Game Strategy Indexing, Learning and Optimization in Real Time Strategy (RTS) Games using Soft Computing Techniques

Ng Hiu Fung

Ph.D.

The Hong Kong Polytechnic University

2013

The Hong Kong Polytechnic University

Department of Computing

Game Strategy Indexing, Learning and Optimization in Real Time Strategy (RTS) Games using Soft Computing Techniques

Ng Hiu Fung

A Thesis Submitted in Partial Fulfillment

of the Requirements for

the Degree of Doctor of Philosophy

April 2012

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Signature:

Name of student: Ng Hiu Fung

Abstract of the thesis titled "Game strategy indexing, learning and optimization in real-time strategy (RTS) games using soft computing techniques", Submitted by Peter H.F. Ng for the degree of Doctor of Philosophy at the Hong Kong Polytechnic University, April 2012.

Abstract

In real-time strategy (RTS) games, players position and maneuver units and structures under their control to secure areas and destroy their opponents' assets. Typical strategies are resource gathering, units formation and positioning, base building, technology development and path finding. All the movements, construction and researches take place in real time, and players have a bird's eye view to control and monitor units using their own strategy. Selecting which strategy to use becomes one of the major challenges. This research focuses on indexing, learning and optimization of these RTS games strategies using soft computing techniques. Three game strategies are selected for the investigation, and original techniques have been developed to tackle this strategy determination problem. Based on our findings, development of RTS computer game software is better understood and supported using soft computing techniques.

The first investigation is to develop a strategy to quickly position game units effectively in a map so that they will create maximum casualty to enemies. A model integrating artificial neural network (ANN), genetic algorithm (GA) and case-based reasoning (CBR) is proposed and tested. The main idea is to evaluate the past strategies using GA, and train up an ANN for fast retrieval of units' locations. When new maps and new conditions are presented, CBR is used to compute the adjustments needed for the new locations. The key contribution here is the formulation of the RTS game strategy selection as CBR planning using a neural-evolutionary model. A number of simulated experiments with different maps and game unit settings are carried out to test the model. The result demonstrated that the model provides an efficient and natural game strategy indexing and determination scheme.

The second investigation is to develop a strategy to determine the types of game units to be selected for production with the purpose to effectively combat with opponents' troops. A contribution is made here by considering how the order of production and feature interaction of game units affect the result of playing RTS games. Due to complicated game rules, extensive terrains and numerous playable items, exhaustive search or explicit description of unit combination effects using analytical models, such as finite state machines, Bayesian networks and decision trees may not be feasible. We developed a machine learning model that extracts and evaluates game unit combination strategy from past data. This model takes into account the sequence in which game units are produced and the interaction among them. We combine fuzzy measure, fuzzy integral and genetic algorithm to develop the model. Warcraft III battle data from real players are used in our experiments. Compared with the traditional Choquet Integral, our new order-based fuzzy integral gives a smaller training and testing error in RTS game strategy selection. A dynamic Bayesian Network is also developed in learning game players' behavior.

The third investigation is optimal path determination. This is complicated because RTS game environment is hostile, dynamic and consists of many different types of game units interacting with each other in the battle field. Traditional path searching algorithms like min-max, alpha-beta pruning, hill climbing and A* are not suitable in such a complicated dynamic game world. We modified the multi-agent potential field model by incorporating the non-linear feature interaction property. The effect of unit cooperation can then be described, and therefore taken into consideration in optimal path determination. Our approach can identify the direction of positive and negative interaction for unit movement planning and team composition in RTS games. A combination of using real data and simulation experimental setting is used in this investigation. The results demonstrated that our path determination method is much better than the traditional methods implemented in Warcraft III.

As a summary, this PhD research focused on the investigation of RTS game strategies. Three original models are developed, namely (i) a neural-evolutionary model for CBR planning, (ii) an order-based fuzzy integral model, and (iii) a model of multi-agent potential field with feature interaction. All these three models are tested experimentally and promising results were obtained. A number of conference papers were published. One journal paper is under second review while another one is under preparation.

Keywords: Real time strategy game; RTS game; Warcraft III; strategy planning; unit formation planning; unit maneuver; fuzzy measure; fuzzy integral; genetic algorithm; CMA-ES; potential field

Acknowledgements

Studying at HKPU COMP has been one of the most valuable and enjoyable time in my life. By that time, I have worked with a great number of people whose contribution in the research and the making of the thesis deserved special mention. It is a pleasure to convey my gratitude to them all in my acknowledgment.

In the first place I would like to express my deepest thanks and gratitude to my supervisor, Dr. Simon Chi Keung Shiu. I would like to thank him for his kind supervision, continuous support and care during my PhD study. His truly scientist intuition and vision has inspired my thinking and sharpened my research skill. The experience as his student in these three years is great benefit to me for the rest of my life.

Next, I want to thank Prof. Xizhao Wang and Prof. Yan Li, who stayed up with our team to teach us. They have given me many useful suggestions and comments about my research project. It is the time that I formally acknowledge their contribution.

I would like to thank all the members of in our research group, Yingjie Li, Haibo Wang and Ben Niu. I appreciate very much their feedbacks, discussions, assistances, advices, and supports.

Thanks also to the board of examiners who spent their time and effort in assessing this research work and provided many good suggestions for me to improve the thesis. They are Prof. Man Leung Wong from Department of Computing and Decision Sciences in Lingnan University, Prof. Ashish Ghosh from Machine Intelligence Unit in Indian Statistical Institute and Dr. Korris Chung from Department of Computing in The Hong Kong Polytechnic University.

Last, but not the least, I wish to express my deepest appreciation to my family for their endless love, unwavering support and encouragement.

List of Publication

1. Peter H. F. Ng, Y. J. Li and Simon C. K. Shiu. Unit Formation Planning in RTS game by using Potential Field and Fuzzy Integral. In: Proceeding of 2011 IEEE International Conference on Fuzzy System (Fuzz-IEEE 2011), Taipei, Taiwan, 27-30 June 2011, pp.178-184.

2. Y.J. Li, Peter H.F. Ng, H.B. Wang, S.C.K. Shiu and Y. Li. Apply Different Fuzzy Integrals in Unit Selection Problem of Real Time Strategy Game. In: Proceeding of 2011 IEEE International Conference on Fuzzy System (Fuzz-IEEE 2011), Taipei, Taiwan, 27-30 June 2011, pp.170-177.

3. Peter H. F. Ng, Y. J. Li, H. B. Wang, Y. Li and Simon C. K. Shiu. Bottom-Up Strategy Planning Model by applying Fuzzy Integral in RTS Game. In: Proceeding of Joint 5th International Conference on Soft Computing and Intelligent Systems and 11th International Symposium on Advanced Intelligent Systems (SCIS & ISIS), Okayama, Japan, 8-12 December 2010, pp.1579-1584. (Best Student Paper Award)

4. Y. J. Li, Peter H. F. Ng, H. B. Wang, Y. Li and Simon C. K. Shiu. Applying Fuzzy Integral for Performance Evaluation in Real Time Strategy Game. In Proceeding of 2010 2nd International Conference on Information and Multimedia Technology (ICIMT), Hong Kong, RPC China. 28-30 December 2010, pp.168-172.

5. Peter H. F. Ng, Simon C. K. Shiu and Haibo Wang. Learning Player Behaviors in Real Time Strategy Games from Real Data. In: Proceeding of Twelve Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing (RSFDGrC), New Delhi, India. 15-18 December 2009, pp.321-327.

6. Haibo Wang, Peter H. F. Ng, Ben Niu and Simon C. K. Shiu. Case Learning and Indexing in Real Time Strategy Games. In: Proceeding of Fifth International Conference on Natural Computation (ICNC), Tianjin, China, 14-16. Aug 2009, pp.100-104.

7. Ben Niu, Haibo Wang, Peter H. F. Ng and Simon C. K. Shiu. A Neural-Evolutionary Model for Case-Based Planning in Real Time Strategy Games. In: Proceeding of Twenty Second International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE), Tainan, Taiwan, 24-27 Jun 2009, pp 291-300.

Table of Content

Abstract 4

Acknowledgements 7

List of Publication 8

List of Tables 12

List of Figures 13

Chapter 1 16

Introduction 16

1.1 Computer games and AI development 16

1.2 Motivation and objective of this research 17

1.4 Methodologies and research 19

1.5 List of contribution 21

1.6 Thesis organization 22

Chapter 2 23

Literature review 23

2.1 Real time strategy game research and development 23

2.2 Soft computing techniques applied to RTS games 29

2.3 Soft computing techniques used in this research 39

2.4 Other techniques used in this research 55

2.5 Summary 58

Chapter 3 59

A fast indexing scheme for identifying game unit’s best location 59

3.1 Introduction 59

3.2 Tower defense problem in RTS game 60

3.3 Neural-evolutionary model 60
3.4 Experimental result and discussion 66
3.5 Summary 71
Chapter 4 72
Learning player behaviors from 72
RTS game data 72
4.1 Introduction 72
4.2 Knowledge discovery problem in RTS game 73
4.3 Player behavior model 74
4.3.1 Behavior acquisition in replay data 74
4.3.2 Dynamic Bayesian network structure and parameters learning 76
4.3.3 Prediction in dynamic Bayesian network 77
4.4 Experimental result and discussion 78
4.5 Summary 82
Chapter 5 83
An order-based fuzzy integral to model feature interactions in RTS games 83
5.1 Introduction 83
5.2 Macro control problem in RTS game 84
5.3 Bottom-up strategy planning model 86
5.4 Evaluating the non-linear property in unit combination 88
5.5 Applying different fuzzy integral in fitness function 99
5.6 Experimental result and discussion 110
5.7 Summary 127
Chapter 6 128
Optimal path determination using 128

directional based fuzzy integral and potential field 128

6.1 Introduction 128

6.2 Micro control problem in RTS game.. 129

6.3 Min-Max strategy 130

6.4 Learning fuzzy measure by evolution strategy... 132

6.4 Combining Choquet Integral and potential field 134

6.5 Combining directional based fuzzy integral and potential field.................. 143

6.6 Experimental result and discussion 155

6.7 Summary... 158

Chapter 7 159

Conclusion and future works.. 159

7.1 Summary of the research problem.. 159

7.2 Summary of the research work 160

7.3 Future work.. 163

Reference.. 167

List of Tables

TABLE 3.1 GA's training time with different generation 68
TABLE 3.2 GA's training time with different population 68
TABLE 3.3 Performance of GA and Neural-Evolutionary Model 70
TABLE 3.4 Training performance on different hidden layers 70
TABLE 4.1 Selected data for DBN structure 75
TABLE 4.2 Time of Learning and Prediction in DBN 78
TABLE 5.1 Data in Warcarft III Replay 92
TABLE 5.2 Sample of Warcarft III Replay 92
TABLE 5.3 Elements in Score$_{\text{Warcraft III}}$ 97
TABLE 5.4 Data nature of testing data cluster 110
TABLE 5.5 Mann-Whitney Test of Data Cluster 1 & 2 111
TABLE 5.6 Mann-Whitney Test of Data Cluster 3 & 4 112
TABLE 5.7 Bivariate correlation of Different Data Cluster 113
TABLE 5.8 Comparison of training error in GA and CMA-ES 119
TABLE 5.9 Comparison of testing error in GA and CMA-ES 119
TABLE 5.10 Comparison of training error in different Fuzzy Integral 121
TABLE 5.11 Comparison of testing error in different Fuzzy Integral 121
TABLE 5.12 Comparison of training time in different Fuzzy Integral 121
TABLE 5.13 Comparsion of AVG Error with Weighted Average 122
TABLE 5.14 Comparsion of AVG Error with Order based FI (CMA-ES) 123
TABLE 5.15 Fuzzy Measure in Data cluster 4 126
TABLE 6.1 Relationship of Armor and Weapon Type in Warcraft III 141
TABLE 6.2 Armor and Weapon Type of unit type (partial) 141
TABLE 6.3 Comparison of different Micro Control 157
TABLE 7.1 Comparison of frequency with different magnitude in subset 163

List of Figures

Figure 2.1 Screen capture of RTS games 25

Figure 2.2 Tech-tree of Warcraft III (Human race) 26

Figure 2.3 Crossover of strategy cases by Ponsen 34

Figure 2.4 Structure of failure detection CBR by Mehta 34

Figure 2.5 Online learning model by Spronck 35

Figure 2.6 Sample potential field of Hagelback 37

Figure 2.7 Lebesgue Integral. w is the measure of a function $f(x)$ 42

Figure 2.8 Choquet Integral. μ is the fuzzy measure of a function $f(x)$ 43

Figure 2.9 Process of Genetic algorithm 46

Figure 2.10 The directional optimization of the CMA-ES algorithm 48

Figure 2.11 The step size improvement of the CMA-ES algorithm 50

Figure 2.12 A simple artificial neural network 51

Figure 2.13 A simple Hidden Markov Model 52

Figure 2.15 Motion of a robot in potential field 56

Figure 3.1 Neural-evolutionary model 61

Figure 3.2 Demonstration of the encoding in GA 62

Figure 3.3 Cannon distribution 62

Figure 3.4 Damage on the enemy 63

Figure 3.5 Encoding of Point A 65

Figure 3.6 ANN structure 66

Figure 3.7 Cannon distribution for different generations 67

Figure 3.8 Fitness on different generations and populations 67

Figure 3.9 Training and testing result of BP ANN 68

Figure 3.10 Training and testing result of RBF ANN 70

Figure 3.11 Training performance of BP 71

Figure 4.1 Work flow of player behavior simulation model 74

Figure 4.2 Structure of a DBN 77

Figure 4.3(a) Accuracy of 10 learning instances 80

Figure 4.3(b) Accuracy of 30 learning instances 80

Figure 4.3(c) Accuracy of 50 learning instances 80
Figure 4.4(a) Accuracy for the learnt DBN for Player B... 81
Figure 4.4(b) Accuracy for the learnt DBN for Player C... 82
Figure 5.1 Bottom-up strategy planning model 87
Figure 5.2 Warcraft III expansion, The Frozen Throne (WIII: TFT) 91
Figure 5.3 Learning fuzzy measure by GA.. 94
Figure 5.4 Learning fuzzy measure by CMA-ES 95
Figure 5.5 Fitness calculation of one chromosome.. 96
Figure 5.6 Estimated score calculation 98
Figure 5.7 Comparison of CI and OI in three replay data.. 102
Figure 5.8 Finesses of GA in different population 116
Figure 5.9 Finesses of GA in different mutation rate 116
Figure 5.10 Finesses of GA in different generation 117
Figure 5.11 Finesses of CMA-ES in different population... 118
Figure 5.12 Finesses of CMA-ES in different iteration 118
Figure 5.13 Training error of different fuzzy integral... 124
Figure 5.14 Testing error of different fuzzy integral... 124
Figure 6.1 Overview of the Min-Max strategy 131
Figure 6.2 Learn the fuzzy measure by GA 133
Figure 6.3 Choquet Integral with non monotonic fuzzy measure................................. 135
Figure 6.4 Effect of the decay function, $\varphi(P(\mathrm{x,y}), P_i)$ 137
Figure 6.5 Potential field of enemy.. 137
Figure 6.6 Effect of progressive function 138
Figure 6.7 Potential field of footman... 139
Figure 6.8 Potential field of priest 140
Figure 6.9 Potential field of sorceress.. 140
Figure 6.10 Maneuver of player units 140
Figure 6.11 Potential field of enemy... 142
Figure 6.12 Potential field of enemy... 143
Figure 6.13 Potential field of enemy... 143
Figure 6.14 Effect of the decay function, $D = 0.5$ 144

Figure 6.15 Effect of the decay function, $D = 2$ 144
Figure 6.16 Effect of the decay function, $\varphi(P_i, P_j)$ 148
Figure 6.17 The shape of the affected area 150
Figure 6.19 Intermediate phase (Diversion Attack) 151
Figure 6.20 Centralized phase 152
Figure 6.21 Potential field with negative interaction 152
Figure 6.22 Potential field of player 153
Figure 6.23 Effect of progressive function, ϕ 154
Figure 6.24 Simulation in Warcraft III 155
Figure 6.25 Diversion and flank attack in Warcraft III 157
Figure 7.1 Usage count of subset selection for order-based FI in data cluster 1 164
Figure 7.2 Usage count of Subset selection for CI in data cluster 1 164

Chapter 1

Introduction

In this chapter, the background of this research is stated. Then, the overview of computer game and AI development is introduced. After that, we focus on real time strategy (RTS) game and its current situation is summarized. The motivation of this research and the main research problem is stated. Finally, the methodologies of research and structure of this thesis is explained.

1.1 Computer games and AI development

The first computer game was written by Douglas in 1952 for his Ph.D. thesis on human-computer interaction in University of Cambridge. It was a tic-tac-toe game. The Artificial intelligence (AI) could play against a human player. It was regarded as the first connection of human and AI in computer games. In 1975, a group of Massachusetts Institute of Technology (MIT) students developed an optimal strategy for the AI to play Tic-Tac-Toe perfectly. Then, the development of computer game and AI has never been stopped. In 1980s, computer gaming industry was started and grew rapidly. Numerous 2D games were published, such as Pac-Man and Donkey Kong. In 1985, Nintendo released the first video game console and brought the computer game into our family. Many 2D action games, such as Super Mario Bros became a success. Nearly all these games consisted of AI to play with humans. As the game play and control were simple at that time, decision tree and rule based system were widely used. AI development for each specific game was easy. 1990s were regarded as an innovation decade of video gaming. 3D computer graphics raised the computer gameplay into another stage. Lots of new gameplay were started, such as first person shooting (FPS), massively multiplayer online role play game (MMORPG) and real time strategy (RTS) game. Thousands of games were published each year afterwards. Therefore, technologies and theories had to be generalized to fulfill the need of market, including computer graphics, multimedia, network and AI. In fact, computer graphics and multimedia showed a significant improvement at that time. In 2010s, the game developers were not only the people in the game industry. The phenomenon of user-created modifications (MOD) games has begun. It extended the life cycle of each game. One of the most important examples is Warcraft III which is a RTS game by Blizzard. The elements that allow players to modify were not only the graphics. Players could design and create new stories, battlefields, game rules and logic in Warcraft III. Two famous gameplays created by the player are tower defense (TD) and defense of ancients (DotA). TD becomes a standalone game in many platforms, especially in mobile device. DotA is predecessor of World of Warcraft which is a well known MMORPG. It occupies two-thirds of the MMORPG market. Another good example of MOD game is Roblox which is a massively multi-player 3D game environment. The players in Roblox build 5.4 million games in 2011. As MOD game players are not professional programmer, a user friendly and powerful editor for adaptable AI was strongly requested. Therefore, AI can be used in different battlefield and gameplay easily.

1.2 Motivation and objective of this research

1.2.1 Driven force in game industry

The Game market grows rapidly in these few years. DFC Intelligence, which is a strategic market research firm focuses on interactive entertainment, stated that the global game market grows from USD$28.5 billion in 2005 to USD $66 billion in 2010. It grows 130% in these 5 years. DFC also forecasted that the market may reach $81 billion in 2016. It is a market with huge potential and the development of game related technologies is fast. In China, there are about ten new online games released each month. A special game AI is necessary to give a special experience to the player as the graphics, physic, gameplay and multimedia improvement are becoming saturated. Most of the games are similar. Online game play could be a solution. It allows lots of human players to play with each other. However, it is difficult to keep a certain amount of players that are available in 24 hours a day and perform different behavior. Moreover, designing strong Artificial Intelligence (AI) for computer games is also a very challenging task as it needs to consider tremendous complexity of games rules. Modern computer game is a complex environment which consists of multiple agents in 3D or even 4D dimension, especially in real time strategy (RTS) game. To fulfill the need of the market, building a scalable and various AI in an efficient way become an important need. “Scalable” means that AI can change its level of intelligence. “Various” means that AI can perform different behaviors in the same situation. “Efficient” means that it is easy to adapt in different games or battle field with minimal human power.

1.2.2 Driven force in game research

RTS game is military simulation. Players fight for the resource, set up economics, expand the base and tech tree, build an army and destroy the enemy base. Other computer games, such as action game or RPG, can rely on graphics and multimedia to attract players. However, successful RTS games mainly rely on game rule design. The role of AI in RTS game is an importance part. Current AI performance in commercial RTS games is not good enough to play with human player. It is lagging behind the gameplay and rule development. AI is not able to handle complex decision making, interacting object, partial information as well as fast-paced units maneuver. Further research and improvement of AI in RTS game is necessary. A search in the IEEE Xplore using the key word “real time strategy game” from 2000 to present gives 21,292 results. Many journals, such as the new issued IEEE transaction on Computation intelligence and AI in games, and many conferences, e.g. The Game Developers Conference, The International Conference on Computer Games, Game AI conference, The International Conference of AI for Interactive Digital Entertainment, etc, totally or partially focus on RST games. Many researchers, such as Buro and Lucas [Buro 2004, Lucas 2009] have called for AI research of RTS game in different conferences. They stated an important thing that RTS game is a well defined environment to conduct experiment for different AI algorithm. There are many low hanging fruits waiting for the researchers. Hence, RTS game is a huge environment and many areas are under study. Researchers can focus on different aspects, such as decision making, path find and scouting algorithm, unit maneuver planning, pattern recognition of player behavior.

1.3.3 Research problem

The research problem of this research can be summed up to two questions. First one is how to improve the Artificial Intelligence (AI) algorithms to learn new tasks and adopt in RTS game. Therefore, various AI can be developed efficiently in both strategy planning and units maneuver.

The second one is how to handle the reasoning about feature interaction in RTS game and improve AI performance. Therefore, the feature interaction about the rules and elements in RTS game can be formulated.

1.4 Methodologies and research

In our work, we focus on strategy indexing, learning and optimization in RTS game. We selected tower defense which is a subgenre of RTS game play as our preliminary study. The goal of tower defense is to set up the towers in suitable positions and stop enemies from crossing a battlefield. As there is no specific rule to guide the setting, we proposed a machine learning approach based on genetic algorithm and artificial neural network and developed a neural-evolutionary model for case-based planning. We encoded the battlefield information into a chromosome in genetic algorithm (GA). Throughout the crossover and mutation, GA could find an optimized solution in different battlefields. However, it is time consuming due to its random walk in new battlefield. It is not applicable in RTS game. We created a model to adopted neural network (NN) into GA to solve this problem.

After the preliminary study, we extended our work to RTS game play. We decoded and extracted the data from the replay of Warcraft III which is a well known RTS game. We adopted a Case-Based Reasoning (CBR) approach to create player behavioral models. The proposed method analyzed and cleaned the data in RTS games. It converted the learnt knowledge of one player into a probabilistic model, i.e., a Dynamic Bayesian Network (DBN), for representation and prediction of player behaviors.

Although we have developed a model to learn player behaviors, there is lack of algorithm to evaluate the performance of cases and handle the feature interaction in RTS game. To overcome this problem, we divided the RTS game play into two types and performed the research. First one is macro control which is about the strategy planning. Second one is micro control which focuses on unit maneuver.

For the macro control, the success of strategy planning is largely determined by an appropriate selection of a suitable mix of game unit types. However, there is no simple optimal strategy existing and it is difficult to decide a mixed army to respond to opponent group as the situation of intransitive superiority often occurs among different unit types. We proposed a bottom-up approach for strategy planning in RTS game. It can avoid defining the complex if-then statement. Action will be generated according to the decision of unit combination. We extracted the strategies from real professional players in Warcraft III and combined GA, CMA-ES, fuzzy measure and integral to learn the performance of unit combination. Fuzzy measure of each subset is guided by the fuzzy integral in the GA or CMA-ES training. Three new fuzzy integrals, Mean based, Max based and Order based fuzzy integral are proposed to describe the feature interaction in RTS game.

For micro control, it is more complicated than macro control, such as building and unit production sequence. The environment is hostile and dynamic in RTS game. It consists of a great quantity of possibilities. Hence, multiple targets and the non-additive property of unit formation lead to a problem in the micro control. Traditional tree searching or A* searching is unable to handle these two properties. They are time consuming in development as there are too many weightings and each of them will interact with the others. Hence, the model did not allow multiple criteria. We applied potential field, fuzzy measure and integral to solve the micro-control. A new fuzzy integral, Directional based fuzzy integral is proposed to descript the interaction in potential field. It optimizes the path finding algorithm and provides the ability to perform flank and diversion attack in RTS game.

1.5 List of contribution

In summary, we have developed the following soft computing based techniques for RTS game.

1. A new location planning model is developed to quickly position game units effectively in a map so that they will create maximum casualty to enemies. The key contribution is the formulation of the RTS game strategy selection as CBR planning using a neural-evolutionary model.

2. A new fuzzy integral, order based fuzzy integral, is developed to evaluate unit combination for production. This integral considers the production sequence of game units and the interaction among them. It reflects the player behavior is RTS game.

3. A new strategy planning model is built to extract and evaluate game unit combination strategy from past data. This model provides effectively combat with opponents' troops.

4. A new fuzzy integral, directional based fuzzy integral, is developed to evaluate unit cooperation. This integral can identify the positive and negative interaction for unit in RTS game.

5. A new path planning model is built to provide effective unit maneuver and team compositions in RTS game. This model provides various unit movement, such as flank and diversion attack.

1.6 Thesis organization

This thesis is divided into seven chapters. Chapter 2 is the literature review. It provides the basic information of RTS game and the soft computing technologies that used in this research. Importance contributes from other researchers and research trend are also summarized and stated in this chapter. Chapter 3 is the preliminary study for identifying game units’ best location problem. We combined GA and NN to create a neural-evolutionary model. Chapter 4 is about how to learn the player behavior in RTS game. We applied DBN to solve this problem. Chapter 5 explains our methodology of fuzzy measure learning with different fuzzy integrals and how to solve the macro control problem of RTS game. Chapter 6 presents our work on fuzzy integral and potential field to solve the micro control problem in RTS game. Conclusion and future work are stated in Chapter 7.

Chapter 2

Literature review

In this chapter, the history of real time strategy (RTS) game is stated with the core elements and game play introduced. Moreover, four main research areas in RTS game are highlighted and some importance contributions by other researchers are summarized. Finally, soft computing techniques which used in this research are explained.

2.1 Real time strategy game research and development

2.1.1 History and gameplay of RTS game

Real time strategy (RTS) game is military simulations. Dune II: The Building of a Dynasty (Figure 2.1(a)) is regarded as the first RTS game. It was published by Westwood Studios in 1992. It consisted of all the key elements and mechanics of modern RTS game and serves as a template of RTS game development. After Dune II's success, two famous RTS game series were then created. One was Westwood's Command and Conquer. Another was Warcraft and Starcraft series produced from Blizzard Entertainment, the world second biggest gaming company in 2010. Their revenue is over 4,622 million. Unlike other gaming companies, their products are limited. There are only 21 games produced in these twenty years, while the biggest gaming companies, Nintendo and the third one, Electronic Arts created more than 1000 games. The 21 games of Blizzard are from three main franchises, Warcraft, Starcraft (Figure 2.1(c)) and Diablo. Each of them provides a huge profit.

In 1994, Blizzard released Warcraft, a RTS game set in the realm of a fantasy. Two years later, Warcraft II (1995) as shown in Figure 2.1(b), obtained one of the biggest successes RTS genre. The game had a long value. Wargus was the academic version of Warcraft II and famous in Game AI research. New RTS games since WARCRAFT II brought the genre to a higher level. In 1998, Blizzard released Starcraft and in 2002, Blizzard released Warcraft as shown in Figure 2.1(d).

We selected Warcraft III : The Frozen Throne in our study case. It was the official expansion pack to Warcaft III: Reign of Chaos and published on 1st July 2003. There are two main advances for being a testing bed of research. First, it provides fruitful environment for AI development. Warcarft III consists of more than 100 units and building types. Each unit consists of 40 to 50 properties. The games rules are complicated and consists of many intransitive superiorities (A beats B, B beats C and C beats A [Watson 2001]) and interactions among the unit. In fact, it won many prizes for its gameplay, including the game of the year in GameSpot 2002 and strategy game of the year in Academy of interactive Arts and sciences. Secondly, there are thousands of cases available in the Internet. Warcraft III provides a replay function which record down all the player input. Player can upload their replay and share with others. Hence, there is no strategy or solutions can absolutely win. The styles of players are various. Therefore, it provides sufficient data for research.

Element in RTS games

Although different RTS games provide different experience to the players, there are common core elements found. Three of them are introduced in the following section.

World

First, a battlefield (or map) is given to all players. It is a virtual world with limited size and consists of different obstacles and landscapes, including grassland, mountain, blight and river. They affect the abilities of different units. For example, units in mountain obtain an extra bonus when they attack the units in the grassland, undead units obtain an extra regeneration bonus on blight area and only the air unit can go through the river, etc. The battlefield also provides limited resources. Raising an army or developing new weapons requires resources. Players have to fight for the resources and maintaining a thriving economy.

Buildings and units

Buildings and units are the basic elements in RTS game. They consist of different attributes and skills, for example, cost, hit points, speed, attack point, attack type, etc.

They have different strengths and weaknesses. One of the tricky points in RTS game is the most expensive and powerful units which can easily be destroyed by some cheap units. According to the game rules, players have a degree of freedom to choose his army mix. Then, players control the units to attack or defense.

(a) Dune II

(b) Warcraft II

(c) Starcraft

(d) Warcraft III

Figure 2.1 Screen capture of RTS games

Rules

RTS games consist of complicated rules. They are different in each game. However, two general rules could be found in all RTS games. They are development rule and intransitive superiority rule. Elements in RTS game follow the development rules, such as, include the construction of buildings, the research of new technologies and combat. A tech tree is used to restrict the unit development. Players cannot create all the unit types at the beginning. An example of tech tree in Warcraft III is shown in Figure 2.2. Players need to flight for the resources and create different building or research to unlock the advance unit.

Another common game rule is intransitive superiority. Each unit or skill is assigned to one or more types. Each type is better or worse versus others. For example, if the opposing player builds ranged attackers, then the natural counter will need to build melee unitswhich have an attack bonus versus them. It encourages unit counters and unit mixing in combat. Therefore, there is no strategy or solutions which can absolutely win.

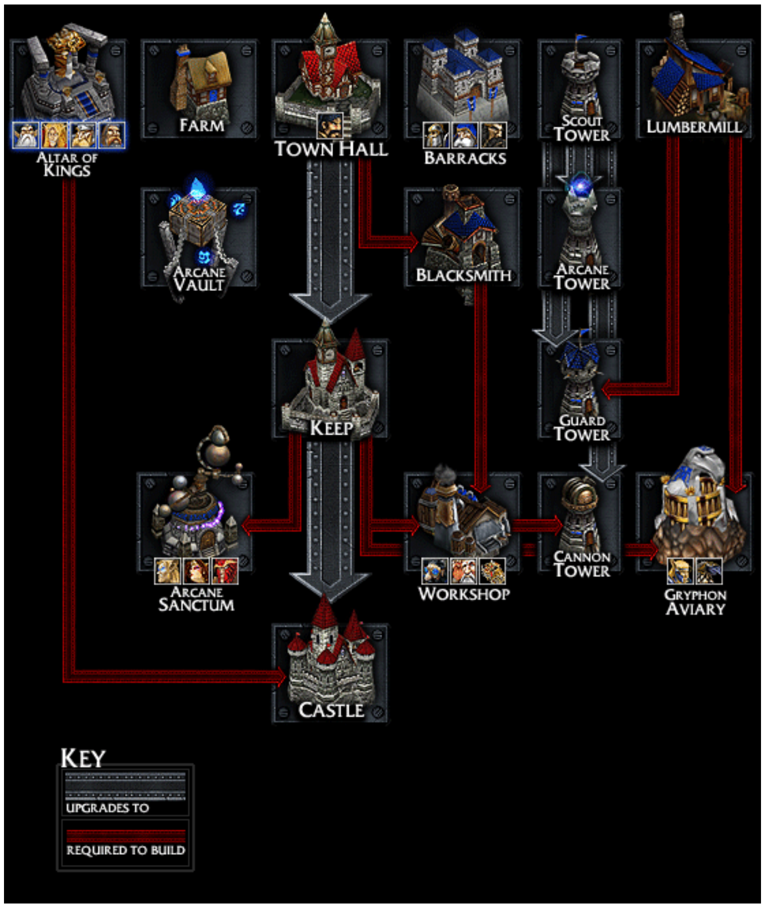


Figure 2.2 Tech-tree of Warcraft III (Human race)

Gameplay of RTS game

The fundamental game play of a typical real time strategy (RTS) game is collecting and allocating resources to build an army and destroy enemy units. Base on the game rules, players are required to decide what buildings or units should be created, what kinds of advanced skill or unit should be unlocked, when and where the units should attack the opponents. We can divide these controls and decision into two types. They are introduced as followings.

Macro control (Strategy)

Macro control or strategy is about the economic model of resource-gathering, base-building and technology development. It tends to predict the future of the battle and the overall situation.

Micro control (Tactics)

Micro control or tactics is about the maneuver of the units, including movement, attack, defense and other special skills. It tends to predict the current situation of the battle and individual status of units.

2.1.3 Research in RTS game

RTS game is a good testing bed for AI research as it consists of complicated game rules and numerous kinds of units. There are many remaining challenges because of its complex decision making under time pressure and uncertainty, such as resources management in macro control or robust terrain analysis in micro control. Hence, the search space of strategies is large and often involves complicated interactions or intransitive superiorities among the game units and corresponding actions. According to Buro [Buro 2003, 2004], there are four main areas of game AI research that is under study. In the following session, a brief history of these research areas is introduced.

Resource management (Macro control)

The first one is resource management which is stated as macro control in our research. It is the main part of RTS game research with most of researchers get interested in this part.

They focus on the decision making for the sequence of actions, including the time to extend the base and upgrade the tech tree in RTS games. Rule based system is dominant in current RTS game and earlier research such as [Jones 2001]. Genetic algorithm and other evolution algorithms have also applied to search for a best fit sequence of actions in macro control such as [Reynolds 2005]. In recent years, Aha and his group have applied Case based reasoning (CBR) and provided many important contributions. CBR looks like a promising starting point for macro control and its details are introduced in the next session. Other researchers, such as Ontanon [Ontanon 2007] and Sharma [Sharma 2007], have also followed CBR approach and continued its development.

Adversarial real time planning (Micro control)

The second one is adversarial real time planning which is stated as micro control in our research. The search space of the units' movement is nearly infinite in RTS game. The first problem is how to determine the destination and the path. Miles [Miles 2006] and Hagelback [Hagelback 2008] adopted potential field to solve this problem. The destination and path can easily be indentified in the dynamic environment. It is easy to adopt and efficient to compute in real time. Then, the research of micro control extends to handle the unit grouping and details maneuver, such as flocking and tracking problem. Potential field still have been widely used such as [Preuss 2010] and [Beume 2008]. In these few years, some researchers improved the traditional path finding algorithm, so that, it can provide inexact solution in a more efficient way for the real time game. Baumgarten [Baumgarten 2009] applied simulated annealing as a fast converge method to locate the destination. Mingliang [Mingliang 2010] has improved the A* path finding to find multiple paths for tracking the enemy. Keaveney [Keaveney 2011] has applied the backward reasoning approach in genetic programming to locate the destination.

Player and opponent modeling and learning

Opponent modeling and learning is mainly combined with the macro control. Usually, the researchers turn the sequence of actions and the conditions from human player into cases and learn by model. One of the advances is to allow the AI system to learn quicker. Hence, by learning the opponent pattern, the performance of counter strategies could be improved. Again, CBR is a common approach. Besides CBR, genetic algorithm, Bayesian network and neural network have been applied to learn player and opponent modeling, such as [Louis 2005, Jack 2006].

Spatial and temporal reasoning

Spatial and temporal relationship among the actions is difficult to investigate. In fact, all the above research involves spatial and temporal reasoning but all of them are designed for particular platform and cannot be generalized. Current RTS game AI has ignored these issues and will be easily confused in common sense reasoning [Forbus 2002]. One of the examples is the temporal reasoning among the action in CBR. CBR approach groups the sequence of action into cases. Decision of actions is affected by time, player and opponent control. They are correlated to each other and the time slices is highly flexible. It caused the difficulty to reform the reasoning.

2.2 Soft computing techniques applied to RTS games

2.2.1 History and characteristics of soft computing techniques

Traditional rule based, game-tree searching and brute force approach is not suitable in this dynamics environment. It is time consuming in both development and run time. In RTS game, an efficient method is needed to provide a best fit solution as all the conditions, targets and destinations will be changed in milliseconds. Soft computing technologies become a reasonable approach. It was first introduced by Zadeh [Zadeh 1994]. The aim of soft computing is to provide inexact or best fit solution to computationally hard tasks which is suitable to describe the situation in RTS game. In the following session, we have selected some key contributions of applying soft computing in RTS game.

2.2.2 Case-based planning in RTS game

Case Based Reasoning (CBR) is a suitable approach to deal with the strategy planning in RTS game as it can handle the inexact strategy planning efficiently. We try to compare three experiments. The first one is written by David W. Aha [Aha 2005]. Platform of his experiment is Wargus, which is an academic version of Warcraft II. Second paper is written by Santiago Ontañón [Ontanon 2007]. Platform is WARGUS, too. The third paper is written by Ji-Lung Hsieh [Hsieh 2008]. Platform is starcraft.

The first challenge of applying CBR is to encode the complex and continuous environment and then turn it into cases for offline learning, i.e., case representation problem. Aha divides the game play into 20 states and 8 different AIs. He defines a case as following.

Case = <Building State, Description, Tactic, Performance>.

Santiago defines a case as following.

Case = < State, Goal, Behavior groups>

Hsieh defines a case as following.

Case = <Building actions, States $\{F_1^a, F_2^a, \ldots F_6^a\}$, Performance>

In general, the authors turn the data from simulation or real game replay into cases. The cases combine conditions, group of game actions and performance. Case clustering is performed by using the game states. It can speed up the case retrieve process and control the number of cases in a reasonable number.

The second challenge is how to compare the case similarity. Case is retrieved by evaluating the similarity of current situation and its performance, which is usually the score inside the game. For example: number of kills or number of destroys. Aha uses the similarity of description and the value of performance to select the building state and tactic in case base as Equation (2.1). Santiago uses similarity of game state and goal to select the behavior groups as Equation (2.2). Goal is the building or tech tree development in the cases. In another words, it is another kind of states. Hsieh uses similarity of game state and the value of performance to select the building action as Equation (2.3).

$$Sim(C,S) = C_{\mathrm{Pr}eformance} / dist(C_{Description}, S) - dist(C_{Description}, S) \tag{2.1}$$

where C is the case in case based

$C_{Description}$ is the vector of case situation

$C_{\mathrm{Pr}eformance}$ is the score of case

S is the situation of new case

$dist$ is the Euclidean distance

$$Sim(C,S) = adist(C_{Description}, S_{Description}) + (1-a)dist(C_{Goal}, S_{Goal}) \tag{2.2}$$

where a is control weighting and $a = 0.5$

$$Sim(C,S) = C_{Performance} / dist(C_{Description}, S_{Description}) \tag{2.3}$$

The three equations are more or less the same. In general, case is retrieved by evaluating the similarity of current situation and its performance. When there is a new situation, a solution has to be seek accordingly. The new situation is compared with all the cases in case based one by one. The case with the highest performance is chosen as the solution. By observing the weighting of the above equations and compare the performance, similarity will usually be dominant in the equation. It leads the number of alternative cases become very little. Another weakness of the equations is that they do not considerate the unit combination. Unit combination is a key element and will directly affect the result of battle. It is difficult to involve in conditions as the combination is numerous. The number of cases will increase sharply.

For the similarity, nearly all authors are using the Euclidean distance to compare the differences between two cases. It is a simple calculation to find out the same cases. However, if the input factors are not weighted and normalized, it may be then easily dominant by some other fields. For example, the number of building in RTS game is less than unit. Unit will be dominant in Euclidean distance but building in RTS game always shows its importance in evaluating the similarity. Some authors choose to balance the weight of input factors but they cannot show their reason of adjustment.

For the performance, it is usually calculated by number of kills or destroys in certain cases. It is a general estimation for traditional game, such as Wargus, which is already 15 years old. In another words, it is an aggressive approach to lead the AI to win in the old game. However, current RTS game contains complex game play; players can perform complex strategy to trap the opponent by misleading. They can win the game with very low scores,for example heroes rush and disturb strategy in Warcraft III. Hence, such kind of performance calculation cannot be easily performed with a various AI in game. Finally, equation is always an ad hoc solution for one game. The factor and the structure need to tune in every RTS game. They do not have any theories or general methodologies behind.

2.2.3 Reinforcement learning in RTS game

Reinforcement learning (RL) in game AI is proposed by Szita, Spronck and Ponsen who work with Aha. It is a framework that based on the process of punishing and rewarding on game action. It is fast and easy to implement. The fitness function is used to evaluate a performance for game AI and expressed in a numeric value that known as weight. The higher the weight is the more suitable action. In another words, if the action is good, reward it by increasing the weight value. If it is a bad action, punish it by decreasing the weight. For each runtime, RL will try to maximize the frequency of rewards and minimize the frequency of punishments. As a result, it tries to perform the best action and condition pairs through the past experiment. Ponsen and Spronck [Ponsen 2005] have shown a solid work about adopting RL in RTS game.

