http://www.ftsm.ukm.my/apjitm
Asia-Pacific Journal of Information Technology and Multimedia
Jurnal Teknologi Maklumat dan Multimedia Asia-Pasifik
Vol. 2 No. 1, June 2013 : 57 - 68
e-ISSN: 2289-2192

EVOLVING CONTROLLERS FOR SIMULATED CAR RACING USING DIFFERENTIAL EVOLUTION

SHI JUN LONG
CHIN KIM ON
JASON TEO
TAN TSE GUAN
RAYNER ALFRED
PATRICIA ANTHONY

ABSTRACT

This paper presents an initial approach for creating autonomous controllers for the car racing game using a hybrid technique. The Differential Evolution (DE) algorithm is combined with Feed-forward Artificial Neural Networks (FFANNs) to generate the required intelligent controllers in a well-known car racing game, namely The Open Racing Car Simulator (TORCS). TORCS is used as a platform in most of the IEEE conference competitions. The main objective of this research is to test the feasibility of the DE implementation in TORCS platform. The literature showed that the application of DE in Real Time Strategy game returned promising results in evolving the required strategy gaming controllers. Interestingly, there is still no study thus far that has been conducted in applying DE into TORCS game platform. This research result shows that DE performed well in TORCS even though a very simple fitness function was used. This indicates that DE has well-tuned the neural network weights to generate optimal and sub-optimal controllers in TORCS.

Keywords: Differential Evolution; TORCS; Artificial Neural Networks, Game AI, Artificial Intelligence, Computational Intelligence, Car Racing Game.

INTRODUCTION

Computer games have become very popular in the recent years. These games provide an interesting real-world simulation to researchers. However, most of the games were built with low level Artificial Intelligence (AI) which sometimes makes the game boring to play. In order to attract and increase players' interest, effective AI technology is required. The purpose of incorporating AI technology in the games are twofolds; to generate human-like Non-Player Character (NPC) controllers as well as to increase the number of players participating in the game, particularly for online games, such as car racing game (Togelius and Lucas, 2005), Real-Time Strategy (RTS) game (Chang, et al., 2011), Mario Bros (Ng, et al., 2011), GNU Go (Tan, et al., 2012), etc.

Car racing is one of the game genres that is available in most of the gaming devices, and TORCS was introduced and developed to be used in competitions (CIG, 2013). TORCS is interesting as it provides real world driving simulation. TORCS provides a number of parameters that can influence the behaviour of the driver, including road curvature, surface friction and characteristics of the track. As for the car itself, speed, acceleration, direction, slipping and skidding of wheels may influence the driving status. Because of the game's complexity, TORCS is also suitable to be used as a platform for testing new algorithm performance.

Over the last decade, a number of researchers have investigated the idea of using Artificial Neural Networks (ANNs) as controllers with various algorithms to optimize the car racing (Togelius and Lucas, 2005; Agapitos, et al., 2007). Car racing was tested using NeuroEvolution of Augmenting Topologies (NEAT) methods (Cardamone, et al., 2009) to evolve the NN for controlling the behaviour of car. Besides, the single-objective evolution algorithms such as Genetic Algorithm (GA), Evolution Strategy (ES), Evolution Programming (EP) and Multi-objective Evolution Algorithm (MOEA) (Munoz, et al., 2010), and Fuzzy Logic have been applied to generate the required controllers for car racing game (Perez, et al., 2009). However, the combination of Differential Evolution (DE) and ANNs into car racing game still remains an open path for research. Hence, this motivates us to test the possibility of implementing DE in TORCS research.

DE was introduced by Storn and Price in 1995 (Storn, R., and Price, K. 1997). The primary idea is to solve Chebychev polynomial fitting problem. Since then, DE has been widely used to solve complicated optimization problems, such as parallel computing and constrained optimization. Initially, DE is a population-based optimization algorithm and was developed to optimize real parameter as well as real valued functions. DE is different from other evolution algorithms, as it is based on global optimization and it is used to solve non-linear, noisy and non-continuous problems. DE has been proven to perform better compared to other acclaimed global optimization methods and has done well in parallel computation (Storn and Price, 1997). It can also be used to train ANNs for autonomous robot and Real-Time Strategy game (Ilonen, et al., 2003; Chang, et al., 2011). In (Liu and Lampinen, 2005), fuzzy logic controllers were incorporated to DE to find the near-optimum solution.

