Jurnal Teknologi

Full Paper

SELF-EVALUATION OF RTS TROOP'S PERFORMANCE

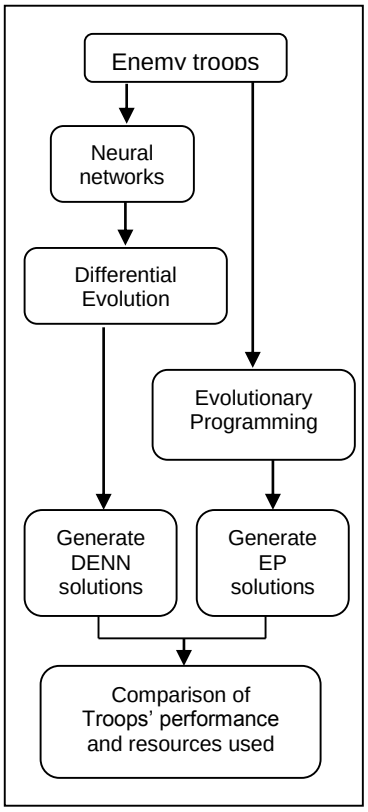
Chin Kim On[a*], Chang Kee Tong[a], Jason Teo[a], Rayner Alfred[a], Wang Cheng[b], Tan Tse Guan[c]

[a]Faculty of Computer and Informatics, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia
[b]School of Economics and Management, Qiqihar University, China
[c]Faculty of Creative Technology & Heritage, Universiti Malaysia Kelantan, Kelantan, Malaysia

Article history
Received
1 June 2015
Received in revised form
13 July 2015
Accepted
20 August 2015

*Corresponding author
kimonchin@gmail.com

Graphical abstract



Abstract

This paper demonstrates the research results obtained from a comparison of Evolutionary Programming (EP) and hybrid Differential Evolution (DE) and Feed Forward Neural Network (FFNN) algorithms in the Real Time Strategy (RTS) computer game, namely Warcraft III. The main aims of this research are to: test the feasibility of implementing EP and hybrid DE into RTS game, compare the performances of EP and hybrid DE, and generate gaming RTS controllers autonomously, an issue primarily of reinforcement/troops balancing. This micromanagement issue has been overlooked since last decade. Experimental results demonstrate success with all aims: both EP and hybrid DE could be implemented into the Warcraft III platform, and both algorithms used able to generate optimal solutions.

Keywords: RTS games, evolutionary computing, evolutionary programming, differential evolution, feed-forward neural network

Abstrak

Kertas ini menunjukkan hasil penyelidikan yang diperolehi daripada perbandingan Pengaturcaraan Evolusi (EP) dan kacukan Evolusi Berbeza (DE) serta algoritma Rangkaian Neural Suap Depan (FFNN) dalam permainan komputer Real Time Strategy (RTS), iaitu Warcraft III. Tujuan utama kajian ini adalah untuk: menguji kebolehlaksanaan EP dan hibrid DE ke dalam RTS, bandingkan prestasi EP dan hibrid DE, dan menjana autonomi pengawal permainan RTS, yang berkenaan dengan isu mengimbangi tentera. Isu pengurusan mikro ini telah diabaikan sejak sedekad lalu. Hasil kajian menunjukkan kejayaan dengan semua matlamat: kedua-dua EP dan hibrid DE boleh dilaksanakan ke dalam platform Warcraft III, dan kedua-dua algoritma yang digunakan mampu menghasilkan penyelesaian yang optimum.

Kata kunci: Permainan komputer RTS, pengkomputeran evolusi, pengaturcaraan evolusi, evolusi berbeza, rangkaian neural suap depan

1.0 INTRODUCTION

Games provided important platform for Artificial Intelligence (AI) research since last decade. It allows testing and comparison of either new or modified algorithms in highly complex environment. Conventional AI research has mainly focused on board games. Slowly the research emphases on modern video games such as Mario Bros [13], [17], First Person Shooting (FPS)[5], Pac-Man [25], Car Racing [21], Go game [24], Tower Defense games [11], and various Real Time Strategy (RTS) games[1], [8]. Modern video games provide challenging and well-defined problems, especially for RTS games. The challenges including real-time planning, decision making under uncertainty, opponent modeling, resource management, path finding, formation, etc. Most of the researchers develop algorithms and methods from the challenges tackled and are mostly focused to the first five challenges [9] but not the formation issue, particularly for Non-Player Character (NPC) formation design.