RL have improved the CBR, especially in the case revision and retain process. It can evaluate the individual action in case. Therefore, actions from different cases can be combined. Ponsen [Ponsen 2005] has considered long term effect of cases, GC/EC , and the military power of player and opponent as shown in Equation (2.4). Every cases contain numerous of action in different states, i. Then, the performance of actions are updated by a weighting, W , as shown in Equation (2.5) one by one. Based on W, he also proposed a method to recombine the game action from different states automatically by Genetic Algorithm. It helps CBR to achieve a better performance case. An example of crossover is shown in Figure 2.3. Szita [Szita 2009] proposed a diverse case retrieve process in game. Therefore, CBR will not perform the exact action in the same situation. The game action is recombined and the fitness is calculated by cross entropy. Mehta [Mehta 2009] also showed similar approach in his research. He proposed to detect the failure pattern and publish the cases, such as continuously repeating behavior or wrong sub-goal of the cases. The structure of his CBR is shown in Figure 2.4.

$$F = \begin{cases} \min\left(\frac{GC}{EC} \times \frac{M_d}{M_d + M_o}, b\right) & d \quad lost \\ \max\left(b, \frac{M_d}{M_d + M_o}\right) & d \quad win \end{cases} \tag{2.4}$$

where F is the fitness of action

M is the military power

o is the opponent, d is the player

GC/EC is used to ensure the case achieve a higher score in long term

b is the control weighting of minimum and maximum value

$$W = \begin{cases} \max\left(W_{\min}, W_{org} - 0.3\dfrac{b-F}{b}P - 0.7\dfrac{b-F_i}{b}P\right) & F < b \\ \min\left(W_{org} + 0.3\dfrac{F-b}{1-b}R + 0.7\dfrac{F-b_i}{1-b}R, W_{\max}\right) & F \geq b \end{cases} \tag{2.5}$$

where W_{org} is the original weighting before learning

$W_{\min}$ and $W_{\max}$ is the control weighting of minimum and maximum value

R is maximum reward and P is maximum punishment

F_i is fitness of the rules at state i

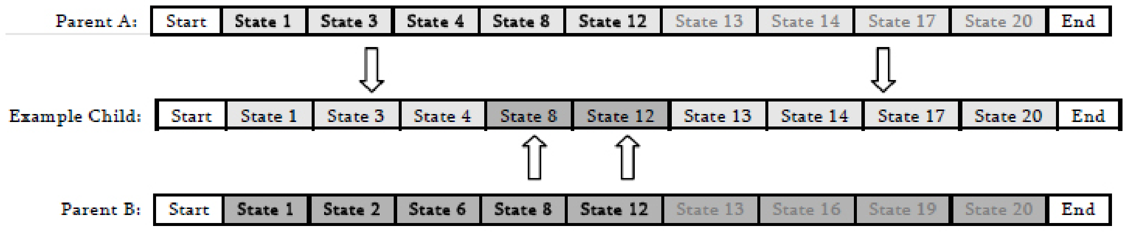


Figure 2.3 Crossover of strategy cases by Ponsen

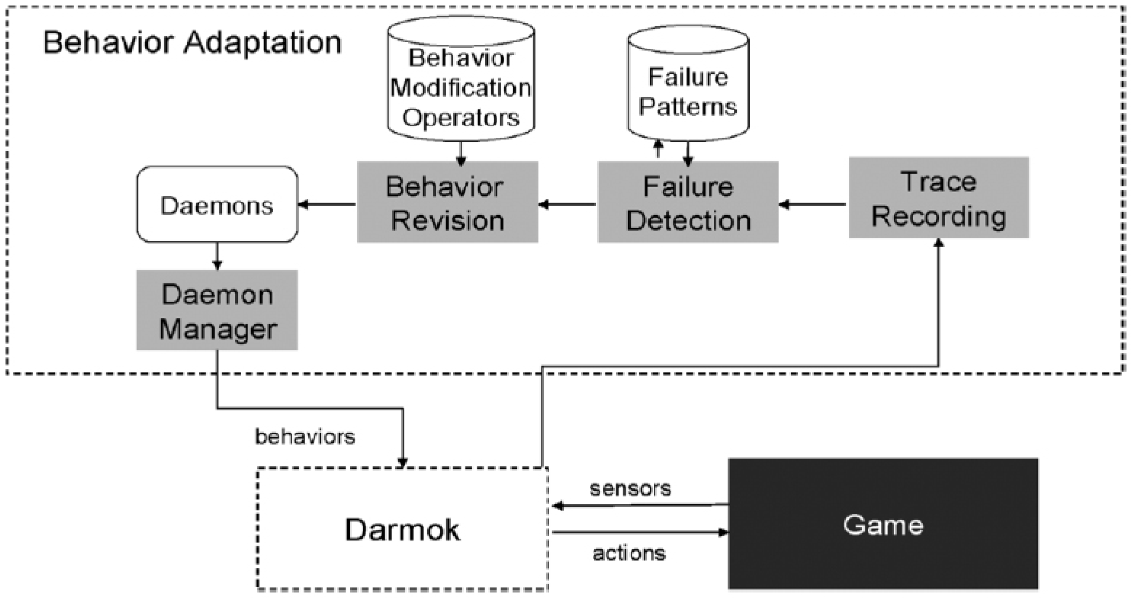


Figure 2.4 Structure of failure detection CBR by Mehta

2.2.4 Online learning in RTS game

Learning could be classified into online and offline. Offline learning is performed after the game finished. Common methods are CBR, GA and Bayesian Network. They have been widely used because they are good classifiers and able to return a higher rate of accuracy. However, they are slow and request heavy computation process. In contrast, online learning is preformed during the game. It is proposed by Spronck. The main structure is shown in Figure 2.5. The whole process is similar to RL. All the action is stored in a ruled base system. The reward value with the action and condition is stored in lookup tables. A reward is given to the correct action. The correct action could be more easier selected by the model. It is fast to achieve in the learning purpose.

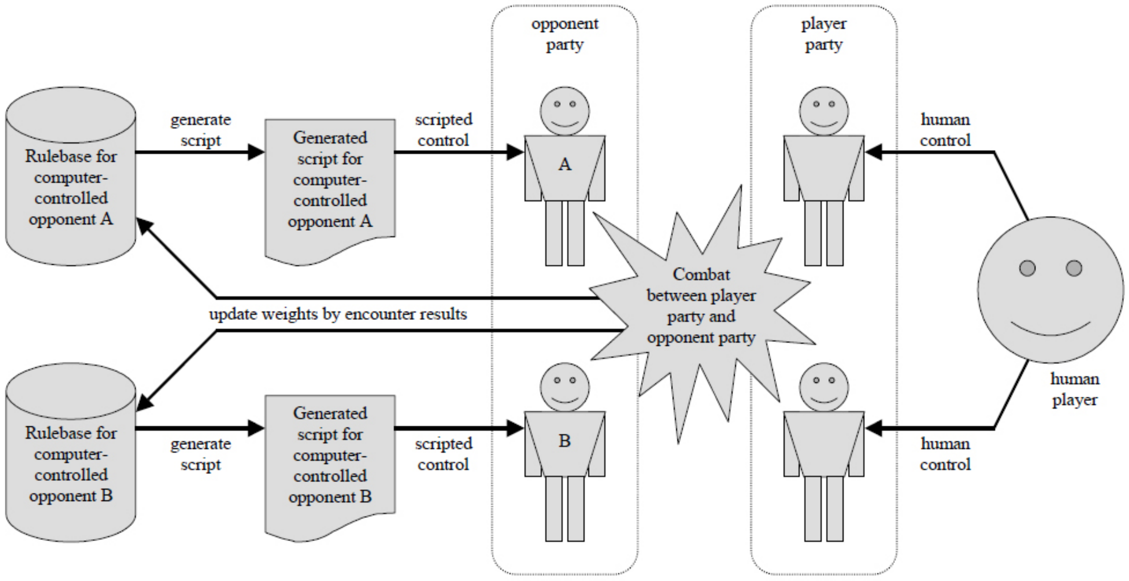

opponent party
player party
Rulebase for computer-controlled opponent A
generate script
Generated script for computer-controlled opponent A
scripted control
A
human control
Combat between player party and opponent party
update weights by encounter results
human player
Rulebase for computer-controlled opponent B
generate script
Generated script for computer-controlled opponent B
scripted control
B
human control

Figure 2.5 Online learning model by Spronck

Online learning could apply to predict the player behavior and regards as a case indexing method. A reward could be given to a player action condition pair. Currently, it is not practical in RTS game as player behaviors vary a lot and the repeated action will not appear so frequently in one battle. Anti-cheating algorithm is another application of online learning especially in first person shooting and online role play game. The action is generated by cheating machine and can be regarded as a special kind of player behavior. The actions of cheating vary in each battle but they perform regular action and condition pair pattern in one battle. Therefore, online learning could be applied to this kind of researches, such as Chamber [Chamber 2005] and Yeung [Yeung 2006] research.

Hence, as the conditions in RTS game are complex, the dimension of the lookup is highly increased. An alternative approach is using ANN to replace lookup table. It is a fast indexing method for CBR. Such kind of process does not require preprocessing.

2.2.5 Multi-agent potentials field in RTS game

Unlike traditional path finding problem which only considers few conditions and a clear destination, such as cost or time, path finding in RTS game does not have a clear destination and need to consider lots of additional conditions, such as avoiding the enemies attack. Hence, inside the dynamic environment of the game, all the conditions are changed in every game cycle which is only few milliseconds. In every game cycle, 30 to 50 units need to update their paths. It is one of the heavy tasks in game AI. Traditional A* path finding is resources intensive and difficult to fulfill the need in RTS game. Hagelback and Johansson [Hagelback 2008, Johansson 2008] worked together to apply potential field in micro control of RTS game. They called it as Multiple Agents Potential Field (MAPF). The testing bed was ORTS which was an open platform for real time strategy game. MAPF shows the ability to avoid colliding with the terrain and getting stuck at other moving objects. It consists of six phases.

First one is the identification of objects, e.g., gold mines or enemies of the game. Second one is the identification of the driving forces. It can be regarded as a weighting of object. Third one is the process of assigning driving forces (or charge) to the objects' coordinate. It generates a potential field around itself. These fields of different objects are summed up to form a total potential field that is used by the agents for navigation. For example, the potential of base is an attractive force and is calculated as Equation (2.6) and Figure 2.6. Each object generates different charges.

$$P_{base}(d) = \begin{cases} 5.25d - 37.5 & d \le 4 \\ 3.5d - 25 & d \in]4, 7.14] \\ 0 & d > 7.14 \end{cases} \tag{2.6}$$

where d is the distance from a point to the closest base

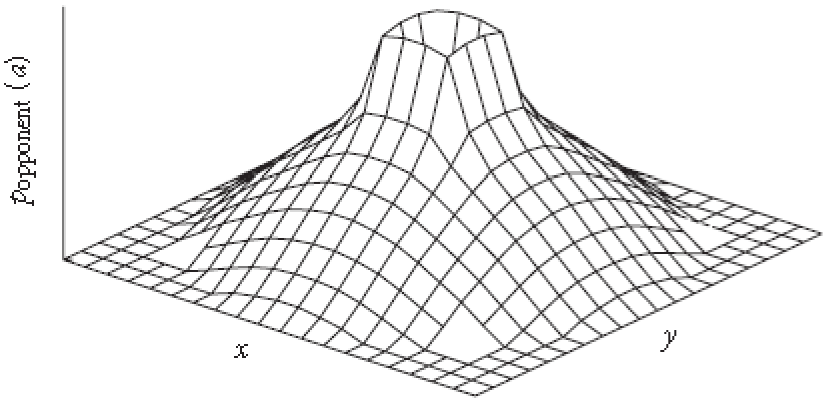


Figure 2.6 Sample potential field of Hagelback

The forth one is the granularity of time and space in the environment. Hagelback did not give a details description about this task. He only pointed out that ORTS is a simple application and his experiment was able to use the full resolution and the time frame without considering any time and space problem. Otherwise, MAPF should be limited in a fixed resolution of battlefield or fixed time slot to implement. The fifth one is the agents of the system. After all the objects are identified in the battlefield, the units that required to perform path finding would be treated as an input and to calculat the distance for each object. The unit will then move from high potential area to low potential area. Final one is the scripting of the agents. It is the interface between agents and game server. In another word, the system generated the actions for the units.

Hagelback compared MAPF with other AI in ORTS 2008 tank battle competition. The averaged winning percentage is 99.25%. His bot won this game in the 2008 years' ORTS competition. MAPF shows a better approach for path finding in dynamic environment of

RTS game. It is not resource intensive but is able to implement in many kinds of computer game. The main weakness is all the equations for the potential field are ad hoc on only certain game play. The driven forces of object are assigned by experts. As a result, it is a time consuming task for AI developers to try out the best parameters set. They need to run the simulation repeatedly. Hence, it is also not practical in real RTS game as it consists of many units. The weighing is highly depends on the developer. Hence, as the equation is not generalized and without any theory to support, it is difficult to adopt into other games. It is a good approach but we suggest using machine learning or other soft computing techniques and theories for the equation and parameter learning.

Preuss [Preuss 2010] improved the efficiency of potential field. His testing bed was Glest which is another open source RTS game published by Figueroa. He applied flocking idea, C_F , into potential field. Therefore, the number of calculation is decreased from the number of units to the number of group (or unit type). As shown in Equation (2.7), the driving force of unit is assigned by its hit point (HP) and attack strength (D), not by human expert. However, the aggregator is still normal additive. Super additive and sub additive could not be shown. They are commonly found in RTS game. Detailed micro control or path finding cannot be shown.

$$C_F = \frac{1}{SEW} \sum_{u \in F} HP_u \sum_{u \in F} D_u \tag{2.7}$$

where u is the unit in the group F

$\frac{1}{SEW}$ is the weighting to scale the aggressiveness of units

HP is the hit point of unit

D is the attack strength of unit

2.3 Soft computing techniques used in this research

2.3.1 Fuzzy measure and fuzzy integral

Fuzzy measure theory and Choquet Integral was introduced by Gustave Choquet [Choquet 1953] in 1953. The concept of its fuzziness and regarded as fuzzy integral was introduced by Sugeno [Sugeno 1974] in 1974. It can be regarded as generalization of the classical probability measure, and has become an effective tool to describe the interaction among the contributions from individual attributes or variables. A value is assigned to each combination of variables, i.e., for n variables, there are $2^n - 1$ parameters to be determined.

There are three kind of fuzzy measures. The first one is non-additive fuzzy measure. They are the original form (basic form) of fuzzy measure. The definition is shown as following.

Definition of non-additive fuzzy measure

Let (X, F) be a measurable space. Set function $\mu : F \to (-\infty, +\infty)$ is called a fuzzy measure (monotone measure) if the following criteria are all fulfilled, i.e., (2.8), (2.9) and (2.10)

$$\mu(\phi) = 0 \tag{2.8}$$

$$\mu(A) \geq 0 \text{ for every } A \in F \tag{2.9}$$

$$\mu(A) \geq \mu(B) \text{ whenever } A \in F,\ B \in F,\ A \subseteq B \tag{2.10}$$

The essential difference between fuzzy measure and traditional measure is that the former one does not need to satisfy the additive property. These conditions can be further elaborated as follows.

Set function μ is called non-monotonic fuzzy measure or efficiency measure [Wang 2008] if it satisfies (2.8) and (2.9). μ is also called a signed efficiency measure if it satisfies (2.8).

Throughout the development of fuzzy measure, there are two simplified fuzzy measure, such as Sugeno-λ and k-additive fuzzy measure. Sugeno-λ fuzzy measure is a special case of fuzzy measure, which can be obtained by determining only one parameter λ. The definition is shown as following.

Definition of Sugeno-λ fuzzy measure

Let (X, F) be a measurable space and let $\lambda \in (-1, \infty)$. Sugeno-λ fuzzy measure is a function g from $P(X)$ to $[0,1]$ with properties (2.8), (2.9), (2.10), (2.11) and (2.12)

$$g(X) = 1 \qquad (2.11)$$

$$g_{\lambda}(A \cup B) = g_{\lambda}(A) + g_{\lambda}(B) + \lambda g_{\lambda}(A) g_{\lambda}(B) \qquad (2.12)$$

It reduces the complexity of fuzzy measure by defining $\lambda \in (-1, \infty)$ and $1 + \lambda = \prod_{i=1}^{n} (1 + \lambda g_i)$ to represent relationship between all the interactions. Therefore, its presentation ability is weaker than original non-additive fuzzy measure. If $\lambda > 0$, then μ is super additive on all the $2^n - 1$ subsets of X , i.e., the interactions among all the variables in X are positive If $\lambda < 0$, then all μ will become sub additive. It is not suitable for RTS game play, where some units work positively while some others work negatively.

K-additive fuzzy measure is another approach to reduce the number of parameters in determining a fuzzy measure to size k. The larger k represents the stronger presentation ability. The definition is shown as following.

Definition of k-additive fuzzy measure

Let (X, F) be a measurable space. k-additive fuzzy measure, μ is defined on X with properties (2.8) and Möbius representation (or inverse) of μ is another set function defined by (2.13) and with properties (2.14), (2.15) and (2.16)

$$m_\mu(A) := \sum_{B \subseteq A} (-1)^{|A \setminus B|} \mu(B) \ , \ \forall A \subseteq X \tag{2.13}$$

The original set function is recovered through Zeta function $\mu(B) = \sum_{B \subseteq A} m(B)$ (2.14)

If its Möbius representation verifies $M(E) = 0$ where $|E| > k$ for any $|E| \subseteq X$ (2.15)

There exists a subset F with k elements such that $M(F) \neq 0$ (2.16)

We have performed some experimental study by using Sugeno-λ and k-additive fuzzy measures. However, the learning result is not good because of their limited representation ability and monotonic assumption, i.e., $A \subset B$ implies $\mu(A) \leq \mu(B)$. In this study, due to the complex interactions among the unit types in RTS games, we suggest adopting the original non additive fuzzy measure. Furthermore, we will elaborate the monotonic assumption of μ when applying to modeling of RTS games.

Fuzzy integral

The fuzziness of fuzzy measure theory is inside fuzzy integral. As the subset of fuzzy measure is huge, subset selection should be preformed. The accuracy of performance is mainly based on the correct subset selection. Fuzzy integral is used to perform the subset selection and provide an effective and efficient aggregation. It can be regarded as a generation of classical Lebesgue integral corresponding to additive measure.

Choquet Integral (CI) is a commonly used fuzzy integral, which is a straightforward expansion of Lebesgue Integral (LI). In a classical measure with no interaction involved, CI is equal to LI as Figure 2.6. and Equation (2.17)

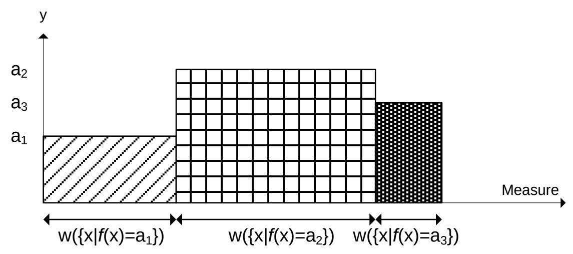


Figure 2.7 Lebesgue Integral. w **is the measure of a function** $f(x)$

$$(1)\int fd\mu = \int fd\mu \tag{2.17}$$

If $X = \{x_1, x_2 ..., x_n\}$ is finite, Lebesgue integral can be written as its discrete form, i.e., the weighted sum as Equation (2.18)

$$\int_X fd\mu = \sum_{i=1}^{n} w(\{x_i\}) f(x_i) \tag{2.18}$$

Here, μ is additive, for any $A \subset X$, $\mu(A) = \sum_{i=1}^{n} w(\{x_i\})$, $x_i \in A$. CI has been a general tool for dealing with multiple criteria decision making problems and is able to model the interactions among different criteria. Sugeno [Ishii 1985] [Murofushi 2000] has applied it to the problem of prediction of wooden strength and plant operator. Both of them show a better result than a linear regression. Peter [Peters 1999] has applied CI in software cost estimation with multiple attribute. Suppose a fuzzy measure $\mu : F \to (0,1)$ and $\mu(\phi) = 0$, definition of CI, see Figure 2.7 and Equation (2.19), is shown as follows and a_i is the sorted increasing sequence of $f(x_i)$.

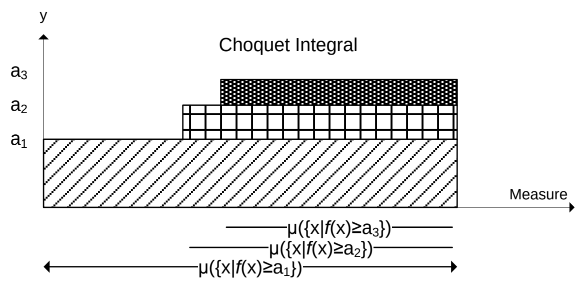


Figure 2.8 Choquet Integral. μ is the fuzzy measure of a function $f(x)$

$(c)\int f(x)\circ\mu(X)=\sum_{i=1}^{n}(a_i-a_{i-1})\cdot\mu(\{x\mid f(x)\geq a_i\})$ or

$$(c)\int f(x)\circ\mu(X)=\sum_{i=1}^{n}(a_i-a_{i-1})\cdot\mu(F_\alpha) \tag{2.19}$$

where $a_0=0$ and $a_0\leq a_1\leq a_2\leq ...a_n$ and $F_\alpha=\{x\mid f(x)\geq a_i\}$

Some CI properties under non-monotonic fuzzy measure are shown as follows,

$$(c)\int_A 1d\mu=\mu(A) \tag{2.20}$$

If $f\leq g$,then $(c)\int fd\mu \leq (c)\int gd\mu$ (2.21)

If a is non-negative real number and b is a real number, then (2.22) $(c)\int(af+b)d\mu=a(c)\int fd\mu+b\mu(X)$

Michio [Michio 1994] and Kwon [Kwon 2000] had proven that CI and its properties were also meaningful when the fuzzy measure was non-monotonic, i.e., efficiency measure. Murofushi [Murofushi 2005] also stated that non-monotonic measures occur when there are limited resources. RTS game is similar to this situation. Time and money of RTS is limited. When the player wastes his resources to develop many unsuitable types, the enemy will advanc and destroy the player's resources. It is also confirmed by many game reviews that: professional players will not create many different kinds of units in one battle, but only concentrate on certain important combinations. This means that using more unit types is not necessarily more powerful than using fewer unit types. Therefore, the monotonity (2.10) is no longer satisfied in this particular application of RTS games. Fuzzy integral is computed with respect to non-monotonic fuzzy measure. Besides, as mentioned in Section II, the units must be built in a predefined sequence, and this information should be taken into account in overall evaluation of unit combinations by fuzzy integral. In this paper, we propose a new type of integral and compare it with Choquet integral.

2.3.2 Genetic algorithm

Genetic Algorithm (GA) was introduced by Barricelli [Barricelli 1957] and Fraser [Fraser 1970]. It is a kind of evolutionary algorithm inspired by the biological evolution.

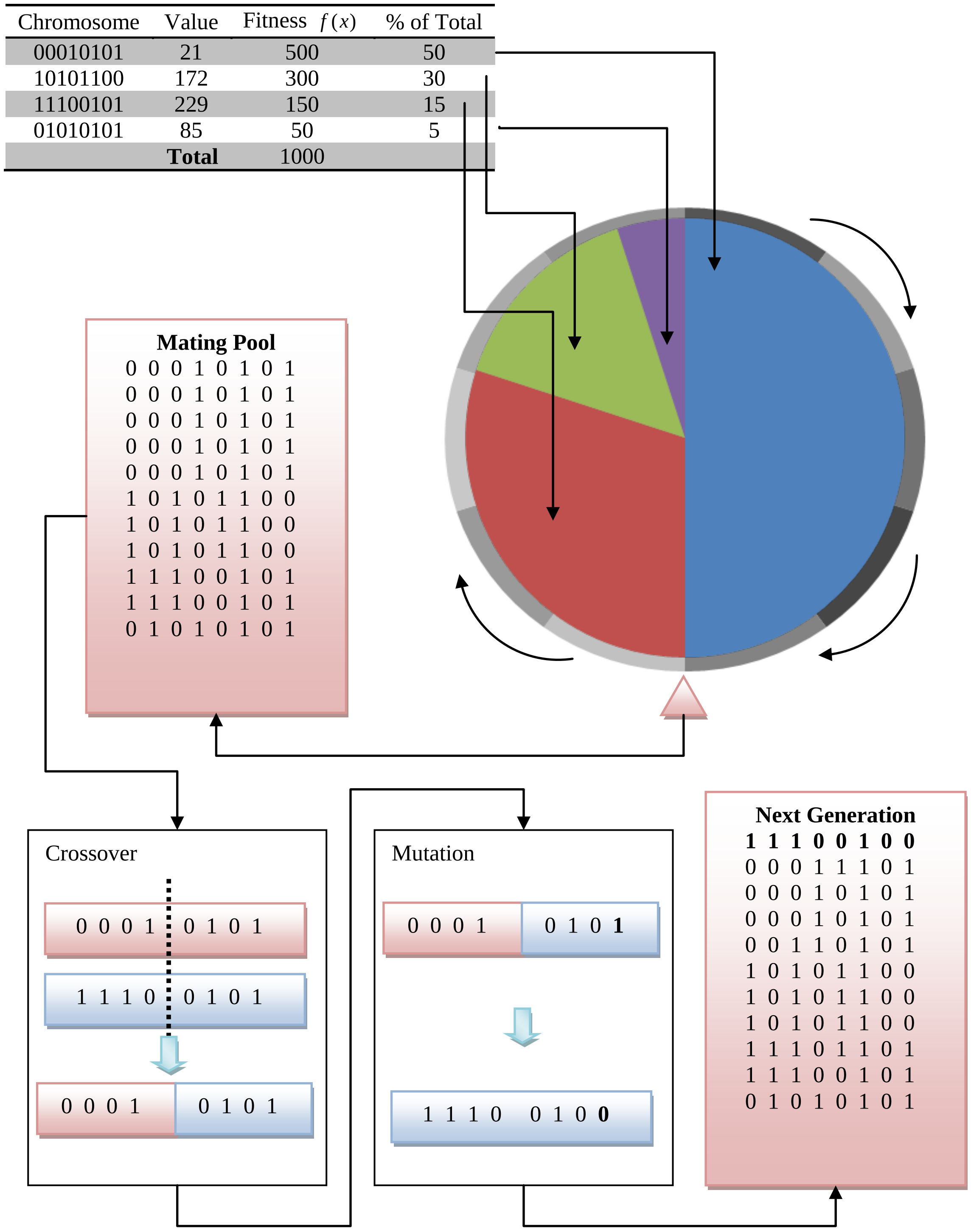
First, a population of string or called chromosomes is randomly generated. Traditionally, each chromosome is represented by a binary string of 0s and 1s. Each chromosome represents a solution. A fitness function, $f(x)$, is given to evaluate all chromosomes.

Based on the fitness value, selection is performed to choose a solution with better quality. In our research, roulette wheel selection is used. The finesses of all chromosomes are summed up. A proportion is then given to each chromosome. Therefore, weaker individuals can be selected with a lower chance. The chromosomes are assigned to a roulette wheel with its proportion. The roulette wheel is spun equal to the size of population. Each time, a chromosome is chosen and put into the mating pool. The solution with better quality will occur more frequently. There are other selection methods, such as Boltzmann selection, tournament selection, rank selection and steady state selection, random selection, etc. Roulette wheel selection was used as it is more natural and with a few parameters. However, it will have a problem when the fitness values differ very much. For example, if one of the chromosome finesses is dominant and controls over 90% of area. The other chromosomes will have less chance to be selected. However, it could be improved by increasing the number of population and mutation.

Then, a genetic operator, crossover is used to generate a solution with better quality for next generation. More than one chromosome is selected as the parent. They are spliced into pieces. The child chromosome is combined by part of its parent. There are numbers of crossover operators, such as one point crossover, two point crossover, cut and splice, uniform crossover, three parent crossover, etc. In our research, one point crossover is used as we can observe clearly which part of the parent chromosomes can give a better fitness. The steps of one point crossover is descripted in Figure 2.8. First, a random position is selected in the parent. All data beyond that point in either organism string is swapped between the two parent organisms. Throughout this crossover operator, global optimization is preformed. All the chromosomes are trend to a better solution.

Then, another genetic operator, mutation, is used to avoid the searching mechanics fall into a local maximum easily. A mutation rate α is given. For each bit, a random number R_i ($0 < R_i < 1$), the bit is converted into the opposite if R_i is greater than α .

Throughout the two genetic operators, a new population of chromosomes is produced. The next generation chromosome repeats the process of fitness calculation, selection, crossover and mutation until the termination condition reached, such as the solution satisfies minimum fitness error or a fixed number of generation reach. The solution is regarded as best fit if the fitness cannot show significant improvement in certain generation.



Chromosome	Value	Fitness $f(x)$	% of Total
00010101	21	500	50
10101100	172	300	30
11100101	229	150	15
01010101	85	50	5
	Total	1000	

Figure 2.9 Process of Genetic algorithm

2.3.3 Covariance matrix adaptation evolution strategy

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) was introduced by Hansen [Hansen 2001]. Similar to GA, it is another evolutionary algorithm for non-linear, non-convex, non-separable and non smooth optimization problems in continuous domain. First, a group of search point, for $k = 1,...,\lambda$, is randomly generated in a n dimensional space where n is the number of features. It is regarded as the initial population. Function, f , is used to calculate the fitness of all search points. All the search points are updated in each generation and moved to a better fitness of solution. The basic equation of each search point at a generation, g , is shown in Equation (2.23)

$$x_k^{(g+1)} \sim m^{(g)} + \sigma^{(g)} N(0, C^{(g)}) \qquad \text{for } k = 1,...,\lambda \tag{2.23}$$

where $\sim$ denotes the same distribution on the left and right side

$\lambda \geq 2$ is the population size

x_k^{g+1} is the k - th search point in generation $g+1$ of feature x

$m^{(g)} \in \mathbb{R}^n$ is the mean value of search distribution at generation g

$\sigma^{(g)} \in \mathbb{R}_+$ is the overall standard deviation of step-size at generation g

$C^{(g)} \in \mathbb{R}^{n \times n}$ is the covariance matrix of search distribution at generation g

$N(0, C^{(g)})$ is a multivariate normal distribution with zero mean and $C^{(g)}$

There are three operators, m , σ and C , for the search points movement. They are updated in each generation. m is the mean value updater for the global search. The equation is shown in Equation (2.24). First, the finesses of all the search points are calculated. The best μ search points are selected for calculating the new mean. w is the positive weight coefficients for re-combination. If w_i is sorted in ascending order, then the movement is dominated by better fitness search points. If $w_{i...\mu} = 1/\mu$, then m will be the mean value of the best μ search points.

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g+1)} \tag{2.24}$$

where $\mu \leq \lambda$ is the parent population size

$w_{i=1...\mu \in \mathbb{R}_+}$ is the positive coefficients for recombination

$\sum_{i=1}^{\mu} w_i = 1$ and $w_1 \geq w_2 \geq ... \geq w_\mu > 0$

$x_{i:\lambda}^{(g+1)}$ is the i - th best search point and $f(x_{1:\lambda}^{(g+1)}) \leq f(x_{2:\lambda}^{(g+1)}) \leq ... \leq f(x_{\lambda:\lambda}^{(g+1)})$

$C^{(g)}$ is the covariance matrix of search distribution. By the adopting the concept of Hessian and covariance matrices, $\sigma^{(g)} N(0, C^{(g)})$ is used to change the shape of search point distribution from circle as shown in Figure 2.10(a) to directional ellipse as shown in Figure 2.10(b). Therefore, CMA-ES provides a faster converge during the evolutions.

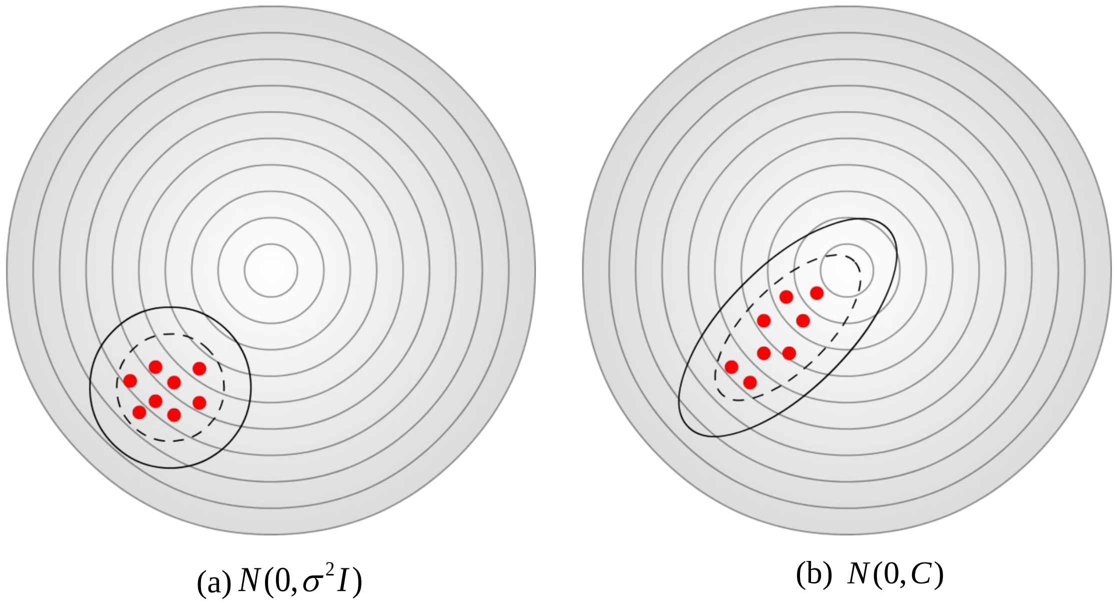


Figure 2.10 The directional optimization of the CMA-ES algorithm

The equation of covariance matrix update is shown in Equation (2.25). It is usually combined by Rank-μ updater as shown Equation (2.26) and Rank one updater as shown Equation (2.27). Rank-μ updater is used for global step-size control with a decay factor,

$(1-c_\mu)$. The sum of the outer products is $\min(\mu, n)$. Rank one updater is used to add the maximum likelihood term into the covariance matrix, C^g. The evolution path in Rank one updater, p_c , is shown in Equation (2.28). It is the search path of each search point. It can be expressed as a sum of consecutive steps of the mean, m . History information is accumulated in the coefficient, μ_{eff} , as shown in Equation (2.29). $\sqrt{c_c(2-c_c)\mu_{eff}}$ is a normalization factor for p_c. By using an evolution path for the Rank one update, the number of function evaluation is decreased from $O(n^2)$ to $O(n)$ [Hansen 2003]. $\sigma^{(g+1)}$ is the standard deviation of step size control as shown in Equation (2.30). The solid line of Figure 2.11 shows the improvement. It tends to make the step conjugated after the adaptation has been successful, i.e., $\left(\frac{m^{(g+2)}-m^{(g+1)}}{\sigma^{(g+1)}}\right)^T C_g^{-1} \frac{m^{(g+1)}-m^{(g)}}{\sigma^{(g)}} \sim 0$. It is increased if and only if $\|p_\sigma\|$ is larger than the expected value and decreased if it is smaller.

$$C^{(g+1)} = (1-c_\mu-c_1)C^g + c_\mu \sum\nolimits_{i=1}^{\mu} w_i x_{i:\lambda}^{(g+1)} x_{i:\lambda}^{(g+1)^T} + c_1 p_c^{(g+1)} p_c^{(g+1)^T} \tag{2.25}$$

where c_μ and c_1 is the control weighting

$$(1-c_\mu)C^g + c_\mu \sum\nolimits_{i=1}^{\mu} w_i x_{i:\lambda}^{(g+1)} x_{i:\lambda}^{(g+1)^T} \tag{2.26}$$

$$(1-c_1)C^g + c_1 p_c^{(g+1)} p_c^{(g+1)^T} \tag{2.27}$$

$$p_c^{(g+1)} = (1-c_c)\,p_c^{(g)} + \sqrt{c_c(2-c_c)\mu_{eff}}\;\frac{m^{(g+1)}-m^{(g)}}{\sigma^{(g)}} \tag{2.28}$$

$$\mu_{eff} = \left(\sum\nolimits_{i=1}^{\mu} w_i^2\right)^{-1} \tag{2.29}$$

$$\sigma^{(g+1)} = \sigma^{(g)} \times \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|p_\sigma\|}{E\|N(0,I)\|} - 1 \right) \right) \tag{2.30}$$

$c_\sigma \approx n/3$ is the backward time horizon for the evolution path p_σ

n is the number of problem dimension

$d_\sigma \approx 1$ is the damping parameter

$E\|N(0,I)\|$ is the Euclidean norm of a distributed random vector, $N(0,I)$

$$E\|N(0,I)\| = \sqrt{2}\Gamma((n+1)/2)/\Gamma(n/2)$$

$$E\|N(0,I)\| \approx \sqrt{n}\left[1 - 1/(4n) + 1/(21n^2)\right]$$

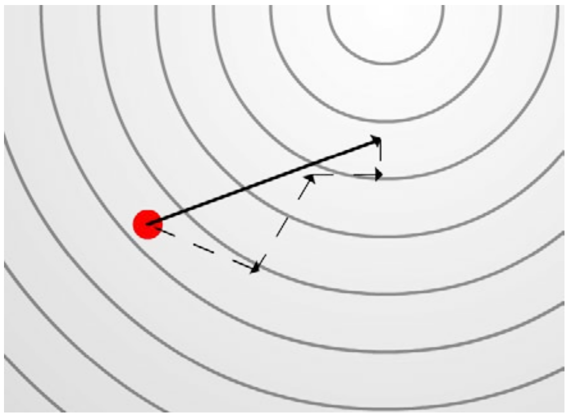


Figure 2.11 The step size improvement of the CMA-ES algorithm

Again, similar to GA, the next generation search point repeats all the process until the termination condition reached, such as the solution satisfies minimum fitness error or a fixed number of iteration reach. The solution is regarded as best fit if the fitness cannot show significant improvement in certain generations.

2.3.4 Artificial neural network

Artificial Neural Network (ANN) was first introduced by Hebb in 1949 [Hebb 1949]. It was concerned into neural networks by Minsky in 1954 [Minsky 1954]. ANN is a simplified emulation of the connections of the human brain, which can be used for learning purposes in an artificial environment.ANN is usually used on offline learning. It learns the relationship of input and output by past experiences throughout the network as shown in Figure 2.12. A hidden layer is used as an aggregator. The output, y_k, is equal to the weighted sum of input and hidden nodes, $\sum_{j=1,k=1}^{j,k} w_{jk} \sum_{n=1,j=1}^{n,j} w_{nj} x_n$. It is flexible and able to solve non-deterministic and non-linear evaluation. ANN is relatively unexplored technique in computer games and is becoming a hot tool for game AI research as they are complicated to understand and is very resource intensive. It is difficult to find a suitable variable. Those reasons create gaps for the AI research to fill up and can be used for many purposes such as learning, classification and pattern recognition.

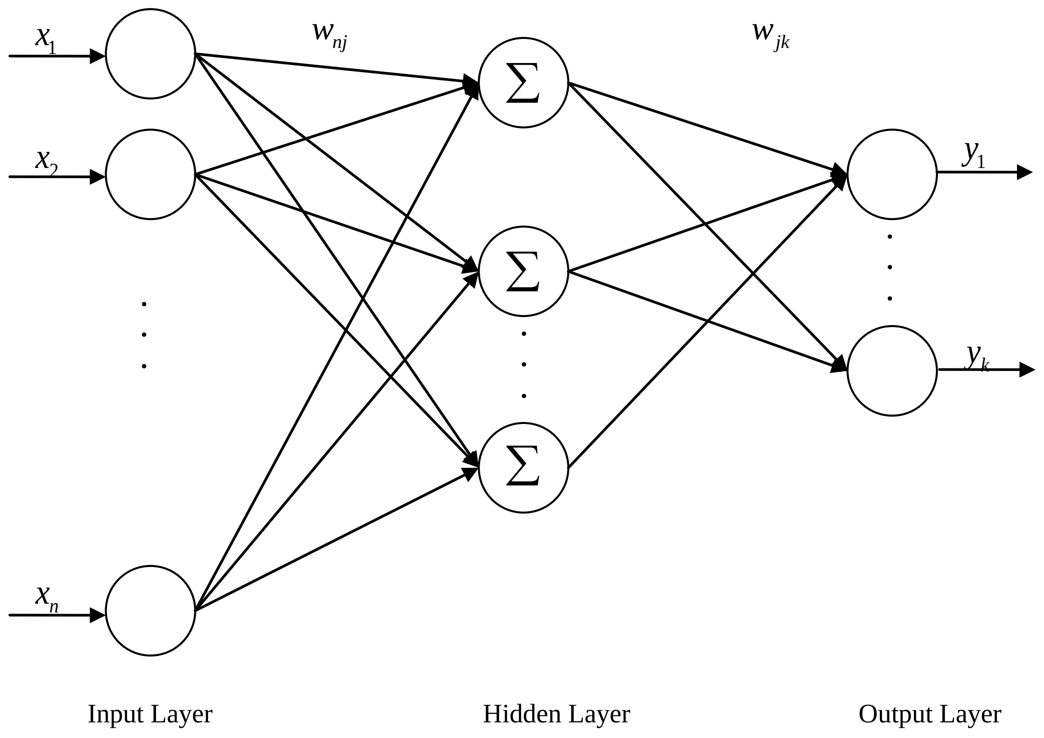


Figure 2.12 A simple artificial neural network

2.3.5 Hidden Markov model and dynamic Bayesian network

Hidden Markov Model (HMM) and Dynamic Bayesian Network (DBN) were first introduced by Baum [Baum 1966]. They are known as directed acyclic graphical model that involves conditional probability distribution (CPD). The simplest kind of DBN is a Hidden Markov Model (HMM), which has one discrete hidden node and one discrete or continuous observed node per slice. The model assumes that each state can be uniquely associated with an observable event as shown in Figure 2.13. CPD is assigned to each relationship.

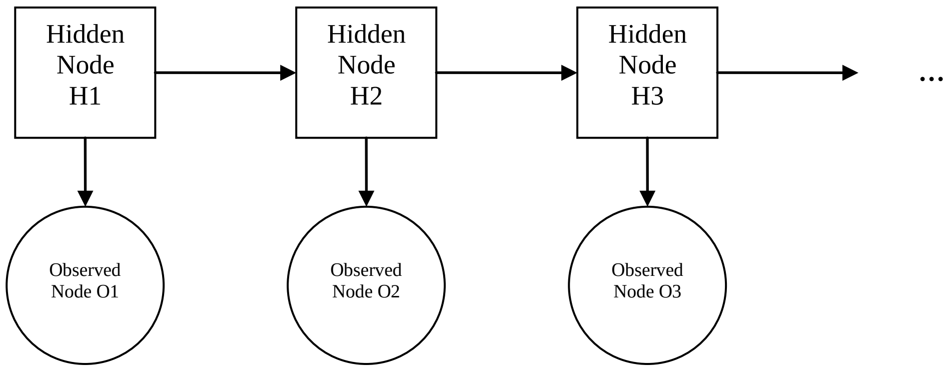


Figure 2.13 A simple Hidden Markov Model

Once an observation is made, the state of the system is then trivially retrieved. This model, however, is too restrictive to be of practically use for most realistic problems. To make the model more flexible, we assume that the outcomes or observations of the model are a probabilistic function of each state. Each state can produce a number of outputs according to a probability distribution, and each distinct output can potentially be generated at any state. These are known a Hidden Markov Models (HMM) asthe state sequence is not directly observable; it can only be approximated from the sequence of observations produced by the system. HMM consists of 5 elements. The first is a set of hidden node, $N = \{n_1, n_2, ..., n_N\}$. The second is a set of output symbols in observation node, $M = \{m_1, m_2, ..., m_M\}$. The third one is an initial state probabilities matrix, $\Pi = \{\Pi_i\}$ and $\Pi_i = P[q_i = n_i]$. We use q_t to represent a state of a hidden node at time t and O_t to represent a state of a observation node. The forth one is a state transition probabilities matrix, $A = \{a_{ij}\}$ and $a_{ij} = P[q_{t+1} = n_j \mid q_t = n_i]$ where q_t is the state of hidden node at time t. The last one is a symbol emission probabilities matrix, $B = \{b_j(k)\}$ and $b_j(k) = P[O_t = m_k \mid q_t = n_k]$. Forward algorithm is the most common algorithm to train up probabilities matrix. It starts at the first node, $t = 1$, and initializes the forward probabilities as the joint probability of state n_1 and initial observation O_1 , i.e., $a_1 = \Pi_1 b_1(O_1)$. Then, induction is performed to each node and calculate the probabilities, $a_{t+1}(j) = \left[\sum_{i=1}^{N} a_{t+1}(i) a_{ij}\right] b_j(O_{t+1})$. It terminates until all the probabilities are calculated, $\sum_{i=1}^{N} a_T(i)$. DBN is similar but consists of more than one hidden nodes and it is usually used to present sequences of variables with time series.

2.3.6 Case-based reasoning

Case based reasoning (CBR) was first introduced by Schank and Abelson [Schank 1977]. It solves the problem from previous human experience. It is useful in dynamic environment and powerful to solve the problem that does not have an actual solution. In brief, it encodes the situations into cases and applies reasoning technique for case retrieval. Then, it performs learning by case revise and retains technique as shown in Figure 2.14.

Before the knowledge can be used, they are converted into a case. This is called case representation.A case usually consists of three elements. First one is situation which contains different feature to describe the problem. The second one is solution which contains the process to solve the problem. The third one is the result which describes the state of situation after the case occurred. Cases may also consist of indices to speed up the process of case retrieval.