In this paper, DE is used to optimize the generated ANNs weights for TORCS game. The objectives of this work are to investigate the feasibility of DE in car racing game and to design a new function for car racing. A three layers perception NN is used as controller to guide the driver. This hybrid technique showed promising results in the evolution and testing phases.

This paper is organized as follow. Section II introduces TORCS, which is the platform used in this research. Section III focuses on the methods used in this research, where DE and ANNs are briefly introduced. Section IV discusses the comprehensive results that we obtained from the experiments. Finally, the summary of the research will be discussed in Section V.

TORCS SIMULATOR

TORCS is an ideal platform for optimization algorithm (Loiacono, Cardamone & Lanzi, 2009). It provides a 3D visualization with various bots and a number of different tracks as shows in Figure 1. The structure is based on a client-server model: the bots are run as external processes connected to the server through UDP connections. That means we can create our own bots as a controller to connect to the server through UDP connection. As the car is simulated with a sophisticated physics engine, many aspects of racing car like traction, aerodynamics, wheel slippage, etc., are taken into account during the gameplay. These characteristics are included in TORCS to make it a meticulous car racing simulator that visualizes real car racing environment.

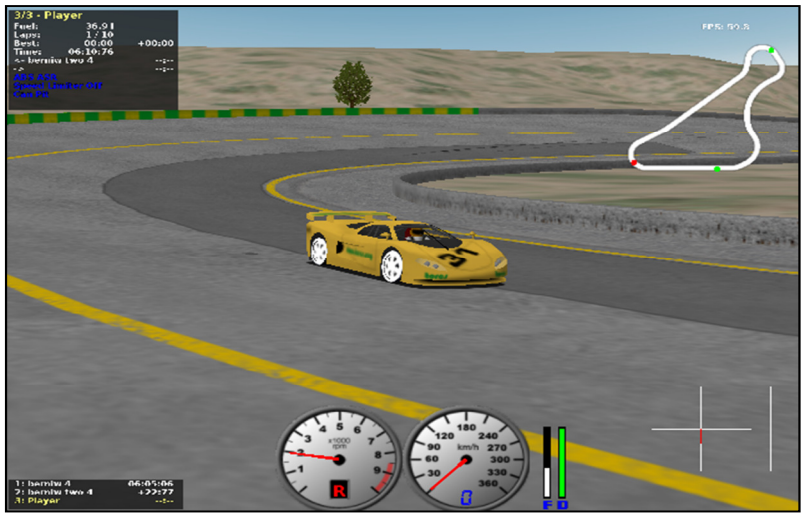

FIGURE 1. The Open Racing Car Simulator (TORCS) screenshot.

Basically, the client bots identify itself to the server and then establish a connection, after that the server sends two basic messages: the number of parameters to optimize and the number of simulation ticks for the whole optimization process. It will spend 20ms of simulated time in each game tick. After the sensory information is sent to the client bots, the client has to wait for 10ms of real-time for a response.

TORCS provides a lot of sensory and parameter information as shown in Table 1. However, only a few of them are used in this research and they are:

1. Angle between the car direction and the direction of the track axis.
2. Distance covered by the car from the beginning of the race till the end of the race.
3. Distance between the car and the track axis.
4. Current damage of the car.
5. Vector of 19 range finder sensors: each sensor represents the distance between the track edge and the car.
6. The velocity of the car.

TABLE 1. Sensors and parameters available in TORCS.

Name	Description
Angle	Angle between car direction and the direction of track axis
Current lap time	Time elapsed during current lap
Damage	Current damage of car
Distance From Start Line	Distance of the car from the start line along the track line
Distance raced	Distance covered by the car from the beginning of the race
Fuel	Current fuel level
Gear	Gear form {-1, 6}, -1 reverse, 0 neutral, 1~ 6 is forward.
Focus	Vector of 5 range finder sensors: each sensor returns the distance between the track edge and the car within a range of 200 meters.
Last Lap Time	Time to complete the last lap
Opponents	Vector of 36 opponent sensors
Rotation per minute	Number of rotation per minute of the car engine
Speed X	Speed of the car along the longitudinal axis of the car.
Speed Y	Speed of the car along the transverse axis of the car.
Speed Z	Speed of the car along the Z axis of the car

Track finder	Vector of 19 range finder sensors: each sensors returns the distance between the track edge and the car within a range of 200 meters
Track Position	Distance between the car and the track axis.
WheelSpinVel	Vector of 4 sensors representing the rotation speed of wheels.
z	Distance of the car mass center from the surface of the track along the Z axis.