NPC formation design is a crucial issue. Play with a weak enemy troop will distract the gamer interest. On the other hand, gamers will not spend their time and money to play an undefeatable game. The design should create balance and interesting environment. A player must be challenged at first but not overwhelmed and, as the player's skill increases, the AI should grow and continue to challenge them. However, this issue has been overlooked in most of the RTS games.

A number of AI methods to the design of game controllers have been developed. Dynamic scripting [18], genetic algorithm [12], case-based reasoning and reinforcement learning [16], fuzzy logic [14], AI planner [19], influence mapping [26], are some methods developed to tackle specific RTS game issue. Hybrid techniques from evolutionary computing and machine learning have also been applied to game controllers as well, whereby Evolutionary Algorithm (EA) used to optimize Artificial Neural Network (ANN) weights. The hybrid approaches had been applied to different game such as Mario and tower defense games [11], [13], [17]. Interestingly, there is a discussion [3] used Evolutionary Programming (EP) to solve formation issue in Wargus game. The research result is remarkable however there is no comparison study has been conducted. Hence, this forms the core motivation of this research.

In this research, the EP algorithm is designed and implemented. The result will be compared with hybrid Differential Evolution (DE) and Feed-Forward Neural Network algorithm (FFNN) for solving formation issue in the Warcraft III platform; one of the famous half-open source RTS game. The optimization algorithm is used to tune the weights of the individuals whilst the FFNN is used to decide on what kind of possible combination of units are to be spawned during gameplay. This is not happen in EP. EP decides the number of units to be spawned without additional guidance.

In this study, a custom map is created in the Warcraft III. There are two teams of units used during gameplay: (1) opponent/enemy: a larger group of randomized units and (2) our AI controller: a group of AI units generated from the hybrid DE or EP. Both hybrid DE and EP will be challenged at first with the opponent, as the performance improved, both AI controllers will be challenged. The final results would represent (1) the shortest time the AI used to defeat the enemy force, (2) the maximum remaining AI units after the battle, and (3) the challenge results.

The remainder of this writing is organized as follows. In section 2.0, the methodology representation will be discussed. This includes the hybrid DE, EP, and an overview of the Warcraft III editor. Then, the experimental setting is presented in section 3.0. It also covers the evaluation function used for both EP and hybrid DE. Furthermore, the results and discussions are included in sections 4.0 and this writing will be concluded with conclusions and future works in section 5.0.

2.0 METHODOLOGY

2.1 EP Algorithm

EP is a paradigm of EAs. It is similar as Genetic Programming but the conventional structure is fixed to be optimized. Later design is not limit to fixed structure or representation and hence it is becoming harder to distinguish from evolutionary strategies [7]. EP involves very low computational cost and this makes EP standouts from other techniques. Most EAs involved more than one operator. But, EP main distinct operation is mutation and this causing no swapping process (crossover operator) between the individuals [7]. As typical EA, EP involves reproduction, mutation and selection.

2.2 Hyrbid DE Algorithm

DE is originally designed by Storn and Price and later there are various versions of modification [22]. In this research, the conventional DE is used and the algorithm is hybrid with neural cognition for solving gaming problems.

The hybrid DE algorithm involves FFNN. DE is almost similar as EAs. It may involve reproduction, crossover operation, mutation operation and selection. However, it is differs to EAs as three vectors are used in the optimization. Hence, DE is highly advantages for searching larger spaces of candidate solutions. DE optimizes problem by maintaining candidate solutions and creating new solutions by combining existing ones according to its formulae. Initially, there are three vectors involved; a main parent and two supporting vectors. In the crossover process, new gene will be generated majority from the main vector combining a uniform probability between two supporting vectors. The uniform mutation process will be accompanied after the crossover. As DE is combined with FFNN, the

FFNN plays important role for deciding the number of units to be spawned.