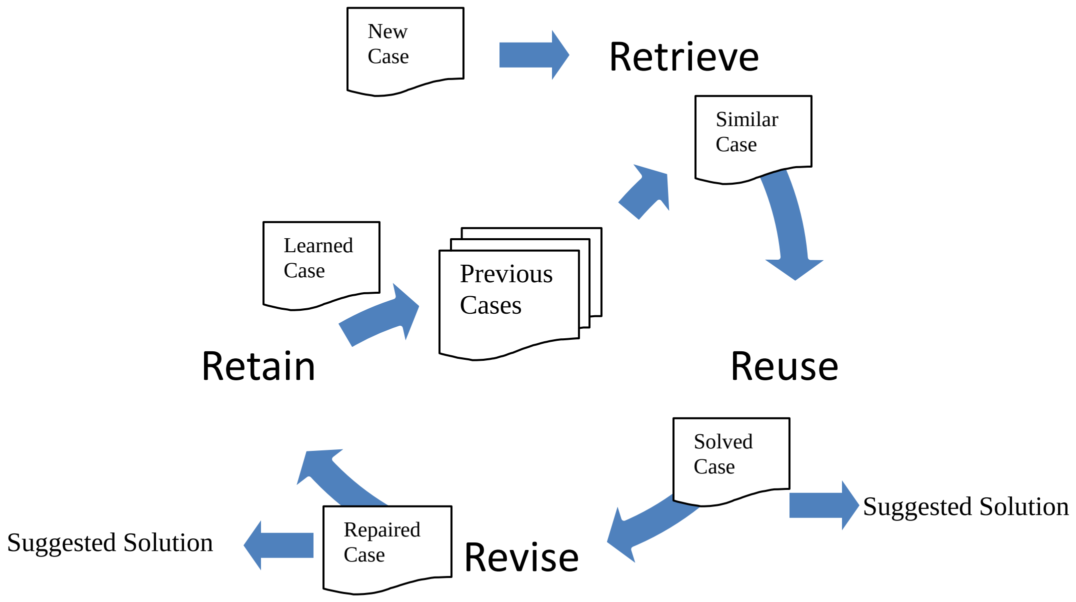


Figure 2.14 Case based reasoning cycle

There are four main processes in CBR. First one is case retrieval. When a new case is given, CBR should retrieve the most similar cases to the current situation. The similarity of each previous case and the new case is computed by Euclidean equation as shown in Equation (2.31). Nearest neighbor is found by comparing the weighted features in the situation.

$$\frac{\sum_{i=1}^{n} w_i \times sim(f_i^N, f_i^R)}{\sum_{i=1}^{n} w_i} \tag{2.31}$$

where sim is the Euclidean equation

f_i^N is the i-th of feature of new case

f_i^R is the i-th of feature of retrieve case

w_i is the weighting of feature

n is the number of feature

The solution of the nearest neighbor is reused in case reuse process or the suggested solution is revised in case revise process and become a new solution in the new world. After the solution has been adapted to the problem, the result is stored as a new case in the case based.

2.4 Other techniques used in this research

2.4.1 Potential field

Potential Field is a concept from robotics that was first introduced by Khatib [Khatib 1986]. The main idea is to use the algorithm to divide the battlefield into grids. Every point of the gird is given a fitness value that describes certain conditions. It navigates multiple units from low potential area to high potential area. For example, assume the robot, target and obstacles are all put in a zero potential field. The target generates the positive force while the obstacles generate the negative force. The force affects the area nearby and all the force is summed up at each point in potential field. Then, the robot is attracted by the positive potential and repulsed by the negative potential. Therefore, a path can be generated. The basic equation of attractive force, U_{att} , is shown in Equation (2.32). It affects the each point in the potential field. While the repulsive force, U_{rep} , is shown in Equation (2.33). It only affects the point at certain distance. The resultant force is calculated by combining U_{att} and U_{rep}. Figure 2.15 shows a sample potential field. Robot moves toward the highest potential and avoid passing through the repulsive force.

$$U_{att} = \frac{1}{2}\xi p_g^2(q) \tag{2.32}$$

where ξ is the gain coefficient

$p_g^2(q)$ is the Euclidean distance between the robot location and the target

$$U_{rep} = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{p(q)} - \frac{1}{p_0}\right)^2 & p(q) \le p_0 \\ 0 & p(q) > p_0 \end{cases} \tag{2.33}$$

where η is the gain coefficient

$p(q)$ is the minimum distance between the robot and obstacle affected area

p_0 is affected area of obstacle

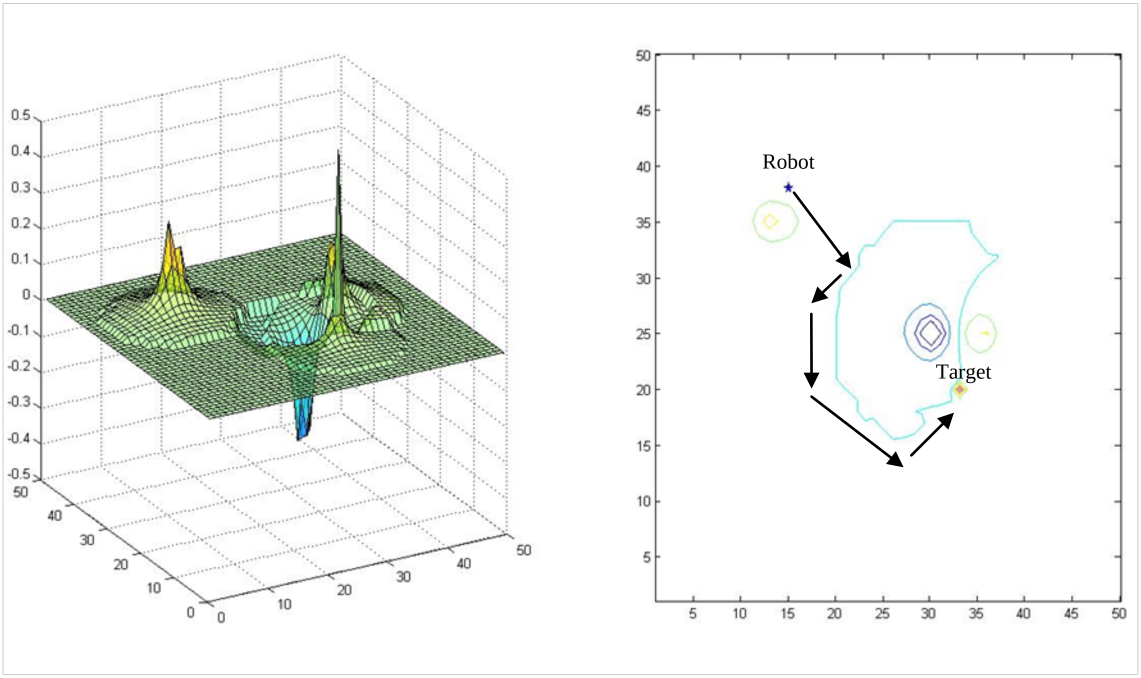


Figure 2.15 Motion of a robot in potential field

2.4.2 Mann-Whitney test

Mann–Whitney U test (also called the Mann–Whitney–Wilcoxon (MWW) or Wilcoxon rank-sum test) was first introduced by Gustav Deuchler in 1914 and extended by Wilcoxon [Wilcoxon 1945] for equal sample sizes. It is a non-parametric statistical hypothesis test to prove two sets of data, a and b , are independent observations to each other. First, the two data sets are combined into a set of $N = n_a + n_b$. All the data is ranked from the lowest to highest for one feature only. Then, the rank is summed up separately as R_a and R_b . The value of U is calculated by $U_a = R_a - \frac{n_a(n_a+1)}{2}$ and $U_b = R_b - \frac{n_b(n_b+1)}{2}$. Two-tailed test is then preformed. If the value of asymptotic 2-tailed is lower than 0.05, we can reject the null hypothesis. The two data sets have significant differences in these features.

2.4.3 Bivariate correlation test

Bivariate correlation is used to measure the strengths of association between two features, X and Y . In another words, there is interaction among the two features. In our research, Pearson correlation, p , is used as shown in Equation (2.34). Correlation is regards a \s significant at the 0.05 level (2-tailed).

$$p_{X,Y} = \mathrm{corr}(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{2.34}$$

where μ_X and μ_Y are the expected values

σ_X and σ_Y are the expected values

E is the expected value operator

cov is covariance

2.5 Summary

Comparing to traditional hard computing schemes, soft computing techniques, such as neural networks, SWARM intelligence, and decision trees induction could help AI to deal with the uncertainly and partial truth environment. However, most of them are mainly formulated on the minimization of Euclidean distance-based error functions. The parameters used in these functions are all normal additive in nature and the model is unable to describe the non-linear effects among parameters which is easily found in RTS game. Hence, there is a lack of methodology to predict the possibilities of win during the RTS game in an efficient way. Current path finding algorithm is unable to provide detaild planning on units maneuver.

Chapter 3

A fast indexing scheme for identifying game unit's best location

Development of real time strategy (RTS) game AI is a challenging and difficult task as real-time constraint and large searching space are required to find its best strategy. This chapter is a preliminary study of game AI in RTS game. We selected tower defense game as our research problem. It is a part of RTS game-plays which focuses on allocating the game unit for a best location and attack. We proposed a machine learning approach based on genetic algorithm (GA) and artificial neural network (ANN) for developing a neural-evolutionary model in tower defense game. This model provides efficient, fair and natural game AI to identify the game unit's best location. With the hybrid GA-ANN approach, the game AI can acquire the knowledge automatically by learning from the previous cases iteratively and memorize the successful experiences to handle future situations just like humans do instead of relying the predefined IF-THEN rules for action.

3.1 Introduction

The goal of tower defense game is to allocate the attack units, e.g., towers, to a suitable location. The player needs to build the tower and kill the enemy units when they pass by. Nowadays, tower defense game is not only found in RTS game. It has been produced as a standalone game in many platforms. However, the AI algorithms now used in computer games are mostly heuristic-based so that the current architecture of game applications cannot fully support in various contents. Thus, the probability of the algorithms is not very well. The game AI can perform well on a predefined map by following the heuristic rules specified by the game developer but it may fail to behave correctly on a new map due to the lack of the rules. To address this deficiency, we proposed a machine learning component for the tower defense game so that the game AI is able to learn from data directly without using the heuristics.

GA is used in this model. However, GA is time-consuming and is not suitable in RTS games. Thus, we proposed a fast indexing technique and encoded the results of GA as the input of ANN for case indexing. As a result, when a new battlefield is given, the solution can be quickly obtained by consulting trained ANN without going through the GA process again. It saves a lot of retrieval time.

3.2 Tower defense problem in RTS game

In the chapter, we simulated a battlefield with following situations for the tower defense game.

1. Two teams are created in a battlefield. One is the attack team, i.e., enemy. Another one is the defense team, i.e., player.
2. Enemy must move to the base of the defense team and attack.
3. Defense team is able to set up a number of cannons in the battlefield to kill the enemy when they are approaching their base.
4. Attack team chooses a path which can minimize the hurt from the towers
5. The goal is to allocate the towers and creates the maximum casualty to the enemy no matter which path they choose to approach the base.

3.3 Neural-evolutionary model

Our proposed neural-evolutionary model is given in Figure 3.1. First, the battlefield is encoded as a chromosome which is the input of GA. GA solves the cannon distribution problem by selection, crossover and mutation. After certain generations, the best off-springs, i.e., cannon distributions, is produced through the fitness evaluation and become the input of ANN for training. Afterwards, cannon location prediction in new terrain can be suggested quickly and directly by the trained ANN for new battlefields.

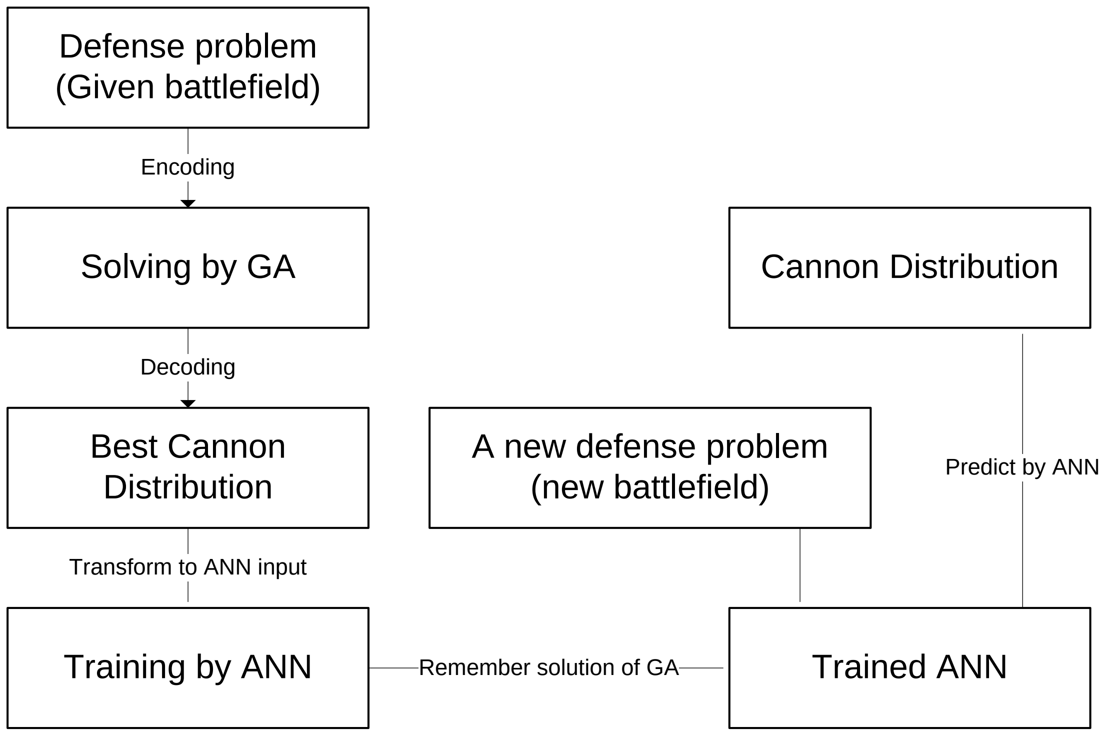


Figure 3.1 Neural-evolutionary model

3.3.1 Chromosome formation in GA

We encoded the solution of tower locations into chromosomes. The length required for the encoding depends on the area of the battlefield and the possible barriers. Here we use an example to demonstrate the encoding idea and is shown in Figure 3.2. A map with 5x5 units is used and five cannons are set in the battlefield. White circles represent the open area that cannon can be set, i.e., location. The black circles represent the barrier which is unable to be set up a cannon. The grey circles represent the positions which the cannons are set up in this solution, i.e., chromosome. The total bits of the encoding depend on the size of the open area, i.e., Encoding bit = Total area – Barrier area. In this example, it is 13 bits. Five cannons are set on the map randomly as shown in Figure 3.2. The chromosome is 0101010100010 in this example.

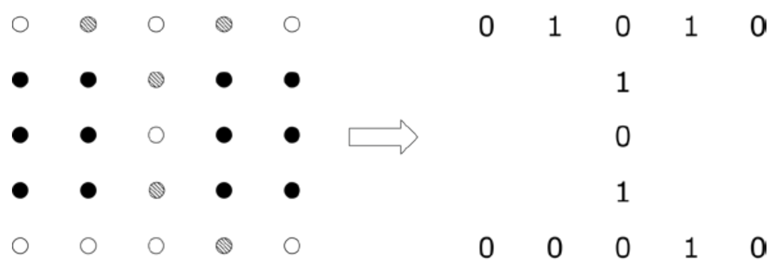


Figure 3.2 Demonstration of the encoding in GA

3.3.2 Fitness value determination in GA

Another crucial component is the fitness function which to evaluate the solution. The map is treated by Voronoi decomposition after all cannons are set. All possible paths are generated. The enemy finds the best path to travel and minimize the damage from the cannons and attack the player's base as shown in Figure 3.3.

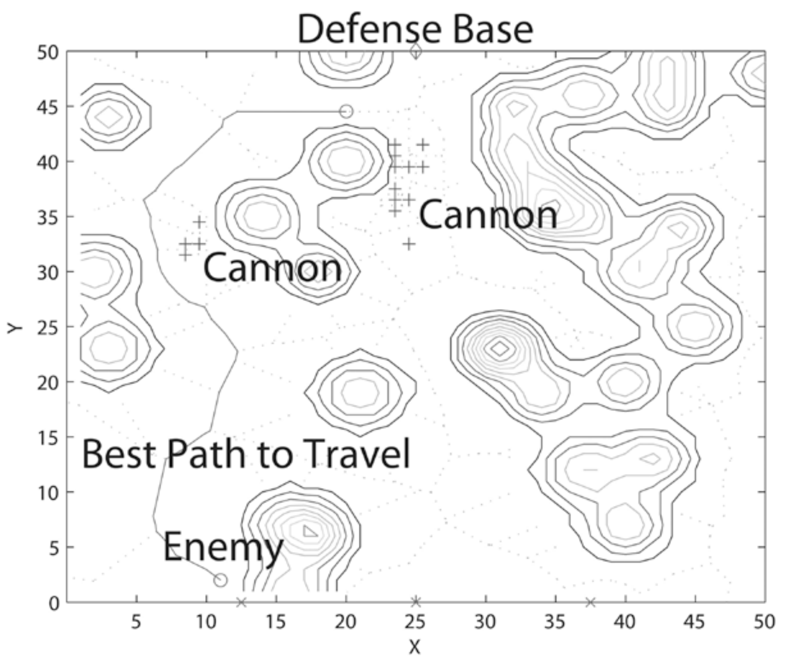


Figure 3.3 Cannon distribution

As shown in Figure 3.3. Defense base is set on the upper part of the map. Enemy moves towards the base from the lower part of the map. A limited number, N , of cannons are set. They try to give maximum damage to the enemy no matter which the travelling path it uses. Enemy is given a certain velocity (v) as well. It receives damage, i.e., fitness function within the attack range of each cannon, i, as shown in Figure 3.4. The degree of damage is calculated using equation (3.1).

$$\text{Fitness} = \sum_{i=1}^{N} \frac{d_i}{v} \times \text{Cannon Power(Hit point per second)} \tag{3.1}$$

where N is the number of cannon

d is the distance in the attack area

v is the velocity of enemy

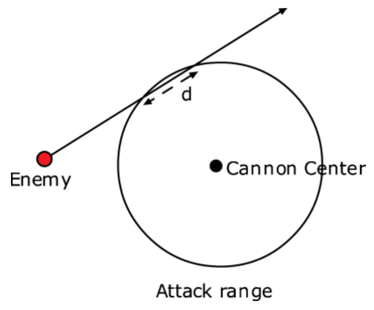


Figure 3.4 Damage on the enemy

The total damage is calculated by summing up all individual damages caused by different cannons. It becomes the fitness value of this simulation. The higher the damage did to the enemies, the better the fitness of the cannons’ positions. Cannon distribution is ranked by this fitness value and used to produce off-springs. Therefore, the fitness value is proportional to the damage on the enemies.

For crossover operator, each bit of off-springs is chosen from corresponding bit of parent chromosome randomly. If the total number of "1" is not equal to the cannon number, we randomly select a bit (If it is greater, then select "1", otherwise select "0") and replace with the opposite bit. The process repeats until the condition is fulfilled.

For mutation operator, we supposed that the mutation rate is α, 0.05. For each bit, we generated a random number R_i ($0 < R_i < 1$), the bit is converted into the opposite if R_i is greater than α. If the total number of "1" is not equal to cannon number, same method as shown above will be used again.

3.3.3 Case indexing by neural network

GA is time-consuming and cannot be implemented in RTS game. To overcome this problem, we suggest using artificial neural network (ANN) to speed up the whole process. Compare with the other traditional indexing methods, such as B+-tree, R-tree or Bayesian Model, Sankar and Simon [Sankar 1994] stated three advantages of using neural network for cases indexing. First, ANN improves the case searching and retrieval efficiency. Moreover, it is very robust in handling noisy case data with incomplete or missing information which is the situation of strategy game. Finally, it is suitable for accumulating data in RTS game.

We use the best solution (cannon distribution) from GA previously as the inputs to the ANN. Every point of the best case with a certain radius is treated as one of the inputs to the ANN. We use an example to illustrate our idea as shown in Figure 3.5. A map with 5x5 units is used again and five cannons are set in the battlefield. White circles represent the open area that cannon can be set. The black circles represent the barrier that is unable to set up the cannon. The grey circles represent the position which the cannons are set up in the best off-spring by GA. Every point of the open area is a training case of ANN. We use point A as an example. The eight points which surround point A is encoded. “-1” represents the barrier while “1” represents the open area. The final digit represents the distance between the point A and the base of the player. In this example, the encoding becomes [-1 1 -1 -1 -1 -1 1 -1 2]. This is the input of training case.

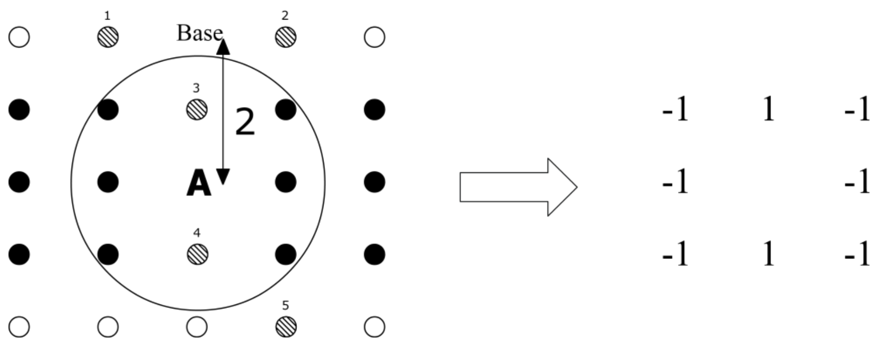


Figure 3.5 Encoding of Point A

$$f(\text{Point A}) = \sum_{i=1}^{n} e^{-\frac{l}{r}} \tag{3.2}$$

where $l = (A_1 - K_1^i)^2 + (A_2 - K_2^i)^2$ and r is a control parameter

(A_1, A_2) is the coordinate of point A

(K_1^i, K_2^i) is the coordinate of cannon i

Equation (3.2) shows the computation of each point A. The objective of the ANN learning is to approximate the best location. Therefore, the equation is based on the relationship of point A and the GA solution. (A_1, A_2) is the coordinate of point A while (K_1^i, K_2^i) is the coordinate of cannon i . l is the distance of point A and cannon i . n is the total number of cannon, i.e., $n = 5$ in this case. r is a parameter for controlling the spread of $f(\text{Point A})$. The higher value of $f(\text{Point A})$, the better location for setting up a cannon. In another word, if point A is nearest to all cannons of GA solution, the distance, l will be smaller and $f(\text{Point A})$ will be larger. Therefore, it is recommended to set up cannon in point A.

The ANN training is using back-propagation and log-sigmoid output function. The number of the hidden layer is calculated as the square root of the encoding string's length. In this example, it is $\sqrt{9}$ as shown in Figure 3.6.

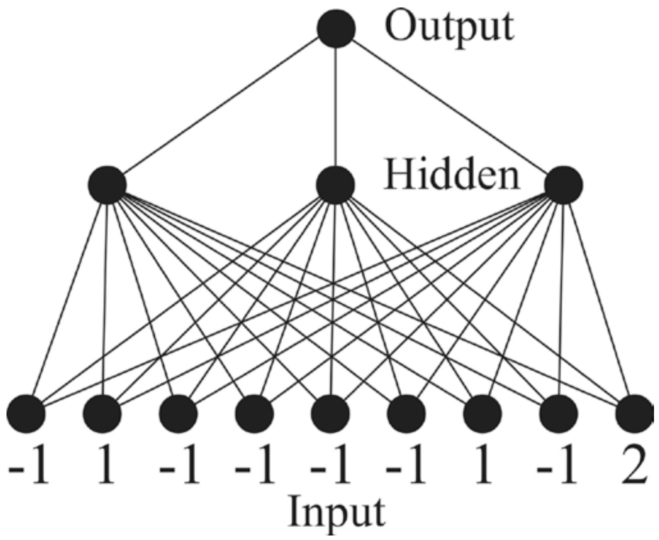


Figure 3.6 ANN structure

3.4 Experimental result and discussion

3.4.1 Experimental result of GA

This simulation is done by using an Intel Pentium IV 2.4 GHz machine with 1.5 GB Ram under Windows XP. MathWorks Matlab 7.0 is used as the simulation tool. Figure 3.7 shows the cannon distribution of different generations. In our simulation, GA does not have a significant change in enemy damages after 60 generations as shown in Figure. 3.8. The result is similar to [Chuen-Tsai 1994] and [Yi 2006]. Population is another concern. The process is very time-consuming if it is poorly designed. In our simulation, 50 populations are chosen because of its faster convergence. If the population size increases, the training time increases exponentially as well, as shown in table 3.2. However, it cannot show a significant improvement on fitness, as shown in Figure. 3.8. The weakness of GA is time-consuming. The run time of GA is around 1000 seconds for 100 generations which is unacceptable in RTS games and real world battlefields.

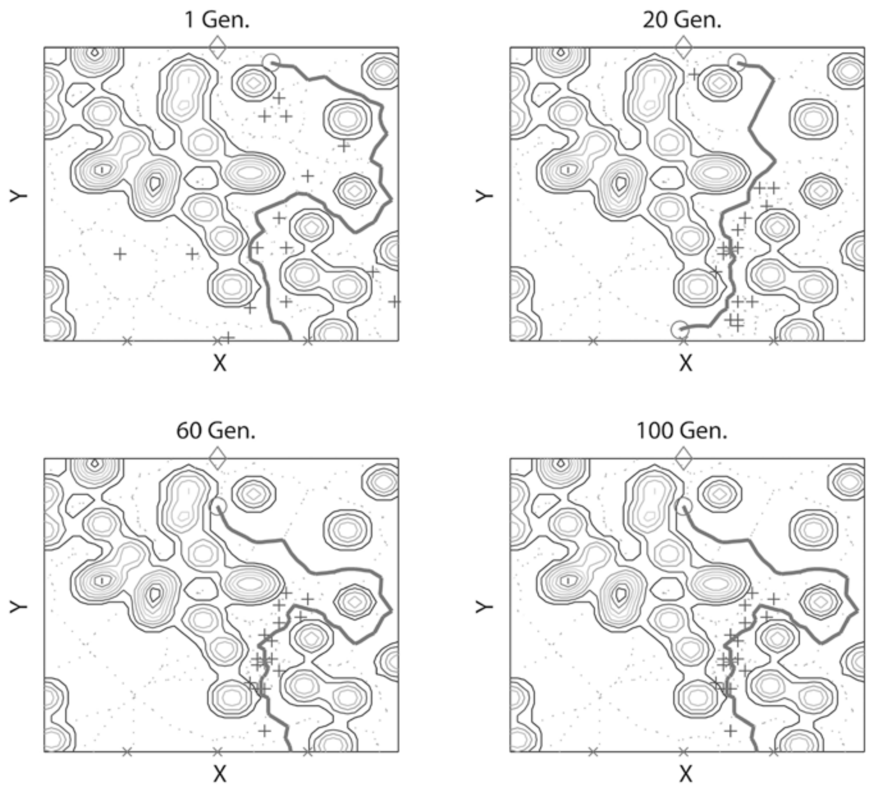


Figure 3.7 Cannon distribution for different generations

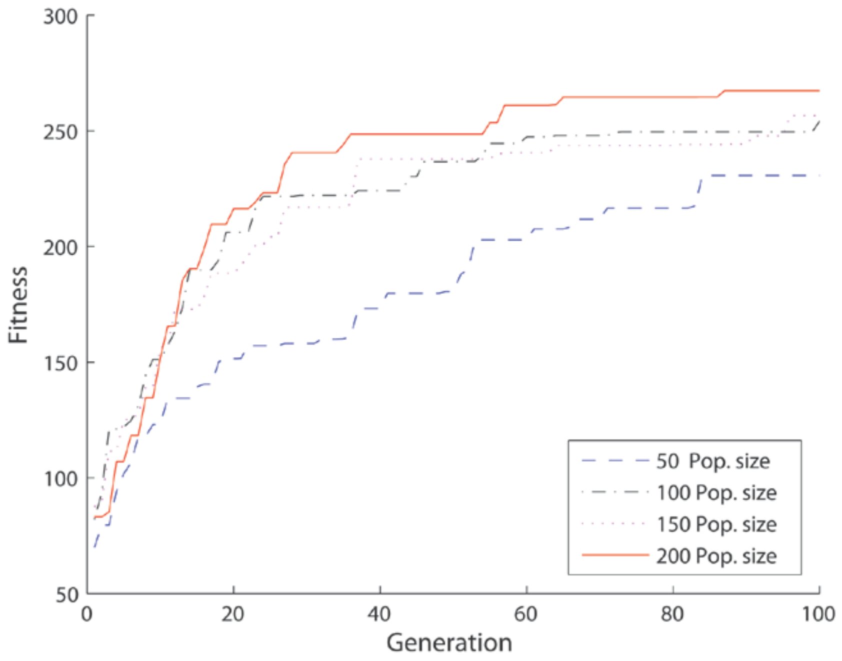


Figure 3.8 Fitness on different generations and populations

TABLE 3.1
GA'S TRAINING TIME WITH DIFFERENT GENERATION

Generations	Training time (second)
1	10
20	205
60	610
100	1012

TABLE 3.2
GA'S TRAINING TIME WITH DIFFERENT POPULATION

Population	Training time (second)
50	856
100	1750
150	2168
200	3443

3.4.2 Experimental result of ANN optimization

After ANN training, our machine learning component becomes very useful. Figure. 3.9 shows an example that is commonly found in RTS games. The training time is around 24 seconds for the ANN to remember the GA solution. When a new terrain is given, the time for distributing cannons in the new battlefield is 0.04 seconds only.

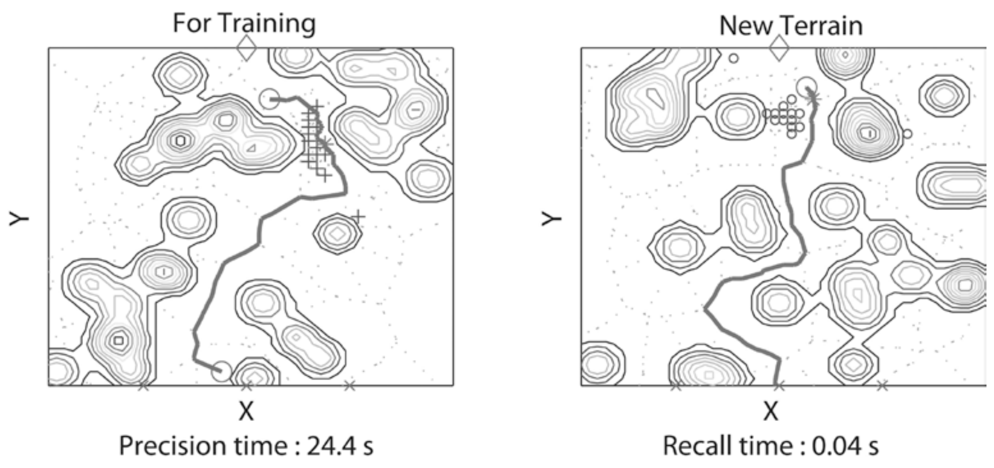


Figure 3.9 Training and testing result of BP ANN

After the neural-evolutionary model is trained, ten battlefields are generated for testing. The time and the damages of enemies are accorded for comparison and shown in the Table 3.14. The higher damages are meant to represent the better solution.

GA obtained a better solution. The average of damages was 150. The neural-evolutionary model provided similar results. The average of damages was 140. The difference was 6.65% ((140.4761-150.4954)/150.4954). The average time for GA to compute a solution was 2949s, while the neural-evolutionary model was only 0.036s. The neural-evolutionary model is fast and practical in RTS game.

TABLE 3.3
PERFORMANCE OF GA AND NEURAL-EVOLUTIONARY MODEL

	Genetic Algorithm		Neural-evolutionary Model	
	Damage	Time (s)	Damage	Time (s)
Map1	246.9430	1996	240.3382	0.0158
Map2	234.6570	2664	237.1426	0.0165
Map3	128.2711	2208	120.0705	0.0133
Map4	127.8631	2707	147.7751	0.0224
Map5	126.4970	2371	101.6419	0.0170
Map6	125.9645	3951	110.5593	0.0724
Map7	147.7750	3521	91.1507	0.0510
Map8	126.1231	3515	127.6389	0.0588
Map9	127.6389	3214	124.1221	0.0416
Map10	113.2131	3345	104.3221	0.0473
Average	150.4954	2949.2	140.4761	0.03561

Training and recall time of BP & RBF models

Back Propagation (BP) and Radial Basis Networks (RBF) models have been commonly used in Neural Network systems. In our experiment, BP results are shown in Figure 3.9. RBF results are shown in Figure 3.10. They show similar results in cannon distribution and fitness.

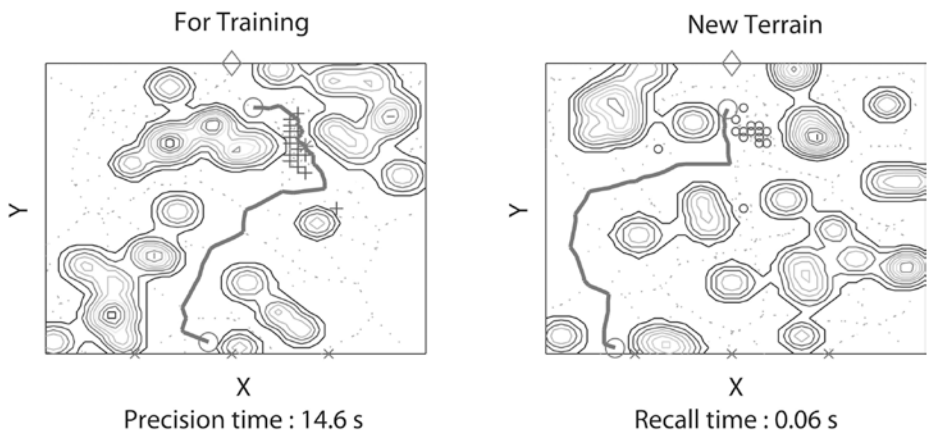


Figure 3.10 Training and testing result of RBF ANN

The main differences of BP and RBF are the training time and recall time. The drawback of BP is the long training time while RBF shows a 40% improvement. [Wang 2006] had performed similar comparison; they showed a 10% improvement on Text Classification. However, the recall time of RBF is 50% more than BP. Although the difference is only 0.02s, it becomes a heavy workload for the RTS game as it needs to complete each game cycle in every 0.02 to 0.03 second. As a result, we suggest using BP for RTS game because of its faster recall time. Figure 3.11 and table 3.3 also shows the training performance of BP using different number of neurons. The results are similar in different number of neurons.

TABLE 3.4
TRAINING PERFORMANCE ON DIFFERENT HIDDEN LAYERS FOR 253 X 1334 INPUT AND 1000 EPOCHS

Hidden Layer	Training time (second)
11	24.3
16	24.6
21	32.0

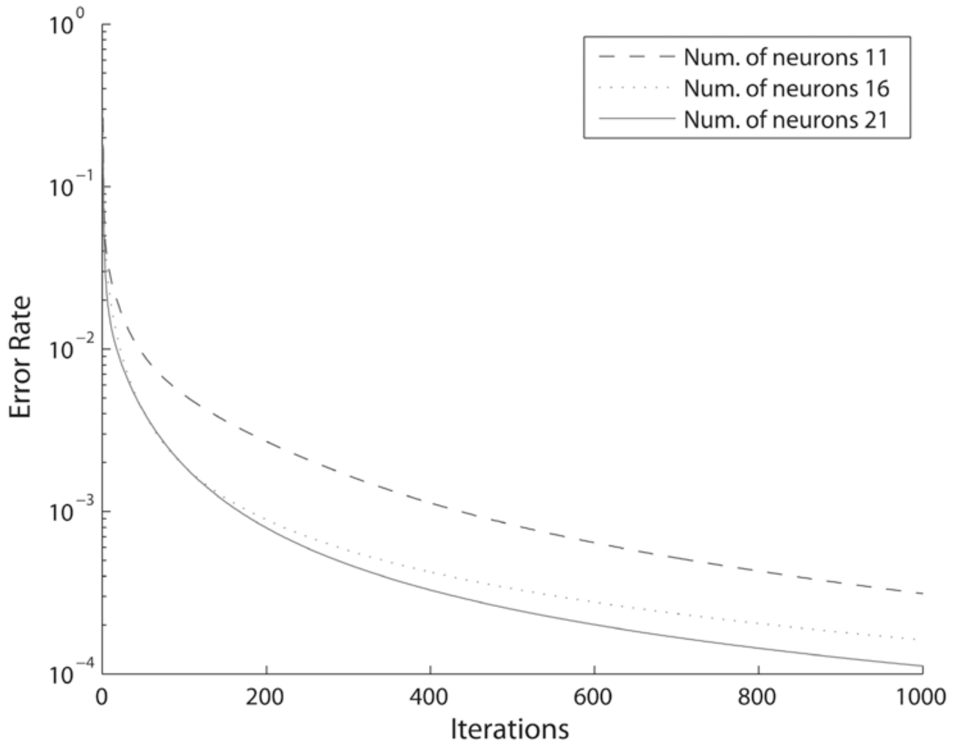


Figure 3.11 Training performance of BP

3.5 Summary

A neural-evolutionary model for case-based planning in real time strategy games is developed and shown in this chapter. We believe that this research direction can provide an efficient, fair and natural AI development for RTS games. Base defense is part of our evaluations in our current and future works. We will extend the idea and combine with other key components in RTS games, such as resource management and battle strategies, in our future research.

Chapter 4
Learning player behaviors from RTS game data

This chapter is another preliminary study of game AI in RTS game. It illustrates our idea of learning and building player behavioral models in real time strategy (RTS) games from replay data by adopting a Case-Based Reasoning (CBR) approach. The proposed method analyzes and cleans the data in RTS games and converts the learnt knowledge into a probabilistic model, i.e., a Dynamic Bayesian Network (DBN), for representation and prediction of player behaviors. Each DBN is constructed as a case to represent a prototypical player's behavior in the game. The use of these cases is to predict the behavior by applying a junction tree mechanism. Sixty sets replay data of a prototypical player are used to test our idea. Fifty cases are used for learning and another ten cases are for testing. Experimental result is shown to prove our work.

4.1 Introduction

Nowadays, multiplayer online games and virtual community are popular as players enjoy the game with other real players in a virtual world. However, it is very difficult for game companies to maintain a huge number of players, with varied styles, online at the same time. Therefore, to improve attractiveness, many avatars and characters in the game or virtual community need to be controlled by AI techniques. However, developing different behavioral styles from scratches to real players' simulation are difficult and time-consuming. To help accomplish this goal, we develop a method to learn them from real data. We used the Blizzard Warcraft III Expansion: The Frozen Throne (WIII: TFT) which is a well known and best selling RTS game in recent years to test our idea. Experimental result is shown and discussed in this chapter.

4.2 Knowledge discovery problem in RTS game

RTS game consists of complex game rules and hence, data of player input are noisy. One of the possible solutions to extract the knowledge is using Case-based reasoning (CBR). CBR has been studied for many years and its applications in computer games are becoming more popular. For example, Hsieh [Hsieh 2008] used professional game players' data of up to 300 replays to train a case-based decision system in one single battlefield. Ontanon [Ontanon 2007] introduced similar case-based reasoning framework for RTS game. Aha [Aha 2005] and Ponsen [Ponsen 2004] focused on strategies of building sequences in their case-based planning model. In general, the predicted accuracy in CBR systems greatly depends on the number of learnt cases and their qualities, i.e., the more the cases with higher qualities, the better the accuracy. However, as response time is critical in RTS games, there is always a tradeoff between accuracy and efficiency.

Bayesian network (BN) can be viewed as a mathematical model that describes the relationships between antecedents and consequences using conditional probabilities. It has been used quite extensively for representing the probabilistic relationships between diseases and symptoms. Dynamic Bayesian network (DBN) is one form of BNs that represents sequences of variables and is usually time-invariant. Some related works of using BNs for user modeling are by Ranganathan [Ranganathan 2003] and Montaner [Montaner 2003]. They implemented BN into their SMART Agents and proved its possibility to analyze user behaviors. Kuenzer [Kuenzer 2001] and Schiaffino [Schiaffino 2000] also did similar user behavior modeling on web applications using BN. Furthermore, Gillies [Gillies 2009] combined BN with finite-state machine to improve the use of motion capture data. Other uses of BN on games include Albrecht's [Albrecht 1998] BN structure to adventure games and Yeung's [Yeung 2006] BN technique to detect cheats in first person shooting games. In this research, we use DBN and junction tree algorithm as a case and similarity calculation respectively for predicting user behaviors. We believe that this is a promising direction for game developers and publishers that require varied styles of avatar behaviors.

4.3 Player behavior model

We proposed a player behavior model as shown as Figure 4.1. First, sets of replay data are collected from the internet. Information inside these replays are filtered, cleaned and summarized. Player actions and useful battle information are gathered from the replays, while the repeated data and useless information are discarded. The information is then used to build a DBN structure. Prediction of user behaviors is carried out afterwards. We use an example of a typical player called “Player A” to illustrate our approach.

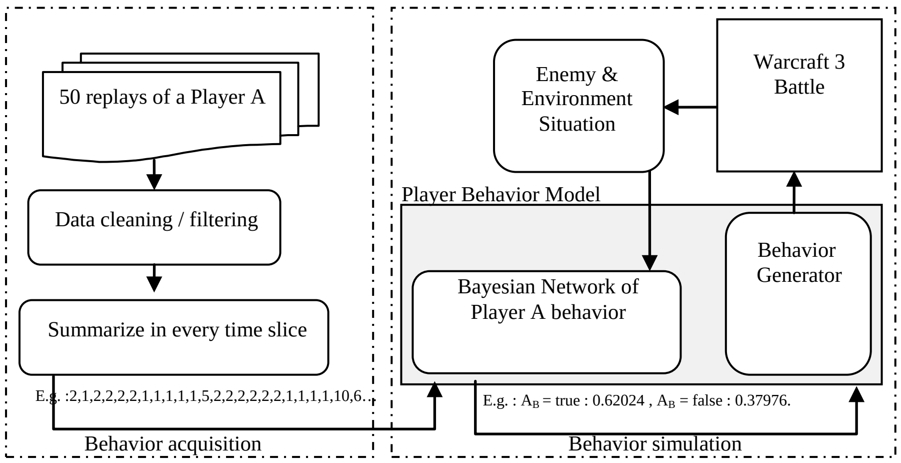

50 replays of a Player A
Data cleaning / filtering
Summarize in every time slice
E.g. :2,1,2,2,2,2,1,1,1,1,1,5,2,2,2,2,2,2,1,1,1,1,10,6…
Behavior acquisition
Enemy & Environment Situation
Warcraft 3 Battle
Player Behavior Model
Behavior Generator
Bayesian Network of Player A behavior
E.g. : A_B = true : 0.62024 , A_B = false : 0.37976.
Behavior simulation

Figure 4.1 Work flow of player behavior simulation model

4.3.1 Behavior acquisition in replay data

Thousands of Warcraft III replays can be collected on the Internet. All the player’s actions are recorded in the replay data. In this chapter, Solo Ladder (1 versus 1) battle in Battlen.net is chosen. It is the official game site and provides a fair environment for the players to fight against each other. Fifty replays of a player called “Player A” (name is removed) are collected. They describe the behavior of Player A against different opponents in different maps. For the purpose of reusing the player behavior model in different maps and games, the data is analyzed with meanings as described in Table 4.1.