The selected sensors affect the outcome of the preliminary test. For this reason, only some of them were selected to be evolved as other sensors and parameters did not affect the driving results. It is also costly in terms of time if all sensors are used in the optimization and testing phases. Although the opponent sensors may also be important, it is not included in this research, as this work only focuses on solo-car racing optimization. Figure 2 shows the various sensors used for the simulator.

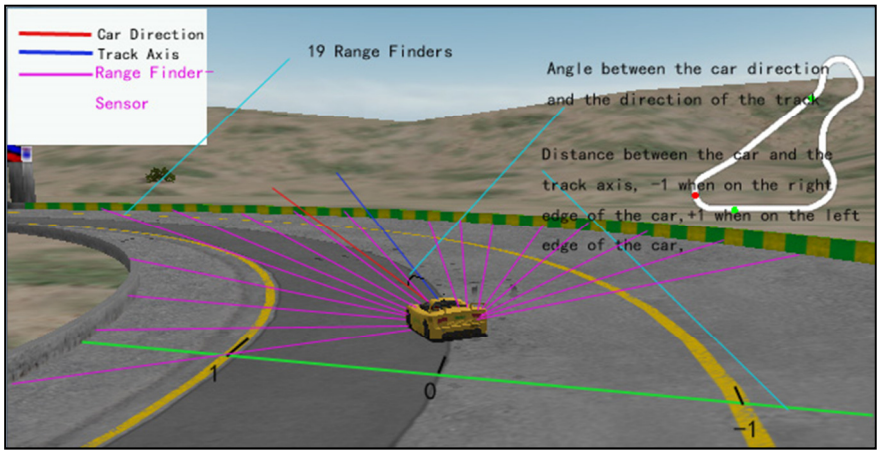

FIGURE 2. Screenshot of sensors used in the simulator.

METHOD

This section briefly describes the methods used in this research. It involves description of the FFANN, DE, and fitness function used.

FEED-FORWARD ARTIFICIAL NEURAL NETWORK (FFANN)

FFANN is proven as a suitable technique for optimization processes in games and robotics (Chin, et al., 2008; 2010; Ng, et al., 2011). Because of its simple architecture, FFANN is used widely for game controlling, such as 3D First Person Shooter game, RTS game, Ms. Pac-man, GNU Go game and Super Mario Bros (Ng, et al., 2011; Chang, et al., 2011; Tan, et al., 2012).

Unlike Multi-Layer Perceptron (MLP) ANN and Feedback ANN, the FFANN only has one straightforward connection without cycle and loops. The simplest architecture of FFANN is referred to as single layer perception (SLP), which includes one input layer and one output layer. In addition to these two layers, hidden layers are located between the input and the output layers. Hidden neurons are used to extract important features contained in the input. Figure 3 shows the structure of FFANN and the methods used. In this research, the FFANN with one hidden layer is used in TORCS. There are 13 input neurons, 16 hidden neurons and 3 output neurons in the network architecture. The input neurons represent the sensors inputs of: 10 range finder sensors, angle between the car direction and the direction of car axis, distance raced and the car's velocity. The output includes values for accelerate, brake and steer.

The weights of the FFANN are real numbers between $N(-1, 1)$ which are represented as the genes of each individual. Binary Sigmoid activation function is used to train the network.