The FFNN [2] is the simplest form of ANN. Thus, it is widely used as ease to design and implement as well as low computational cost. Initially, FFNN consists of weights and the weights could be represented in input, hidden and output layers. The sum of the products of the weights and the inputs is calculated in each node. The neuron fires from the input to hidden layer. The activated value will be further fired from the hidden to output layer. Figure 1 shows the flow chart of hybrid DE.

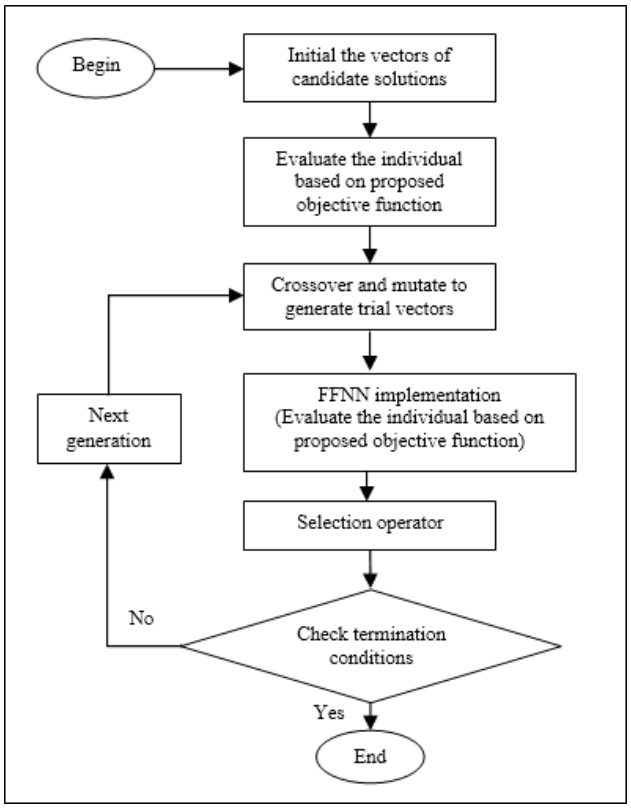


Figure 1 Flowchart of Hybrid DE

2.3 Warcraft III Editor

Warcraft III is a RTS game released by Blizzard Entertainment in its Warcraft game series. This platform has been widely used for research purpose [4], [15], [20], [23]. The Warcraft III provides easier customization (allow editing, modifying and including new features/components), user friendly environment, and interesting interfacing design [10]. Thus, it has been chosen as the test bed in most RTS gaming research. World Editor comes together with Warcraft III and it allows map customization and programming scripting. The World Editor also provides readily building and units. It allows resetting or modifying building's or unit's properties. The programming task is carried out using open source software, namely JassCraft programming language. Detail info related to Warcraft III could be found in their official web site [27].

3.0 EXPERIMENTAL SETTING

Both of the EP and Hybrid DE share some general experimental settings. Each experiment involves 10 runs. The termination conditions are: (1) when no foods remained in the gameplay, (2) simulation time reached 180 seconds, and (3) reached a maximum of 200 generations. Both algorithms share a same fitness function. Detail discussion of the fitness function is accessible in Section 3.4. The uniform mutation operator is involved and the rate is 0.02.

3.1 Gameplay Setting

In the game, the troops in an army are members of four possible races, each with different strengths and weaknesses. However, in this research, only the Human race was considered as it has the highest armor skill. We believe that the evolved Human controller would simply defeat other races during the gameplay even though it has never been evolved against other races.

The Human race consists of 13 different types of units: Peasant, Footman, Rifleman, Knight, Spell Breaker, Mortar Team, Priest, Sorceress, Siege Engine, Flying Machine, Gryphon Raider, Dragonhawk Raider and Militia. The Knight is the toughest ground unit with extremely high hit point and armor levels. The Gryphon Raider is an air unit with the highest attack damage and the Mortar Team is a ground unit having the highest siege attack damage. Footman, Knight and Mortar Team units are limited for ground attack. Hence, the combination of units rather than their number is the key to winning the game. In this research, only eight unit types were used during the experimentation. The Peasant, Priest, Sorceress, Siege Engine and Militia units were rejected due to having extremely low hit point, armor and attack damage. Our preliminary experimentation results showed that the inclusion of these weak units would slow down the optimization processes.

