Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 24 (2013) 114 – 120

Procedia Computer Science

17th Asia Pacific Symposium on Intelligent and Evolutionary Systems, IES2013

Hybrid of Rule-based Systems Using Genetic Algorithm to Improve Platform Game Performance

Hyun-Tae Kim[a], Kyung-Joong Kim[b,*]

[a,b]Cognition and Intelligence lab, the Computer Science and Egineering, Sejong University, Seoul, South Korea

Abstract

Several attempts in the field for various games have been made using Genetic Algorithm. “Geometry Friends” is one of many platform games. This paper shows performance improvement of the platform game by applying GA. We attempt to determine the parameters for rule-based systems in the “Geometry Friends” game. We perform experiments for the problem not to be solved based on rule-based systems. We divide experimental cases and apply GA to each case. As a result, we could get a set of optimal parameters. Parameters tuning of rule-based systems using GA is more effective to improve performance than rule-based systems.

Keywords: Genetic Algorithm, Rule-Based System, Game Artificial Intelligence, Platform Games, Game Play Agent, Geometry Friends;

1. Introduction

A platform game is one of the most popular genre of video games. Platform games normally have platforms and obstacles. The player can jump over obstacles or get specific items. Famous platform game is “Super Mario”, ”Donkey Kong”, and so on. An important point is that the platform game needs very delicate movement. For example, it results in very large risk that the player makes a mistake like to move to specific location or jump to avoid obstacle. The delicate movement is one of the important elements of the platform game. In this paper, we will handle the “Geometry Friends” game for explaining our approach.

“Geometry Friends” is one of CIG (IEEE Conference on Computational Intelligence in Games) 2013 Competitions. It is a two players co-operative computer game. It is a platform puzzle game in a 2D environment

* Corresponding author.
E-mail address: kimkj@sejong.ac.kr.

with simulated physics (with gravity and friction). The goal of the players is to gather a set of diamonds in a short time [1].

The AI-agent in "Geometry Friends" game is performed by rule-based systems, which have many various parameters in order to determine character's behavior. This paper tunes the parameters using Genetic Algorithm (GA) in the "Geometry Friends" game. GA is used to solve various problems in other areas [2]. Actually, A lot of effective methods using GA have been studied [2,3]. Chishyan Liaw *et al*. [4] introduced evolving a team in a FPS game by using GA. The method that selects meta-level action in response to environments using GA was proposed by Ken Hasegawa *et al*. [5] in platform game.

This work attempts to determine the parameters for rule-based systems in game. At first, we grasp ambiguous part that could not be resolved only rule-based systems and point out limitation about this system. After that, we actualize the delicate movement in the game. Our experiment attempts to compare various situations for more effective learning.

The purpose of our experiment is to apply GA to a platform game and to improve the game performance. This paper is organized as follow. We will introduce about "Geometry Friends" at session 2. Genetic algorithm used in our experiments is introduced at session 3. Task experiment and experimental results are given at session 4. In session 5, conclusion about out experiment and more future work are shown

2. Geometry Friends

2.1. Score system

The AI players will be tested with a set 10 levels. The score system will rank using the total of collectibles they gather by level and by the time they take to complete a level. The AI players will be tested in each level 10 times in order to minimize the submissions. The score of each level will be the average of the score about the 10 runs. The score of a run will be calculated in the following way at each level:

$$Score = (N_{get} * V_{get}) + V_{bonus} * (maxTime - playTime)/maxTime \quad (1)$$

V_{get} is the score value attributed to the player for each diamond it collects. N_{get} is the total number of diamonds that were collected in the level. V_{bonus} is a fixed bonus score attributed to the player (e.g. collects all diamonds). If we collect all diamonds, we can get it. If not, V_{bonus} is zero (0). Each level will be given the time limit ($maxTime$). The values of V_{get} and V_{bonus} and $maxTime$ will be given to the participants with the information of each level [1].

The valuation function in GA is acquired by this score system. The reason why the score system is very important is that V_{bonus} is high when level difficulty is hard, and we can get more point when V_{get} is high. Then, getting all diamonds in the early time is very important.