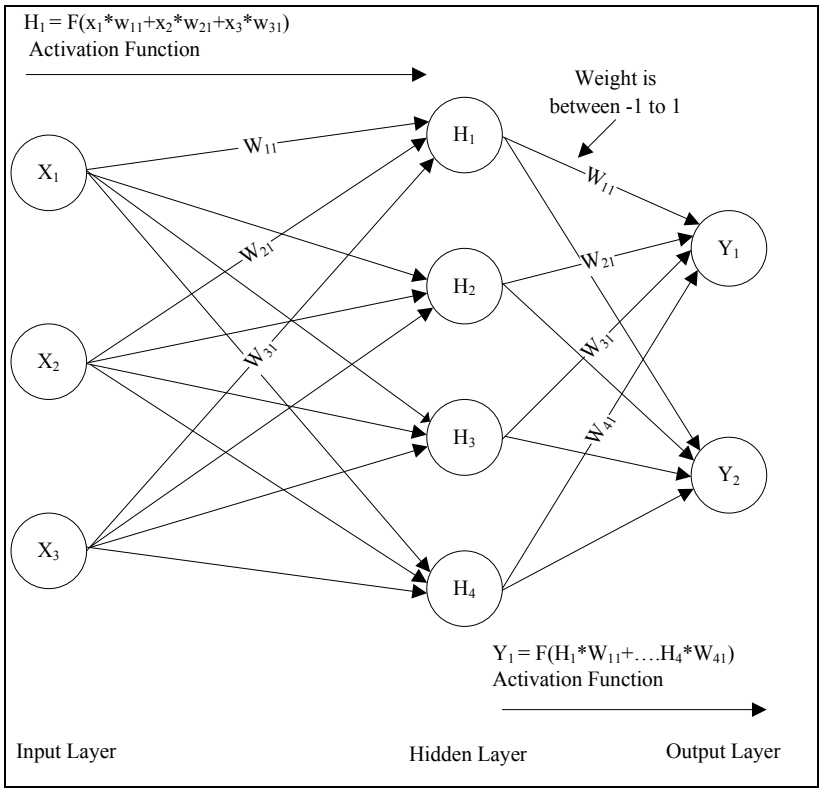

FIGURE 3. Structure of FFANN and methods used.

DIFFERENTIAL EVOLUTION (DE)

DE is a population-based optimization algorithm. Based on DE's greedy scheme, after generating a new population through crossover and mutation, the parents will be mated to produce its own offspring for a better solution. DE differs from other single-objective optimization algorithm such as GA, ES, EP, etc. in that there are a minimum of three parents that are used in DE. DE uses one main parent with two supporting parents in generating a new offspring. This is an advantage because this kind of DE solutions will not easily converge in the environment.

In this work, we conducted 10 runs for the optimization process with a crossover rate of 0.7 and a mutation rate of 0.02 (Chang, et al., 2005; Chin & Teo, 2010; Chin, et al., 2008). Using these values, our DE solution was able to successfully generate the optimal controllers. The population size is set at 10 and the termination condition is 1000 generations. The DE algorithm is shown in Figure 4.