A customized map is designed. The AI-units spawned at the bottom centre of the map whilst the enemy unit is positioned opposite to the AI-unit. All units will be ordered to march to the centre of the map and the units will automatically fight and attack the enemy once the attack range is within the visualization area of the units. Figure 2 shows the designed customized map.

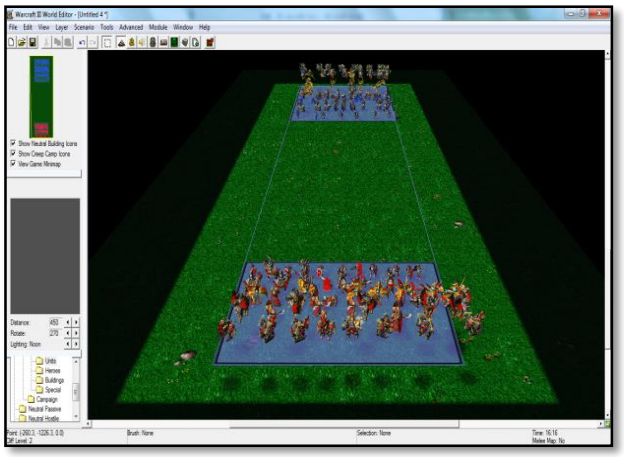


Figure 2 Custom map

Food is the main key in the game since different units hold some foods (capacity). In the experiment, the EP's and hybrid DE's are given 200 foods, 100 for each. The randomized enemy holds 120 foods. It is not an easy task in defeating a troop with larger foods.

3.2 EP's Setting

In this research, the (μ + μ) survivor selection is used; each parent generates an offspring. The EP's chromosome representation is formed by eight pairs of integer array. Each pair represents the type of unit and the number of units to be spawned. Table 1 shows the chromosome representation.

Table 1 Chromosome Representation for EP

Chromosome Representation	Unit Types
1	Footman
2	Rifleman
3	Knight
4	Mortar Team
5	Spell Breaker
6	Gryphon Raider
7	Flying Machine
8	Dragonhawk Raider

3.3 Hybrid DE's Setting

The hybrid DE's experimental setting is slightly different compared to EP as FFNN is involved in the DE. The input layer consists of 8 + 1 neurons (number of opponent unit by type plus one bias neuron). The hidden layer involves 17 + 1 neurons (number of hidden neurons plus one bias neuron). There are eight neurons in the output layer. The output neurons represent the number of units by types to be spawned. A binary sigmoid is used during the learning process. The uniform crossover operator is used with a rate of 0.7.

3.4 Evaluation Function

The evaluation function involved maximizing remaining foods after each match. The process means maximizing the number of units survives in any gameplay. Only stronger group of army will retain on the ground. This objective function will guide the controllers to spawn stronger combination of units.

$$F_1 = F_{U1} - F_{E1} \quad (1)$$

where F_1 represents the fitness value of an individual. F_{UI} represents the remaining foods of proposed AI units (either EP's or hybrid DE's controller). F_{EI} is representing remaining food of opponent units. F_1 yield a positive value if the controller wins the match. Otherwise, it generates negative value. Table 2 below shows the summary of the experimental setting.

Table 2 Summary of experimental setting

Descriptions/ Operators	EP Setting	Hybrid DE Setting
Number of experiments	10	10
Termination condition	200	200
Crossover operator	Uniform	Uniform
Crossover rate	-	0.7
Mutation operator	Uniform	Uniform
Mutation rate	0.02	0.02
AI-units food	100	100
Enemy food	120	120
Number of input neurons	-	8 + 1(bias)
Number of hidden neurons	-	17 + 1(bias)
Number of output neurons	-	8
Activation function	-	Binary sigmoid
Termination condition for each match	0 foods / 180 seconds	0 foods / 180 seconds
Type of neural network	-	Feed-forward

4.0 RESULTS AND DISCUSSIONS

4.1 Optimization Results

Figure 3 shows the experimental result obtained from one of the evolutionary run. The EP's controller generated lots of ground units at the beginning. The controller lost during early stage of the gameplays as the combination of units was too weak against the opponent troop. Most of these combinations involved big number of mortal team with massive riflemen and a few footmen and knights. In Warcraft III, mortal team is a unit that comes with low hit points and low armour but equipped with long range and high attack damage that is almost similar as a rifleman. Although they are very powerful, they are weak in defence and hence a group of knights or footmen are required as defender or protector. Nevertheless, the troops

generated by EP were very weak against the opponent troop as the opponent generated better combination of units.