TABLE 4.1
SELECTED DATA FOR DBN STRUCTURE

PLAYER ACTION

Set Name	Description
Attack	All kinds of attack commands with target data. Data of player A (A) and opponent (A') is both collected. For example: attack unit (A_U, A'_U), attack base (A_B, A'_B), etc. Each element of $A \in [True, False]$.
Create Building	All kinds of create building commands and their numbers in the same time slice. Data of player A (B) and opponent (B') is both collected. For example: build base (B_B, B'_B), build research centre (B_R, B'_R), etc. Each element of $B \in [0,1,2...\text{Upper Limit}]$ where upper limit is the maximum number of buildings that are created in the time slice.
Create Unit	All kinds of create unit command and their numbers in the same time slice. Data of player A (U) and opponent (U') is both collected. For example: create piercing unit (U_P, U'_P), create siege unit (U_S, U'_S), etc. Each element of $U \in [0,1,2...\text{Upper Limit}]$ where upper limit is the maximum number of units that are created in the time slice.

DEMOGRAPHIC INFORMATION

Set Name	Description
Race	There are 4 races that are provided by Warcraft III, where $R \in [1,2,3,4]$. Data of player A (R) and Opponent (R') is both collected
Unit	Current numbers of the alive units in time slice Δt. Rounding up to the nearest 10, where $N \in [0,1,2...\text{Upper Limit}]$, with upper limit equals to the maximum number of units that are created in the replay. Data of player A (N) and Opponent (N') is both collected
Map (M)	There are 13 official battle fields that are provided by Battle.net, where $M \in [0,1,2...13]$

As an observation, professional players in Warcraft III usually focus on a few types of units in the battles. They seldom create many different kinds of unit because they want to save the resources for upgrading the power. Therefore, to reduce the complexity of DBN, all unused commands of player A are filtered, i.e., if player A does not create any siege unit in 50 replays, the field of siege unit (U_s) is neglected and will not become a component of the DBN. Unit Number (U) is suggested to estimate the player situation. Our model focuses on the relationship of different commands with respect to the game play.

Then, actions of Player A are summarized in every fixed time slice (Δt). In this study, Δt is set as 15 seconds which is the minimum time to create a unit in Warcraft III. Selected fields for each Δt becomes the components in DBN are represented as a set of numeric data (a sample: 2,1,2,2,2,2,1,1,1,1,1,5,2,2,2,2,2,2,1,1,1,1,10,6…). The average time of Player A' replay is 20 minutes. Therefore, there are around 80 instances in each replay. Upper limit for field (B), (U) and (N) is also be set according to the maximum number of productions in Δt. It can reduce the parameters of the tabular nodes in the DBN.

4.3.2 Dynamic Bayesian network structure and parameters learning

A DBN consists of a structure and a number of parameter. The analyzed field in the previous process becomes the tabular nodes of the DBN structure. Intra-slice topology (within a slice) and inter-slice topology (between two slices) are defined according to the game play of Warcraft III. For example, if the player wants to create certain units, he needs to build certain buildings first. The relationship between the nodes is shown in Figure 4.2.

Parameters of DBN are represented as conditional probability distribution (CPD). It defines as the probability distribution of a node given by its parents, i.e., $P(A_{t+1}, U_{t+1}, \ldots) = P(A_{t+1} \mid \text{parent}(A_{t+1}))P(U_{t+1} \mid \text{parent}(U_{t+1}))\ldots$ All instances from 50 replays are used to perform parameters learning. As the data from the replays are fully observed and the structure is known, maximum likelihood estimation algorithm is used to compute a full DBN. A DBN that contains multidimensional CPD in each node is considered as a “case” to represent player A’s behavior.

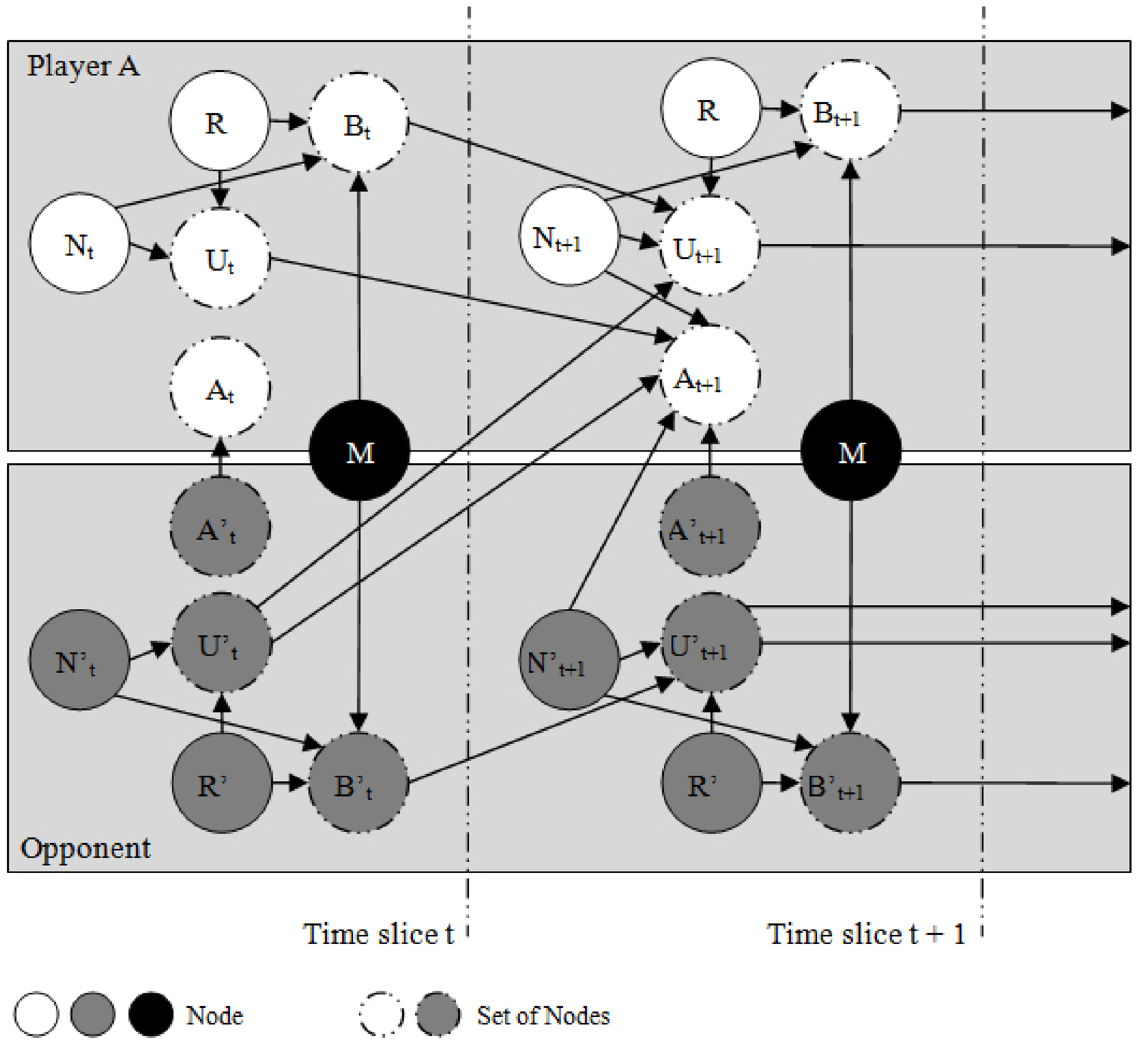


Figure 4.2 Structure of a DBN

4.3.3 Prediction in dynamic Bayesian network

Having created the DBN of Player A, it can be used for prediction. In this research, junction tree algorithm is used. Its purpose is to find out the probability of attack ($P(A_{t+1})$), create building ($P(B_{t+1})$) and create unit command ($P(U_{t+1})$) of Player A based on his previous behavior. The calculation of probabilities is based on all the previous time slices of their parents, i.e., $P(A_{t+1} | \text{parent}(A_{t+1}))$, $P(B_{t+1} | \text{parent}(B_{t+1}))$ and $P(U_{t+1} | \text{parent}(U_{t+1}))$. In every Δt, enemy and environment situation information ($M, R, R', N, N', A', B', U'$) are summarized and sent to the DBN as a fact to compute the marginal distribution for each node (A, B, U). Marginal distribution contains the probabilities of all parameters in each node. For example, the attack base command (A_B) only contains 2 parameters ($A \in [True, False]$). Therefore, the marginal distribution of A_B are represented as $A_B = true = 0.62024$ land $A_B = false = 0.37976$. The parameters with the highest probability is chosen and passed to the behavior generator, e.g., building base.

4.4 Experimental result and discussion

To calculate the prediction accuracy, ten new cases of Player A are prepared for testing. The simulation was run by using Matlab version 2008b with the BN toolbox that was written by Kevin Murphy. The machine used was a Core 2 Duo 2.13GHz with 4 GB Ram PC. In this simulation, 18 nodes (8 nodes in create buildings commands (B), 5 in create units (B) and 5 in attack actions (A)) of Player A was required to be predicted in every Δt. The running times for the learning and the average prediction for each Δt are shown in Table 4.2. The prediction time is stable as the time depends on the complexity of the BN structure (Dimensions of the CPD) and is independent of the number of learning instances. The constant performance of this prediction time fits the game implementation requirement.

TABLE 4.2
TIME OF LEARNING AND PREDICTION IN DBN

Number of cases	10	20	30	40	50
Learning Time (S)	245.15	516.35	649.90	770.38	968.72
Prediction Time (S)	0.1143	0.1373	0.1025	0.1033	0.1038

The highest prediction probability of each node is taken as the predicted command which was then compared with the actual command of Player A. The accuracy is calculated in every Δt using equation (4.1) which is similar to [Albrecht 1998].

$$\frac{1}{n}\frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{N}\begin{cases}1 & \text{Predicted command = actual command of Player A}\\ 0 & \text{Otherwise}\end{cases} \tag{4.1}$$

where n is number of testing replays, i.e., $n=10$ and N is the number of nodes that is required to be predicated in every Δt, i.e., $N=18$.

The accuracy against time slices with different numbers of learning instances are plotted in Figure. 4.3. We observed that if the number of learning instances are insufficient, the probabilities of computed marginal distribution would be closed to 1/(number of choices in the node) , e.g., $A_B = true = 0.5391$ and $A_B = false = 0.4609$ where $A \in [True, False]$). As a result, the accuracy decreases. In this case, we suggested executing the predicted command if the probability reaches a certain threshold level in Warcraft III. For example: 20 % increment, i.e., $P(A_B) > 0.6$ or 40 % increment, i.e., $P(A_B) > 0.6$. DBN could be improved accordingly if the number of learning instances increases. As shown in Fig 4.3c, the curves are similar in shapes and getting and closer to each other.

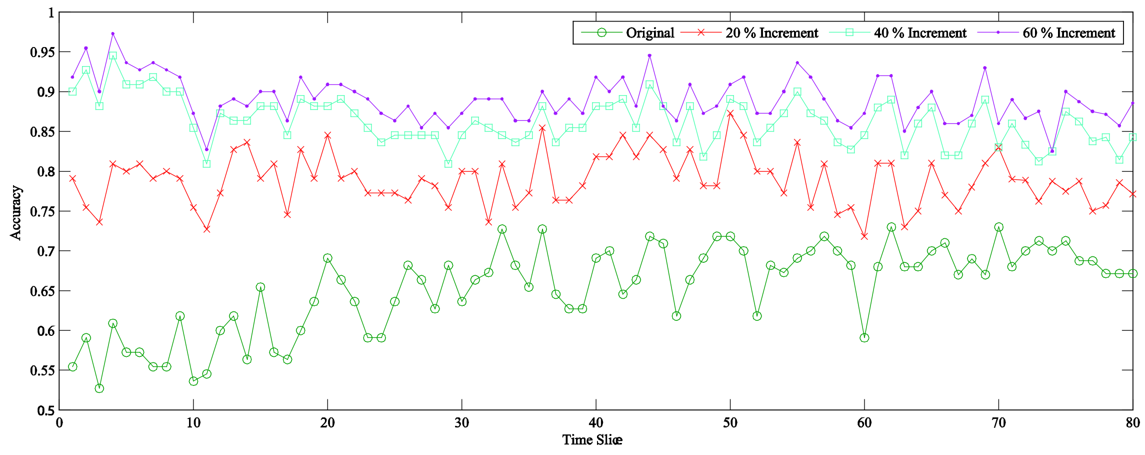


Figure 4.3(a) Accuracy of 10 learning instances

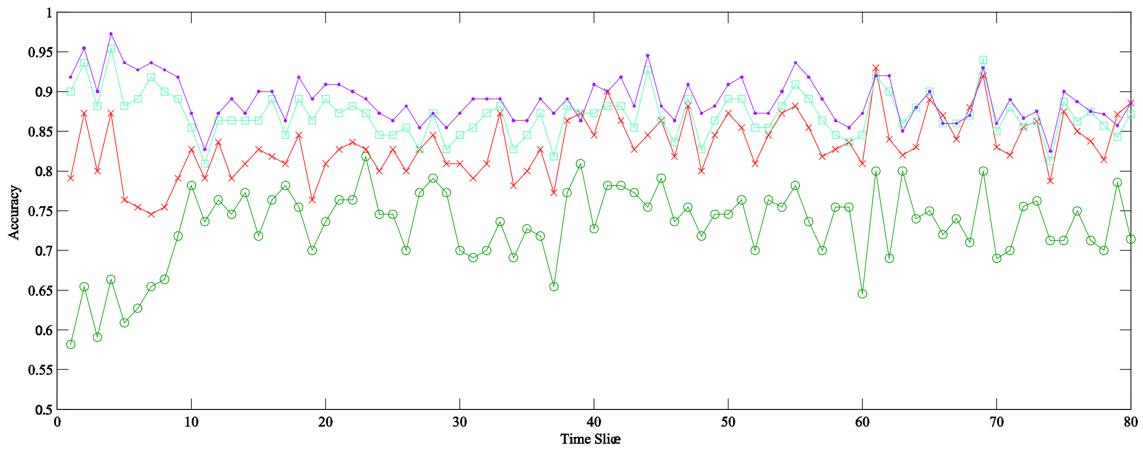


Figure 4.3(b) Accuracy of 30 learning instances

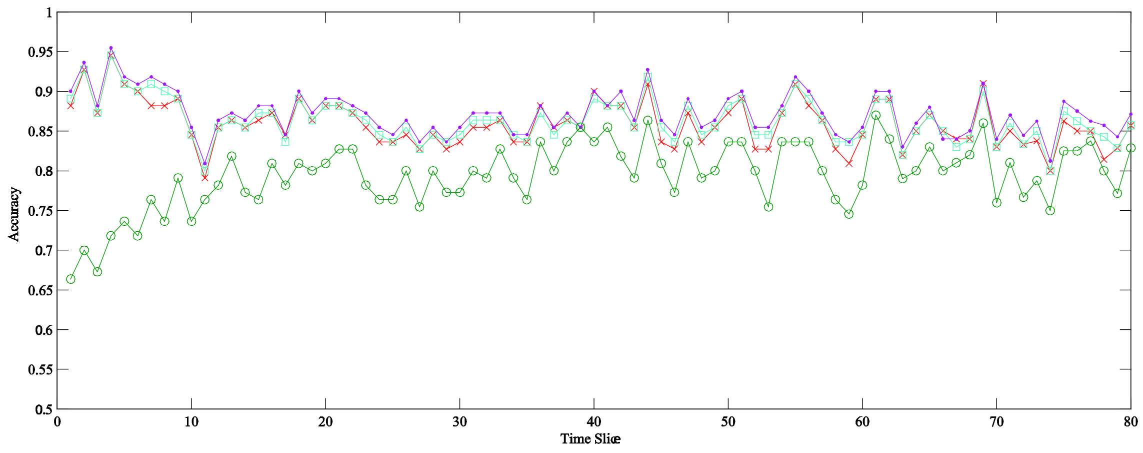


Figure 4.3(c) Accuracy of 50 learning instances

The learnt DBN represented one player (Player A) behavior. The predictability decreased for other players.

We have collected the replays from two professional players (Player B, WE.Pepsi.TED and Player C, YzU.weishawang) to test the learnt DBN. Three players used the same race but their behaviors are different, such as the building sequence, strategy and movement. In Figure 4.4(a), the lower blue line shows accuracy of the learnt DBN for Player B while the upper red line shows accuracy for Player A. The predictability of Player B was much lower than Player A. It is below 50% most of the time. The situation of Player C was similar as shown in Figure 4.4(b).

Computing a DBN for one player is time consuming and cannot be easily applied to other players. This is the driven force to develop an efficient model to evaluate the strategy combat as stated in Chapter 5.

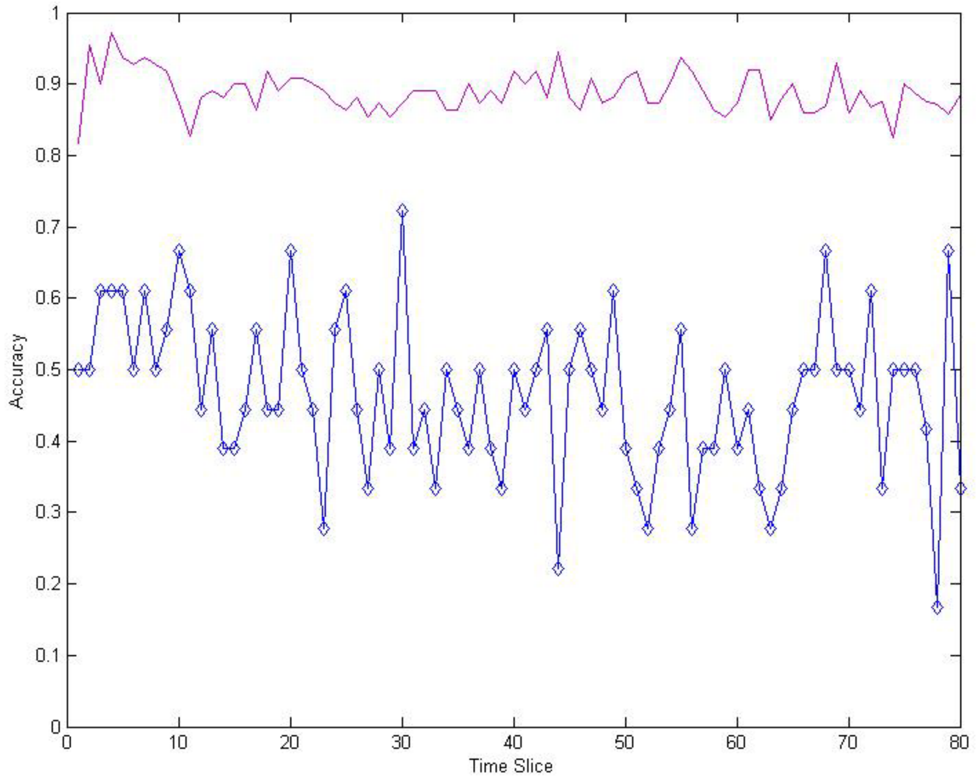


Figure 4.4(a) Accuracy of the learnt DBN for Player B

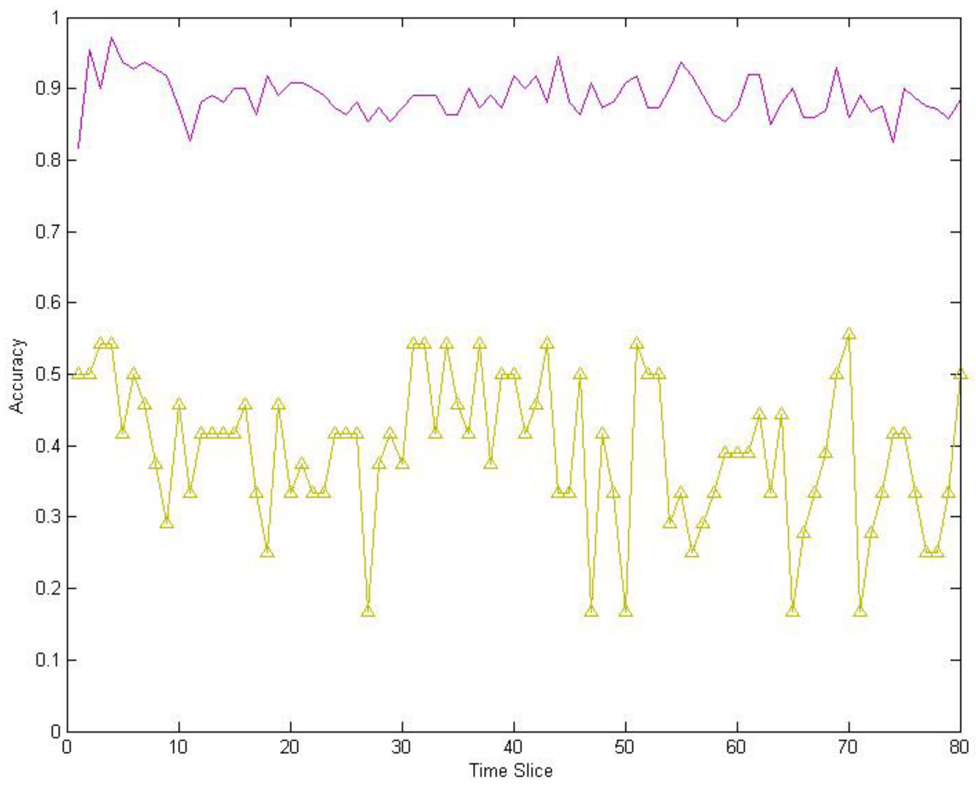


Figure 4.4(b) Accuracy of the learnt DBN for Player C

4.5 Summary

In this paper, we presented a Case-based reasoning framework to learn players' behavior in RTS games. The advantages of applying DBN and junction tree technique to RTS games are shown. Many players in multiplayer online games or virtual communities are not looking for challenging AIs but varied AIs. Our approach shows a possibility to develop varied AIs in games as well as the virtual world which is efficient and useful. Future work of our research will focus on a larger scale of learning in more different types of replays and to construct a useful case library for simulating various players of different styles. We plan to combine DBN with other soft computing techniques, such as artificial neural networks and genetic algorithms for faster learning, indexing and similarity calculations in player models.

Chapter 5

An order-based fuzzy integral to model feature interactions in RTS games

A contribution is made in this chapter by considering how the order of production and feature interaction of game units affect the result of playing RTS games. Explicit description using analytical models, such as finite state machines, Bayesian networks and decision trees may not be feasible due to complicated game rules, extensive terrains and numerous playable items. We present a machine learning model that extracts and evaluates game unit combination strategy from data in the past. This model takes the sequence into account which game units are produced and the interaction among them. We combine fuzzy measure and fuzzy integral with two different evolutionary algorithms to develop the learning model. The first one is genetic algorithm (GA) and the second one is covariance matrix adaptation evolution strategy (CMA-ES). Warcraft III battle data from real players are used in our experiments. Compared with the traditional Choquet Integral, our new order-based integral gives a smaller training and testing error in RTS game strategy selection.

5.1 Introduction

The fundamental game play of a typical real time strategy (RTS) game is collecting and allocating resources to build an army to destroy enemy units. From this perspective, the game play can be divided into two types: (1) macro control which consists of the development of resource gathering plans, base building decisions and technology upgrade paths, and (2) micro control which involves directing unit movements, path selection and the combats encounter. The success of these actions is largely determined by an appropriate selection of a suitable mix of game unit types. Many players may consider using a balanced army with many unit types. However, professional players seldom use this strategy in real game competition because such approach is unable to gain massive destroy power in a short period of time. Therefore, they develop “unbalanced” army units, which possess excellent killing power to certain kinds of enemy units but these units might be easily killed by another special kind of enemy units. Hence, it is very difficult to rate the combined action of different unit types as they may have completely different skills, e.g., healer units Thus, knowing the effect of unit combination becomes one of the major learning exercises and challenges to real game players in macro control. There are two more considerations when deciding what units to build. First, game units can only be produced following some specific orders of production sequence, i.e., some cheaper units must be produced first before the advance units can be unlocked. Second, the combined power of units cannot be simply computed using weighted average. This gains difficulty on how to model and understand the non-additive properties of unit combinations. In this research, we present a machine learning model that extracts and evaluates game unit combination strategy from past data.

This chapter is organized in seven main sections. This section is introduction. The next section is the problem statement. Section three is bottom-up strategy planning model. Section four explains how to evaluate the non-linear property in unit combination. Section five explains the proposed fuzzy integral, section six explain our experimental design and the conclusion is presented in section seven.

5.2 Macro control problem in RTS game

Macro control (macro management) can be regarded as game strategy development and selection. It consists of resource gathering plans; base building decisions and technology upgrade paths. It focuses on economic development and the future of the game, while micro control is focus on game unit control and the present of the game.

Aha [Aha 2005] used case-based reasoning to construct cases for strategy prediction. He used the stage of base development to cluster relevant cases and developed a network of cases in the case library for future use. Hsueh [Hsueh-Min 2009] constructed a Belief measure network to control nonplayer characters (NPC). Preuss's [Preuss 2010] combat strength of team composition and Keaveney's [Keaveney 2011] spread coordination measure are all important metrics. All of the above researchers share a same view on strategy selection. Comparing the power of unit combination is one of the main criterions. However, it remains a challenge as the combined power of units consists of non-additive properties. Hence, the intransitive superiority (A beats B, B beats C and C beats A [Watson 2001]) always occurs in RTS game play. It is difficult to evaluate a unit combination which consists of different unit types.

Our approach is applying fuzzy measure and integral to describe the interaction in RTS game. Although there have been many reports about the application of non-linear integrals in machine learning algorithm design, for example, constructing classifiers by non-linear integral projections in [Xu 2003] and Choquet integral with fuzzy-valued integrand in [Yang 2007]. However, there are very few reports in RTS game. Some recent advances can be found in [Avery 2010] where authors successfully demonstrated the use of coevolving influence maps to generate coordinating team tactics for a RTS game and in [Preuss 2010], authors specified that team composition for battling spatially distributed opponent groups can be supported by a learning self-organizing map (SOM) that relies on an evolutionary algorithm (EA) to adapt it to the game. In addition, authors in [Keaveney 2011] showed that evolutionary computation techniques (genetic programming in this case) can be used to evolve coordination in RTS games.

Our new idea which is differentiating from existing ones is that in a RTS game the cooperation and interaction within a team are measured by a non-linear integral which defines the maximum potential fighting-power of the team and helps / guides to evolve the team coordination.

It is really important to identify the opponents' strategies as fast and accurate as possible in RTS games so that an effective response can be scheduled. Regarding to the opponent's strategy identification, there are very few reports [Kabanza 2010, Genter2011] in the literature. In [Kabanza 2010], the authors conducted a preliminary behavior recognition based on the probability of behavior and influence map. In [Genter2011], inductive learning was used to perform the recognition of opponents' strategies which were represented with the extracted learning rules. In addition, authors in [Wang 2010] studied a reinforcement learning NPC team for playing domination games in which a Q-learning-style algorithm was used to learn the optimal decision-making policy. Our proposed idea here is to measure the opponent power by the previously defined non-linear integral and to learn the opponent’s inner interaction (positive or negative) by fuzzy measure, fuzzy integral and genetic algorithm [Wang 2001, Wang 2009, Wang 2011]. The final goal is to show that inductive learning with evolutionary computing techniques can lead to robust, flexible, challenging opponents that learn from human game-play.

5.3 Bottom-up strategy planning model

Traditional top-down strategy planning divides the battle into different time slices or stages [Aha 2005]. It compared the situation and select a suitable action for the next stage and so on. This counter strategy is easily to be developed but it cannot provide a large-scale strategic maneuvering. The idea of strategy, such as rushing strategy or defense strategy cannot be shown easily. We proposed bottom-up strategy planning model. First, a unit combination representing the idea of chosen strategy is selected. Then, all the corresponding actions are generated to achieve this unit combination. This planning model could perform a various behaviors in RTS game. The flow of the bottom-up strategy planning is shown in Figure 5.1.

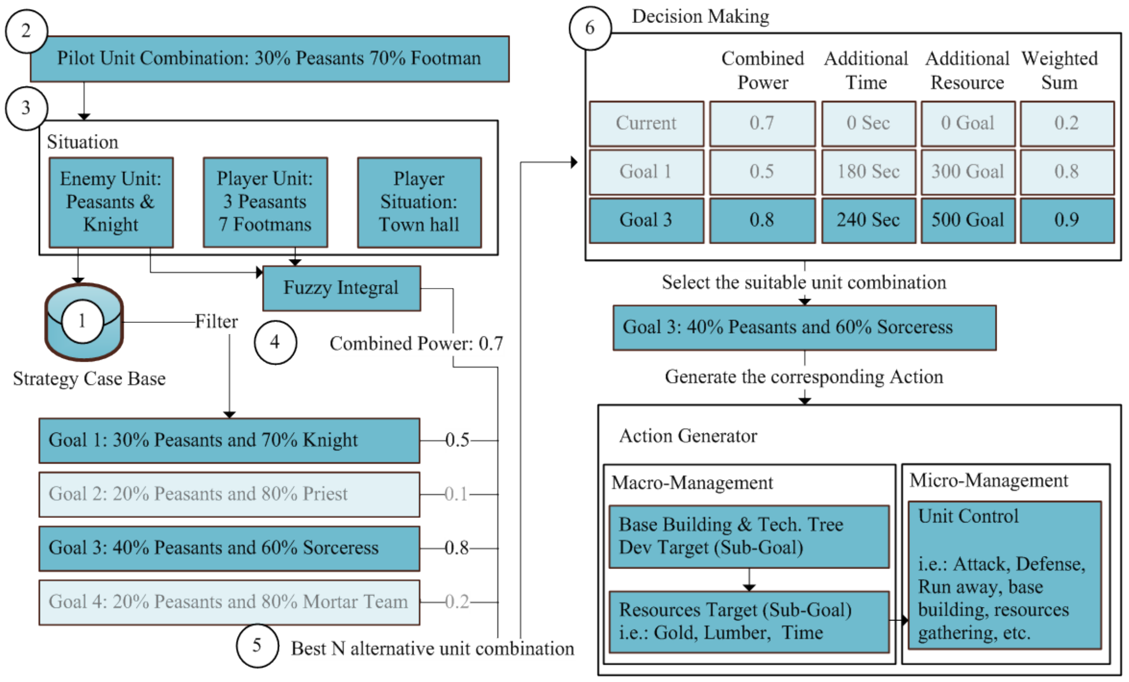

2
Pilot Unit Combination: 30% Peasants 70% Footman
3
Situation
Enemy Unit: Peasants & Knight
Player Unit: 3 Peasants 7 Footmans
Player Situation: Town hall
Fuzzy Integral
1
Filter
Strategy Case Base
4
Combined Power: 0.7
Goal 1: 30% Peasants and 70% Knight
0.5
Goal 2: 20% Peasants and 80% Priest
0.1
Goal 3: 40% Peasants and 60% Sorceress
0.8
Goal 4: 20% Peasants and 80% Mortar Team
0.2
5
Best N alternative unit combination
6
Decision Making
Combined Power
Additional Time
Additional Resource
Weighted Sum
Current
0.7
0 Sec
0 Goal
0.2
Goal 1
0.5
180 Sec
300 Goal
0.8
Goal 3
0.8
240 Sec
500 Goal
0.9
Select the suitable unit combination
Goal 3: 40% Peasants and 60% Sorceress
Generate the corresponding Action
Action Generator
Macro-Management
Base Building & Tech. Tree Dev Target (Sub-Goal)
Resources Target (Sub-Goal) i.e.: Gold, Lumber, Time
Micro-Management
Unit Control
i.e.: Attack, Defense, Run away, base building, resources gathering, etc.

Figure 5.1 Bottom-up strategy planning model

First, all previous cases are converted into the strategy case base and used to train the fuzzy measures as the label 1 in Figure 5.1. When there is a new battle case, pilot unit combination is used to start the game as label 2. Then, at each certain time step, situation of player and enemy are extracted and pass to the decision making modules as shown in label 3. Combined power of units is calculated by fuzzy integral, trained fuzzy measures and enemy unit types. Other alternative unit combination which contains similar situation of enemy and player are filtered out and the best N unit combinations are selected and shown in label 4. Decision making module evaluates the additional resources, time and combined power that are required for each case as shown in label 6. Weighted sum calculation could be applied here and to perform a vary strategies control. For example, if we want to select an aggressive unit combination or rushing strategy, the weighting of additional time should be increased. Finally, the action generator is responsible to perform all the actions according to the game play, e.g., massive basic unit will be produced if rushing strategy is selected. Programmer can only focus on this part and program the actions for each kind of unit type without handling any strategy planning. Thus, the bottom-up strategy model can reduce the complexity of AI programming.

5.4 Evaluating the non-linear property in unit combination

The objective of our chapter is to model feature interactions of different unit type combinations in RTS game using real replay data. If this problem is viewed from the statistical machine learning perspective, for example using Bayesian networks, the model developed describes the probabilistic state transitions among different unit types but not their non-linear interactions. Besides, the learning of the joint probability distributions among unit combinations requires very detailed temporal information about the changes of states and their frequencies. This is very difficult to extract from the replay data. Furthermore, we also need to associate the effectiveness of using these unit combinations in the game. This association is difficult to represent in a Bayesian network. There are many learning algorithms including neural networks, SWARM intelligence, decision trees induction techniques and the traditional searching approaches. All these techniques are formulated on the minimization of some Euclidean distance-based error functions. The parameters used in these functions are all additive in nature and the model is unable to describe the non-linear effects among parameters. For example, the Back-propagation learning algorithm in multi-layer perceptron neural network sums up all the weighted inputs and projects the answer in the output space using various activation functions and the gradient descent technique. The concept of power set and feature combinations will be extremely difficult to encode in these commonly used neural network models.

Our motivation of using fuzzy measure and integral are listed as follows:

1. Fuzzy integral is an efficient aggregation operator to sum up all the fuzzy measure. This integral can be seen as the effect of using the chosen unit type combinations in the game.
2. Feature interactions can be determined by trying out various fuzzy integrals. The best integral then describes the additive, super-additive, and sub-additive properties in the chosen unit type combinations.

3. We also need to consider the quantity of each unit produced in each game. These quantities are measured by the resources needed to produce them rather than their physical count. This transformation can also be captured by the integral function as a weight to each fuzzy measure.

We have mentioned that fuzzy measure is defined as a mapping: $\mu : P(X) \to [0,1]$, where $P(X)$ is the power set of X, i.e., all the ($2^n - 1$) subsets of X. Here in this RTS game research, a subset of X denotes a possible unit combination. Thus, after learning the fuzzy measure, the contribution of each unit combination can be obtained. Fuzzy integral is then be used to sum up all the subsets' fuzzy measures. The final integral can be regarded as the "outcome" of all the unit combinations being used.

Our main idea is to formulate a suitable fuzzy integral which only sums up all those meaningful subsets, i.e., all the unit combinations that have been used in a particular game play. Moreover, the fuzzy measures of these subsets demonstrate the unit's interaction properties, i.e., super-additive, additive or sub-additive. Our methodology is briefly described as following.

Unit type combination

We define unit type combination in a game as the creation of a suitable army mix. For example, given three unit types {peasants}, {footman}, {rifleman}, the power set is: {peasants}, {footman}, {rifleman}, {peasants, footman}, {peasants, rifleman}, {footman, rifleman}, {peasants, footman, rifleman}. If only these three unit types were produced in a game, and assuming that they were used to attack similar types of enemies, then we can hypothesize that the "outcome" of the battle shall be similar. Therefore, the learning of the effectiveness of different unit combinations is possible.

Clustering of game data

A total of 2,649 Warcraft III game logs were collected and clustered based on the player and enemy unit types. Since the battles in Warcarft III involve combats among different races, the players are fighting with each other using different types of units from different races. Therefore, the learnt fuzzy measure and integral tell us what unit type combinations in a particular race is outperform what unit type combinations in another race.

Learning fuzzy measure

Each chromosome in GA or search point in CMA-ES encodes a power set of fuzzy measures, (i.e., $2^n - 1$ fuzzy measures in one chromosome, where *n* is the total number of unit types). We used genetic algorithm and covariance matrix adaptation evolution strategy to search the best fuzzy measures guided by the fitness function. In Warcraft III, each game log provides a final score to both players. This score tells how many enemies were killed, buildings were destroyed, resources being collected and the lands being conquered. Therefore, the higher the score, the better the performance, and it also tells who wins the game. We use this score as the measure of the fitness of a chromosome in GA or search point in CMA-ES. Throughout the iterations, the optimized unit type combination is identified.

Design fuzzy integral

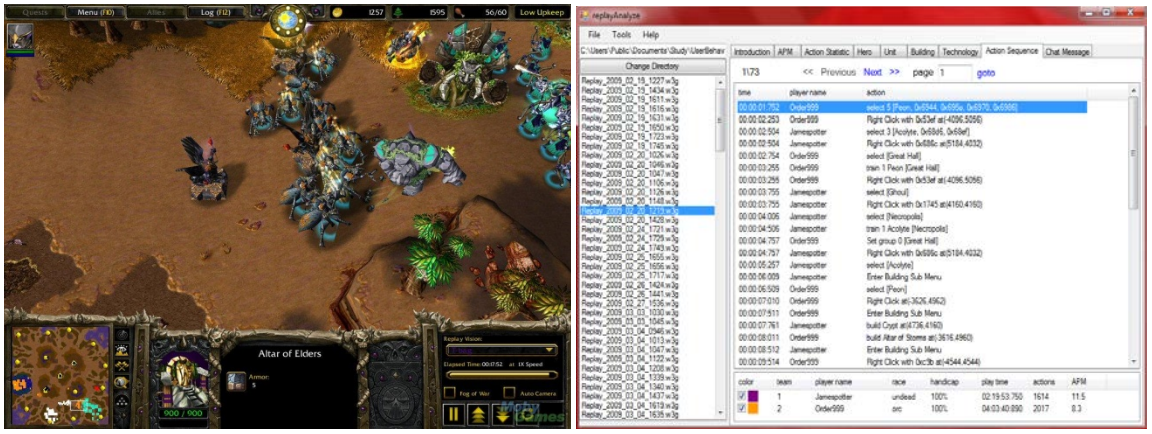
Fuzzy integral can be regarded as an aggregation operator. Different integrals perform the aggregation differently based on the problem on hand. Some common fuzzy integrals include Choquet Integral and Sugeno Integral. In this research, we design some new integrals that are suitable to aggregate the fuzzy measures.

Performance based on feature interaction

We use 2,649 sets of Warcraft III data to perform the experiments. They were selected from a larger set of 8,130. The selected ones are more homogenous in unit type combinations which are also commonly used by players. These data are clustered into several clusters based on the similarities of the unit types that the enemy used. Learning of fuzzy measures is carried out using genetic algorithm. Aggregation of these measures is done by fuzzy integral. We use 70% of the data for training and 30% for testing. 20 cross validation cycles are performed. The result shows that there are feature interactions among unit types. New fuzzy integral are defined to model such interaction in RTS games. Details are explained in the section 5.6.

5.4.1 Data collection and preprocessing

In our research, we select the Warcraft III replays in professional one versus one competition from 2007 to 2010, see Figure 5.2(a). We created a program to decrypt and extract data from these replays, see Figure 5.2(b).



```
C:\Users\Public\Documents\Study\UserBehav
Replay_2009_02_19_1227.w3g
Replay_2009_02_19_1434.w3g
Replay_2009_02_19_1611.w3g
Replay_2009_02_19_1616.w3g
Replay_2009_02_19_1631.w3g
Replay_2009_02_19_1650.w3g
Replay_2009_02_19_1723.w3g
Replay_2009_02_19_1745.w3g
Replay_2009_02_20_1026.w3g
Replay_2009_02_20_1046.w3g
Replay_2009_02_20_1047.w3g
Replay_2009_02_20_1106.w3g
Replay_2009_02_20_1126.w3g
Replay_2009_02_20_1148.w3g
Replay_2009_02_20_1219.w3g
Replay_2009_02_20_1428.w3g
Replay_2009_02_24_1721.w3g
Replay_2009_02_24_1729.w3g
Replay_2009_02_24_1743.w3g
Replay_2009_02_25_1655.w3g
Replay_2009_02_25_1656.w3g
Replay_2009_02_25_1717.w3g
Replay_2009_02_26_1424.w3g
Replay_2009_02_26_1441.w3g
Replay_2009_02_27_1536.w3g
Replay_2009_03_03_1030.w3g
Replay_2009_03_03_1045.w3g
Replay_2009_03_04_0946.w3g
Replay_2009_03_04_1013.w3g
Replay_2009_03_04_1047.w3g
Replay_2009_03_04_1122.w3g
Replay_2009_03_04_1208.w3g
Replay_2009_03_04_1339.w3g
Replay_2009_03_04_1340.w3g
Replay_2009_03_04_1437.w3g
Replay_2009_03_04_1619.w3g
Replay_2009_03_04_1635.w3g
```

1\73

time	player name	action
00:00:01.752	Order999	select 5 [Peon, 0x6944, 0x695e, 0x6970, 0x6986]
00:00:02.253	Order999	Right Click with 0x53ef at(-4096.5056)
00:00:02.504	Jamespotter	select 3 [Acolyte, 0x68d5, 0x68ef]
00:00:02.504	Jamespotter	Right Click with 0x686c at(5184.4032)
00:00:02.754	Order999	select [Great Hall]
00:00:03.255	Order999	train 1 Peon [Great Hall]
00:00:03.255	Order999	Right Click with 0x53ef at(-4096.5056)
00:00:03.755	Jamespotter	select [Ghoul]
00:00:03.755	Jamespotter	Right Click with 0x1745 at(4160.4160)
00:00:04.006	Jamespotter	select [Necropolis]
00:00:04.506	Jamespotter	train 1 Acolyte [Necropolis]
00:00:04.757	Order999	Set group 0 [Great Hall]
00:00:04.757	Jamespotter	Right Click with 0x686c at(5184.4032)
00:00:05.257	Jamespotter	select [Acolyte]
00:00:06.009	Jamespotter	Enter Building Sub Menu
00:00:06.509	Order999	select [Peon]
00:00:07.010	Order999	Right Click at(-3626.4962)
00:00:07.511	Order999	Enter Building Sub Menu
00:00:07.761	Jamespotter	build Crypt at(4736.4160)
00:00:08.011	Order999	build Altar of Storms at(-3616.4960)
00:00:08.512	Jamespotter	Enter Building Sub Menu
00:00:09.514	Order999	Right Click with 0xc3b at(-4544.4544)

color	team	player name	race	handicap	play time	actions	APM
✓ 🟪	1	Jamespotter	undead	100%	02:19:53.750	1614	11.5
✓ 🟧	2	Order999	orc	100%	04:03:40.890	2017	8.3

(a) (b)

Figure 5.2 Warcraft III expansion, The Frozen Throne (WIII: TFT)

The replay data is a binary file which consists of a header which contains some demographic data and some actions blocks. Each block stores all the player actions at 250 milliseconds' interval. All the random events are generated by a random seed that is given at the beginning of the battle. Table 5.1 shows the elements in Warcraft III replays. Table 5.2 shows a detailed example of it. The example shows that there are some building actions with XY coordinates, technological upgrade, unit productions, unit selections, unit movement commands with XY coordinate, and unit skills, etc. We have created a C# program to decrypt and extract the data from these replays, see Figure 5.2.