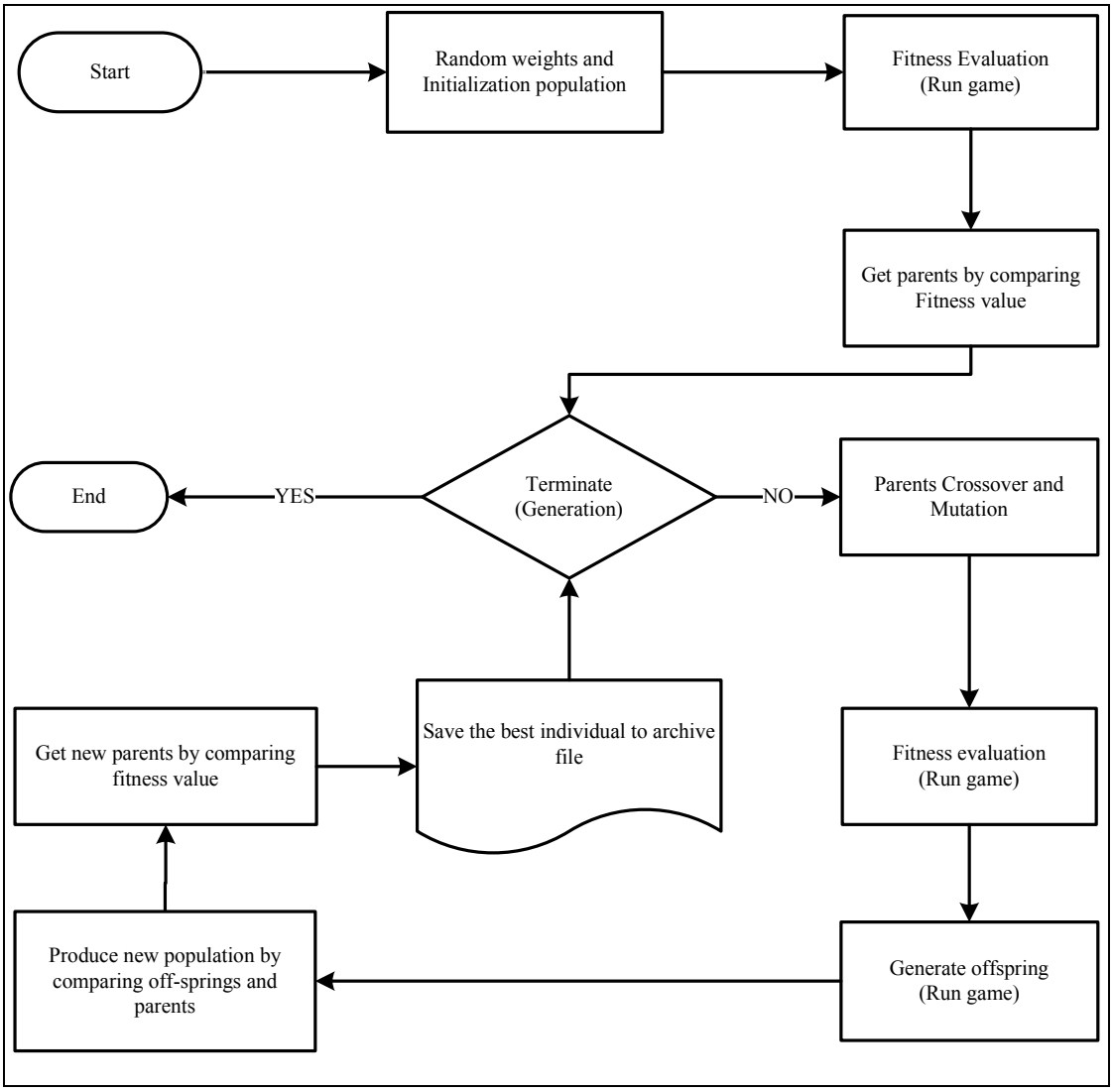

FIGURE 4. Flowchart of DE.

FITNESS FUNCTION USED

The Fitness Function used is a very simple function that is solely based on the distance of the race. A comprehensive comparison of different fitness functions will be part of our future work. The fitness function is given by Equation 1:

$$F = D_{raced} \tag{1}$$

Fitness function is important in the EAs cognition. Different results can be obtained with different fitness functions. In some cases, the computation time can be costly if a complex fitness function is used. On the other hand, the training can fail if the combination of the functions is too simple. In this work, we used a single component function which is the maximum distance of the race within 60 seconds during the gameplay. Surprisingly, the hybrid DE and ANN was able to generate the required controllers using this simple function.

The track used for this research is shown in Figure 5, which is a normal track used in most TORCS competitions, without many curves. The track length is 2,057.56 meters.

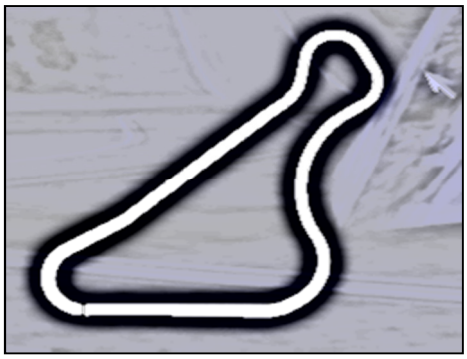

FIGURE 5. Screenshot of track used in this experiment.

RESULTS AND DISCUSSIONS

The results are shown in Figures 6, 7, and 8 which clearly show that DE has successfully optimized the required controllers for TORCS game. In can be seen in Figure 6 that the fitness value increases rapidly from the 1st generation to the 115th generation. Then, this fitness value increases slowly until the 822nd generation and there is no further increase after the 822nd generation. The optimal controller reached a fitness score of 9241.8 at the end of the run.

Figure 7 shows that the optimal solution learnt slower than the controllers generated in Figures 6 and 8. The optimal controllers reached fitness score of 7500 -8900 between generations 25th and 680th. Then, the optimal solution reached a high score of 9023.99 after generation 680th and maintains it till the end.

Figure 8 shows that the optimal controller learnt very fast and reaches a high fitness score before the 30th generation. Then, the fitness score increases slowly from 30th generation onwards. Finally, the solution found was able to reach a fitness score of 9057.13 after the 819th generation. Table 2 shows the tabulated fitness scores obtained during each set of experiment.