In the 12th generation, the EP's controller was improved as it spawned a group of flying units combined with a few knights for defeating its opponent and it obtained high fitness score. Then, the EP's controller continued to adjust the involvement of knights and hence generated fluctuation fitness scores. Until the 34th generation, the EP's controller finally learned to remove all the knights and spent all resources to spawn a combination of troops formed only by flying units. The combination included gryphon raiders, flying machines and dragonhawk raiders. In later generations, the EP's controller started to spawn less flying machine and replaced with gryphon raiders or dragonhawk raiders that would help to obtain higher fitness score. This process continued until end of the optimization process.

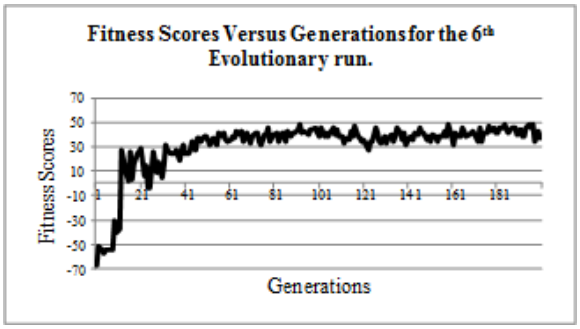


Figure 3 One of the EP's Optimization results – the 6th evolutionary run

Figure 4 presents the experimental result obtained from one of the DE's evolutionary run. The graph shows the DE controller obtained low fitness scores on the first half of the evolutionary runs. The DE's controller generated a combination of troops consists of majority knights or mortal teams and very few mortar teams or riflemen or footmen in the early phases. Such combination suffers a great lost as the troops were weak against flying units such as gryphon raider or dragonhawk raider. Until 105th generation, the DE's controller included flying units in the troop. The troop was formed with mostly gryphon raiders and very few flying machines. The DE's controller won the games afterward and obtained high fitness scores. Such solution helped the DE's controller in gaining high fitness score.

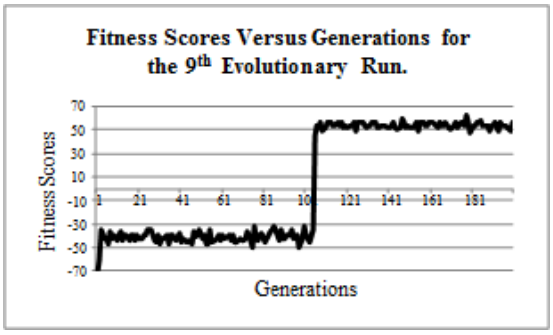


Figure 4 One of the DE's Optimization results – the 9th evolutionary run

4.2 Testing Results

Ten tests had been conducted for both EP and DE controllers. The results showed all generated controllers were able to win all of the game tests. EP controllers generated few combinations of troops. The combinations of troops involved:

- Majority of gryphon raiders, very few flying machines.
- Majority of gryphon raiders, very few spell breakers.
- Majority of gryphon raiders, very few dragonhawk raiders.
- Majority of dragonhawk raiders, some gryphon raiders and few flying machines.
- Majority gryphon raiders, some knights and few riflemen.

The solution provided by DE controllers for all test turned to one last solution although all generated initial controllers were different. The generated DE controllers proposed 15 gryphon raiders and two flying machines.

4.3 Statistics Analysis and Comparison of Reinforcement

Table 3 below shows all of the controllers obtained positive maximum fitness scores. It means all of the generated controllers can defeat their opponents and won the games. Table 3 also shows the generated DE's controllers obtained highest average maximum fitness score, 60.2 score compares to EP's controllers which obtained 49.6 average maximum fitness scores. Furthermore, DE's controllers obtained very low standard deviation compared to EP's controllers, 1.40 versus 3.63 respectively. It happened probably the generated DE's controllers always converged into a single solution that performed similar results in most of the evolutionary runs. That means the solutions found by the DE's controllers are stronger and highly survivable after gameplays.