2.2. Features of the game

This game exists the two "characters" a yellow ball and a green square. Both players control motion and shape of its character, subject to certain restrictions. Both characters can move horizontally, and the ball can jump and resize. The square can change its shape to an horizontal or vertical square. Each characters show their movement in Fig.1 [1].

This game involves physical element (with gravity, friction, speed, and acceleration). Many parts of this game are very similar to the latest popular game "Angry-Bird". The player isn't directly control delicate movement. If the player presses a command, a character shows the movement which is right for this. For example, when the player press a lot of right command, square's acceleration and speed increases. Then, it is necessary to control properly such behavior. In rule-based systems, standard work must be clear. However, it is a extremely hard work that choice all regulations by himself or herself. We solve the problems through utilizing the GA in this paper.

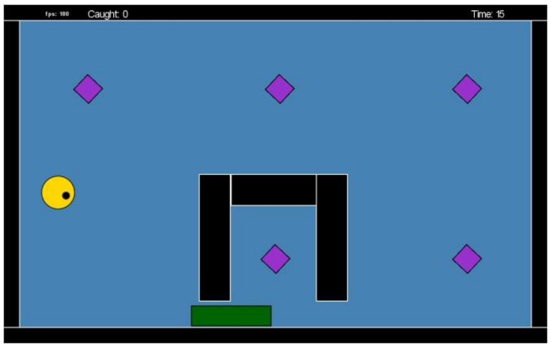


Fig. 1. Example of Geometry Friends map

3. Hybrid of Rule-based Systems and Genetic Algorithms

3.1. Genetic Algorithm (GA)

As shown in Fig.3, A classical Genetic Algorithm [6] is used in this paper. One iteration, called a generation, consists of N populations. At first, initial parents (P) are selected randomly. Parameters of each population are initialized to a random value between given minimum and maximum value in initial parents (P).

After that, we evaluate the population from score system (in section 2.1) by simulating the game. The $u \times N$ parents (P) from the current population are selected. New $(1-u) \times N$ offspring (p) are created by parents through mutation and crossover. The value of u is considered a good answer ratio of the population, be kept to the next generation. GA repeats again for maximum number of iteration.

As a result, we are able to get optimal parameters in game. If the agent wants to go the goal in game, we have to define parameters such as agent's velocity, break point, direction, and etc. But, it is difficult to define all parameters point by point. Therefore, we search for appropriate values about these parameters using GA. One of population is created by gathering these parameters

3.2. Rule-based systems (RBS)

Rule-based systems (RBS) are made up many functions. The game agent is able to get a diamond (the goal of the game) through RBS. In this paper, we focus on the *stop()* function that is the most important problem in the game. The *stop()* function is one of many functions. The *stop()* function is that the character's speed decreases gradually as it is close to target point. Conclusively, the square agent stops at target point. para[0] to para[4] in Fig.2 are tuned parameters by performing GA. If human modify them, the game performance is very low. We are able to make artificial intelligence using only RBS. But, only RBS agent is very difficult to expect high game performance.

```
private int stop(float target_point)
{
   x ← the x-coordinate of square
   x_vel ← the x-velocity of square
   absdistance ← Math.Abs(target_point - x)

   If (absdiastance <= para[0])
           If (Random(1,para[1]) ==1)
                    If (x_vel > para[2]) move_left;
                    Else if (x_vel < -para[2]) move_right;
           Else move_stop;
      Else if (absdiastance <= para[3])
           If (x_vel > para[4]) move_left;
           Else if (x_vel < -para[4]) move_right;
}
```

Fig. 2. Pseudo code of rules

3.3. *Optimization of RBS using GA*

We need to use hybrid of RBS using GA. RBS constitute our main systems. We focus on the optimization of main RBS using GA. Fig.4 presents the simulation procedure per iteration in running GA. Likewise, para[0] to para[# of parameters] is changed by repeating GA. this procedure perform total 10 simulation per iteration and finally acquire total 10 fitness. As soon as game ended, $playTime$ and N_{get} are put into the fitness function. Then, we can acquire each population's fitness value. If we successfully perform this procedure, we expect to acquire optimized parameters of RBS.