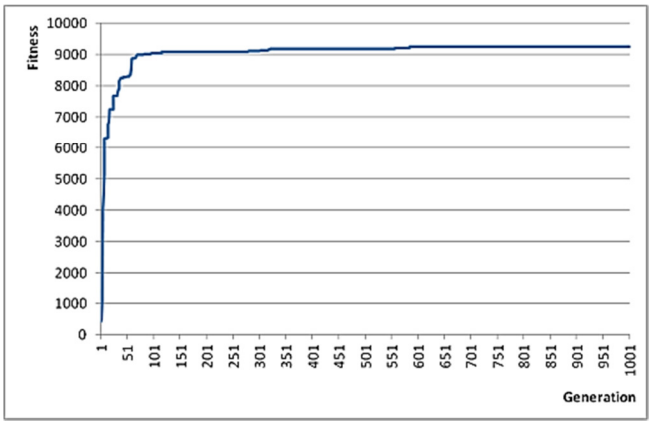

FIGURE 6. Fitness score versus generation for Run 6.

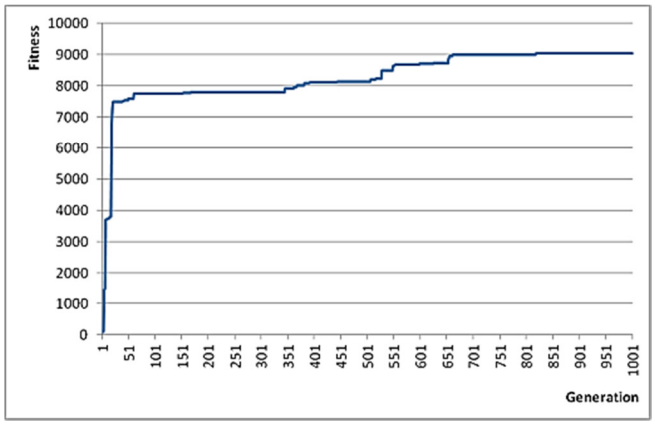

FIGURE 7. Fitness score versus generation for Run 4

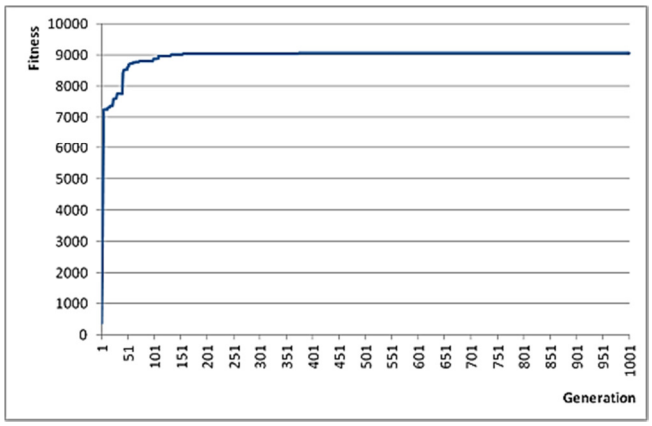

FIGURE 8. Fitness score versus generation for Run 9

TABLE 2. Highest fitness scores obtained for all optimized controllers

Run	Highest Fitness
1	7,400.57
2	8,768.69
3	8,725.92
4	**9,023.99**
5	8,716.94
6	**9,241.48**
7	8,476.62
8	8,155.17
9	**9,057.13**
10	8,242.68
Average	8,712.07
Max	9,241.48
Min	8,155.17

Based on the result shown in Table 2, the combination of DE and FFANN approach successfully generated the required controllers. Runs 4, 6 and 9 (in bold) show the generated controllers are able to obtain a fitness score of more than 9000 during the optimization processes, which is a highly interesting result as it is not easy to score 9000 in TORCS game with a single component fitness function. Usually, a complex single fitness function or a combination of several fitness functions and a good AI strategy are required in order to generate controllers that are capable of obtaining a fitness score higher than 9000 in any TORCS competition (Togelius, and Lucas, 2005). Thus, based on Table 2, we can conclude

that the controllers generated in Run 6 outperformed the controllers generated in Run 9 and Run 4 and the controllers generated in Run 9 outperformed the controllers generated in Run 4.

We observed and noted that some of the generated controllers evolved interesting driving skills. One of the best controllers drove with a maximum car speed without releasing the accelerator at all even when it is manoeuvring a curve. But, the controllers pressed both accelerator and brake together when it is negotiating the curved section. In the second case, the generated controller drove without using brake at all. The controllers slightly released the accelerator pedal when approaching the curve. Hence, the car slowed down when it reaches the curve but it sped immediately after the curve. This can happen because there is no big curve or U-turn in the map. This is probably the reason why, the controller was able to drive at a high speed to overcome the curve. In another case, the controller moved with a maximum car speed in the curved sections without pressing the brake or releasing the accelerator pedal. As a result, the car lightly bumped to the wall when it slowed down.