Table 3 Comparison of fitness scores

Test set	EP			DE		
	Avg	Max	Min	Avg	Max	Min
1	3.84	56	-66	24.89	58	-61
2	17.20	48	-55	26.38	60	-61
3	28.96	45	-63	23.22	60	-59
4	20.59	52	-57	35.55	58	-55
5	32.66	48	-48	17.93	61	-65
6	31.28	49	-67	12.46	60	-58
7	34.80	48	-19	6.62	61	-64
8	28.21	51	-65	31.87	62	-65
9	29.02	45	-58	4.20	62	-71
10	-3.12	54	-55	17.81	60	-64
Average	22.34	49.6	-55.3	20.09	60.2	-62.3
Standard Deviation	11.75	3.63	14.09	10.29	1.40	4.50

The generated EP's controllers produced two commonly used combinations: (1) gryphon raiders + flying machines + dragonhawk raiders (This solution showed EP's controllers generated all flying units during the gameplays), and (2) gryphon raiders + knights+ riflemen (This solution showed the controllers learned to spawn some knights as defender and utilized remained resources to spawn a few riflemen to attack from a far distance). This reveals the generated EP's controllers learned to spawn different type of troops to eliminate the opponents.

In other case, the generated DE's controllers only spawned flying units such as gryphon raiders, flying machines and very unlikely dragonhawk raiders in the troop. It happened as the DE's controllers found that spawning stronger units in a gameplay could generate higher fitness score. The possible solutions suggested by EP's and DE's controllers are tabulated in Table 4.

Table 4 Possible Solutions Obtained in EP's And DE's Gameplays

Solutions	EP		DE
Common combination	gryphon raiders + flying machines + dragonhawk raiders	gryphon raiders + knights + riflemen	gryphon raider + flying machines
Possible solution	8 gryphon raiders + 7 Flying machines + 7 dragonhawk raiders.	12 gryphon raiders + 3 knights +1 rifleman	15 gryphon raiders + 2 flying machines

Table 5 show the resources consumed and time required to spawn the combination of troops proposed by EP's and DE's controllers. Table 5 clearly shows there were 5620 gold, 1780 woods and 1252 seconds were required for spawning a group of units consisting 8 gryphon raiders, 7 flying machines and 7 dragonhawk raiders. The other solution found by the EP's controller utilized 5540 gold, 1740 woods, and 1246 seconds for spawning 12 gryphon raiders with 3 knights and 1 rifleman. For the DE's controller, it used 5600 gold, 1880 woods, and 1346 seconds for spawning 15 gryphon raiders with 2 flying machines.

Table 5 Resources required for spawning selected troop from generated EP's and DE's controllers

Resources/ Algorithms	EP		DE
Total Gold used	4270 (to spawn unit) + 1350 (technology expenses) = 5,620 gold	4,300 (to spawn unit) + 1240 (technology expenses) = 5,540 gold	4,380 (to spawn unit) + 1220 (technology expenses) = 5,600 gold
Total Woods used	980 (to spawn unit) + 800 (technology expenses) = 1,780 woods	1050 (to spawn unit) + 690 (technology expenses) = 1,740 woods	1110 (to spawn unit) + 770 (technology expenses) = 1,880 woods
Total Foods	59	51	47
Time required (s)	647 (to spawn unit) + 605(technology build time) = 1,252s	701 (to spawn unit) + 545(technology build time) = 1,246s	701 (to spawn unit) + 645 (technology build time) = 1,346s
Unit Level	5	5	5
Technology required	Lumber mill (180 gold + 60s), Keep (320 gold + 210 woods + 140s), Castle (360 gold + 210 woods + 140s), Gryphon Aviary (140 gold + 150 woods + 75s) Arcane Vault (130 gold + 30 woods + 60s)	Lumber mill (180 gold + 60s), Keep (320 gold + 210 woods + 140s), Castle (360 gold + 210 woods + 140s), Gryphon Aviary (140 gold + 150 woods + 75s), Blacksmith (140 gold + 60 woods + 70s), Barracks (160 gold + 60 woods + 60s)	Lumber mill (180 gold + 60s), Keep (320 gold + 210 woods + 140s), Castle (360 gold + 210 woods + 140s), Gryphon Aviary (140 gold + 150 woods + 75s), Blacksmith (140 gold + 60 woods + 70s), Workshop (140 gold, 140 woods, 60s)
Unit spawned	8 gryphon raiders + 7 Flying machines + 7 dragonhawk raiders.	12 gryphon raiders + 3 knights +1 rifleman	15 gryphon raiders + 2 flying machines

