200			02	3
1143	111	2	11	8	6100	610	2	61	3	2213	264	2			1411			20	3
1311			16	3	1006	100	2	10	3	4221					2041			00	5
5161			61	3	0600	006	2	0	3	3223					0102			01	2
1611					6603					2642					0021				
1411					0230					1124					2014				
1111					2016					0264					0000				
1611					3061					2246					0110				
Proposed Algorithm																			
4144	161	3	14	3	1616	161	2	16	6	6160	616	2	61	3	0614	614	2	06	3
1143	611	2	11	7	6166	616	4	61	6	1606	160	2	16	3	6140	140	2	61	3
1311	111	2	16	3	6111			11	3	6060	606	3	60	6	1403	403	2	14	3
5161			61	3	1164			66	3	2602	061	2	06	4	4030	630	2	40	3
1611					6162					0120					6303			03	3
1411					3666					1002					0630			30	3
1111					4361					0616					6026				
1611					1613					6061					1061				

Fig.10. Similarities between Genes

6 Conclusion

We have taken a first step to improve a game based on gamers' propensity. If proposed algorithm was concerned to the beginning of project, it was had very robust and powerful performance and it will make the various game changed by player's adaptness. Developer most knows that the factor of difficulty and fitness. During a game, simple or randomly appearing iterations of events occur, thus we make the game in such a way that a game be able to self-change the level of difficulty according to user's game skill. We extend the Searching Space of limited time and sequential event driven in the environments. And so we propose a way to quickly process the operations of Genetic Algorithm. Through our research, we proved decrease the gap from beginners to experts, thus we can effectively adapt Genetic Algorithm into a game.

References:

[1] DeJong K. "*An Analysis of the Behavior of a Class of Genetic Adaptive Systems*," University of Michigan, Ann Arbor.1975

[2] Goldberg, D. "*Genetic algorithms in search optimization and machine learning,*" Reading, MA: Addison-Wesley.1989.

[3] David E. Goldberg, Kumara Sastry. "*A Practical Schema Theorem for Genetic Algorithms Design and Tuning,*" IlliGAL Report No. 2001017

[4] Riccardo Poli andW.B. Langdon. "*A New Schema Theory for Genetic Programming with One-point Crossover and Point Mutation*". School of Computer Science, The University of Birmingham

[5] Itamar Faybish, "*Applying the Genetic Algorithm to the Game of Othello.*" VRIJE Universiteit BRUSSEL Faculty of Science Computer Science Departement

[6] David Carmel and Shaul Markovitch. "*Learning Models of Opponent's Strategy in Game Playing,*" CIS Report #9318.1993.

[7] John E. Laird and Michael van Lent. "*Human-level AI's Killer Application : Interactive Computer Games*," Proc. AAAI 2000, pp. 1171-1178. AAAI Press / The MIT Press. 2000.
[8] John E. Laird. "*Using a computer game to develop advanced AI*". IEEE Computer. July. 2001
[9] Steven Woodcock, "*Game AI: The State of the Industry*", Gamasutra Magazine Nov. 2000.
[10] By David C. Pottinger and John E. Laird, "Game AI: The State of the Industry Part 2," Nov. 2000
[11] S. W Um, T. Y Kim and J. S Choi, "*Player dependent difficulty control algorithm using G.A,"* Journal of the Korean Society for Computer Game, Vol. 01, Oct 2002.
[12] S. W Um, T. Y Kim and J. S Choi, "*User Adaptive Algorithm for Game Difficulty Control*," ADCOG 2003.