TABLE 5.1
DATA IN WARCARFT III REPLAY

	Element
Header	Player Record, Game Name, Map Settings, Map Record, Map & Creator Name, Player Count, Game Type, Language ID, Player List, Game Start Record, Slot Record, Random Seed
Actions Block	Player ID, Action ID, Action Arguments

TABLE 5.2
SAMPLE OF WARCARFT III REPLAY

Time	Action
00:00:02:002	Player 1 train 1 Peasant
00:00:02:253	Player 1 select 5 [Peasant]
00:00:02:503	Player 1 Right Click with 0x58a8 at(7296,2432)
00:00:15:241	Player 2 build Altar of Darkness at(-1376,6240)

Player 1 produces 1 Peasants, then select 5 Peasants and move to (X: 7296, Y: 2432). Player 2 builds Altar of Darkness at(X: -1376, Y: 6240)

The data that extracted from the replay are (1) unit type combination, (2) enemy units and player races and (3) the final scores. These three attributes are called one case for our training. Its definition is as follows.

Case = {Unit Type Combination, Situation, Scores}

Unit Type Combination refers to a suitable army mix, i.e., a suitable unit combination with certain proportion, e.g., 10% peasants, 40% footman and 50% rifleman. Situation refers to some common circumstances the player is dealing with. This information is used for clustering cases. Score refers to the points obtained after the game. It is used for evaluating the outcome of the Warcraft III battle, and is used to guide the training of fuzzy measure.

5.4.2 Learning fuzzy measure by GA

We used Genetic Algorithm (GA) to obtain the fuzzy measure. The technique used is similar to [Wang 1997, Wang 1998, Wang 1999, Cheng 2000]. We selected GA to obtain fuzzy measure as it has been widely used with its efficiency proven. The chromosome can be defined to represent all the subsets in a power set, i.e., all unit type combinations can be represented. Fuzzy measures can be associated with each subset easily. Genes in the chromosome are grouped in pairs while each pair represents the unit type combination and its associated fuzzy measures.

The chromosome consists of many subsets of fuzzy measure and they are highly related. A strong unit combination in RTS game is usually come from a success unit type. For example, $\{x_1, x_2\}$ is usually high if $\{x_1\}$ and $\{x_2\}$ is high. The crossover concept of GA provides a natural way to produce solutions with high qualities by two successful fuzzy measures within a few generations. However, the search space of fuzzy measure is large and our replay data is insufficient. We can only obtain the partial information from the replays. The strength of GAs is providing a parallel global search in fuzzy measure. Through roulette wheel selection and the mutation operators, even weak fuzzy measure may have chances to be part of the future candidate solutions. It can avoid the local minimum problem.

We combined fuzzy measure and integral with GA by the following steps. First, an initial population of chromosome is randomly generated and their fitness is computed by fuzzy integral and compared with the real game data. Selection, crossover and mutation operators are applied to these chromosomes to generate children for the next generation. After certain amount of generations or meeting the termination condition, optimized fuzzy measures is obtained and used for testing, see Figure 5.3.

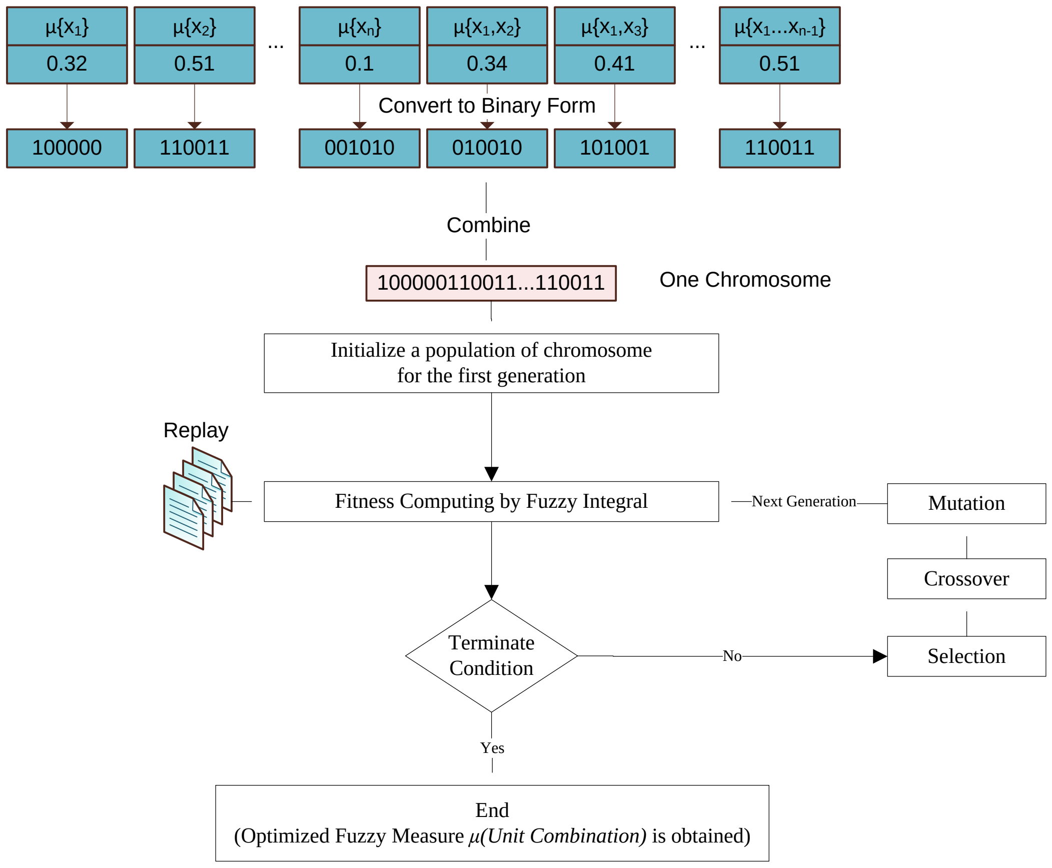


Figure 5.3 Learning fuzzy measure by GA

5.4.3 Learning fuzzy measure by CMA-ES

We also use Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to obtain the fuzzy measure. The technique has been used to evaluate different functions [Hansen 2004, Shir 2006, Liang 2005]. In Black-Box Optimization Benchmarking (BBOB) competition of Genetic and Evolutionary Computation Conference (GECCO) 2009 [Hansen 2010] and 2010 [Auger 2010], CMA-ES has been proven to be better than other evolutionary algorithms in about 90% of the cases and is more effective to evaluate non-linear function. Therefore, we combine fuzzy measure and integral with CMA-ES.

CMA-ES provides the information in global search by the mean of selected points and local information by the covariance matrix of each search point. We combine fuzzy measure and integral with CMA-ES by the following steps. First, an initial population of search points is randomly generated in a $2^n - 1$ dimensional matrix, where n is the number of unit type. Again, their fitness is computed by fuzzy integral and compared with the real game data. A group of better solution is selected to computer mean vector, step size and covariance matrix. The search points are updated by these three parameters for the next generation. After certain amount of generations or meeting the termination condition, the fuzzy measures are used for testing, see Figure 5.4.

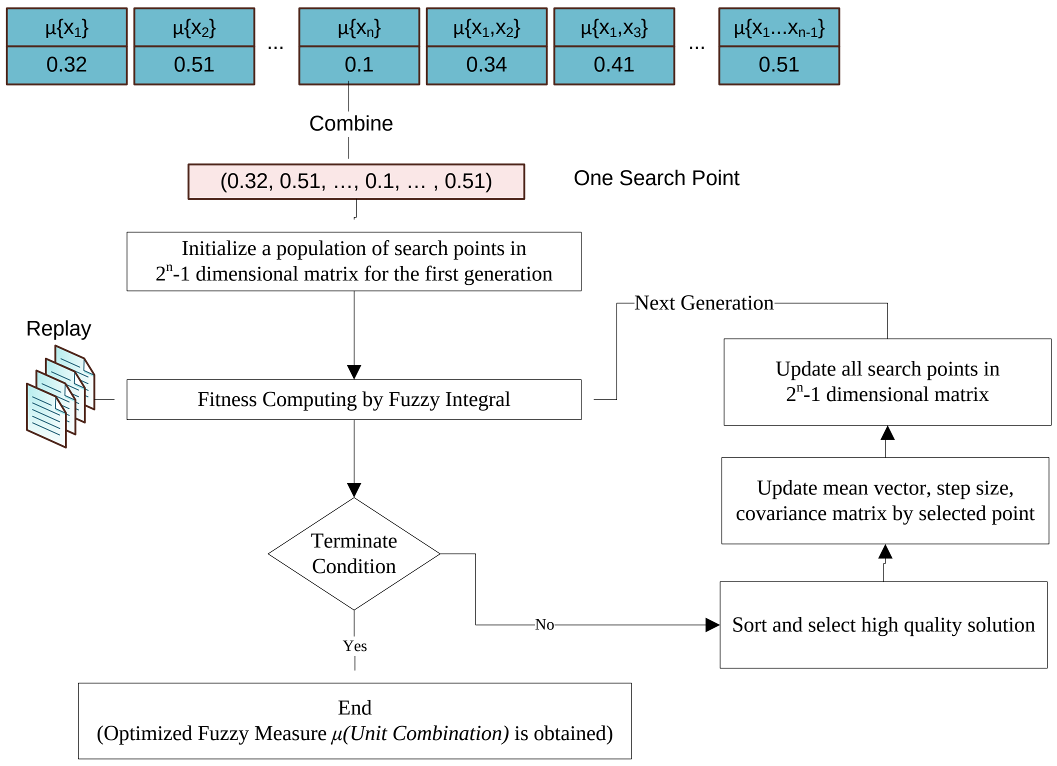

μ{x1}
0.32
μ{x2}
0.51
...
μ{xn}
0.1
μ{x1,x2}
0.34
μ{x1,x3}
0.41
...
μ{x1...xn-1}
0.51
Combine
(0.32, 0.51, …, 0.1, … , 0.51)
One Search Point
Initialize a population of search points in 2n-1 dimensional matrix for the first generation
Replay
Fitness Computing by Fuzzy Integral
Next Generation
Update all search points in 2n-1 dimensional matrix
Update mean vector, step size, covariance matrix by selected point
Terminate Condition
No
Sort and select high quality solution
Yes
End
(Optimized Fuzzy Measure μ(Unit Combination) is obtained)

Figure 5.4 Learning fuzzy measure by CMA-ES

5.4.4 Fitness value determination

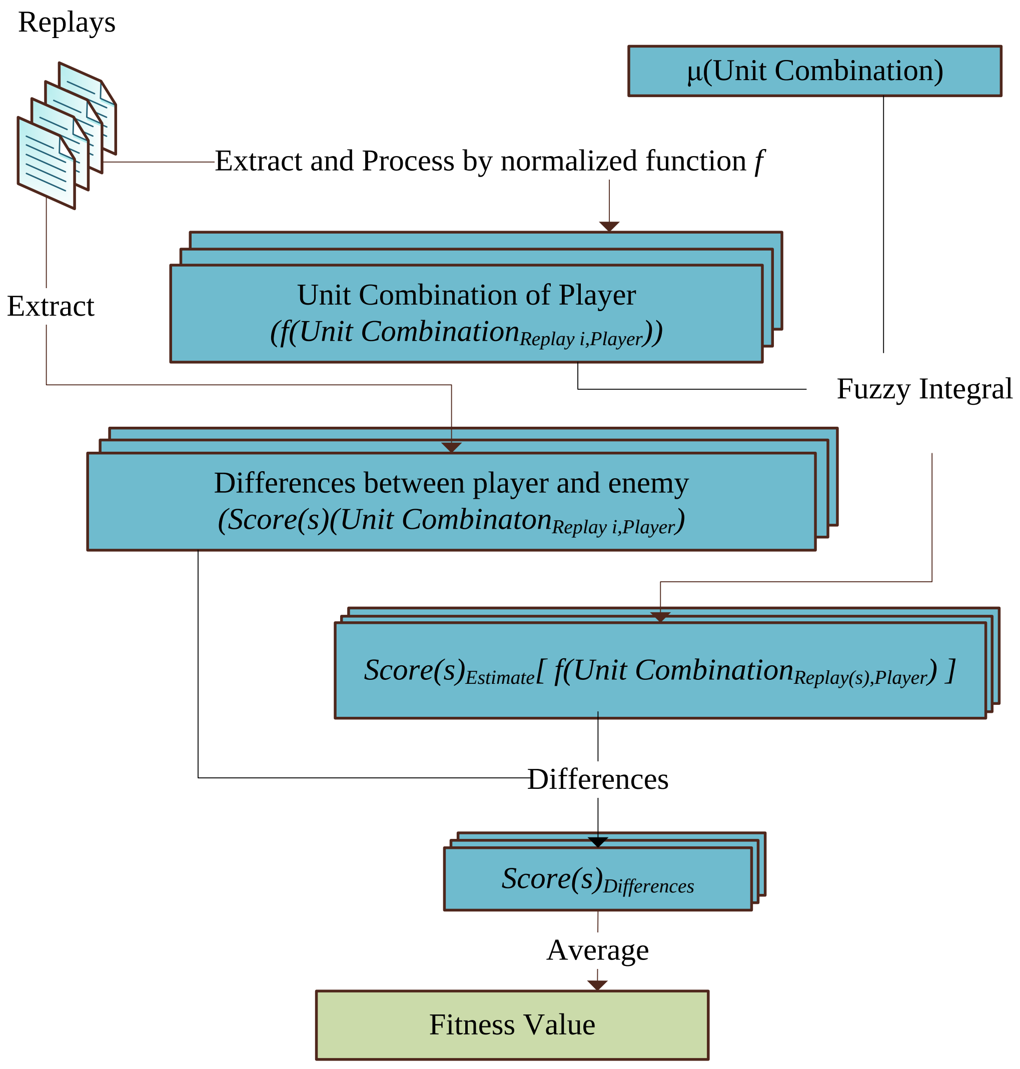

Replays
μ(Unit Combination)
Extract and Process by normalized function *f*
Extract
Unit Combination of Player
(f(Unit Combination$_{\text{Replay i,Player}}$))
Fuzzy Integral
Differences between player and enemy
(Score(s)(Unit Combinaton$_{\text{Replay i,Player}}$)
Score(s)$_{\text{Estimate}}$[f(Unit Combination$_{\text{Replay(s),Player}}$)]
Differences
Score(s)$_{\text{Differences}}$
Average
Fitness Value

Figure 5.5 Fitness calculation of one chromosome

Figure 5.4 shows the step of fitness calculation. First, the exact quantity produced by each unit type is extracted from the game data. These quantities are normalized by a normalization function f. For example, given three unit types: 10% peasant, 40% footman and 50% rifleman, let $X = \{x_1, x_2, x_3\}$, and x_1, x_2, x_3 denotes peasant, footman and rifleman respectively. Then $f(x_1) = 0.1$; $f(x_2) = 0.4$ and $f(x_3) = 0.5$. Noted that the quantity here is measured by the amount of resources used instead of the physical quantity count. For example, the resources needed to produce a footman is twice as much as a peasant, therefore even the physical count is one footman and one peasant, f the values of them is 2 versus 1.

Next, the total score defined as $Score_{WarcraftIII}(r)$ is extracted from each replay, r. Table 5.3 and Equation (5.1) shows its detail components.

$$Score_{WarcraftIII}(r) = \text{Unit score + Resource score + Hero score} \tag{5.1}$$

where r stands for replay r

TABLE 5.3

ELEMENTS IN SCORE$_{\text{WARCRAFT III}}$

Type	Element
Unit score	Units Produced, Units Killed, Buildings Produced, Buildings Razed
Resource score	Gold Mined, Lumber Harvested
Hero score	Experience Gained

We define $Score(r)$, as the differences between player's and enemy's scores, as shown below in Equation (5.2).

$$Score(r) = Score_{WarcraftIII,player}(r) - Score_{WarcraftIII,enemy}(r) \tag{5.2}$$

Positive $Score(r)$ means the player is performing better in the battle than his enemy and vice versa. $Score(r)$ is also normalised in our study. Estimated score is calculated by the fuzzy integral based on the learned fuzzy measures and the quantity produced in each unit type, i.e., $f(x)$. See Figure 5.5.

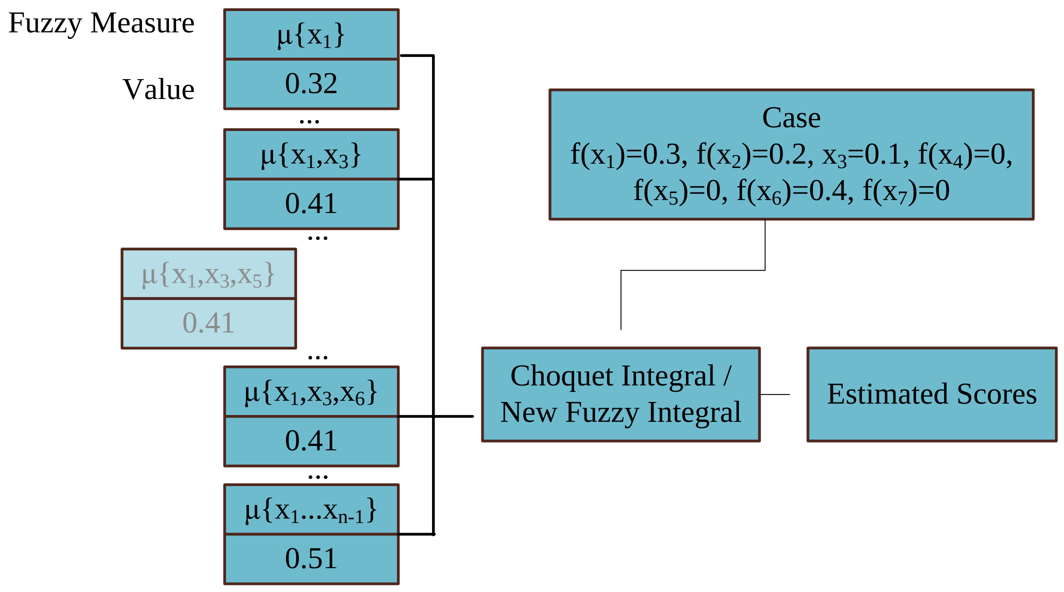


Figure 5.6 Estimated score calculation

$Score_{Estimate}[\int f(x) \circ \mu(X)]$ is denoted as the estimated score. The difference between the real score (from data) and the estimated score is denoted as Equation (5.3).

$$Score_{Different}(r) = Score_{Estimate}[\int f(x) \circ \mu(X)] - Score(r) \tag{5.3}$$

The fitness of each chromosome is defined in Equation (5.4). It is the root mean square average of the score differences between the actual score and the estimated score.

$$\text{Fitness value} = -\frac{1}{1+e} \text{ where } e = \left(\frac{1}{N}\sum_{i=1}^{N} score^2_{Different}(i)\right)^{\frac{1}{2}} \tag{5.4}$$

5.5 Applying different fuzzy integral in fitness function

5.5.1 Choquet Integral

We demonstrate our above idea by using the following example. Given that the number of unit type is 7, i.e., $n = 7$ and the unit combination of the replay is:

$$f(x_1) = 0.1,\ f(x_2) = 0.3,\ f(x_3) = 0.6,$$
$$f(x_4) = 0,\ f(x_5) = 0,\ f(x_6) = 0,\ f(x_7) = 0$$

For the Choquet Integral, the ascending order sorting of $f(x)$ is $a_1 = f(x_4) = 0$, $a_2 = f(x_5) = 0$, $a_3 = f(x_6) = 0$, $a_4 = f(x_7) = 0$ $a_5 = f(x_1) = 0.1$, $a_6 = f(x_2) = 0.3$, $a_7 = f(x_3) = 0.6$ where $a_1 \le a_2 \le ... \le a_n$. Suppose, the fuzzy measure obtained are $\mu(X) = \{..., \mu(\{x_3\}) = 0.74, \mu(\{x_2, x_3\}) = 0.98, ... , \mu(\{x_1, x_2, x_3\}) = 0.71, ...\}$

If we use Choquet integral,

$$
\begin{aligned}
&(c)\int f(x) \circ \mu(X) \\
&= \sum_{i=1}^{7} (a_i - a_{i-1}) \cdot \mu(\{x \mid f(x) \ge a_i\}) \\
&= ((a_1 - a_0) \cdot \mu(\{x \mid f(x) \ge a_1)\}) + ... \\
&= 0 + ((a_5 - a_4) \cdot \mu(\{x \mid f(x) \ge a_5)\}) + ... \\
&= 0.1 \cdot \mu(\{x_1, x_2, x_3\}) + 0.2 \cdot \mu(\{x_2, x_3\}) + 0.3 \cdot \mu(\{x_3\}) \\
&= 0.1 \times 0.71 + 0.2 \times 0.98 + 0.3 \times 0.74 \\
&= 0.489
\end{aligned}
$$

The original score provided by Warcraft III is 0.41. Therefore, the error is 0.079.

5.5.2 Motivation to develop new fuzzy integrals

CI is useful in many applications because of its generalised mean operator and the assumption that every combination of features is considered to be possible. Furthermore, from its definition, the aggregation put more emphasis on those subsets which have smaller values of a_i. This assumes that the feature interactions are considered more important if the set contains features with smaller values of a_i. We have some doubt that whether this will work well in RTS games in which the larger the values of a_i the more important the unit w.r.t. and the ability to fight with enemy. This is the motivation why we want to develop other aggregation operators and compare them with CI.

There is usually a sequence in building up resources in a typical RTS game, i.e., labour units will be produced before the other simple military units being generated.. Different kinds of advance unit will be unlocked further after some resources threshold. We believe that the advance units are important and shall carry heavier interactions with other units. We developed two new integrals for investigation. They are explained in the following sections.

5.5.3 Mean based fuzzy integral

Mean based Fuzzy Integral (Mean based FI or (m)) which is defined in Equation (5.5).

Suppose a fuzzy measure μ on X. Mean based FI of a function $f : X \Rightarrow \mathbb{R}^+$ can be written as following form.

$$(m)\int f(x) \circ \mu(X) = \sum_{i=1}^{n} x_i \times \left(\frac{1}{m_i}\sum_{j=1}^{m_i} \mu(S_{ij})\right) \tag{5.5}$$

where $x_i \in S_{ij}$ and $\forall x \in S_{ij}, x \neq 0$, n is number of unit type,

m is the number of sets which consist of x_i

Mean based FI tries to find out all the interactions that involve the selected unit type with an average being taken. Compared with CI, it shows a better result in both training and testing. One of the weaknesses is the heavier computed load with longer training time as it needs to search the corresponding subset, $\mu(S_{ij})$, for m time in $2^n - 1$ search space. Compare it with CI, no additional searching is required.

5.5.4 Max based fuzzy integral

Max based Fuzzy Integral (Max based FI or (M)) which is defined in Equation (5.6).

Suppose a fuzzy measure μ on X. Max based FI of a function $f : X \Rightarrow \mathbb{R}^+$ can be written as following form.

$$(M)\int f(x) \circ \mu(X) = \sum_{i=1}^{n} x_i \times \max(\mu(S_{ij})) \tag{5.6}$$

where $x_i \in S_{ij}$ and $\forall x \in S_{ij}, x \neq 0$, n is number of unit type,

m is the number of sets which consist of x_i

As we mentioned before, we concern with the powerful unit type in unit combination. We design Max based FI and consider the highest value subset. Again, it obtains a better result but it is more time consuming.

5.5.5 Order based fuzzy integral

As the resource weighting in the advance unit is higher, usually the proportion is usually dominated by them as shown in Figure 5.6 as they are the core of the army mix. Other units should collaborate with the advance units. We put this in priority order with the most important as the top consideration. Order based FI focuses on the highest proportion units (i.e., highest resources units) first and calculates their interactions with all other units and so on.

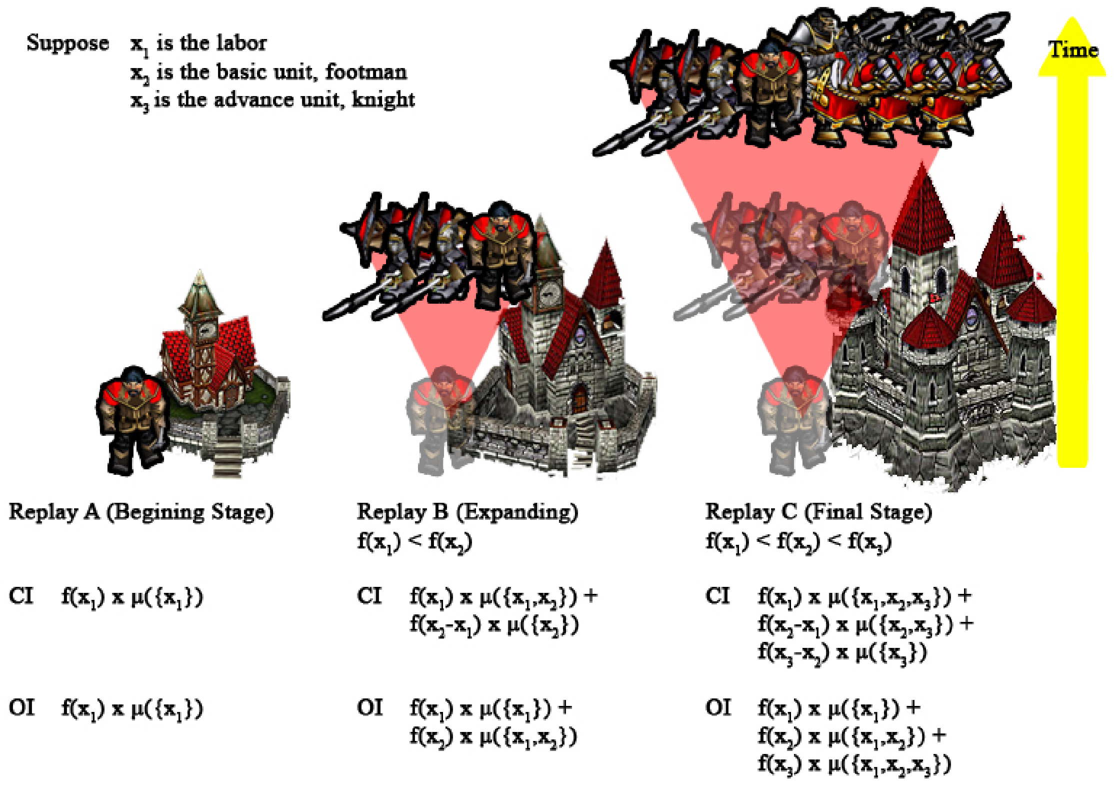

Suppose x_1 is the labor
x_2 is the basic unit, footman
x_3 is the advance unit, knight
Time
Replay A (Begining Stage)
CI $f(x_1) \times \mu(\{x_1\})$
OI $f(x_1) \times \mu(\{x_1\})$
Replay B (Expanding)
$f(x_1) < f(x_2)$
CI $f(x_1) \times \mu(\{x_1,x_2\}) + f(x_2\text{-}x_1) \times \mu(\{x_2\})$
OI $f(x_1) \times \mu(\{x_1\}) + f(x_2) \times \mu(\{x_1,x_2\})$
Replay C (Final Stage)
$f(x_1) < f(x_2) < f(x_3)$
CI $f(x_1) \times \mu(\{x_1,x_2,x_3\}) + f(x_2\text{-}x_1) \times \mu(\{x_2,x_3\}) + f(x_3\text{-}x_2) \times \mu(\{x_3\})$
OI $f(x_1) \times \mu(\{x_1\}) + f(x_2) \times \mu(\{x_1,x_2\}) + f(x_3) \times \mu(\{x_1,x_2,x_3\})$

Figure 5.7 Comparison of CI and OI in three replay data

Order based Fuzzy Integral (Order based FI or (o)) considers the order of unit type production and is defined in Equation (5.7).

Suppose a fuzzy measure μ on X. Max based FI of a function $f: X \Rightarrow \mathbb{R}^+$ can be written as following form.

$$(o)\int f(x) \circ \mu(X) = \sum_{i=1}^{n} a_i \cdot \mu(\{x \mid 0 < f(x) \le a_i\}) \text{ or} \tag{5.7}$$

$$(o)\int f(x) \circ \mu(X) = \sum_{i=1}^{n} a_i \cdot \mu(F_{a^-})$$

Where $a_1, \ldots, a_n$ is the sorted $f(x_1), \ldots, f(x_n)$, $a_1 \ge \ldots \ge a_{n-1} \ge a_n > 0$ and F_{α^-} is the complement set of F_{α^+} and $F_{\alpha^-} = \{x \mid 0 < f(x) \le a_i\}$

Here we used an example to illustrate our idea. Given the number of unit type is 7, i.e., $n = 7$ and the unit combination of this replay is

$$f(x_1) = 0.1, f(x_2) = 0.3, f(x_3) = 0.6,$$
$$f(x_4) = 0, f(x_5) = 0, f(x_6) = 0, f(x_7) = 0$$

Then, descending order sorting is performed, i.e. $a_1 = f(x_3) = 0.6$, $a_2 = f(x_2) = 0.3$, $a_3 = f(x_1) = 0.1$, where $a_1 \geq a_2 \geq a_3$. As the fuzzy measure is trained by Order based FI, the values are shown below,

$\mu(X) = \{..., \mu(\{x_1\}) = 0.54, \mu(\{x_1, x_2\}) = 0.42, ... \mu(\{x_1, x_2, x_3\}) = 0.43, ...\}$

The Order based FI will give

$$\begin{aligned}
&(o)\int f(x) \circ \mu(X) \\
&= \sum_{i=1}^{4} (a_i) \cdot \mu(\{x \mid 0 < f(x) \leq a_i\}) \\
&= a_1(\mu(\{a_1, a_2, a_3\})) + a_2(\mu(\{a_2, a_3\})) + a_3(\mu(\{a_3\})) \\
&= x_3(\mu(\{x_1, x_2, x_3\})) + x_2(\mu(\{x_1, x_2\})) + x_1(\mu(\{x_1\})) \\
&= 0.6 \times 0.43 + 0.3 \times 0.42 + 0.1 \times 0.54 \\
&= 0.438
\end{aligned}$$

The original score that given by Warcraft III is 0.41 and the error is 0.028. Compared with the error computed by CI, 0.079, Order based FI gives a better performance prediction in this case.

5.5.6 Properties of order based fuzzy integral

Let $X = \{x_1, x_2, ..., x_n\}$ and $\mu : P(X) \to [0, \infty)$ is a non-monotonic measure, i.e., efficiency measure on the power set of X. Ordered based fuzzy integral is a non-linear integral. It also satisfies some basic properties of common fuzzy integrals, which are listed as following.

$$(o)\int_X 1d\mu = \mu(X) \tag{5.8}$$

For any $c \in [0,\infty)$, $(o)\int c \cdot fd\mu = c \cdot (o)\int fd\mu$ (5.9)

$(o)\int fd\mu < (o)\int gd\mu$ if $f(x) \le g(x)$ (5.10)

for every $x \in X$ and $f(x) \le f(y) \Rightarrow g(x) \le g(y)$

Proof:

For finite set $X = \{x_1, x_2, ..., x_n\}$ and $f(x_i)$, then it is sorted in a descending order, i.e., $f(x_1^*) \ge f(x_2^*) \ge ... \ge f(x_n^*)$, where $\{x_1^*, x_2^*, ..., x_n^*\}$ is a permutation of X .

Since $f(x) \le f(y) \Rightarrow g(x) \le g(y)$, we have $g(x)$ also maintains the order of function value $g(x_1^*) \ge g(x_2^*) \ge ... \ge g(x_n^*)$

Based on the definition of Order based FI,

$$
\begin{aligned}
&(o)\int fd\mu \\
&= \sum_{i=1}^{n} f(x_i^*) \cdot \mu(\{x \mid 0 < f(x) \le f(x_i^*)\}) \\
&= f(x_1^*)\mu(x_1^*, x_2^*, ..., x_n^*) + f(x_2^*)\mu(x_2^*, ..., x_n^*) + ... + f(x_n^*)\mu(x_n^*)
\end{aligned}
$$

$$
\begin{aligned}
&(o)\int gd\mu \\
&= \sum_{i=1}^{n} g(x_i^*) \cdot \mu(\{x \mid 0 < f(x) \le f(x_i^*)\}) \\
&= g(x_1^*)\mu(x_1^*, x_2^*, ..., x_n^*) + g(x_2^*)\mu(x_2^*, ..., x_n^*) + ... + g(x_n^*)\mu(x_n^*)
\end{aligned}
$$

Because $f < g$, we have $f(x) < g(x)$, for every $x \in X$, then $f(x_i^*) \le g(x_i^*)$, for $i = 1, 2, ..., n$

It is obvious that $(o)\int fd\mu < (o)\int gd\mu$ holds.

If $f(x) \le f(y) \Rightarrow g(x) \le g(y)$ does not hold, $(o)\int fd\mu < (o)\int gd\mu$ may not be true even if $f(x) \le g(x)$, for every $x \in X$

Example:

Let $X = \{x_1, x_2, x_3\}$, $f(x_1) = 10, f(x_2) = 7, f(x_3) = 3$,

$g(x_1) = 2, g(x_2) = 6, g(x_3) = 5$,

$\mu(x_1) = 1, \mu(x_2, x_3) = 3, \mu(x_1, x_3) = 10, \mu(x_1, x_2, x_3) = 5$

$$\begin{aligned}
&(o)\int f d\mu \\
&= f(x_1)\mu(x_1, x_2, x_3) + f(x_2)\mu(x_2, x_3) + f(x_3)\mu(x_3) \\
&= 10(5) + 7(3) + 3(1) \\
&= 74
\end{aligned}$$

$$\begin{aligned}
&(o)\int g d\mu \\
&= g(x_2)\mu(x_1, x_2, x_3) + g(x_3)\mu(x_1, x_3) + g(x_1)\mu(x_1) \\
&= 6(5) + 5(10) + 2(1) \\
&= 82
\end{aligned}$$

$(o)\int f d\mu < (o)\int g d\mu$ does not hold even if $f(x) \le g(x)$, for every $x \in X$

If α is a non-negative real value, b is a real value, (5.11)

then $(o)\int (af + b) d\mu = a(o)\int f d\mu + b\mu(X)$

$(o)\int f d\mu \le (o)\int f d\nu$ if $\mu(A) \le \nu(A)$ for every $A \subseteq X$ (5.12)

$(o)\int_A f d\mu \le (o)\int_B f d\mu$, where $A \subseteq B$ and μ is a monotonic fuzzy measure (5.13)

Proof:

Let $A = \{x_1, x_2, ..., x_n\}$, $B = \{y_1, y_2, ..., y_m\}$ and $A \subseteq B$, $n < m$

A and B is then sorted in descending order ,

i.e., $A = \{x_1^*, x_2^*, ..., x_n^*\}$ and $f(x_1^*) \ge f(x_2^*) \ge ... \ge f(x_n^*)$

i.e., $B = \{y_1^*, y_2^*, ..., y_n^*\}$ and $f(y_1^*) \ge f(y_2^*) \ge ... \ge f(y_n^*)$

Since $A \subseteq B$, we have $\forall x_j^* \in A, \exists y_i^* \in B$, such that $x_j^* = y_i^*$, $j \le i$, i.e., $f(x_j^*) = f(y_i^*)$ as μ is monotonic,

$$
\begin{aligned}
&(o)\int_A f d\mu \\
&= \sum_{j=1,2,\dots,n} f(x_j^*) \cdot \mu(\{x_1^*, x_2^*, .., x_j^*\}) \\
&\le \sum_{i|x_j^*=x_i^*} f(x_i^*) \cdot \mu(\{x_1^*, x_2^*, .., x_i^*\}) \\
&\le \sum_{i=1,2,\dots,m} f(x_i^*) \cdot \mu(\{x_1^*, x_2^*, .., x_i^*\}) \\
&= (o)\int_B f d\mu
\end{aligned}
$$

If μ is non-monotonic, the above property is not necessarily true.

Example:

Let $A = \{1,3\}$, $B = \{1,2,3\}$, $A \subseteq B$

$\mu(1) = 1$, $\mu(1,2) = 3$, $\mu(1,3) = 10$, $\mu(1,2,3) = 5$, μ is non-monotonic

$$
\begin{aligned}
&(o)\int_A f d\mu \\
&= 1(1) + 3(10) \\
&= 31
\end{aligned}
$$

$$
\begin{aligned}
&(o)\int_B f d\mu \\
&= 1(1) + 2(3) + 3(5) \\
&= 22 \\
&\le 31 \\
&\le \int_A f d\mu
\end{aligned}
$$

$(o)\int f d\mu = 0 \xLeftrightarrow{\text{if and only if}}$ for $\forall A \subseteq X$ with $\mu(A) > 0$, there exists $x \in A$ such that $f(x) = 0$, that is $\mu(\{x \mid f(x) > 0\}) = 0$ (5.14)

$\forall a \in [0,\infty)$, $(o)\int (f + a) d\mu \ge (o)\int f d\mu + (o)\int a d\mu$ (5.15)

Proof:

Assume there exits $f(x_1^*) \geq f(x_2^*) \geq ... \geq f(x_n^*)$,

then $f(x_1^*) + a \geq f(x_2^*) + a \geq ... \geq f(x_n^*) + a$

$$
\begin{aligned}
&(o)\int (f+a)d\mu \\
&= \sum_{i=1}^{n} f(x_i^* + a) \cdot \mu(\{x \mid 0 < f(x) \leq f(x_i^*)\}) \\
&= f(x_1^* + a) \cdot \mu(x_1^*, ..., x_n^*) + f(x_2^* + a) \cdot \mu(x_2^*, ..., x_n^*) + .. + f(x_n^* + a) \cdot \mu(x_n^*) \\
&= f(x_1^*) \cdot \mu(x_1^*, ..., x_n^*) + f(x_2^*) \cdot \mu(x_2^*, ..., x_n^*) + .. + f(x_n^*) \cdot \mu(x_n^*) + a[\mu(x_1^*, ..., x_n^*) + ... \mu(x_n^*)] \\
&\geq f(x_1^*) \cdot \mu(x_1^*, ..., x_n^*) + f(x_2^*) \cdot \mu(x_2^*, ..., x_n^*) + ... + f(x_n^*) \cdot \mu(x_n^*) + a\mu(X) \\
&= (o)\int f d\mu + (o)\int a d\mu
\end{aligned}
$$

$\int (f+g)d\mu = \int f d\mu + \int g d\mu$ may not hold as Order based FI is non-linear. Based (5.16) on the priority sorting, the subset selection a different.

In the experiment of this chapter, μ is set as a bounded variation, and $f(x)$ as a normalized unit proportion within 0 and 1. The resulting upper bound and lower bound of Order based FI are 0 and 1 respectively.

5.5.7 Subset selection for different fuzzy integral

The main difference of the three fuzzy integrals is the subset selection. They select different sets for summation. Finding the subset selection could provide a better understanding of fuzzy integral. Hence, the training of fuzzy measure is a time consuming task. In the RTS game, some unit combination will never be used due to the game play design. Finding the subset selection can reduce the length of chromosome and the dimension of search point.

According to [Wang 1996], all the subsets in the fuzzy measure could be generated by the following equation as shown in (5.17). K_i is generated for each i and represents a subset in fuzzy measure, μ . The order is $\{x_1\}$, $\{x_2\}$, $\{x_1, x_2\}$, $\{x_3\}$, $\{x_1, x_3\}$, $\{x_2, x_3\}$, $\{x_1, x_2, x_3\}$, $\{x_4\}$,…, $\{x_1, x_2, ..., x_n\}$. $\overline{K_i}$ represents the complement set of K_i . The advantage for using this equation is that the set can be directly indicated by using i .

(5.17)

$$K_i = \left\{ k : \frac{i}{2^k} - \left\lfloor \frac{i}{2^k} \right\rfloor \geq 0.5,\ 1 \leq k \leq n \right\}$$

$$\overline{K_i} = \{1, 2, \cdots, n\} - K_i$$

Where $i = 1, 2, \cdots, 2^n - 1$ and $k = 1, 2, ...n$, n is the number of feature, $\lfloor a \rfloor$ denotes integer part for a non-negative number a

Base on Equation (5.17), the set selection of CI could be expressed in Equation (5.18), where δ_i is the set selection operator for each set. In our study, we try to convert the Equation (5.18) into Equation (5.19) for better understanding and comparison. Subset selection of Order based FI is shown in Equation (5.20). Mean Based FI is shown in Equation (5.21) and Max Based FI is shown in Equation (5.22) for reference.

$$\delta_i = \max(\min_{k\in K_i} f(x_k) - \max_{k\in \overline{K_i}} f(x_k), 0) \tag{5.18}$$

$$\delta_i = \begin{cases} \min_{k\in K_i} f(x_k) - \max_{k\in \overline{K_i}} f(x_k), & \min_{k\in K_i} f(x_k) > \max_{k\in \overline{K_i}} f(x_k) \\ 0, & \min_{k\in K_i} f(x_k) \le \max_{k\in \overline{K_i}} f(x_k) \end{cases} \tag{5.19}$$

Assume $\min_{k\in\phi} f(x_k) = \max_{k\in\phi} f(x_k) = 0$

$$\delta_i = \begin{cases} \max_{k\in K_i} f(x_k), & \max_{k\in K_i} f(x_k) < \min_{k\in \overline{K_i}} f(x_k) \\ 0, & \max_{k\in K_i} f(x_k) \ge \min_{k\in \overline{K_i}} f(x_k) \end{cases} \tag{5.20}$$

$$\delta_i = \operatorname{avg}_{k\in K_i} f(x_k) \tag{5.21}$$

$$\delta_i = \max_{k\in K_i} f(x_k) \tag{5.22}$$

5.5.8 Extrapolation of fuzzy measures for missing points

As stated before, some unit type combinations do not exist. We called such subset as missing data set. We need a method to extrapolate such fuzzy measures. The extrapolation of fuzzy measure is performed. First, starting from the set with the smallest number of elements, each missing data set, α , is extrapolated by taking the average of its neighborhood, $\frac{1}{|E|}\sum_{\alpha\subset E}\mu_\alpha$ where $\forall E \subset X$ and $|E| = |\alpha| - 1$. For example, the fuzzy measure of set {4,6,8} can be estimated by the taking an average of set {4,6}, {4,8} and {6,8}. The missing data set could be found by Equation (5.19) and Equation (5.20).

5.6 Experimental result and discussion

5.6.1 Brief description of testing data

2,649 strategy cases of one versus one battle are collected. Useful information is decrypted and extracted from the replay data. In Warcraft III, there are 73 unit types which consist of different attributions and skills. They are regarded as different attributions and are divided into four different races. Each race has a unique set of units, structures, technologies, and base-building methodologies. Clustering is performed based on player and enemy unit type as the following two reasons. First, unit combination is intransitive superiority; the combined power is affected due to the enemy unit type. Second, it can reduce the number of unit types, n in fuzzy measure. Top five unit combinations are selected for the experiment as shown in Table 5.4.