As a summary, both EP and DE controllers utilized less than 5700 gold. However, the total woods used by DE controller were slightly higher compared to EP solutions. The EP controllers required more foods compared to DE controller. The time required for spawning DE solution was slightly higher compared to EP. It happened as the DE solution involved higher level of technology during gameplay. Hence, in term of resources management, the generated EP solutions performed better compared to the generated DE's controllers.

5.0 CONCLUSION AND FUTURE WORKS

In term of fitness score, the experimental results showed all of the generated controllers could generate good solutions to defeat a larger and stronger group of opponent. The DE's controller obtained highest fitness score and lowest standard deviation value compared to the generated EP's controllers.

In other perspective, both EP's and DE's controllers included stronger and higher level of units such as gryphon raider, dragonhawk raider and knights in the gameplay. It means in term of troop's performance, the solutions found by EP's and DE's controllers provided low defensive in the early stage of the gameplay. Furthermore, the process of preparing sufficient resources to spawn high level of units is time consuming and highly complex during each gameplay. Hence, the solutions provided are extremely risky and costly in practice.

As a conclusion, there is no significant different between the generated EP's and DE's controllers in the testing phase. Both algorithms used could generate the required gaming controllers. The generated controllers were capable to win all tests. However, the generated DE's controllers could eliminate the opponent units faster and there were more units remained after each match. The testing results showed all generated controllers were too superior to be defeated and this is not a good practice as no player want to play an undefeatable game.

As future works, probably applying different type of evolutionary computing techniques such as multi-objectives optimization algorithms, incremental evolution, interactive evolution and hybrid different type of neural networks might be an alternative solution.

In the extended works we designed and implementing the Pareto-Differential Evolutionary algorithm. The results will be compared with the EP and hybrid DE data. The suggested Pareto-Differential Evolutionary algorithm has been proven to be superior in designing simulated robot controllers [6].

Acknowledgement

This work has been partly supported by the Exploratory Research Grant Scheme (ERGS) project funded by the Ministry of Higher Education, Malaysia under Grants No. ERGS0045-ICT-1-2013.

References

[1] Bakkes, S., Kerbusch, P., Spronck, P., and van den Herik, J. 2007. Automatically Evaluating the Status of an RTS Game. *Proceedings of the Annual Belgian-Dutch Machine Learning Conference*. 143-144.

[2] Baptista, D. and Morgado-Dias, F. 2013. A Survey of Artificial Neural Network Training Tools. *Neural Computing and Applications.* 23(3-4): 609-615.

[3] Ch'ng, S. H., and Teo, J. 2012. Online Evolution of Offensive Strategies in Real-time Strategy Gaming. *IEEE Congress on Evolutionary Computation.* 1-8.

[4] Chambers, C., Feng, W. C., Feng, W. C., and Saha, D. 2005. Mitigating Information Exposure to Cheaters in Real-Time Strategy Games. *Proceedings of the international workshop on Network and operating systems support for digital audio and video.* 7-12.

[5] Chang, K. T., Ong, J. H., Teo, J., and Chin, K. O. 2011. The Evolution of Gamebot for 3D First Person Shooter (FPS). *IEEE The Sixth International Conference on Bio-Inspired Computing: Theories and Applications.* 21-26.

[6] Chin, K.O., and Teo, J. 2010. Evolution and Analysis of Self-Synthesized Minimalist Neural Controllers for Collective Robotics using Pareto Multi-objective Optimization. *IEEE World Congress on Computational Intelligence.* 2172-2178.

[7] Eiben, A. E., and Smith, J. E. 2003. *Introduction to Evolutionary Computing*. Springer Science & Business Media.

[8] Gagné, A. R., El-Nasr, M. S., and Shaw, C. D. 2011. A Deeper Look at the Use of Telemetry for Analysis of Player Behavior in RTS Games. *Entertainment Computing*–ICEC. 247-257.