We conducted tests for all of the generated optimal controllers and the results are shown in Table 3. All optimal controllers were tested for 10 races, and a ranking system and best lap time were considered in these tests. The result shows the optimal controller generated from Run 4 was able to complete all the tests and won the competition with the best lap time of 45.85s, although it was slightly slower compared to the controllers generated from Run 6 and 9. Surprisingly, the controller generated from Run 6 completed only 2 out of 10 tests and the controller generated in Run 7 outperformed Run 6. The controller generated from Run 7 was not worse off than the controller generated from Run 6 during the optimization stages. On the other hand, the controller generated from Run 9 failed to complete any of the tests. The controllers generated from Run 2, 7 and 10 outperformed the controller generated from Run 9. The optimal controllers generated during the optimization stage performed worst in the testing phase due to the noise inclusion in the testing phases.

TABLE 3. Competition results for all generated optimal controllers

Run	Test(10)						Ranking
	Laps					Best Lap Time(s)	
	Zero	one	two	three	four		
1	1	4	3	2	0	55.06	6th
2	5	1	0	3	1	46.15	4th
3	6	2	2	0	0	48.30	9th
4	0	0	0	0	10	45.85	1st
5	3	1	4	2		46.15	7th
6	7	0	0	1	2	45.36	3rd
7	0	0	0	6	4	49.13	2nd
8	10	0	0	0	0	00.00	10th
9	2	2	5	1		45.73	8th
10	5	3	0	1	1	50.42	5th

CONCLUSIONS

This results of these experiments showed that the application of Differential Evolution combined with Feed-Forward Artificial Neural Network performed well in generating optimal car racing controllers. The generated controllers were able to control the simulated car even though a simple single component fitness function was used. Some of the generated controllers were superior as they were able to complete four laps of the race with extremely high speed without bumping to any wall or obstacle during the races. Other sub-optimal controllers (which achieved 5000-8500 average fitness score in the optimization stage) performed slightly worst. However, these controllers were able to move steadily during the testing phases and played the game with human-like feature.

A more complex fitness function will be considered in the future in order to generate better controllers that can be used in different maps. Besides, different types of AI methods will be implemented and tested as well.

ACKNOWLEDGEMENT

This research work is funded under the project ERGS0045-ICT-1/2013 granted by the Ministry of Higher Education, Malaysia.

REFERENCES

Agapitos, A., Togelius, J., and Lucas, S. M. 2007. Evolving controllers for simulated car racing using object oriented genetic programming. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation.* London: ACM, 1543-1550.

Cardamone, L., Loiacono, D., and Lanzi, P. L. 2009. Evolving competitive car controllers for racing games with neuroevolution. *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. Canada:* ACM, 1179–1186.

Chang, K. T., Chin, K. O., Teo, J., and Lii, C. B. 2011. *Automatic generation of real time strategy tournament units using differential evolution. Proceedings of the Conference on Sustainable Utilization and Development in Engineering and Technology.* Semenyih, Malaysia: IEEE Malaysia Section SMC Chapter, 101-106.

Chang, K.T., Ong, J.H., Teo, J., and Chin, K.O. 2011. The evolution of gamebots for 3D First Person Shooter (FPS). *Proceedings of the Sixth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011.* Penang, Malaysia: IEEE Malaysia Society, *21-26.*

Chin, K. O. and Teo, J. 2010. Evolution and analysis of self-synthesized minimalist neural controllers for collective robotics using pareto multiobjective optimization. *Proceedings of Evolutionary Computation (CEC).* Barcelona, Spain: IEEE Society, 2172-2178.

Chin, K. O. Teo, J., and Azali, S. 2008. Multi-objective artificial evolution of RF-localization behavior and neural structures in mobile robots. *Proceedings of Evolutionary Computation (CEC).* Hong Kong: IEEE Society*,* 350-356.

CIG. 2013. TORCS Competition. http://cig.dei.polimi.it/ [10 September 2013].

Ilonen, J., Kamarainen, J.K., and Lampinen, J. 2003. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, (17): 93–105.

Liu, J., and Lampinen, J. 2005. A fuzzy adaptive differential evolution algorithm. Soft Computing, 9(6): 448-462.

Loiacono, D., Cardamone, L., and Lanzi, P. L. 2013. Simulated car racing championship: competition software manual. *arXiv preprint arXiv:1304.1672*.