TABLE 5.4
DATA NATURE OF TESTING DATA CLUSTER

Data Cluster	1	2	3	4	5
Player race	Undead	Undead	Orc	Orc	Elf
Enemy unit	Fm, P	Dr, Fm, P	Pr, So, Sb, P	Fm, P	Fm, P
No. of unit type (n)	7	7	11	8	8
Fuzzy measure Size ($2^n - 1$)	127	127	2047	255	255
No. of case	162	65	1004	549	869
No. of Combination	23	16	60	31	31

Fm – Footman, P – Peasant, Dr, Dragon rider, Pr – Priest, So – Sorceress, Sb – Spell breaker

We preformed the U-Test on the same race. By observing asymptotic significance, 65% of the features are below 0.05. Results of the Mann-Whitney Test of are shown in Table 5.5 and 5.6. Therefore, the data set is not identical. To prove if there are interactions among the attributions, bivariate correlation is used to measure the relationship between the two variables as shown in Table 5.7. 40% of correlation is significant in our data set.

TABLE 5.5
MANN-WHITNEY TEST OF DATA CLUSTER 1 & 2

RANK

Unit Type	Data Cluster	No of Case	Mean Rank	Sum of Ranks
Acolyte	1	174	103.07	17933.50
	2	65	165.33	10746.50
	Total	239		
Ghoul	1	174	115.59	20113.50
	2	65	131.79	8566.50
	Total	239		
Fiend	1	174	121.26	21098.50
	2	65	116.64	7581.50
	Total	239		
Gargoyle	1	174	101.91	17732.50
	2	65	168.42	10947.50
	Total	239		
Wagon	1	174	115.48	20093.00
	2	65	132.11	8587.00
	Total	239		
Obsidian	1	174	111.64	19425.50
	2	65	142.38	9254.50
	Total	239		
Destroyer	1	174	116.38	20250.50
	2	65	129.68	8429.50
	Total	239		
Score	1	174	126.81	22064.50
	2	65	101.78	6615.50
	Total	239		

TEST STATISTICS

Unit Type	Mann-Whitney U	Wilcoxon W	Z	Asymp. Sig. (2-tailed)
Acolyte	2.708E3	1.793E4	-6.333	.000
Ghoul	4.888E3	2.011E4	-1.621	.105
Fiend	5.436E3	7.582E3	-.512	.609
Gargoyle	2.508E3	1.773E4	-8.154	.000
Wagon	4.868E3	2.009E4	-3.446	.001
Obsidian	4.200E3	1.943E4	-3.251	.001
Destroyer	5.000E3	2.025E4	-1.690	.091
Score	4.470E3	6.616E3	-2.491	.013

TABLE 5.6
MANN-WHITNEY TEST OF DATA CLUSTER 3 & 4

RANK

Unit Type	Data Cluster	No of Case	Mean Rank	Sum of Ranks
Peon	3	1004	802.53	805737.00
	4	549	730.32	400944.00
	Total			
Grunt	3	1004	884.83	888373.00
	4	549	579.80	318308.00
	Total			
Troll	3	1004	783.56	786696.00
	4	549	765.00	419985.00
	Total			
Demolisher	3	1004	744.40	747380.00
	4	549	836.61	459301.00
	Total			
Raider	3	1004	983.04	986976.50
	4	549	400.19	219704.50
	Total			
Tauren	3	1004	778.64	781755.00
	4	549	774.00	424926.00
	Total			
Shaman	3	1004	782.72	785846.00
	4	549	766.55	420835.00
	Total			
Doctor	3	1004	777.82	780931.50
	4	549	775.50	425749.50
	Total			
Spirit Walker	3	1004	939.28	943040.50
	4	549	480.22	263640.50
	Total			
Kodo Beast	3	1004	858.81	862246.50
	4	549	627.39	344434.50
	Total			
Wind Raider	3	1004	766.89	769954.00
	4	549	795.50	436727.00
	Total			
Score	3	1004	796.07	799256.00
	4	549	742.12	407425.00
	Total			

TEST STATISTICS

Unit Type	Mann-Whitney U	Wilcoxon W	Z	Asymp. Sig. (2-tailed)
Peon	2.500E5	4.009E5	-3.120	.002
Grunt	1.673E5	3.183E5	-13.004	.000
Troll	2.690E5	4.200E5	-3.650	.000
Demolisher	2.400E5	7.400E5	-5. 547	.000
Raider	6.873E4	2.197E5	-24.803	.000
Tauren	2.740E5	4.249E5	-1.814	.070
Shaman	2.699E5	4.208E5	-2.445	.014
Doctor	2.748E5	4.257E5	-1.282	.200
Spirit Walker	1.127E5	2.636E5	-19.789	.000
Kodo Beast	1.935E5	3.444E5	-13.874	.000
Wind Raider	2.654E5	7.700E5	-2.557	.011
Score	2.564E5	4.074E5	-2.266	.023

TABLE 5.7

BIVARIATE CORRELATION OF DIFFERENT DATA CLUSTER

DATA CLUSTER 1

		Acolyte	Ghoul	Fiend	Gargoyle	Wagon	Obsidian	Destroyer
Acolyte	Pearson Correlation	1	.092	.083	-.100	.090	.057	.056
	Sig. (2-tailed)		.228	.276	.190	.237	.458	.465
Ghoul	Pearson Correlation	.092	1	-.422**	-.019	.092	.013	.089
	Sig. (2-tailed)	.228		.000	.800	.225	.866	.244
Fiend	Pearson Correlation	.083	-.422**	1	-.273**	-.003	-.137	-.248**
	Sig. (2-tailed)	.276	.000		.000	.968	.072	.001
Gargoyle	Pearson Correlation	-.100	-.019	-.273**	1	-.034	-.238**	-.177*
	Sig. (2-tailed)	.190	.800	.000		.655	.002	.020
Wagon	Pearson Correlation	.090	.092	-.003	-.034	1	.168*	.210**
	Sig. (2-tailed)	.237	.225	.968	.655		.027	.005
Obsidian	Pearson Correlation	.057	.013	-.137	-.238**	.168*	1	.886**
	Sig. (2-tailed)	.458	.866	.072	.002	.027		.000
Destroyer	Pearson Correlation	.056	.089	-.248**	-.177*	.210**	.886**	1
	Sig. (2-tailed)	.465	.244	.001	.020	.005	.000	

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

DATA CLUSTER 2

		Acolyte	Ghoul	Fiend	Gargoyle	Wagon	Obsidian	Destroyer
Acolyte	Pearson Correlation	1	.021	.000	.074	-.208	.154	.109
	Sig. (2-tailed)		.866	.996	.559	.096	.221	.387
Ghoul	Pearson Correlation	.021	1	.065	-.047	-.171	.094	.029
	Sig. (2-tailed)	.866		.604	.709	.174	.457	.816
Fiend	Pearson Correlation	.000	.065	1	-.112	.443**	-.196	-.235
	Sig. (2-tailed)	.996	.604		.373	.000	.118	.059
Gargoyle	Pearson Correlation	.074	-.047	-.112	1	-.126	-.128	-.126
	Sig. (2-tailed)	.559	.709	.373		.319	.310	.319
Wagon	Pearson Correlation	-.208	-.171	.443**	.029	1	-.126	-.128
	Sig. (2-tailed)	.096	.174	.000	.817		.319	.310
Obsidian	Pearson Correlation	.154	.094	-.196	-.515**	-.126	1	.923**
	Sig. (2-tailed)	.221	.457	.118	.000	.319		.000
Destroyer	Pearson Correlation	.109	.029	-.235	-.508**	-.128	.923**	1
	Sig. (2-tailed)	.387	.816	.059	.000	.310	.000	

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

DATA CLUSTER 4

		Peon	Grunt	Troll	Demolisher	Raider	Doctor	Kodo	Wind
Peon	Pearson Correlation	1	-.006	.215**	-.072	.018	-.069	-.008	-.135**
	Sig. (2-tailed)		.893	.000	.091	.680	.107	.847	.002
Grunt	Pearson Correlation	-.006	1	-.026	-.068	.026	.039	-.053	-.146**
	Sig. (2-tailed)	.893		.540	.111	.542	.360	.216	.001
Troll	Pearson Correlation	.215**	-.026	1	-.049	-.034	-.059	-.036	.011
	Sig. (2-tailed)	.000	.540		.250	.433	.171	.405	.803
Demolisher	Pearson Correlation	-.072	-.068	-.049	1	.054	.180**	.303**	-.010
	Sig. (2-tailed)	.091	.111	.250		.207	.000	.000	.815
Raider	Pearson Correlation	.018	.026	-.034	.054	1	-.012	.173**	-.031
	Sig. (2-tailed)	.680	.542	.433	.207		.776	.000	.473
Doctor	Pearson Correlation	-.069	.039	-.059	.180**	-.012	1	-.022	-.109*
	Sig. (2-tailed)	.107	.360	.171	.000	.776		.613	.010
Kodo	Pearson Correlation	-.008	-.053	-.036	.303**	.173**	-.022	1	-.031
	Sig. (2-tailed)	.847	.216	.405	.000	.000	.613		.474
Wind	Pearson Correlation	-.135**	-.146**	.011	-.010	-.031	-.109*	-.031	1
	Sig. (2-tailed)	.002	.001	.803	.815	.473	.010	.474	

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

Data Cluster 5

		Wisp	Archer	Huntress	Glaive	Dryad	Claw	Talon	Chimaeras
Wisp	Pearson Correlation	1	.023	-.007	.073*	.268**	.161**	-.118**	.027
	Sig. (2-tailed)		.501	.837	.031	.000	.000	.000	.424
Archer	Pearson Correlation	.023	1	-.311**	-.151**	-.084*	-.092**	.097**	-.004
	Sig. (2-tailed)	.501		.000	.000	.013	.006	.004	.897
Huntress	Pearson Correlation	-.007	-.311**	1	.173**	-.142**	-.090**	-.112**	-.044
	Sig. (2-tailed)	.837	.000		.000	.000	.008	.001	.197
Glaive	Pearson Correlation	.073*	-.151**	.173**	1	-.037	-.077*	-.063	-.025
	Sig. (2-tailed)	.031	.000	.000		.282	.023	.064	.469
Dryad	Pearson Correlation	.268**	-.084*	-.142**	-.037	1	.393**	-.078*	-.032
	Sig. (2-tailed)	.000	.013	.000	.282		.000	.021	.344
Claw	Pearson Correlation	.161**	-.092**	-.090**	-.077*	.393**	1	-.054	-.021
	Sig. (2-tailed)	.000	.006	.008	.023	.000		.112	.534
Talon	Pearson Correlation	-.118**	.097**	-.112**	-.063	-.078*	-.054	1	.137**
	Sig. (2-tailed)	.000	.004	.001	.064	.021	.112		.000
Chimaeras	Pearson Correlation	.027	-.004	-.044	-.025	-.032	-.021	.137**	1
	Sig. (2-tailed)	.424	.897	.197	.469	.344	.534	.000	

*. Correlation is significant at the 0.05 level (2-tailed).

**. Correlation is significant at the 0.01 level (2-tailed).

5.6.2 GA operators

Roulette wheel selection is used to avoid local minimum and provide chances for weak candidates to crossover. One-point crossover has also been applied. It is easier to observe the development of the next generation.

All the fuzzy integrals are evaluated by different populations (from 5 to 100) and different mutation rates (from 0.01 to 0.20). The finesses are similar by increasing the number of mutation rate and the population (after 50) in Figure 5.7 and Figure 5.8. For the generation, all fuzzy integrals cannot show significant improvement after 100 generation as shown in Figure 5.9.

Therefore, the mutation rate of this research used is 0.05. The GA process is repeated until the fitness value is stable or the maximum generations reached. The maximum generation used is 100 and the population size is 50 in each generation.

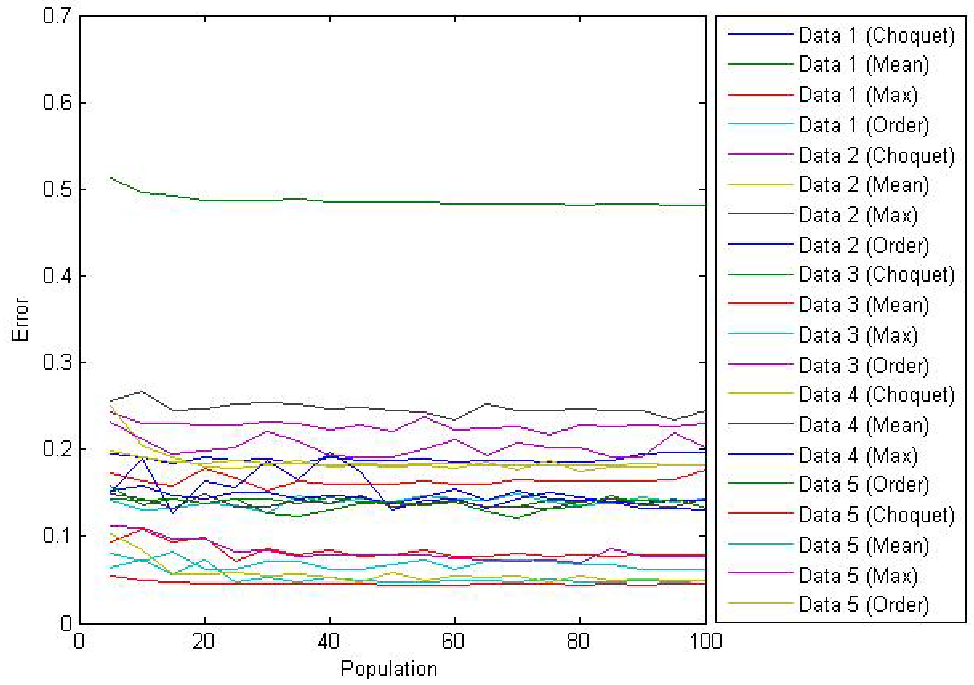


Figure 5.8 Finesses of GA in different population

(Mutation rate: 0.05, Generation: 100)

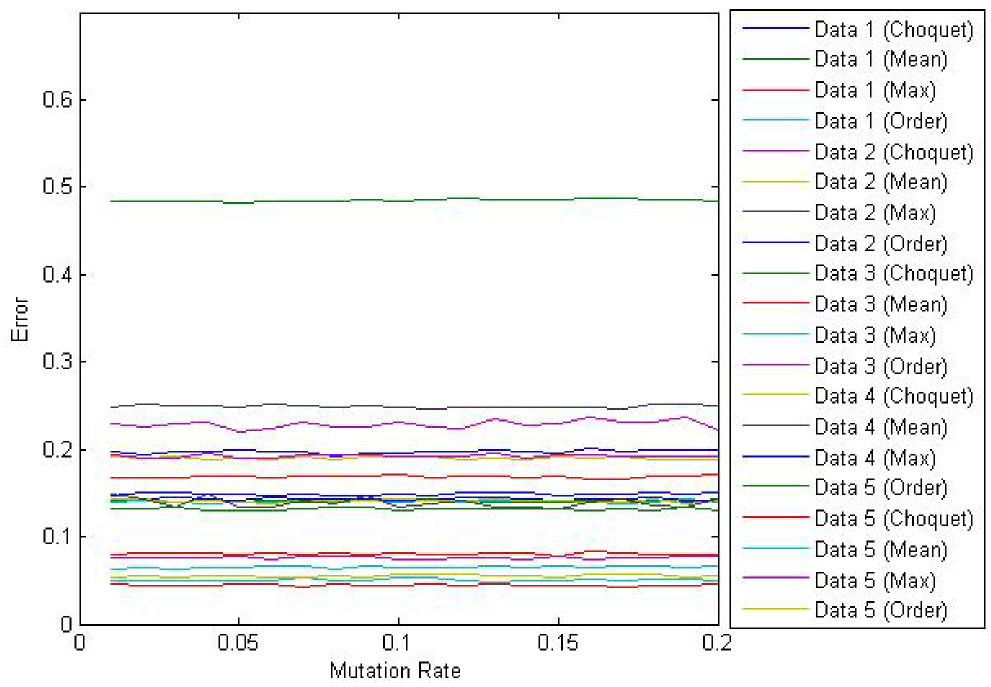


Figure 5.9 Finesses of GA in different mutation rate

(Population: 50, Generation: 100)

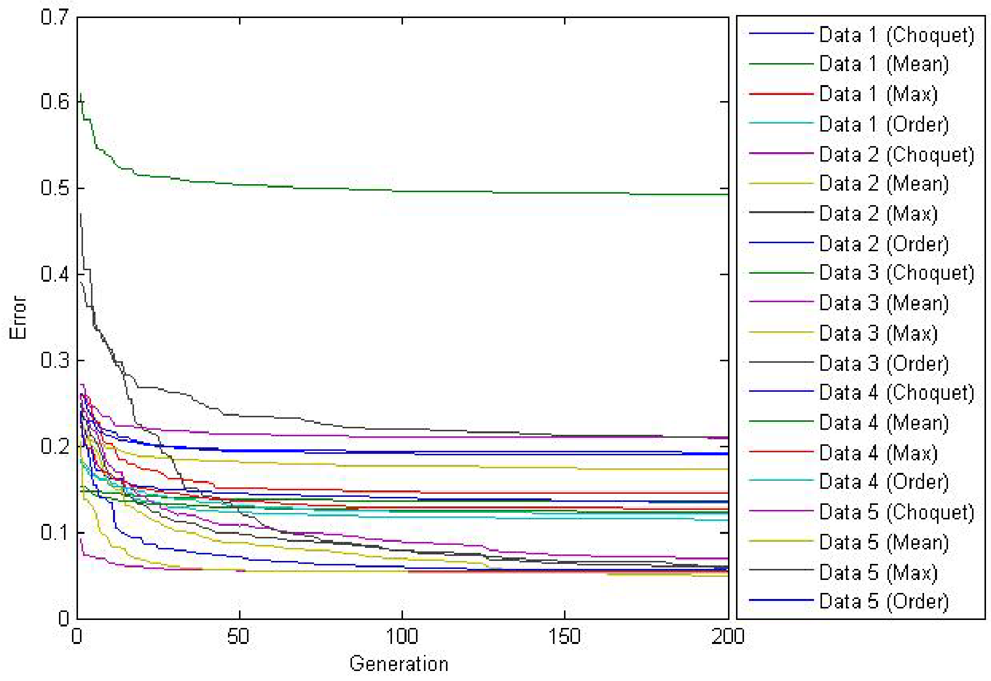


Figure 5.10 Finesses of GA in different generation

(Mutation rate: 0.05, Population: 50)

5.6.3 CMA-ES operators

CMA-ES with rank one and rank mu update is used. All the fuzzy integrals are evaluated by different populations (from 5 to 100) and different iterations (from 50 to 5000). The finesses are similar by increasing the number of the population (after 50) in Figure 5.10. Half of the best populations are then selected for next generation. The initial step size is set to a half of the initialization intervals, i.e., 0.5. For the iteration, all fuzzy integrals cannot show significant improvement after 2000 in Figure 5.11.

Therefore, the population of CMA-ES used is 50. The run is stopped at 2500 function evaluations as the time cost is similar to the GA test.

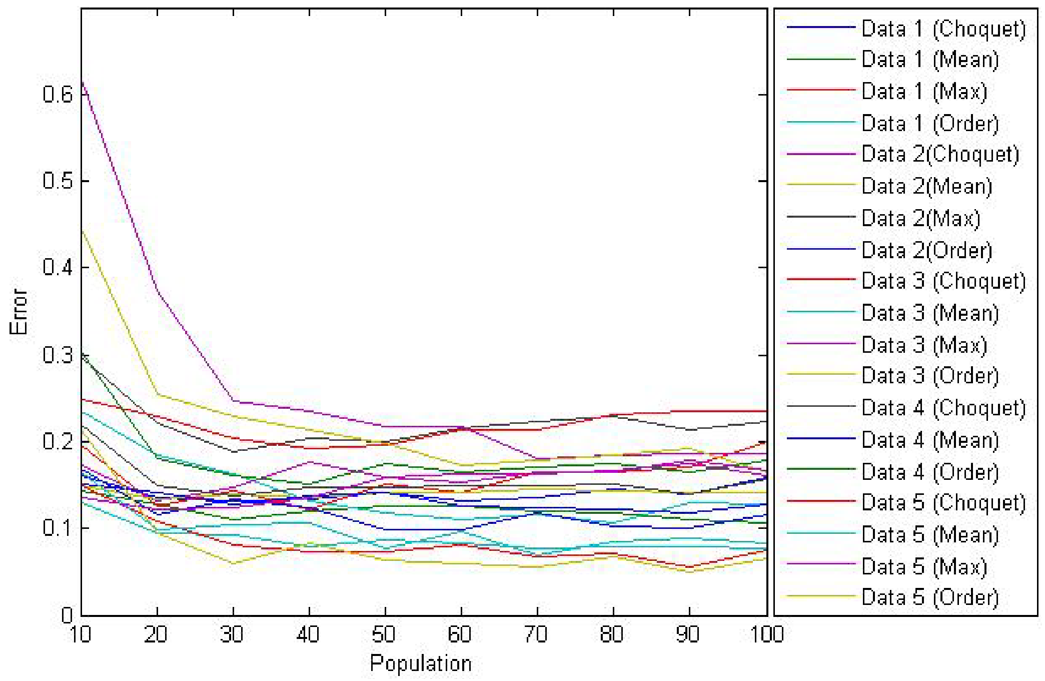


Figure 5.11 Finesses of CMA-ES in different population

(Iteration: 1000)

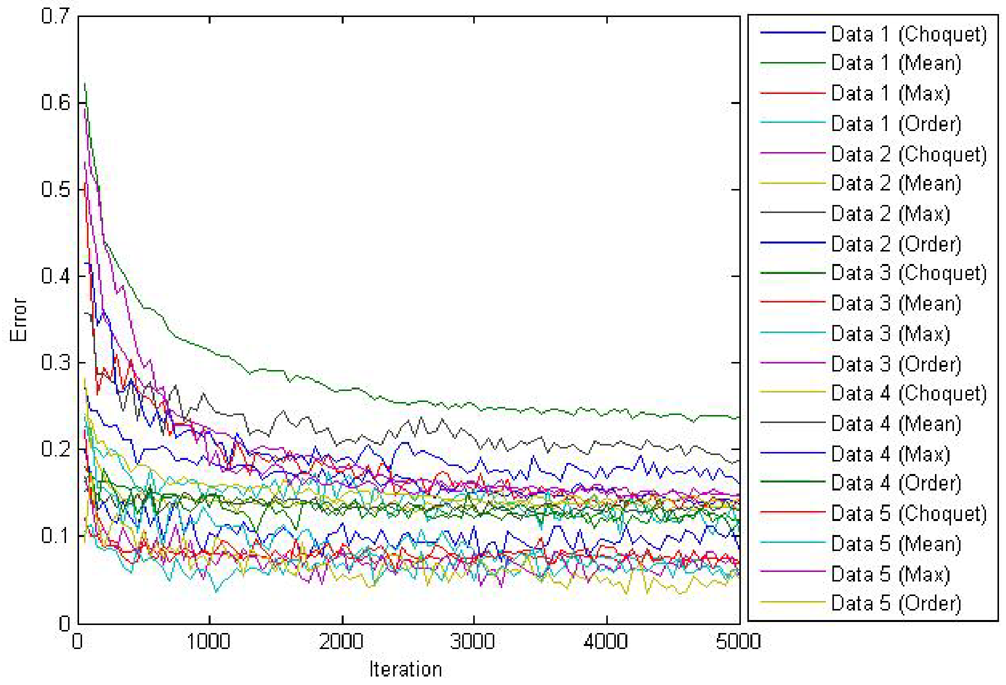


Figure 5.12 Finesses of CMA-ES in different iteration

(Population: 50)

5.6.4 Comparison of GA and CMA-ES

TABLE 5.8
COMPARISON OF TRAINING ERROR IN GA AND CMA-ES

Data Cluster	1	2	3	4	5	Average
GA	0.15225	0.18975	0.16375	0.148	0.0925	0.14925
CMA-ES	0.13075	0.141	0.1129	0.14325	0.089	0.12338

TABLE 5.9
COMPARISON OF TESTING ERROR IN GA AND CMA-ES

Data Cluster	1	2	3	4	5	Average
GA	0.18825	0.26175	0.16875	0.159	0.0925	0.17405
CMA-ES	0.1865	0.284	0.141	0.16275	0.0925	0.17335

Table 5.8 and 5.9 show that the training and testing error of GA and CMA-ES. CMA-ES can obtain better quality solution with better computation efficiency.

GA and CMA-ES begin with a randomly generated population, a fitness value is given to evaluate each chromosome or search point in the population. GA provides the global search by combining two successful chromosomes. This concept is suitable for searching a fuzzy measure as mentioned in Chapter 5.4.2. We can observe the change of fuzzy measure easily. However, the fuzzy measure of next generation is heavily based on their parents. Although mutation and the roulette wheel selection provide the ability to select other possible solutions, the improvement is limited in our data set as the dimension is high and GA cannot provide a guided value in the random walk. Therefore, it will be easily converge to an optimized solution and the fitness is stop. In Figure 5.9, the finesses of all the fuzzy integrals are stopped nearly at the first 20 generations.

CMA-ES provides a de-randomized concept by using a mean updater in global search and covariance matrix updater in local search. These two parameters provide extra information during the search. Hence, the mean updater of selected points provides information sharing mechanism for the convergence. Thus, CMA- obtain solution with better quality and computation efficiency. In our case, the mean updater is affected by the population size. CMA-ES cannot perform very well when the population is less than

$(2^n - 1)/5$ as shown in Figure 5.10, where n is the number of unit type and $2^n - 1$ is the number of subset in fuzzy measure. A large initial population is therefore recommended. CMA-ES Restarts [Auger & Hansen 2005] could also improve this problem by increasing population size in each iteration.

5.6.5 Comparison of different fuzzy integral

We use 70% of the data for training and 30% for testing. 20 cross validation cycles are performed. Error is calculated as the average of the difference between the actual scores in Warcraft III and estimated scores by fuzzy integral. Then experiments are done as follows.

Experiment 1 is designed for testing weighted average. It is similar as Yeung's [Yeung 2004] methodology. We assign a weighting, w, to each unit type. Then, we computed the combined power of unit combination by summing up all the weighted unit proportions together and took an average as shown in Equation (5.23). Neural Network with 3 hidden layers and 9 neurons was also performed for comparison.

$$f(X) = \frac{1}{n}\sum_{i=1}^{n} w_i x_i \tag{5.23}$$

Then, we set up four more experiments by using Choquet Integral, Mean based Fuzzy Integral, Mas based Fuzzy Integrals and Order based Fuzzy Integral respectively. GA is used to obtain the fuzzy measure. Another four sets of experiment is done by using CMA-ES. The average of training error, testing error and training time of our new fuzzy integral and CI are summarised in Table 5.10, 5.11 and 5.12 and provided the following experimental conclusions.

TABLE 5.10

COMPARISON OF TRAINING ERROR IN DIFFERENT FUZZY INTEGRAL

Data Cluster	1	2	3	4	5	Average
Weighted Average	0.512	0.576	0.799	0.497	0.463	0.569
Neural Network	0.152	0.214	0.047	0.146	0.057	0.123
Choquet Integral (GA)	0.181	0.200	0.456	0.187	0.054	0.215
Mean based FI (GA)	0.138	0.181	0.078	0.133	0.061	0.118
Max based FI (GA)	0.16	0.245	0.048	0.14	0.197	0.158
Order based FI (GA)	0.130	0.133	0.073	0.132	0.058	0.105
Choquet Integral (CMA-ES)	0.140	0.146	0.2535	0.146	0.067	0.150
Mean based FI (CMA-ES)	0.127	0.133	0.074	0.136	0.066	0.107
Max based FI (CMA-ES)	0.140	0.195	0.058	0.167	0.167	0.145
Order based FI (CMA-ES)	0.116	0.090	0.0661	0.124	0.056	0.090

TABLE 5.11

COMPARISON OF TESTING ERROR IN DIFFERENT FUZZY INTEGRAL

Data Cluster	1	2	3	4	5	Average
Weighted Average	0.505	0.683	0.752	0.498	0.490	0.586
Neural Network	0.166	0.228	0.064	0.148	0.046	0.130
Choquet Integral (GA)	0.198	0.328	0.461	0.191	0.048	0.245
Mean based FI (GA)	0.185	0.237	0.067	0.145	0.054	0.219
Max based FI (GA)	0.187	0.234	0.064	0.15	0.212	0.157
Order based FI (GA)	0.183	0.248	0.083	0.150	0.056	0.167
Choquet Integral (CMA-ES)	0.203	0.337	0.312	0.177	0.068	0.245
Mean based FI (CMA-ES)	0.181	0.290	0.081	0.155	0.080	0.137
Max based FI (CMA-ES)	0.178	0.248	0.077	0.168	0.168	0.169
Order based FI (CMA-ES)	0.184	0.261	0.094	0.151	0.054	0.144

TABLE 5.12

COMPARISON OF TRAINING TIME IN DIFFERENT FUZZY INTEGRAL

Data Cluster	1	2	3	4	5
Choquet Integral	22.906s	10.644s	183.96s	64.723s	96.047s
Mean based FI	1712s	1572s	34450s	3727s	4014s
Max based FI	1698s	1474s	32340s	3511s	3973s
Order based FI	23.809s	10.960s	190.26s	67.722s	99.701s

1. Improvement was calculated by percentage change of the average error in the five data sets. For example, the improvement of the training error by Order based FI (CMA-ES) was 84.2% ((0.56-0.09)/0.56). Compared with the weighted average model, all fuzzy integrals with CMA-ES show a decrease in both training and testing error on all the five data clusters as shown in Table5.13.Therefore, fuzzy integral may have a better ability on predicting the power of unit combination in Warcraft III data. In another word, interactions exist in these data.

TABLE 5.13
IMPROVEMENT OF FUZZY INTEGRAL (COMPARED WITH WEIGHTED AVERAGE)

Data Cluster	Training Error	Testing Error
Choquet Integral (CMA-ES)	73.6%	58.2%
Mean based FI (CMA-ES)	81.2%	76.6%
Max based FI (CMA-ES)	74.5%	71.2%
Order based FI (CMA-ES)	84.2%	75.4%

2. Mean based FI presented a better result in all data as it considered more options in set selection. The performance was stable among all five sets of data. The average training errors were 0.118 in GA and 0.107 in CMA-ES. Comparing with Choquet Integral (CMA-ES), Mean base FI (CMA-ES) has 28.6% ((0.15-0.107)/0.15) improvement in training error and 44.0% ((0.245-0.137)/0.245) improvement in testing error.
3. The main weakness of Mean based FI is time consuming as shown in Table 5.12. The training time of Mean based FI is 40 to 180 times longer than CI. Mean based FI needs to search for the whole power set of unit types, $P(X)$ while, the subset selection process of CI is directed by Equation (5.19). There is no additional search.
4. Training time of Order based FI remains the same as CI because it is also directly addressing the required subset by Equation (5.20). Thus, no additional searching is required. The training and testing error were similar to Mean based FI. Compared with Mean based FI (CMA-ES), Order based FI (CMA-ES) has 15.9% improvement in training error and difference of testing error is 5.1% as shown in Table 5.14. Although the test error of Order based FI (CMA-ES) was not as good as Mean based FI (CMA-ES), Order based FI (CMA-ES) provide an efficient evaluation. It is 40 to

180 times faster than Mean based FI (CMA-ES) and it is more importance in RTS game.

TABLE 5.14
IMPROVEMENT OF ORDER BASED FI (CMA-ES)

Data Cluster	Training Error	Testing Error
Weighted Average	84.2%	75.4%
Choquet Integral (CMA-ES)	66.7%	41.2%
Mean based FI (CMA-ES)	15.9%	-5.1%
Max based FI (CMA-ES)	37.9%	14.5%
Neural Network	26.7%	-10.1%

5. Testing error of Order based FI is similar to training error. It shows the generalization ability of order based FI is good.
6. Comparing with fuzzy measure with GA, fuzzy measure with CMA-ES could obtain a better solution in most of the case. According to Table 5.8, CMA-ES has 17% ((0.14925-0.12338)/0.14925) improvement in average training error. It shows the improvement in all the testing examples. According to Table 5.9, CMA-ES has 0.4% ((0.17405-0.17334)/0.17405) improvement in average testing error.
7. Comparing with neural network (NN), Order based FI (CMA-ES) has 26.7% of improvement in training error and 10.7% difference in testing error. Although, testing error of Order based FI (CMA-ES) was not as good as neural network, there are two main reasons to use the fuzzy approach instead of ANN. Firstly, ANN is a black box optimization. The weightings of ANN may be difficult to interpret by game developer and involves the human reasoning. In contrast, the fuzzy measure stated all the unit combination in the subset. It is easier for a player / AI developer to understand the relationship of interactions. Secondly, the ANN required data for training. It is hard to gather the data before the game is launched. Therefore, it is impossible to use ANN at an early stage of game AI development. Instead, fuzzy measure could be assigned by game designers before the game is launched. It could be used without any training data.

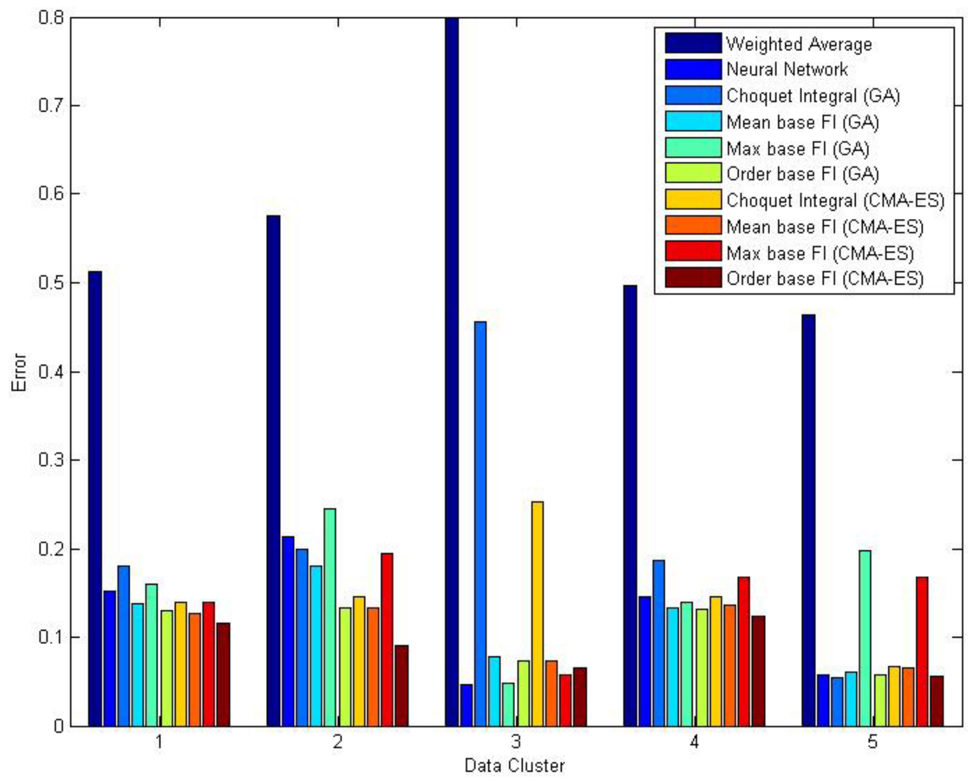


Figure 5.13 Training error of different fuzzy integral

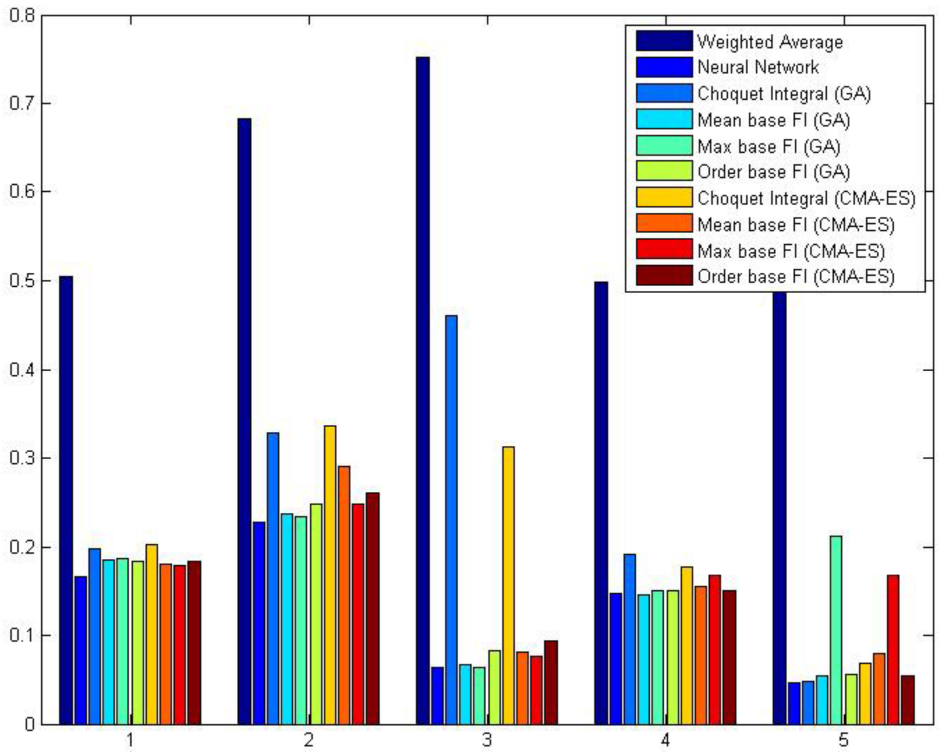


Figure 5.14 Testing error of different fuzzy integral

5.6.6 Observe the usage count of subsets in different fuzzy integrals

If the subset is selected by fuzzy integral during the training process once, the number of counts will be increased by one. By observing table IX, the number of counts in subset selection with more elements drops sharply in both CI and Order based FI. Most of the unit type subsets cannot be trained by our replay data as professional players prefer using an unbalanced army with 3 to 4 different unit types in each battle instead of developing a balanced army with many different kinds of units.

The main difference between Choquet Integral and the Order based fuzzy integral is the way in set selection as shown in Equation (5.19), $\min_{k\in K_i} f(x_k) > \max_{k\in \overline{K_i}} f(x_k)$ and Equation (5.20), $\max_{k\in K_i} f(x_k) < \min_{k\in \overline{K_i}} f(x_k)$. The number of sets is the same in each integral calculation but they select different sets. Data cluster 4 is used to show a concept of subset selection in Table 5.14.

Set selection of Order based FI, equation (5.20), select the unit type with the smallest proportion first, i.e., {1} (labour) which then shows the unit production sequence in Warcraft III. The subset {1, 2} is much higher than other sets that contain two unit types and then so on. Super-additive, such as {1,2} and sub-additive, such as {1,3} could be easily found in the learned fuzzy measure. Importance unit combination, such as, {1,2,4,5} could be identified by its highest fuzzy measure. It helps the analyser to identify which unit type is better at unit production development.

In contrast, CI could not capture the production sequence. It is assumed that every combination of unit types is possible to be produced in the game play, this is not reasonable in RTS games as the ultimate unit costs many resources and needs longer time to unlock. Some sets that represent the ultimate unit, such as {8}, {2,8} {4,8} , are not produced alone in RTS game but they are used in CI.

TABLE 5.15
FUZZY MEASURE IN DATA CLUSTER 4

Fuzzy Measure	Order based FI		Choquet Integral	
	Weighting	No of count	Weighting	No of count
{1}	0.196875	412	0.5	23
{2}	0. 32751	20	0.617188	343
{3}	0.34375	23	0.929688	37
{4}	0.046875	14	0.742188	56
{5}	0.421875	3	0.867188	0
{6}	0.765625	18	0.992188	6
{7}	0.25	0	0.476563	0
{8}	0.492188	0	0.789063	37
{1,2}	0.67821	297	0.679688	241
{1,3}	0.539063	88	0.796875	14
{2,3}	0.476563	3	0.992188	84
{1,4}	0.53125	57	0.851563	1
{2,4}	0.46875	2	0.96875	80
{3,4}	0.19532	0	0.976563	12
{1,5}	0.679688	4	0.683594	0
{2,5}	0.374692	0	0.984375	1
{3,5}	0.039063	2	0.898438	0
{1,6}	0.828125	63	0.746094	0
{2,6}	0.65625	2	0.101563	36
{3,6}	0.875	1	0.976563	1
{4,6}	0.375	11	0.796875	5
{1,7}	0.117188	4	0.488281	0
{4,7}	0.359375	0	0.609375	1
{1,8}	0.539063	15	0.289063	8
{2,8}	0.671875	0	0.375	39
{4,8}	0.539063	0	0.71875	6
		…		
{1,2,3}	0.453125	110	0.992188	116
{1,2,4}	0.625	63	0.859375	48
{1,3,4}	0.015625	14	0.28125	4
{2,3,4}	0.380211	0	0.882813	24
{1,2,5}	0.632813	5	0.40625	2
{1,3,5}	0.367188	2	0.79296	0
{2,4,5}	0.890625	0	0.0625	3
{1,2,6}	0.625	45	0.921875	33
{1,3,6}	0.09375	6	0.839844	0
{2,3,6}	0.039063	1	0.757813	8
{1,4,6}	0.578125	49	0.90625	1
		…		
{1,2,3,4}	0.867188	28	0.984375	31
{1,2,3,5}	0.023438	1	0.890625	2

{1,2,4,5}	0.992188	1	0.53125	1
{1,3,4,5}	0.804688	1	0.710938	0
{1,2,3,6}	0.507813	16	0.390625	14
{1,2,4,6}	0.164063	77	0.992188	71
		…		
{1,2,3,4,5}	0.695313	1	0.164063	1
{1,2,3,4,6}	0.828125	17	0.945313	20
{1,2,3,4,7}	0.023438	1	0.921875	1
{1,2,4,5,7}	0.210938	1	0.8125	1
{1,2,4,6,7}	0.335938	1	0	1
{1,2,3,6,8}	0.96875	1	0.515625	0
{1,2,4,6,8}	0.796875	1	0.664063	1
{2,3,4,6,8}	0.257813	0	0.265625	1
		…		
{1,2,3,4,6,8}	0.679688	1	0.820313	1
		…		
{1,2,3,4,5,6,7,8}	0.03906	0	0.523438	0

1 – Peon (Labour), 2 – Grunt (Basic Melee unit), 3 – Demolisher (Advance Siege unit), 4 – Raider (Advance Melee Unit), 5 Shaman (Advance Magic Unit), 6 – Spirit Walker (Advance Support Unit), 7 – Kodo Beast (Advance Support Unit).), 8 – Wind Rider (ultimate Air Unit) Number of count of Order based FI and CI = 0 are omitted in this table.

5.7 Summary

In this chapter, we have developed an evaluation model of unit combinations in RTS games which uses the concepts of fuzzy measure and fuzzy integral. GA and CMA-ES algorithm are applied to learn fuzzy measure from collected replay data of Warcraft III. Based on the obtained fuzzy measure, fuzzy integral is then used to compute the overall power of combined unit types. Since classical Choquet integral does not consider the characteristics of unit production ordering in RTS games, i.e., basic units must be produced before advanced units, we defined a new type of fuzzy integral called ordered-based FI. This new integral can evaluate the unit combinations and analyse ordering data well. Experiments are carried out to compare ordered-based FI, mean-based and CI integrals. Different interactions such as super-additive or sub-additive can be observed. For the scores estimation, Order based FI is 80% better than weighted average and 41 % better than Choquet Integral. The future work will focus on applying the result of feature interaction to the potential field technique.