[9] Hagelback, J., and Johansson, S. J. 2010. A Study on Human like Characteristics in Real Time Strategy Games. *In IEEE Symposium Computational Intelligence and Games (CIG).* 139-145.

[10] Isberg, J. 2004. Using Interactive Computer Games for AI Research. Master Thesis. Lund University.

[11] Leow, C. L., Gan, K. S., Tan, T. G., Chin, K. O., Alfred, R., and Anthony, P. 2013. Self-Synthesized Controllers for Tower Defense Game using Genetic Programming. *IEEE 2013 International Conference on Control System, Computing and Engineering (ICCSCE'2013)*. 487-492.

[12] Miles, C., and Louis, S.J. 2006. Towards the Co-Evolution of Influence Map Tree Based Strategy Game Players. *In IEEE Symposium on Computational Intelligence and Games*. 75-82.

[13] Ng, C. H., Niew, S. H., Chin, K. O., and Teo, J. 2011. Infinite Mario Bross AI using Genetic Algorithm. *IEEE Conference on Sustainable Utilization and Development in Engineering and Technology.* 85-89.

[14] Ng, P. H. F., Li, Y., and Shiu, S. C. K. 2013. New Fuzzy Integral for the Unit Maneuver in RTS Game. *In Pattern Recognition and Machine Intelligence*. 256-261.

[15] Niu, B., Wang, H. B., Ng, P. H. F., and Shiu, S. C. K. 2009. A Neural-Evolutionary Model for Case-Based Planning in Real Time Strategy Games. *IEA/AIE*. 291-300.

[16] Ontanon, S. 2012. Case Acquisition Strategies for Case-Based Reasoning in Real-Time Strategy Games. *Twenty-Fifth International FLAIRS Conference.*

[17] Pedersen, C., Togelius, J., and Yannakakis, G. N. 2009. Modeling Player Experience in Super Mario Bros. *In IEEE Symposium on Computational Intelligence and Games.* 132-139.

[18] Ponsen, M., Spronck, P., Munoz-Avila, H., and Aha, D. W. 2007. Knowledge Acquisition for Adaptive game AI. *Science of Computer Programming*. 67(1): 59-75.
[19] Rogers, K., and Andrew, S. 2014. A Micromanagement Task Allocation System for Real-Time Strategy Games. *Computational Intelligence and AI in Games, IEEE Transactions on.* 6(1): 67-77.
[20] Sheldon, N., Girard, E., Borg, S., Claypool, M., and Agu, E. 2003. The Effect of Latency on User Performance in Warcraft III. *Proceedings of the 2nd workshop on Network and system support for games.* 3-14.
[21] Shi, J. L., Tan, T. G., Teo, J., Chin, K. O., Alfred, R., and Anthony, P. 2013. Evolving Controllers for Simulated Car Racing Using Differential Evolution. *Asia-Pacific Journal of Information Technology and Multimedia.* 2(1): 57-68.
[22] Storn, R. and Price, K. 1995. Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. *International Computer Science Institute, Berkeley*. Technical Report No. TR-95-012.
[23] Szczepanski, T., and Aamodt, A. 2009. Case-Based Reasoning for Improved Micromanagement in Real-Time Strategy Games. *Proceedings of the Workshop on Case-Based Reasoning for Computer Games, 8th International Conference on Case-Based Reasoning.* 139-148.
[24] Tan, K. B., Teo, J., Chin, K. O., and Anthony, P. 2012. An Evolutionary Multi-objective Optimization Approach to Computer Go Controller Synthesis. *PRICAI 2012: Trends in Artificial Intelligence Lecture Notes in Computer Science,* 7458: 801-806.
[25] Tan, T. G., Teo, J., Chin, K. O., and Anthony, P. 2012. Pareto Ensembles for Evolutionary Synthesis of Neurocontrollers in a 2D Maze-based Video Game. *Applied Mechanics and Materials.* 3173-3177.
[26] Uriarte, A., and Ontañón, S. 2012. Kiting in RTS Games Using Influence Maps. *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference.*
[27] Warcraft III, 2008. [Online]. From: http://www.blizzard.com/us/war3/. [Accessed on 24 November 2013].