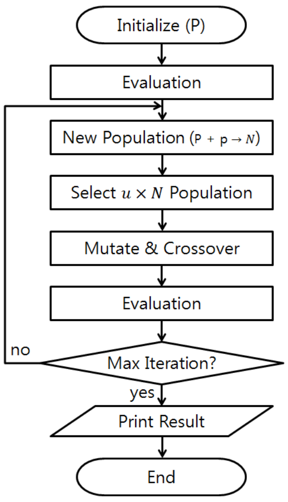
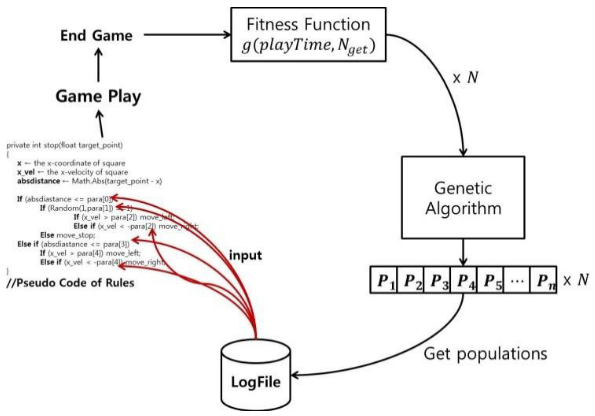


Fig. 3. Flow diagram of Genetic Algorithm



```
private int stop(float target_point)
{
   x ← the x-coordinate of square
   x_vel ← the x-velocity of square
   absdistance ← Math.Abs(target_point - x)

   If (absdiastance <= para[0])
         If (Random(1,para[1]) ==1)
                  If (x_vel > para[2]) move_left;
                  Else if (x_vel < -para[2]) move_right;
         Else move_stop;
   Else if (absdiastance <= para[3])
         If (x_vel > para[4]) move_left;
         Else if (x_vel < -para[4]) move_right;
}
//Pseudo Code of Rules
```

Fig. 4. Simulation procedure per iteration

4. Task Experiment

We perform experiment for the problem not to be solved (not clear the map perfectly in the game) based on RBS not human. We design such cases and perform GA. And then, this search examines that which kind of case shows the performance improvement by tuning parameters.

4.1. Experimental environment

As shown in Fig.5, the white box is a green square and the black box is obstacles. The point c in Fig.5 is the center of hole between obstacles. l_1 to l_3 are an arbitrary distance in which center is c. The level for our experiment would clear that the white box which is a character passes into the hole. But, this work is very delicate work that difficult to employ RBS, because it is not easy to control delicate movement of this game by the player.

The *stop()* function in section 3.2 is used to solve this problem, where target point is c. The competition opens five of these levels previously. If the *stop()* function is completed, we will expect good score about four levels of the five levels. l_1 to l_3 are a boundary line which is baseline to reduce speed. Human needs trial and error to get a lot of parameters. However, it is very inefficiency. So, we get the optimal performance about each situation using GA.

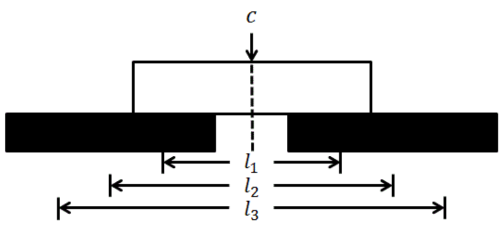


Fig. 5. The imaging of the map

4.2. Experimental design

Our experiments investigated three cases; each experiment is called A1, A2, and A3.

Table 1. Experiment case

Case	# of parameters	Used the boundary (l_n)
$\boldsymbol{A1}$	4	Only l_1
$\boldsymbol{A2}$	6	l_1 and l_2
$\boldsymbol{A3}$	8	l_1 and l_2 and l_3

GA's condition was as follows: Max iteration is 50; the value of u is 0.3; the value of N is 10. Each case run GA 500 times (Total simulation is $500 * 3 = 1500$).