Chapter 6
Optimal path determination using directional based fuzzy integral and potential field

Unit formation planning and target of attack is the cores of micro control in real time strategy (RTS) game. It is more complicated than macro control, such as building and unit production sequence. It consists of a great quantity of possibility. Multiple targets and the intransitive superiority of unit formation lead the micro-control to remain a problem. Traditional tree searching or A* searching is unable to handle these two properties. There are too many weightings and each of them will interact with the others. In this chapter, we applied potential field, fuzzy measure and integral to solve the micro control problem. Potential field is suitable for complicated and various environment with multiple targets. However, it does not consider non-additive property. We integrated it with fuzzy measure and integral to extend simple additive property to non-additive property. It provides the ability to handle interaction among different targets. We have also proposed a new fuzzy integral, directional based fuzzy integral, to support the flank and diversion attack in micro control. It can avoid the trap and siege of enemy units, and maximize the combined power of player units.

6.1 Introduction

As stated in chapter 5, control of RTS game can be classified into two types. We have applied fuzzy measure and integral to improve the macro control problem, i.e., the unit production planning. In this chapter, we extended the usage of trained fuzzy measure and applied to the micro control, i.e., tactics. Micro control involves planning of the unit movement and control. We combined the potential field and fuzzy integral to solve planning for the unit formation and target of attack. Hence, by using the proposed

Directional based fuzzy integral. Detail micro control could be performed in potential field, such as flank and diversion maneuver. It improves the capabilities of risk analysis in potential field. We simulated our experiment in Warcraft III. Detail steps and experimental result are included in this chapter.

This chapter is organized in seven main sections. This section is an introduction. The next section is the problem statement. Section three is the proposed min-max strategy. Section four explains the steps of combined Choquet integral (CI) and potential field. Section five explains proposed Directional based fuzzy integral. Section six explains our experimental design and the conclusions are presented in section seven.

6.2 Micro control problem in RTS game

More and more researchers such as [Buro 2004], [Lucas 2009] and [Alexander 2007], are interested in RTS game and have recently generated many important outputs in the community. They shared the same view on RTS game environment. It is hostile and dynamic. Counter maneuvers need to be implemented under uncertainty and time pressure. Potential Field is designed for searching the next movement location in multiple targets and various environments under the time pressure. It is suitable for RTS game environment. The classic potential field consists of two virtual forces [Weijun 2010, Yin 2008]. One is attractive potential force, $U_{att}(q) = \frac{1}{2}\xi p_g^2$, where ξ is a positive constant scaling factor and $p_g = \left\| q - q_g \right\|$ is the Euclidean between the object and the target. Another force is repulsive potential force, $U_{rep}(q) = \frac{1}{2}\eta(\frac{1}{p(q)} - \frac{1}{p_0})^2$ where η is a positive constant scaling factor, $p(q)$ is the minimum distance from the object to the obstacle and p_0 is a positive constant that presents the influence distance of the obstacles. Multi-agents potential fields of Hagelback [Hagelback 2008, Johansson 2008] and Johan [John 2009] showed the importance to micro control and tactics development. Their experiment result had been proven in the event, such as Open Real Time Strategy (ORTS) AI competitions. They showed a possible solution on micro control in RTS game. However, potential field is normal additive. Therefore, it may not have the ability to handle interaction among different targets, especially the sub additive problem, i.e., $f(X) \le \sum_{i=1}^{n} f(x_i)$. Details unit maneuver, such as flank and diversion cannot be implemented easily.

Many researches have been done to prove the ability of interaction handling [Sugeno 1985, Murofushi 2000, Peter 1999]. We have also implemented fuzzy integral to evaluate the combined power of units. It is an efficient way to handle the interaction problem in RTS game. However, the aggregator of fuzzy integral has not been investigated and visualized very well. Our proposed idea here is to measure the opponent power by the fuzzy integral and provide a planning model for micro control.

6.3 Min-Max strategy

We proposed a Min-Max strategy for the micro control planning. The main ideas of this model are to minimum the interaction of enemy units and maximize the interaction of player units. For minimizing the enemy interaction, area of diversion and flank maneuver are to be predicated. Diversion is used to block the area of enemy units. Flank maneuver is used to evade the high risk area and avoid the siege of enemy units. The flow of the Min-Max strategy is shown in Figure 6.1.

First, fuzzy measure of enemy and player is trained individually by GA as shown in label 1. The working step is similar to chapter 5.4. All the cases are clustered by the four races. Only the unit combination and scores are be extracted to evaluate the combined power. The reason of this filtering is to simplify the training procedure. Micro control in RTS game is highly dynamic and consists of lots of objects. The possibilities are numerous. If we involve the terrain information or other game rules in the training, the subset of the fuzzy measure will be increased exponentially. Therefore, we assume interaction only occurs in the unit type. The effect of terrain information and other games rules is concerned in the potential field only.

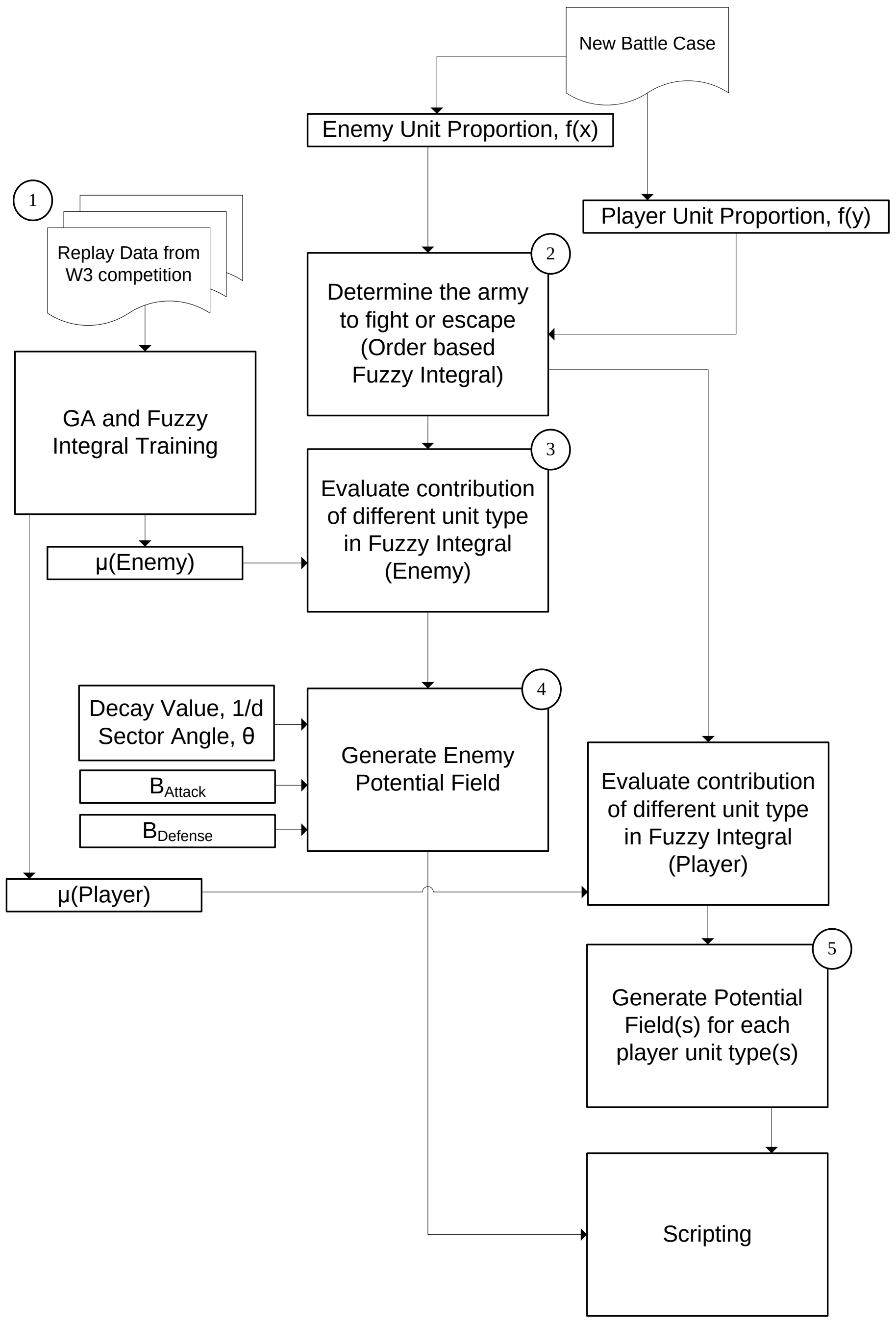

New Battle Case
Enemy Unit Proportion, f(x)
Player Unit Proportion, f(y)
1
Replay Data from W3 competition
2
Determine the army to fight or escape (Order based Fuzzy Integral)
GA and Fuzzy Integral Training
3
Evaluate contribution of different unit type in Fuzzy Integral (Enemy)
μ(Enemy)
4
Decay Value, 1/d Sector Angle, θ
B Attack
B Defense
Generate Enemy Potential Field
Evaluate contribution of different unit type in Fuzzy Integral (Player)
μ(Player)
5
Generate Potential Field(s) for each player unit type(s)
Scripting

Figure 6.1 Overview of the Min-Max strategy

Second, when there is a new battle case, player and enemy unit combination are extracted. Order base fuzzy integral is used to compare the combined power by the fast aggregator as shown in label 2. If the combined power of player is low, the army should escape and search for another unit combination as stated in bottom-up strategy planning in chapter 5.2. If the combed power of player is high, the min-max strategy should be preformed to generate the decision of target and unit formation of the player.

Then, the contribution of subset in fuzzy integral is investigated as shown in label 3 and becomes a point of interest in potential field. Potential field of enemy and player are generated individually. Then the potential fields are combined with the effect of terrain and game rules as shown in label 4. Two potential fields are compared. Finally the target of attack and the path are generated by the scripting.

6.4 Learning fuzzy measure by evolution strategy

Usually, the charge in potential field is provided by expert or game designer [Hagelback 2008, Johansson 2008]. However, it is difficult to assign all the weighting and interaction of units one by one. Learning the charge by simulation is also a time consuming task. Therefore, we use the Warcraft III replay and learn the interaction first and assign the result into the potential field.

6.4.1 Data collection and preprocessing

All the replays of data cluster 3 and 4 that were stated in chapter 5 are selected. They are the competition of two races. The race of enemy is orc while the race of player is human. The number of cases is 1553. For each replay, r, unit proportions and scores are extracted. In this chapter, $X = \{x_1, x_2, ... x_n\}$ is used to represent the enemy unit proportion, i.e., orc. $Y = \{y_1, y_2, ... y_n\}$ is used to represent the player unit proportion, i.e., human. n is total number of unit type and $n = 12$. These quantities here are measured by the amount of resources used instead of the physical quantity count. Then it is normalized by a normalization function f .The total score defined as $Score_{WarcraftIII,Player}(r)$ and $Score_{WarcraftIII,Player}(r)$ is extracted from each replay, r.

6.4.2 Setting and operators of evolution strategy

Fuzzy measure is used to represent the combined power. Fuzzy measure of enemy is defined as a mapping: $\mu : P(X) \rightarrow [0,1]$, where $P(X)$ is the power set of X, i.e., all the ($2^n - 1$) subsets of X. The fuzzy measure of player is defined as a mapping: $\mu : P(Y) \rightarrow [0,1]$ with the same condition. $\mu(X)$ and $\mu(Y)$ are trained separately by GA or CMA-ES as shown in Figure 6.2. The steps are the same as Chapter 5.4.

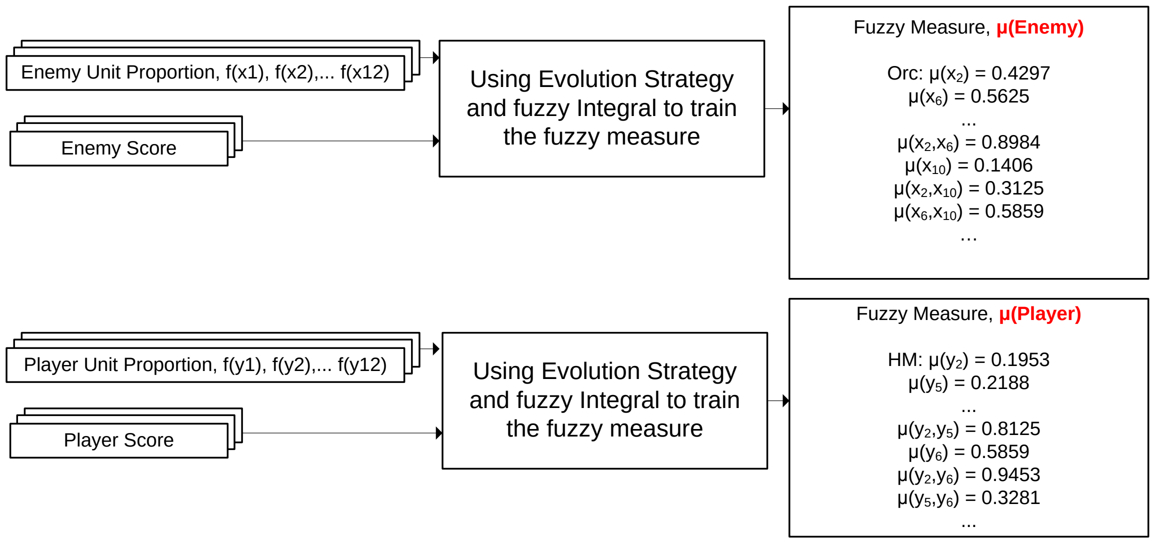


Figure 6.2 Learn the fuzzy measure by GA

For GA, a number of chromosomes are generated randomly. Each chromosome represents a fuzzy measure with $2^n - 1$ subset. Fuzzy measure is combined with unit proportion by using fuzzy integral. Estimated score is calculated and compared with the real score case by case. Finally, the sum of difference, as shown in Equation (5.4) becomes the fitness value. The population is set as 50, the mutation rate is 0.05 and generation is 100. Roulette wheel selection is used. 20 cross validation cycle have been run.

For CMA-ES, 50 search points are generated randomly in a $2^n - 1$dimensional matrix. Again, their fitness is computed by fuzzy integral and compared with the real game data.

25 better solutions are selected to computer mean vector, step size and covariance matrix. The search points are updated by these three parameters for the next generation. After 2500 iteration, the fuzzy measures will be used for testing.

All the cases, i.e., 1553 are used for training. The fuzzy measures of GA and CMA-ES are compared. The fuzzy measures with the lower training error are selected to generate the potential field.

6.4 Combining Choquet Integral and potential field

6.4.1 Evaluating the contribution of each unit type

We have obtained the fuzzy measure of enemy, $\mu(X)$, by GA and Choquet Integral. Then the contribution of each unit type in Choquet Integral should be investigated and defined as a point of interest (POI) in potential field. The purpose is to determine which type of enemy unit contributes the most in the Choquet Integral. Then we should attack it first and minimize the interaction effect of enemy.

We demonstrate our above idea by using the following example. Suppose the enemy army is combined by three unit types, i.e., Grunt (x_2), Raider (x_6) and SP-Walker (x_{10}). The unit proportion is given as follows, $f(x_2) = 0.5$, $f(x_6) = 0.3$ and $f(x_{10}) = 0.2$. Sorting is performed and suppose $a_1 \le a_2 \le a_3$, thus, $a_1 = f(x_{10}) = 0.2$, $a_2 = f(x_6) = 0.3$ and $a_3 = f(x_2) = 0.5$. The Choquet Integral is computed as follows.

$$
\begin{aligned}
&(c)\int f(X)d\mu \\
&= (c)\int f(x) \circ \mu(X) \\
&= \sum_{i=1}^{3}(a_i - a_{i-1}) \cdot \mu(x \mid f(x) \ge a_i) \\
&= f(a_1)\mu(a_1, a_2, a_3) + [f(a_1) - f(a_2)]\mu(a_2, a_3) + [f(a_3) - f(a_2)]\mu(a_3)
\end{aligned}
$$

The contribution of a_1 , a_2 and a_3 is distributed in different places. It is difficult to present in potential field. Therefore, we converted Choquet Integral into another form as shown in Equation (6.1). $C(a_i)$ is used to represent the contribution of unit type a_i. Its equation is shown in Equation (6.2).

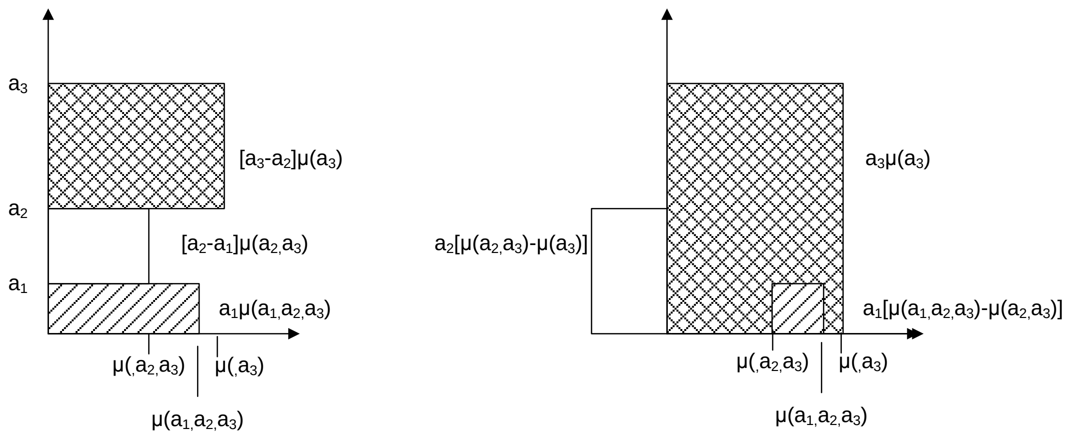
$$\sum_{i=1}^{n}(a_i)\cdot[\mu(x\mid f(x)\geq a_i)-\mu(x\mid f(x)\geq a_{i+1})] \tag{6.1}$$

$$C(a_i)=(a_i)\times\left[\mu(x\mid f(x)\geq a_i)-\mu(x\mid f(x)\geq a_{i+1})\right] \tag{6.2}$$

Now, the Choquet Integral of the example is converted as follows.

$$(c)\int f(X)d\mu$$
$$=a_1[\mu(a_1,a_2,a_3)-\mu(a_2,a_3)]+a_2[\mu(a_2,a_3)-\mu(a_3)]+a_3\mu(a_3)$$

Therefore, $C(a_1)=a_1\times\left[\mu(a_1,a_2,a_3)-\mu(a_2,a_3)\right]$, $C(a_2)=a_2\times\left[\mu(a_2,a_3)-\mu(a_3)\right]$ and $C(a_3)=a_3\times\mu(a_3)$.



(a) (b)

Figure 6.3 Choquet Integral with non monotonic fuzzy measure

The left side of Figure 6.3 shows the visual meaning of Choquet Integral. The right hand side shows the visual meaning of another form, Equation 6.1. The area represents the contribution of different unit types. As the fuzzy measure is non-monotonic, the contribution of unit type, $C(a_i)$, may be negative. For example, if $\mu(a_2, a_3) \le \mu(a_3)$, then $C(a_2) = \mu(a_2, a_3) - \mu(a_3)$ will be negative. It is presented as the white area in Figure 6.3(b). Therefore, the weak interaction of enemy could be identified.

6.4.2 Assigning charge to potential field

Potential field of enemy

The contributions of enemy unit type are assigned to potential field with its coordinate. This charge is in a ring around the object with a radius. All the potential is summed up: the highest potential, and the highest contribution in the CI. It is the most attractive destination. When the player attacks this unit type, it can minimize the enemy interaction in a short time.

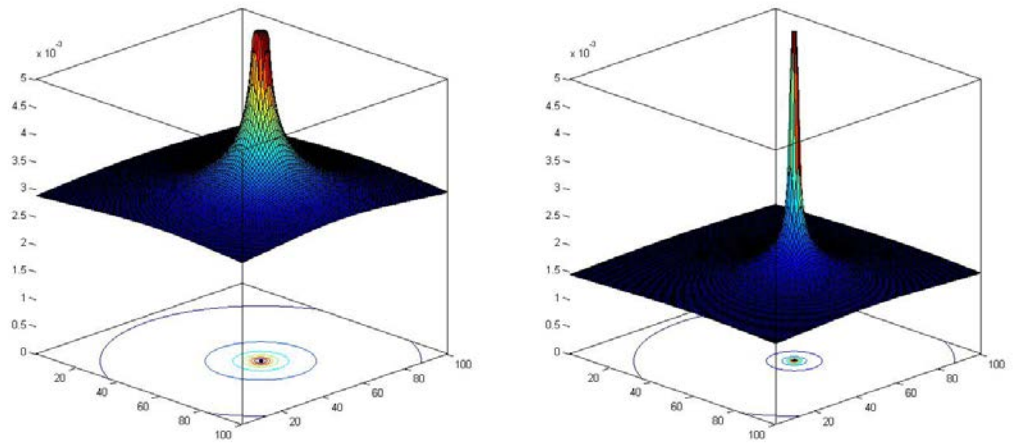
A potential field with size $S \times S$ is generated. The equation of each point in potential field is shown in the following Equation (6.3). The contribution of each unit type, $C(a_i)$, is used as a point of interest (POI). Its coordinate, P_i, is extracted in the battle. A decay function, $\varphi(P(x,y), P_i)$, is added to each POI. If the point, $P(\mathrm{x,y})$ is far away from the P_i, the power will be decreased sharply. $(D \times \log(\sqrt{[P(\mathrm{x}) - P_i(\mathrm{x})]^2 + [P(\mathrm{y}) - P_i(\mathrm{y})]^2}))^{-1}$. Euclidean distance is used for calculation. D is a weighting to control area of the affected. The larger the D, the smaller affected area. Two examples are shown in Figure 6.4 for $D = 0.5$ and $D = 2$. Figure 6.5 shows the potential field of example in Chapter 6.4.1. The highest value in the potential field represents the greatest contribution in the Chqouet Integral. Player should attack this point. Therefore, combined power of enemy could be minimizing in a short time.

$$P(\mathrm{x,y}) = \sum_{i=1}^{i=n} C(a_i) \times \varphi(P(\mathrm{x,y}), P_i) \tag{6.3}$$

where n is the total number of unit type, P_i is the coordinate of a_i

$$\varphi(P(\mathrm{x,y}), P_i) = (D \times \log(\sqrt{[P(\mathrm{x}) - P_i(\mathrm{x})]^2 + [P(\mathrm{y}) - P_i(\mathrm{y})]^2}))^{-1}$$

D is the weighting to control area of affected, $D = 1$



(a) $D = 0.5$ (b) $D = 2$

Figure 6.4 Effect of the decay function, $\varphi(P(\mathrm{x,y}), P_i)$

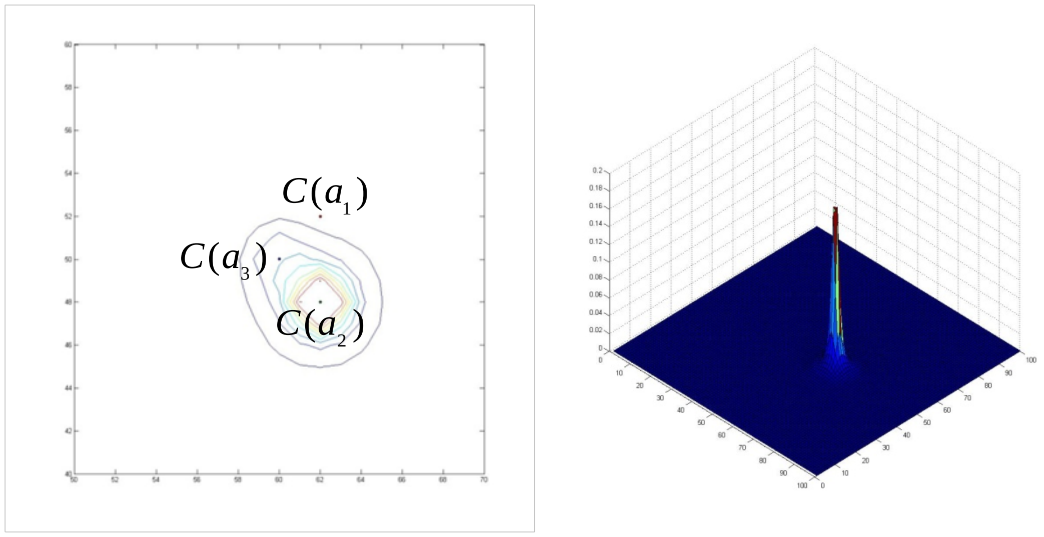


Figure 6.5 Potential field of enemy ($a_3 = 0.5$, $a_2 = 0.3$, $a_1 = 0.2$ and $\mu(X)$ **)**

Potential field of player

We have generated the potential field of enemy. Then, new potential field should be generated to each player unit type. The purpose is to determine which player unit types should get closer together for cooperation. In other words, it is the formation of player unit.

For each player unit type p, the contributions of other unit type are determined into a new potential field. Its own contribution, $C(b_p)$, is not concerned. It is because we are concerning the movement of unit type p. The potential field and scripting will be confused if its own value is added. The equation is shown in (6.4). A progressive function, $\phi(P_p, P_q)$, is added. Figure 6.6(a) shows the original potential field. If the unit is far away from the others, the potential will be increased as shown in Figure 6.6(b). The potential of teammate become more attractive. Therefore, the unit will not get away and the enemy will not break the formation easily.

$$P(\mathrm{x,y}) = \sum_{i=1}^{i=n} C(a_i) \times \varphi(P(\mathrm{x,y}), P_q) \times \phi(P_p, P_q) \qquad (6.4)$$

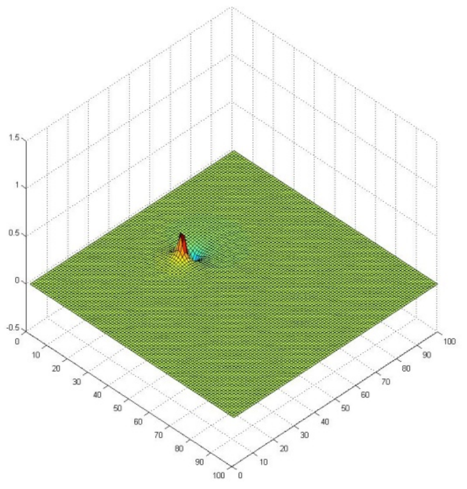
where n is the total number of unit type,

P_p is the coordinate of b_p , P_q is the coordinate of other unit type

$$\varphi(P(\mathrm{x,y}), P_q) = (D \times \log(\sqrt{[P(\mathrm{x}) - P_i(\mathrm{x})]^2 + [P(\mathrm{y}) - P_i(\mathrm{y})]^2}))^{-1}$$

$$\phi(P_p, P_q) = \sqrt{[P_p(\mathrm{x}) - P_q(\mathrm{x})]^2 + [P_p(\mathrm{y}) - P_q(\mathrm{y})]^2}$$

D is the weighting to control area of affected, $D = 1$


(a)

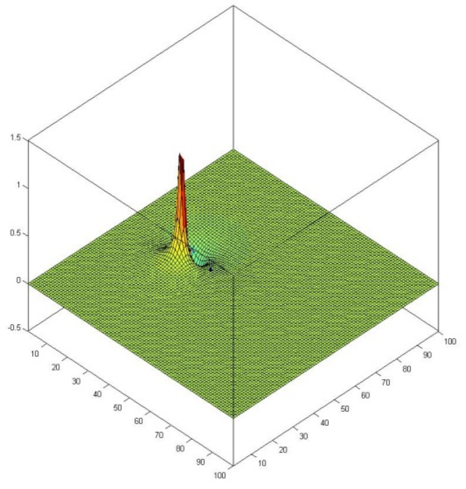

(b)

Figure 6.6 Effect of progressive function, $\phi(P_p, P_q)$

We demonstrate our above idea by using the following example. Suppose the player army is combined by Footman (y_2), Priest (y_5) and Sorceress (y_6). Unit proportion is stated as $b_3 = f(y_2) = 0.5$, $b_2 = f(y_5) = 0.3$ and $b_1 = f(y_6) = 0.2$ where $b_1 \leq b_2 \leq b_3$. The potential field of b_3, b_2 and b_1 is computed by Equation (6.4) and shown in Figure 6.7 , 6.8 and 6.9. Positive and negative contributions are shown in the figures. Player should go to the point with highest potential. It is the most attractive destination to maximize the player interaction. Therefore, greater combined power could be obtained. By observing Figure 6.7 and 6.8, Footman, b_2 and Priest, b_3 should work together. Sorceress, b_1 is not a good combination with Footman or Priest. They should stay behind the Priest as shown in Figure 6.9. Maneuver of player units is shown in Figure 6.10.

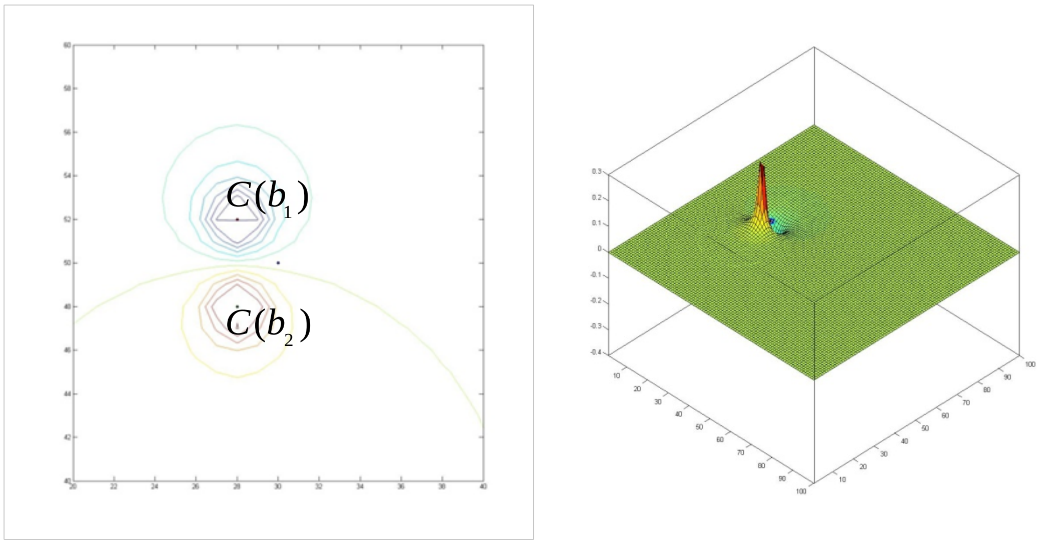


Figure 6.7 Potential field of footman (b_1, b_2 and $\mu(Y)$)

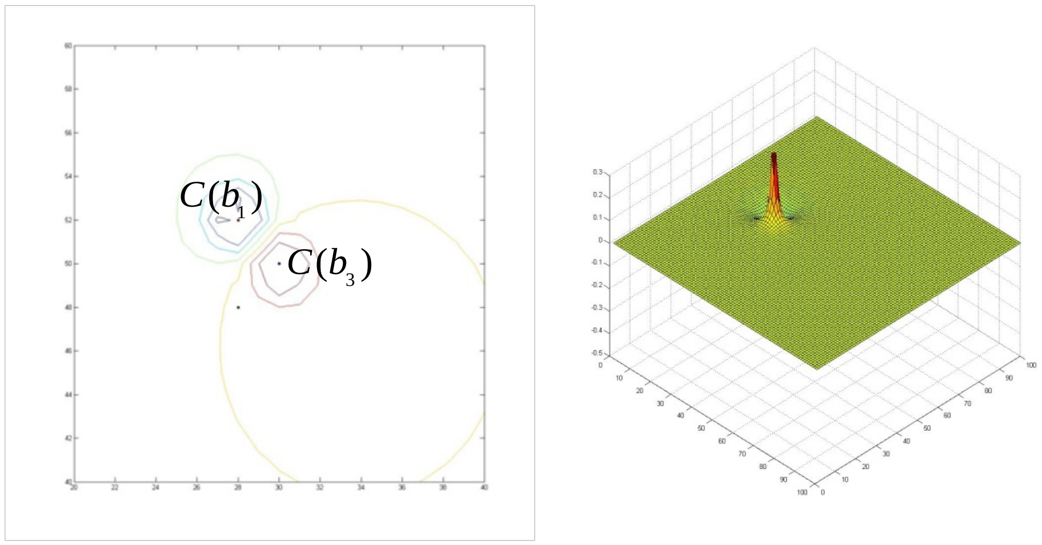


Figure 6.8 Potential field of priest (b_1 , b_3 and $\mu(Y)$)

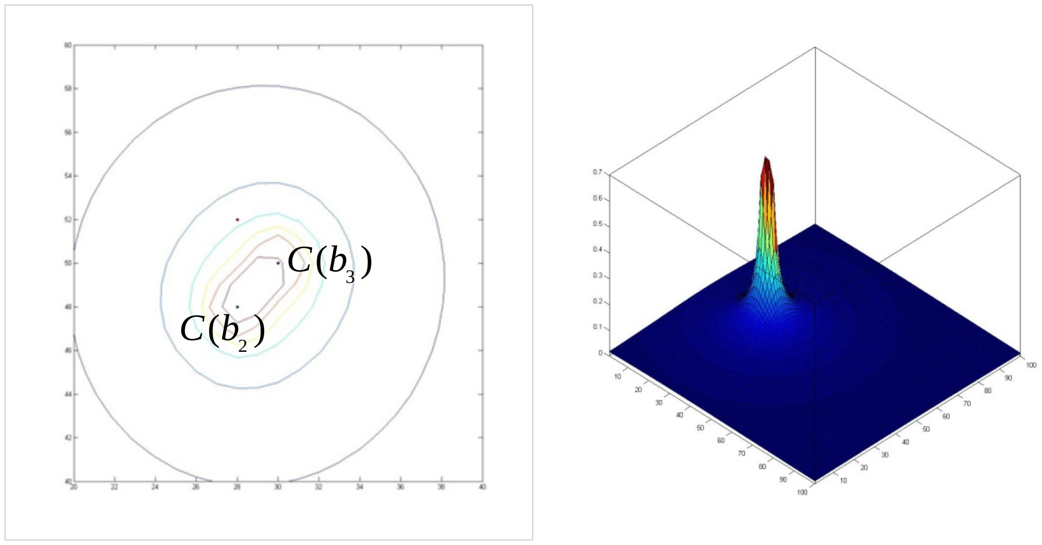


Figure 6.9 Potential field of sorceress (b_2 , b_3 and $\mu(Y)$)

$$\begin{tikzcd}
C(b_3) \arrow[dd] & \\
 & C(b_3) \arrow[dl, bend left] \\
C(b_2) \arrow[ur] &
\end{tikzcd}$$

Figure 6.10 Maneuver of player units

6.4.3 Assigning game rule to potential field

As mentioned before, we assumed that the relationship of game rule and the unit are linear. We use the armor and weapon type as an example. Every unit in the Warcraft III was assigned to an armor type and attack type. Each attack type is better or worse versus other armor types as shown in Table 6.1. It is known as bonus. For example, the attack type of Grunt is normal, which does 150% damage versus medium armor units like the Archer, Rifleman, Troll Headhunter, and Crypt Fiend. The Archer has a Pierce attack, which does 100% extra damage versus light armor units like the Gryphon Rider. It is used to perform intransitive superiority and commonly found in computer games. It encourages unit counters and unit mixing in combat. If the opposing player builds ranged attackers, then the natural counter would be to build melee units, which have an attack bonus versus them.

TABLE 6.1
RELATIONSHIP OF ARMOR AND WEAPON TYPE IN WARCRAFT III

		Armor Type				
		Light	Medium	Heavy	Fort	Hero
Weapon Type	Normal	100%	150%	100%	70%	100%
	Pierce	200%	75%	100%	35%	50%
	Siege	100%	50%	100%	150%	50%
	Magic	125%	75%	200%	35%	50%
	Chaos	100%	100%	100%	100%	100%
	Spells	100%	100%	100%	100%	70%
	Hero	100%	100%	100%	50%	100%

TABLE 6.2
ARMOR AND WEAPON TYPE OF UNIT TYPE (PARTIAL)

Symbol	Name	Armor Type	Weapon Type
a_1	Grunt	Heavy	Normal
a_2	Raider	Medium	Siege
a_3	SP-Walker	Unarmored	Magic
b_1	Footman	Heavy	Normal
b_2	Priest	Unarmored	Magic
b_3	Sorceress	Unarmored	Magic

The armor and weapon type of the unit that mentioned in Chapter 6.4.1 and 6.4.2 are shown in Table 6.2. $B_{i,p}$, is selected from Table 6.1 to present the bonus for enemy unit type i and player unit type p. Equation (6.3) is modified. For each player unit type p, a new potential field of enemy is combined with the bonus as shown in Equation(6.5). The equation of each point in potential field is shown in the following equation. The new potential fields for enemy are shown in Figure 6.11, 6.12 and 6.13. In this case, enemy unit, a_2, is the attack target.

$$P(\mathrm{x,y}) = \sum_{i=1}^{i=n} B_{i,p} \times C(a_i) \times \varphi(P(\mathrm{x,y}), P_i) \tag{6.5}$$

where n is the total number of unit type, P_i is the coordinate of a_i

$$\varphi(P(\mathrm{x,y}), P_i) = (D \times \log(\sqrt{[P(\mathrm{x}) - P_i(\mathrm{x})]^2 + [P(\mathrm{y}) - P_i(\mathrm{y})]^2}))^{-1}$$

D is the weighting to control area of affected, $D = 1$

$B_{i,p}$ is the bonus for enemy unit type i and player unit type p

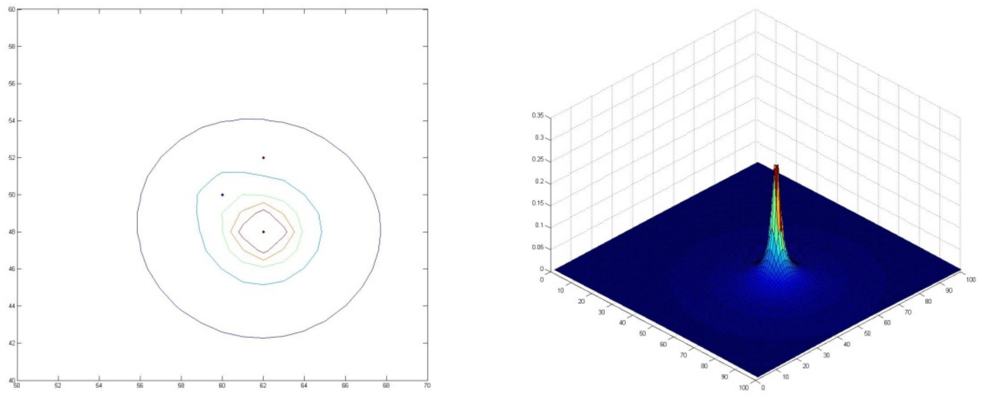


Figure 6.11 Potential field of enemy (for player unit type b_3)

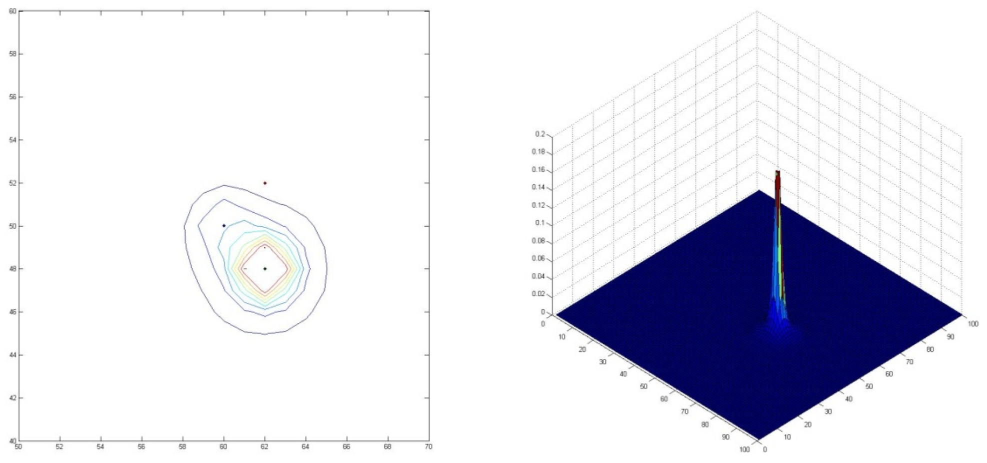


Figure 6.12 Potential field of enemy (for player unit type b_2)

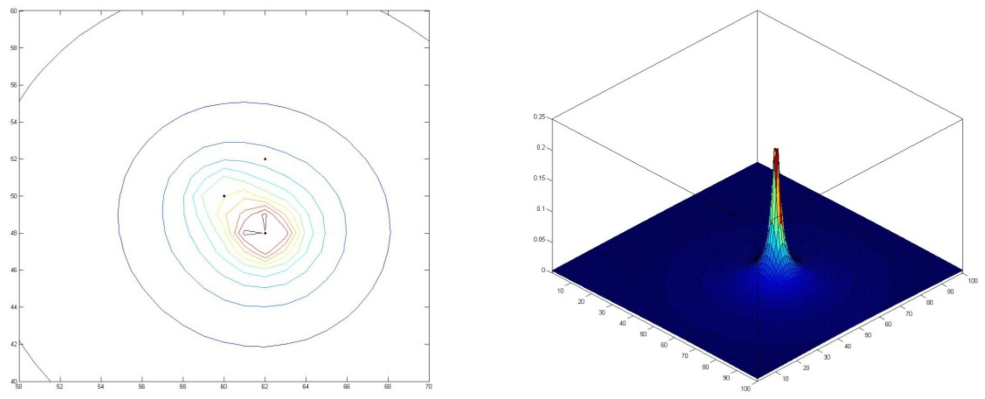


Figure 6.13 Potential field of enemy (for player unit type b_1)

For each player unit type, two potential fields are generated. One is for player unit formation, the other is for the target of attack. The highest points of the two potential fields are compared. The highest one is chosen as the target of attack or destination of cooperation.

6.5 Combining directional based fuzzy integral and potential field

Potential field is designed for dynamic and complex environment. It provides an efficient way to search for the target and path. We proposed a methodology to learn the potential of enemy unit. However, details movement or maneuver still could not perform in attack planning even the interaction is determined by CI. The reason is the decay function is difficult to define and the normal additive aggregator cannot fulfill the representation of interaction. Figure 6.14 and 6.15 shows an example. If the decay function is too small, the highest potential will become the center of all units as shown in Figure 6.14. It is damager as player unit will be easily surrounded by the enemy. On the other hand, if the decay function is too large, only the centers of unit are significant in the potential field as shown in Figure 6.15. The idea of interaction cannot be presented. Potential field will be meaningless as the path will be a straight line to one of the units. The destination and path are not optimized. Flanking or diversion attack cannot be planned.