Munoz, J., Gutierrez, G., and Sanchis, A. 2010. Multi-objective evolution for car setup optimization. *UK Workshop In Computational Intelligence (UKCI)*. UK: IEEE Society, 1-5.

Ng, C. H., Niew, S. H., Chin, K.O and Teo,J. 2011. Infinite Mario Bros AI using genetic algorithm. *Proceedings of the Conference on Sustainable Utilization and Development in Engineering and Technology.* Semenyih, Malaysia: IEEE Malaysia Section SMC Chapter, 96-100.

Perez, D., Recio, G., Saez, Y., and Isasi, P. 2009. Evolving a fuzzy controller for a car racing competition. *Proceedings of Computational Intelligence and Games, (CIG) 2009.* Milano: IEEE Computational Intelligence Society, 263-270.

Storn, R., and Price, K. 1997. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*: 341–359.

Tan, K.B., Teo, J., Chin, K.O., and Anthony, P. 2012. An evolutionary multi-objective optimization approach to computer go controller synthesis. *Proceedings in 12th Pacific Rim International Conference on Artificial Intelligence (PRICAI'12),* Kuching, Malaysia: Springer-Verlag Berlin, 801-806.

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. 284-287:

3173-3177.
Togelius, J., and Lucas, S. M. 2005. Evolving controllers for simulated car racing. *Proceedings of Conference on Evolutionary Computation (CEC).* IEEE Society, (2):1906-1913.

BIBLIOGRAPHY

Chin Kim On research interests included gaming AI, evolutionary computing, evolutionary robotics, neural networks, image processing, semantics technology, agent technology, evolutionary data mining and biometric security system with mainly focused on fingerprint and voice recognition. He has more than 60 articles in the forms of journals, book chapters and conference proceedings. He is a member of IEEE and IAENG.

Jason Teo received his doctorate in information technology from the University of New South Wales, Australia, researching Pareto artificial evolution of virtual legged organisms. He has over 150 publications in the areas of differential evolution, artificial life, evolutionary robotics, multi-objective optimization and swarm intelligence. His current research focuses on the applications of evolutionary computation in games, robotics and optimization. He is a senior member of IEEE, professional member of ACM and program evaluator for ABET.

Patricia Anthony received her PhD in Computer Science from the University of Southampton in 2003. She is currently working as a Senior Lecturer at the Department of Applied Computing, Lincoln University, New Zealand. Her research interest is in semantic agents and multi-agent systems and how these agents can interact with each other within an open domain to solve problems. She is also interested in investigating how agents can communicate with each other at the semantic level using semantic technology. To date, she has published more than 80 articles in the forms of journals, book chapters and conference proceedings. She is a member of IEEE and ACM.

Rayner Alfred leads and defines projects around knowledge discovery and information retrieval at Universiti Malaysia Sabah. Rayner's works address current advanced intelligent techniques to model and optimise the complex and dynamic processes of knowledge discovery and information retrieval for structured/unstructured data. He holds a PhD degree in Computer Science from York University (UK), a Master degree in Computer Science from Western Michigan University (USA) and a Computer Science degree from Polytechnic University of Brooklyn, New York (USA). He has authored and co-authored more than 75 journals/book chapters, conference papers, and served on the organising committees of numerous international conferences and workshops.

Shi Jun Long is pursuing his Master Degree in Computer Science with the School of Engineering and Information Technology, Universiti Malaysia Sabah. His research interests included artificial neural networks, computer and video games, and differential evolution algorithm.

Tan Tse Guan received the bachelor of computer science (software engineering) and master degree in artificial intelligence from Universiti Malaysia Sabah, Malaysia, in 2006 and 2008. He is working toward the PhD degree in computer science at the Universiti Malaysia Sabah, Malaysia in the field of gaming AI research.

Shi Jun Long,
Chin Kim On,
Jason Teo,
Tan Tse Guan,
Rayner Alfred
School of Engineering and Information Technology,
Universiti Malaysia Sabah, 88400, Kota Kinabalu, Sabah, Malaysia.
shijunlong.stan@gmail.com, kimonchin@ums.edu.my, jtwteo@ums.edu.my,
tseguantan@gmail.com, ralfred@ums.edu.my

Patricia Anthony
Department of Applied Computing, Faculty of Environment, Society and Design,
Lincoln University, Christchurch, New Zealand
patricia.anthony@lincoln.ac.nz