As shown in Fig.6, each candidate save to log file every time one generation ends while running GA. Top 8~10 candidates are selected from log file. The reason why the number of selected candidates is between 8 and 10 is that candidates more than reasonable standard is only 8 in case $A1$ and only 9 in case $A2$. Each selected candidate runs simulation 50 times. The simulation systems save scores which is obtained from each candidate. To measure the success ratio of each candidate, the highest candidate is extracted among top 8~10 candidates. The highest candidate (a set of parameters) is the optimal parameters of case $(A1, A2, A3)$.

The reason why the last candidate obtained by GA not use to the highest candidate is that they have some risk falling into local optima. So, we select top 8~10 candidate and evaluate which one fall into the hole more than the others. Best performance can be obtained with highest candidate.

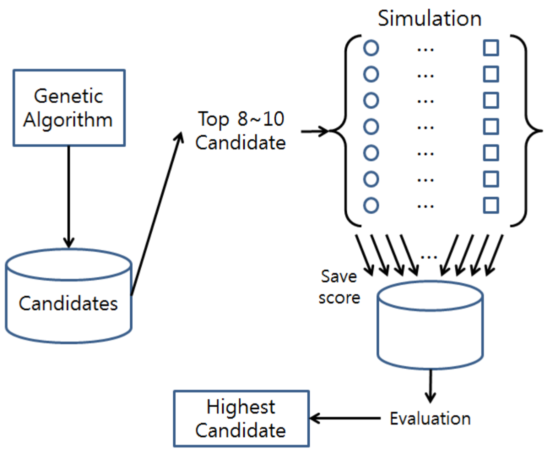


Fig. 6. The schematic processing of experiment

4.3. Results

Table 2. The success probability of cases

Case	Probability of last candidate	Probability of highest candidate
$A1$	0.18	0.24
$A2$	0.6	0.8
$A3$	0.08	0.18

As shown in Table 2, the last candidate for all cases always not obtained a good result. They showed the above-average performance at each case but the other of selected candidates became the highest candidate. $A2$ was 56% higher than $A1$ and 62% higher than $A3$. This experiment identified GA can get a set of optimal parameters in *stop()* function.

5. Conclusion and Future work

This paper demonstrates the optimization for game parameters using GA. The results of comparison between human, RBS, and RBS-Tuned are shown in Table 3. The scores measure using the score system in this game was performed each level 5 times. We measure scores about level 1,2,3,5 because the *stop()* function effects the levels.

Parameter tuning using GA improves the real game performance signally. V_{bonus} has very high value when level is 3 and 5. As mentioned in Section 2, difference between rule-based and RBS-Tuned is obvious. Our future work is to apply this process for another platform game. We will also study to improve performance efficiently and quickly in little iteration.

Table 3. The results table in levels

Levels	Human	RBS	RBS-Tuned
1	640	200	400
2	422	346	328
3	1885.5	560	1069
5	1552.78	600	1008.33

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (2013R1A2A2A01016589, 2010-0018950).

References

[1] http://gaips.inesc-id.pt/geometryfriends/

[2] Kim, K. J., and Cho, S. B., "Automated synthesis of multiple analog circuits using evolutionary computation for redundancy-based fault-tolerance," Applied Soft Computing, vol. 12, no. 4; 2012, pp. 1309-1321

[3] Grefenstette, and John J., "Optimization of control parameters for genetic algorithms," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 16, no. 1; 1986, pp. 122-128.

[4] Liaw, C., Wang, W. H., Tsai, C. T., Ko, C. H., and Hao, G., " Evolving a team in a first-person shooter game by using a genetic algorithm," *Applied Artificial Intelligence*, vol. 27, no. 3; 2013, pp. 199-212.

[5] Hasegawa, K., Tanaka, N., Emoto, R., Sugihara, Y., Ngonphachanh, A., Ichino, J., and Hashiyama, T., "Action selection for game play agents using genetic algorithms in platform game computational intelligence competitions," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 17, no. 2; 2013, pp. 201-207.

[6] Goldberg, D. E., *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass. : Addison-Wesley, 1989.