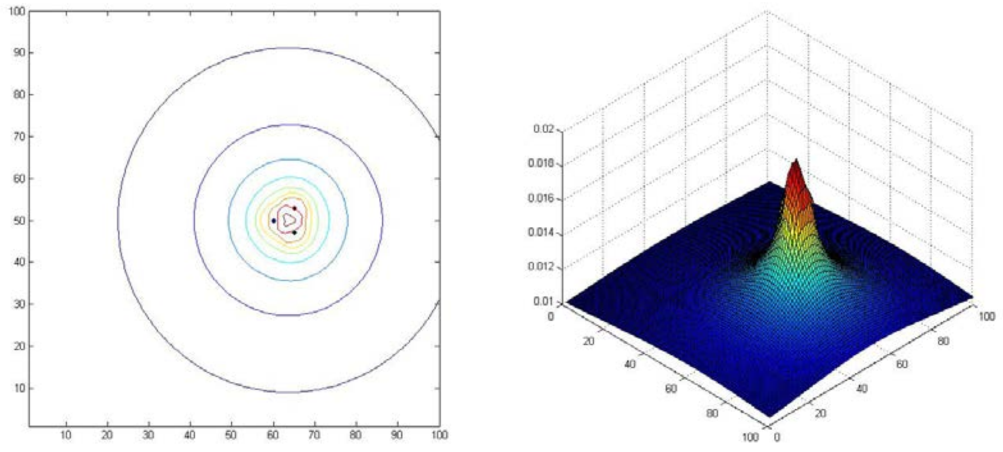


Figure 6.14 Effect of the decay function, $D = 0.5$

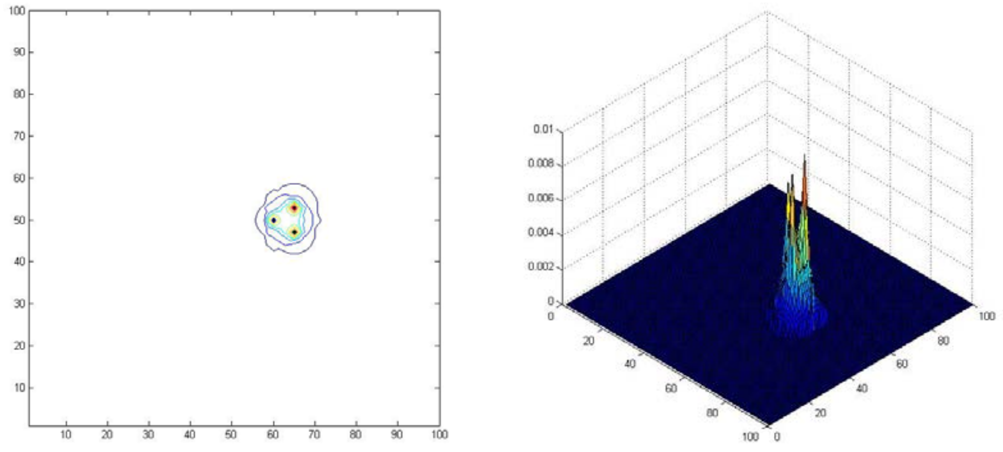


Figure 6.15 Effect of the decay function, $D = 2$

6.5.1 Evaluating the individual contribution and interaction

Although the Choquet Integral could present the positive and negative interaction among the unit type, the priory sorting of $f(x)$, it is difficult for the game developer to understand the meaning of $C(a_i)$ where $C(a_i) = \mu(x \mid f(x) \geq a_i) - \mu(x \mid f(x) \geq a_{i+1})$. The contribution is related to the subset of a_i and its upper term a_{i+1}. The contribution of a_i and the interaction to the other unit are mixed together. Therefore, we developed Directional base fuzzy integral (Directional base FI or $(d)\int f(x)d\nu$). It is used to describe the correlation of two unit types. The entire equation is shown in Equation (6.6).

$$
\begin{aligned}
&(d)\int f(x)d\nu \\
&= \sum_{i=1}^{n} \left\{ f(x_i)\nu(x_i) + w\sum_{j=1,\, j\neq i}^{n} f(x_i)[\nu(x_i, x_j) - \nu(x_j)] \right\} \\
&= \sum_{i=1}^{n} f(x_i) \left\{ \nu(x_i) + w\sum_{j=1,\, j\neq i}^{n} [\nu(x_i, x_j) - \nu(x_j)] \right\}
\end{aligned}
\tag{6.6}
$$

or

$$
\begin{aligned}
&(d)\int f(x)d\nu = \sum_{i=1}^{n} (O + I) \\
&O = f(x_i)\nu(x_i) \\
&I = w\sum_{j=1,\, j\neq i}^{n} f(x_i)[\nu(x_i, x_j) - \nu(x_j)]
\end{aligned}
$$

$$\text{where } f(x_i) \neq 0, f(x_j) \neq 0$$

The integral is divided into two parts. The first part, $O = f(x_i)\nu(x_i)$, is its individual contribution. The second part, $I = w\sum_{j=1,\, j\neq i}^{n} f(x_i)[\nu(x_i, x_j) - \nu(x_j)]$, is used to describe all the correlation of unit types. No priority sorting is required. w is the weighting to enlarge effect of interaction effect. In our experiment, it is set as 10. New fuzzy measure of player and enemy are trained again. The producers are the same as Chapter 6.4.2.

6.5.2 Properties of directional based fuzzy integral

Let $X = \{x_1, x_2, ..., x_n\}$ and $\mu : P(X) \rightarrow [0, \infty)$ is a non-monotonic measure, i.e., efficiency measure on the power set of X. Directional based fuzzy integral is a non-linear integral. It also satisfies some basic properties of common fuzzy integrals, which are listed as follows.

$(d)\int_X 1 d\nu = \nu(X)$ (6.7)

For any $c \in [0, \infty)$, $(d)\int c \cdot f d\nu = c \cdot (d)\int f d\nu$ (6.8)

$(d)\int f d\nu < (d)\int g d\nu$ if $f(x) \leq g(x)$ (6.9)

If α is a non-negative real value, b is a real value, (6.10)

then $(d)\int (af + b) d\nu = a(d)\int f d\nu + b\nu(X)$

$(d)\int f d\mu \leq (d)\int f d\nu$ if $\mu(A) \leq \nu(A)$ for every $A \subseteq X$ (6.11)

$(d)\int_A f d\nu \leq (d)\int_B f d\nu$, where $A \subseteq B$ and ν is a monotonic fuzzy measure (6.12)

$(d)\int f d\mu = 0 \xrightleftharpoons{\text{if and only if}}$ for $\forall A \subseteq X$ with $\mu(A) > 0$, there exists $x \in A$ such that $f(x) = 0$ (6.13)

$\forall a \in [0, \infty)$, $(d)\int (f + a) d\nu \geq (d)\int f d\nu + (d)\int a d\nu$ (6.14)

$\int (f + g) d\nu = \int f d\nu + \int g d\nu$ (6.15)

6.5.2 Assigning charge to enemy potential field

The individual contribution and interaction can be indicated independently and combined with the decay function, φ. Finally, it is assigned to the potential field. The equation of each point in potential field is expressed as Equation (6.16).

$$P(\text{x,y}) = \sum_{i=1}^{n} B_{i,p} \times (O \times \varphi(P(\text{x,y}), P_i)) + I \times \varphi(P_i, P_j)) \tag{6.16}$$

where n is the total number of unit type, P_i is the coordinate of a_i

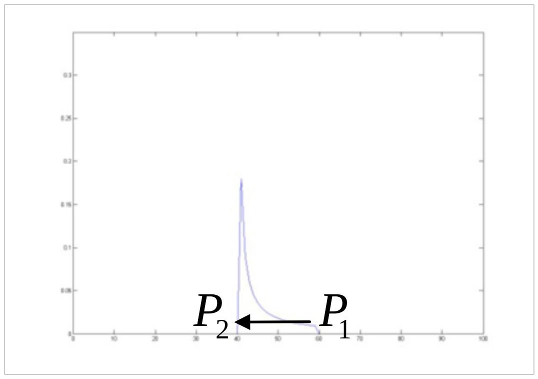
$B_{i,p}$ is the bonus for enemy unit type i and player unit type p

$$\varphi(P_i, P_j) = (D \times \log(\sqrt{[P_i(\text{x}) - P_j(\text{x})]^2 + [P_i(\text{y}) - P_j(\text{y})]^2}))^{-1}$$

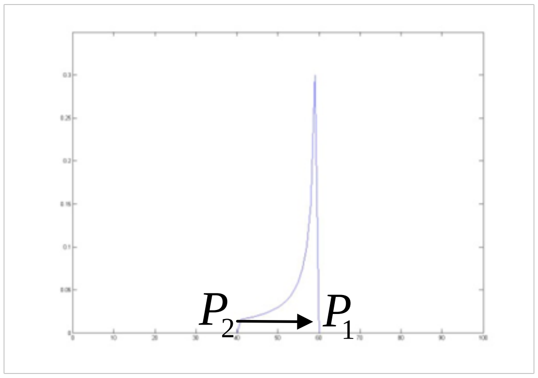
D is the weighting to control area of affected, $D = 1$

w is the weighting to enlarge effect of interaction effect, $w = 10$

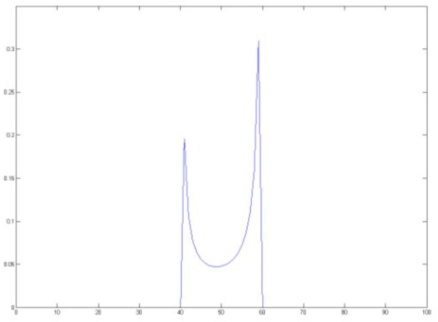
The potential, I, represents the contribution of x_i to x_j. $\varphi(P_i, P_j)$ is a decay function which is used to enlarge the potential if the enemy unit is closer to each other. Figure 6.16 (a) shows the potential generate for P_1 to P_2, the potential, $f(x_1)[v(x_1, x_2) - v(x_2)]$, is enlarged when it is closer to P_2. Similarity, Figure 6.16 (b) shows the potential generate for P_2 to P_1, the potential, $f(x_2)[v(x_1, x_2) - v(x_1)]$, is enlarged when it is closer to P_1. Figure 6.16 (c) and (d) show the combined potential. If P_1 and P_2 is getting away, the potential will drop sharply as shown in Figure 6.16 (e) and (f).



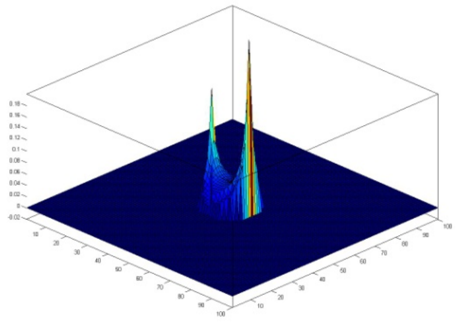
(a)



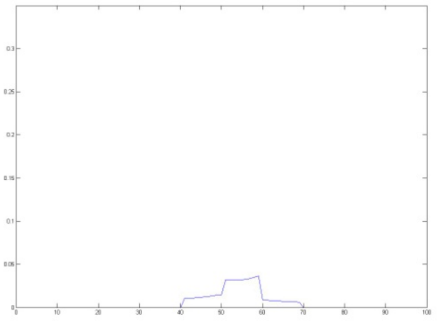
(b)



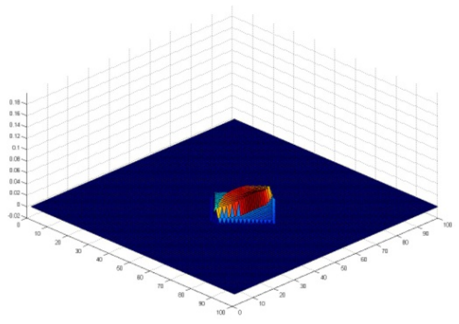
(c)



(d)



(e)



(f)

Figure 6.16 Effect of the decay function, $\varphi(P_i, P_j)$

The affected area of individual contribution, $f(x_i)v(x_i)$, is presented as a circle which is similar to the original potential field. However the affected area of interaction part, $I = w\sum_{j=1, j\neq i}^{n} f(x_i)[v(x_i, x_j) - v(x_j)]$, is presented as a sector of circle and points to another unit. The Equation (6.16) is modified to Equation (6.17). The interaction part is generated in the area in between two points as the expression (6.18) and within a degree, α , as the expression (6.19). Figure 6.17 shows the interaction part of x_1 to x_2 and x_3 .

$$P(\mathrm{x,y}) = \begin{cases} \sum_{i=1}^{n} B_{i,p} \times (O \times \varphi(P(\mathrm{x,y}), P_i)) + I \times \varphi(P_i, P_j)) & \text{(6.18), (6.19) is true} \\ \sum_{i=1}^{n} B_{i,p} \times (O \times \varphi(P(\mathrm{x,y}), P_i)) & \text{Otherwise} \end{cases} \qquad (6.17)$$

where n is the total number of unit type, P_i is the coordinate of a_i

$B_{i,p}$ is the bonus for enemy unit type i and player unit type p

$$\varphi(P_i, P_j) = (D \times \log(\sqrt{[P_i(\mathrm{x}) - P_j(\mathrm{x})]^2 + [P_i(\mathrm{y}) - P_j(\mathrm{y})]^2}))^{-1}$$

D is the weighting to control area of affected, $D = 1$

w is the weighting to enlarge effect of interaction effect, $w = 10$

$$\sqrt{[P(\mathrm{x}) - P_j(\mathrm{x})]^2 + [P(\mathrm{y}) - P_j(\mathrm{y})]^2}) \leq \sqrt{[P_i(\mathrm{x}) - P_j(\mathrm{x})]^2 + [P_i(\mathrm{y}) - P_j(\mathrm{y})]^2}) \qquad (6.18)$$

$$\theta(P_i, P_j) - \alpha \leq \theta(P(x,y), P_i) \leq \theta(P_i, P_j) + \alpha \qquad (6.19)$$

where $\theta(P_1, P_2) = \arctan\left(\frac{P_1(y) - P_2(y)}{P_1(x) - P_2(x)}\right)$ and $\alpha = \pi / 4$

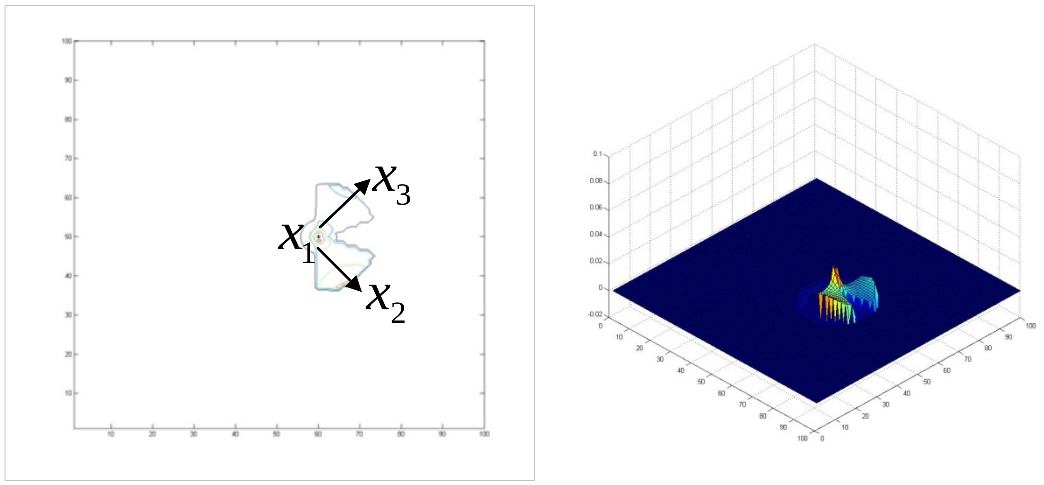


Figure 6.17 The shape of the affected area

6.5.3 Flanking and diversion attack

Based on the above methodology, the interaction can be indicated easily in the potential field. The highest potential is the destination. Flanking and diversion attack can be suggested. Suppose all the interaction is positive. Figure 6.18 shows the enemy formation is in a decentralized phase. They are far away from each other. Thus, $\varphi(P_i, P_j)$ is small and the interaction, $I \times \varphi(P_i, P_j)$ is not significant. We regarded the interaction in this case does not occur. The highest potential tends to be the greatest individual contribution of enemy units, i.e., highest potential $= \max(f(x_i)\nu(x_i))$ and become the target of attack. To avoid the siege of enemy, the area of the sector of circle, potential $\geq I \times \varphi(P_i, P_j)$, is not recommended to be passing through. Flanking attack could be performed based on this setting as shown in the path of Figure 6.18.

Figure 6.19 shows the enemy formation is in an intermediate phase. They are getting closer to each other. Thus, $\varphi(P_i, P_j)$ is increasing and the interaction, $I \times \varphi(P_i, P_j)$ is growing up sharply. We regarded the interaction or cooperation of enemy is going to happen. The highest potential is much higher than the greatest individual contribution of enemy units, i.e., highest potential $\geq \max(f(x_i)\nu(x_i))$. If we do not prevent this situation, enemy will get close to each other and the combined power will be sharply increased. To avoid this situation, diversion attack should be performed. Player unit is suggested to go to the highest potential, i.e., I , and attack the enemy as shown in Figure 6.19.

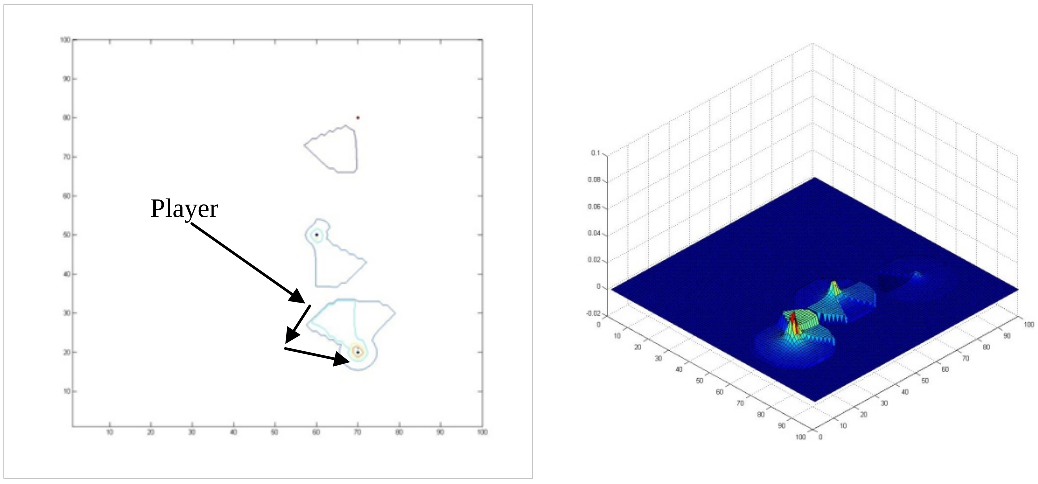


Figure 6.18 Decentralized phase (Flank attack)

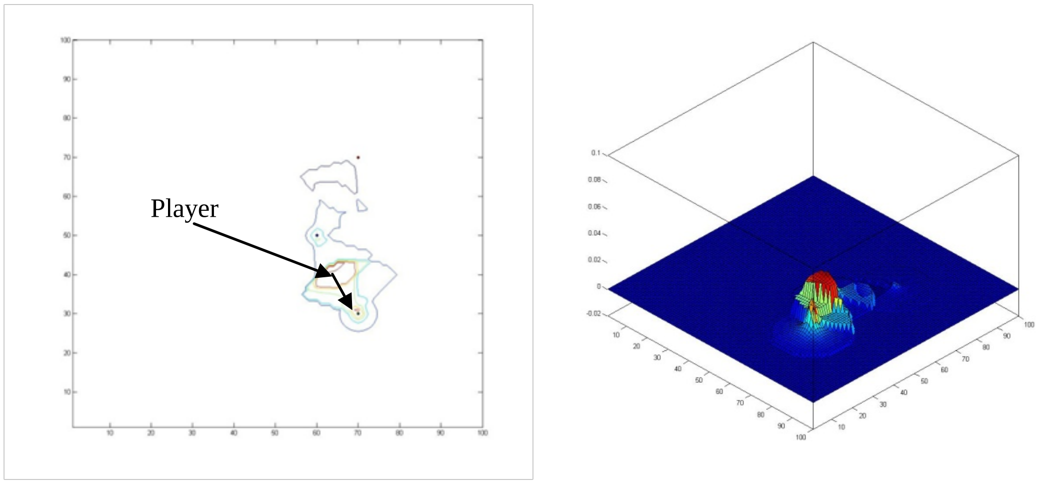


Figure 6.19 Intermediate phase (Diversion Attack)

Figure 6.20 shows the enemy formation is in a centralized phase. They are close to each other. Thus, $\varphi(P_i, P_j)$ is large and the interaction, $I \times \varphi(P_i, P_j)$ is high. We regarded the interaction or cooperation of enemy occurs and the combined power is high. The highest potential is higher than the greatest interaction of enemy, i.e., highest potential $\geq \max\left(f(x_i)\left[\nu(x_i, x_j) - \nu(x_i) \right] \right)$. It is difficult to break the enemy formation and prevent any flank or diversion attack. Player unit is suggested to attack the highest potential as soon as possible. Therefore, the interaction of enemy can be minimized in a short time.

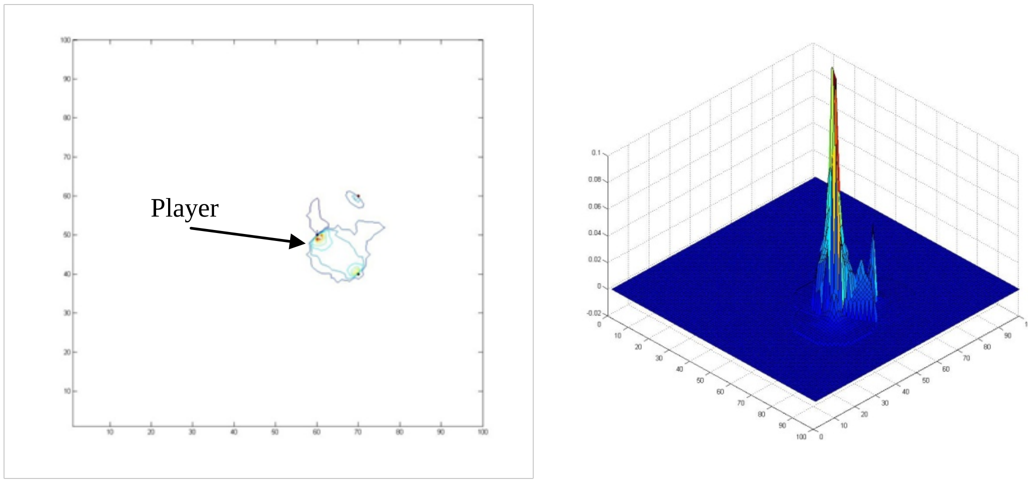


Figure 6.20 Centralized phase

Figure 6.21 shows the enemy potential field with negative interaction. Negative interaction is regarded as the bad unit combination. Player cannot obtain high reward when they attack these areas. It wastes the time. Therefore, the player unit is not suggested to go through any area of negative potential.

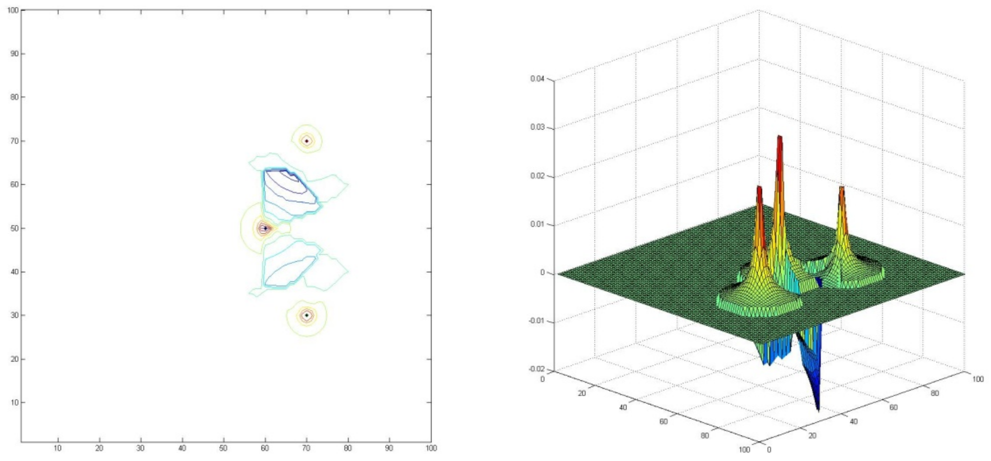


Figure 6.21 Potential field with negative interaction

6.5.3 Assigning charge to player potential field

The procedure of assigning the Directional based FI to player potential field is similar to previous session. There are two main differences. First, the individual contribution, O, is not considered as the value will confuse the scripting and maneuver. The other is progressive function, ϕ, is used instead of a decay function. The reason is that we want to keep the good unit combination close together. The equation is shown in (6.20). Figure 6.22 shows the player potential field. The highest potential that is generated by Equation (6.10) is able to keep the player unit in a reasonable distance, i.e., the area in between the two units. If the units are getting away from each other, ϕ will be increased sharply and will attract the units as shown in Figure 6.23.

$$P(\mathrm{x,y}) = \begin{cases} \sum_{i=1}^{n} (I \times \phi(P_p, P_q)) & \text{(6.18), (6.19) is true} \\ 0 & \text{Otherwise} \end{cases} \tag{6.20}$$

where n is the total number of unit type, P_i is the coordinate of a_i

$$\phi(P_p, P_q) = \sqrt{[P_p(\mathrm{x}) - P_q(\mathrm{x})]^2 + [P_p(\mathrm{y}) - P_q(\mathrm{y})]^2}$$

D is the weighting to control area of affected, $D = 1$

w is the weighting to enlarge effect of interaction effect, $w = 10$

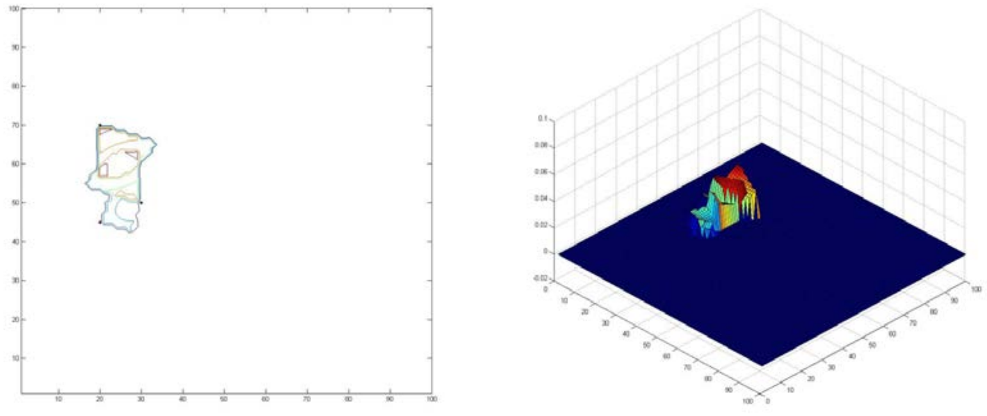


Figure 6.22 Potential field of player

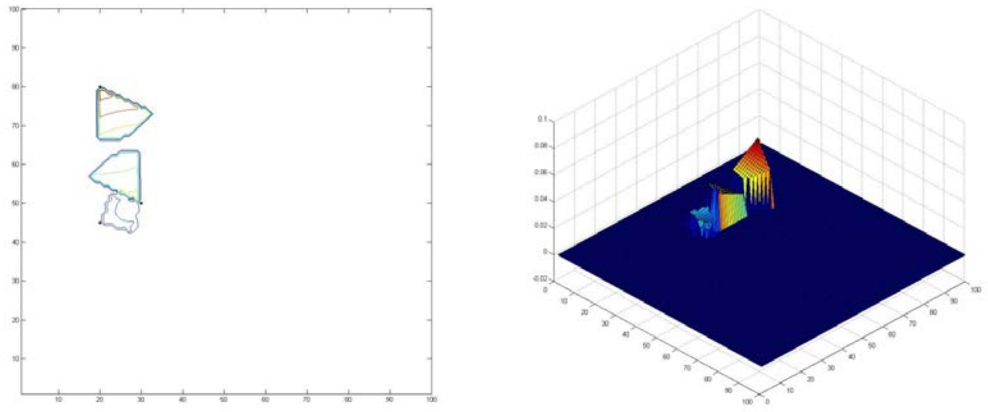


Figure 6.23 Effect of progressive function, ϕ

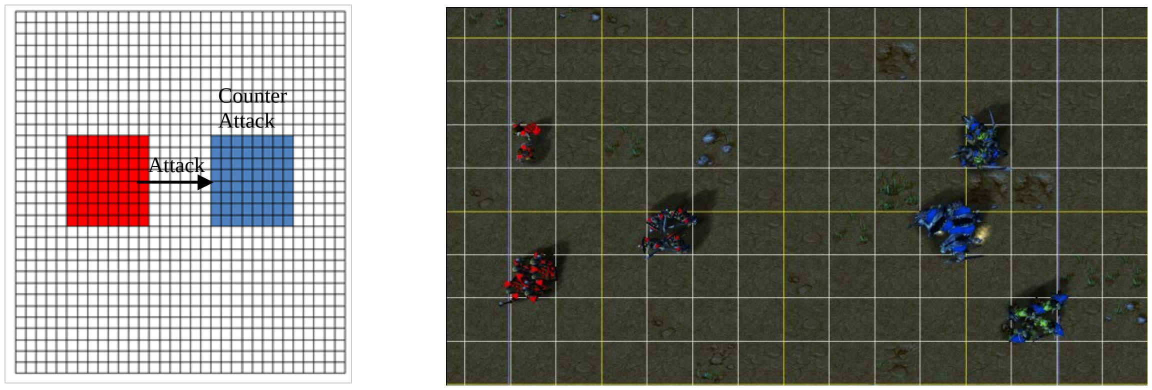
6.5.4 Scripting for micro control

After the enemy and player potential field are generated, scripting is needed to decide the action of each unit type, such as flanking, diversion, direct attack or cooperation. First the highest potential of enemy and player are compared. One of them is selected. If the potential of player is higher, cooperation is performed to maintain the unit formation. On the other hand, if the potential of enemy is higher, the player will perform the flank attack (highest potential $= \max(f(x_i)\nu(x_i))$) or diversion attack (highest potential $\geq \max(f(x_i)\nu(x_i))$) or direct attack (highest potential $\geq \max(f(x_i)\left[\nu(x_i, x_j) - \nu(x_i)\right])$). The highest potential will become the destination and the path is generated. All the process is updated at certain time slot. Destination and path is continually updated.

6.6 Experimental result and discussion

6.6.1 Experiment Setting

To prove the performance of potential field and fuzzy integral, we used Warcraft III to simulate the experiment. First, all the replays of data cluster 3 and 4 that were stated in chapter 5 are selected. They are the competition of two races, orc and human. The number of cases is 1553. The number of unit types is 12 for each race. Therefore, the size of each fuzzy measure is $2^{12}-1=4095$. The number of unit, $f(x)$, and scores for each case are extracted and used to train the fuzzy measure. GA and CMA-ES are both tested. The result of CMA-ES is used for better training and testing result. μ_{Orc} and μ_{Human} are the fuzzy measure trained by Choquet integral while ν_{Orc} and ν_{Human} are the fuzzy measure are trained by Directional based fuzzy integral.



(a) (b)

Figure 6.24 Simulation in Warcraft III

The initial setting of one verse one battle is presented as following. First, a battle field with 32×32 unit is generated as shown in Figure 6.24(a). Three unit types are randomly selected for each side. The number of unit in each type is randomly assigned but the total amount of the army is fixed to 20, i.e., $\sum_{i=1}^{3} f(x_i)=20$. Their locations are randomly assigned in the red square and blue square as shown in shown in Figure 6.24(b). The units in the blue square are regarded as enemy. They are control by the rule based system of Warcraft III. They will not perform any actions until the units in red square move into the blue square. The unit in the red square is regarded as player and will move into the blue square and attack. The battle is terminated until all units of one side are killed or the time of battle exceeds ten minutes. Fifty battles are set up for testing. Half of them are orc attack human and another half are human to attack orc. For each battle, three experiments are set up for testing.

For the first experiments, player is controlled by original rule based system. The player will directly go into the blue square. Therefore, there is no planning for the unit maneuver. During the movement, the unit will attack the nearest enemy until the enemy is dead and control by the scripting of Warcraft III.

For the second experiments, it is guided by potential field with Choquet Integral. The highest potential will become the destination. Again, the unit will attack the nearest enemy until the enemy is dead. Potential field will be updated for each 10 seconds.

For the third experiments, it is guided by potential field with Directional based fuzzy integral. The highest potential will become the destination. Scripting is stated in Chapter 6.5.3. Flanking and diversion attack will perform if the condition meets. Potential field will be updated for each 10 seconds.

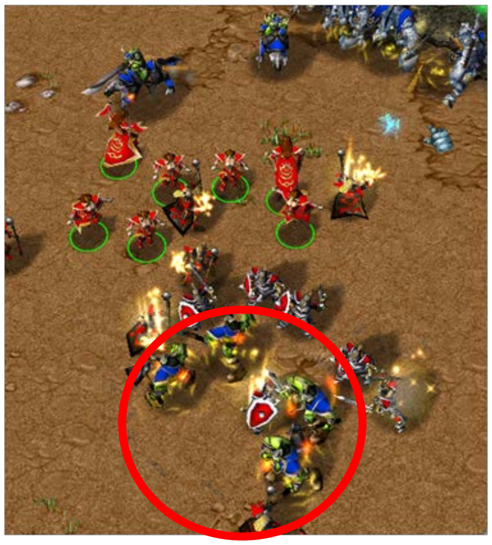
6.6.2 Results and visualization

For each experiment, 50 battles are preformed. The result has been stated in Table 6.3. The winning percentage of potential field with Directional based fuzzy integral is the highest. Compared with rule based system, potential field with Choquet Integral has a 29% improvement and the potential field with Directional based fuzzy integral has a 48% improvement. The performance of micro control is optimized. By observing the battle, both potential fields with fuzzy integral can locate the support unit, such as Priest in Human and Raider in Orc. The combined power will be decreased by stopping their support. In another word, kill them at once can increase the possibility of win.

Potential field with Directional based fuzzy integral cannot show significant different when the unit is closed together, i.e., potential $\geq \max\left(f(x_i)\left[\nu(x_i, x_j) - \nu(x_i) \right] \right)$. In another word, interaction has already occurred in the enemy, However, it showed a significant improvement when flank and diversion attack can be performed. Diversion attack can isolated some enemy units as shown in the circle of Figure 6.24(a). As the enemy support is blocked, the isolated enemy will be easily killed. The player can retain more units for the remaining battle. Flank attack could also improve the possibility of win as the player does not need to face all the enemy units at the same time Figure 6.24(b).

TABLE 6.3
COMPARISON OF DIFFERENT MICRO CONTROL

Micro Control	Wining
Rule based System	42%
Potential field and Choquet Integral	54%
Potential field and Directional based fuzzy Integral	62%


(a) Diversion

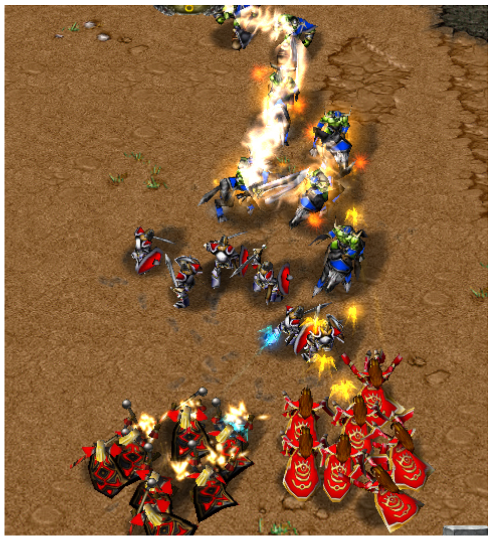

(b) Flank Attack

Figure 6.25 Diversion and flank attack in Warcraft III

6.7 Summary

In this chapter, we have developed an adversarial real time planning model for micro control in RTS game. We have extended the normal additive properties to non-linear for potential field. Interaction of different units in the battle has been considered. Since classical Choquet integral cannot be visualized the interaction in potential field easily. We defined a new type of fuzzy integral called Directional-based FI. This new integral can evaluate the correlation of different unit combinations. Individual contribution and interaction of different unit type can be determined and assigned to potential field easily. Experiments are carried out to compare rule based system, potential field with CI and potential field with Directional based FI. By using the proposed integral, different interactions such as super-additive or sub-additive can be visualized in potential field. The performance of path finding in micro control is optimized. Details unit maneuver, such as flank and diversion attack can be performed. For the winning percentage, compared with rule based system, potential field with Choquet Integral has a 29% improvement and the potential field with Directional based fuzzy integral has a 48% improvement. The future work will focus on extending the result of feature interaction in perform more advance unit maneuver in RTS game, such as tracking problem.

Chapter 7

Conclusion and future works

7.1 Summary of the research problem

The game market and related technologies grow rapidly in these few years. Graphics improvements are becoming saturated. However, artificial intelligence (AI) development in game remains a grand challenge for researchers, especially in real time strategy (RTS) game. The fundamental game play of a typical RTS game is collecting and allocating resources to build an army and destroy enemy units. Strategy refers to a sequence of above actions to achieve this goal. It depends heavily on the current opponent and spatial information. It is difficult to formulate a model as it consists of complicated game rules and numerous kinds of interacting units. The search space is large and often involves complicated interaction among the game units and corresponding actions. In the game industry, strategy planning is usually handled by rule based system or tree searching. It is hard to manage and easily discovered by human players. Moreover, it could only support the action level and lack for abstract thinking to control the sequence of actions.

Soft computing techniques are efficient to search the inexact solution under time pressure and uncertainty. However, current techniques, such as neural networks, SWARM intelligence, and decision trees induction are formulated on the minimization of Euclidean distance-based error functions. The parameters used in these functions are all normal additive in nature and the model is unable to describe the non-linear effects among parameters. It is not suitable to present the intransitive superiority situation in RTS game.

Case based reasoning (CBR) is another important tracks in game AI development. It reduced the workload and the development time for strategy planning. Unlike, NN, GA or other AI technologies, expert assistance could be easily involved. Decisions and actions could be learnt from human players. There are two main disadvantages. First one is the huge quantities of data which are required at the beginning stage. Another problem is the algorithm for case retrieval has not yet been formulated very well in RTS game. The algorithm will tend to a constant number when the number of cases or the dimension of the problem spaces increases. It cannot identify the difference among situations and find a most suitable one.

7.2 Summary of the research work

In our work, we focus on strategy indexing, learning and optimization in RTS game. For the indexing problem, we improved the random walk of GA and reduced the recall time for searching an optimized solution. We have also created a model to learn the player decisions and actions. It improved the learning ability by considering the interaction inside the features. Finally, we developed four fuzzy integrals to optimize the case retrieval of strategy planning model and action behavior of object in RTS game.

We selected tower defense which is a subgenre of RTS game play as our preliminary study. We created a model to adopt neural network (NN) into GA to solve this problem. The optimized solutions that provided by GA are encoded, memorized and indexed by NN. Comparing with GA, the training time of the new model has a 49.8% of improvement. GA required 800 to 1000 seconds to obtain the solution of tower distribution, while the recall time of the new model is only 0.04 second to 0.06 second. This model provides efficient, fair and natural game AI to tackle the game problems. Simulation results are provided to support our idea.

After the preliminary study, we extended our work to RTS game play. We decoded and extracted the data from the replay of Warcraft III which is a well known RTS game. We adopted a Case-Based Reasoning (CBR) and Bayesian Network (DBN) approach to create player behavioral models. The model achieved the average accuracy of 84.0%. The prediction time is around 0.1 second in a Core 2 Duo 2.13GHz machine with 4 GB Ram. It is efficient in runtime. However, most of the unit types and attributes of RTS game cannot be classified into discrete or Gaussian distributions. The structure of DBN is huge and the joint distributions are unmanageable. The player behavioral models are hard to turn into a strategy planning model.

Although we have developed a model to learn player behaviors, there is lack of algorithm to evaluate the performance of cases and handle the feature interaction in RTS game. To overcome this problem, we divided the RTS game play into two types and performed the research. First one is macro control. It consists of the development of resource gathering plans, base building decisions and technology upgrade paths. Another is micro control which involves in directing unit movements, path selection and the combats encounter.

For the macro control, we extracted the strategies from real professional players in Warcraft III. A total of 2,649 replay files of professional one-versus-one competitions are selected for the experimental training and testing. We combined GA, CMA-ES, fuzzy measure and integral to learn the performance of unit combination. Fuzzy measure of each subset is guided by the fuzzy integral in the GA or CMA-ES training. The value of each subset can be fully observed and managed. Finally, the model is able to evaluate the new situation in the complex environment and gives a score for the strategy. Fuzzy measure and technique also optimized the strategy planning model by considering super-additive and sub-additive in the feature. Thus, the model is able to search an optimized strategy in the intransitive superiority situation.

Traditional Choquet Integral is useful in many applications because of its generalized mean operator. However, it could not obtain a good result in the scores estimation as the aggregation put more emphasis on those subsets that have smaller values. Therefore, three new fuzzy integrals have been proposed as an aggregation operator to sum up all the fuzzy measures and to model the interaction among the feature in RTS game. We developed mean based and max based fuzzy integral to evaluate the performance of strategies in macro-control. Mean based fuzzy integral considers all the interactions that involve the selected unit type in the fuzzy measure and then take an average. It has 78% and 38% improvement, compared with weighted average and Choquet Integral. Max based fuzzy integral considers the maximum value of the subset. It has 73.0% and 20.0%

improvement, compared with weighted average and Choquet Integral. One of the weaknesses of these two integral is the training time. It is about 40 to 180 times more than Chouqet Integral. We also developed Order based fuzzy integral to reduce the training time of Mean based fuzzy integral. It considers resource weighting and development of RTS game units. The aggregation operator focuses on the highest proportion units, i.e., highest resources units first and calculates their interactions with all other units and so on. Compared with Mean based fuzzy integral, the training time and memory complexity is reduced from $O(n^2)$ to $O(n)$ where n is the number of unit type. The performance is similar to the Mean based fuzzy integral. It has 80% and 40% improvement, compared with weighted average and Choquet Integral. Compare with NN, it has 7.5% improvement and the concept of power set and feature combinations can be easily shown to programmer or AI developer. We also transformed the fuzzy integral into different mathematical forms to provide a better understanding of sub-set selection.

For micro control, we applied potential field, fuzzy measure and integral to solve the micro-control. Potential field is suitable for complicated and various environment with multiple targets. However, aggregation operator does not consider non-additive property. It is unable to perform the flanking attacks and diversions. We integrated potential field with fuzzy measure and Choquet integral and provided the ability to handle interactions among different targets. It improved the behavior of the unit formation. Cooperative behavior emphasizes the importance of interaction between the units. Hence, we developed directional based fuzzy integral for potential field. It could identify the direction of positive and negative interactions for movement planning and team composition in micro-control. Flank and diversion attack can be preformed. This planning model removes the needs for designing complicated rule set or finite state machine. Compared with rule based system, potential field with Choquet Integral has a 29% improvement and the potential field with Directional based fuzzy integral has a 48% improvement.

7.3 Future work

Both RTS game and interaction are under study problem in research area. Our research can be extended to different aspect for further study. We have the following suggestions.

Simplify fuzzy measure

Throughout the development of fuzzy measure, there are two simplified fuzzy measure, such as Sugeno-λ and k-additive fuzzy measure. K-additive fuzzy measure reduces the number of parameters in determining a fuzzy measure to size k. For example, for k equals to 1, the subset with magnitude equal to 1, such as $\{x_1\}$, $\{x_2\}$, $\{x_3\}$, etc, is found or set up by expert first. The remaining interactions, such as $\{x_1, x_2\}, \{x_2, x_3\}, \{x_1, x_3\}$, $\{x_1, x_2, x_3\}$, are then described by $\{x_1\}$, $\{x_2\}$, $\{x_3\}$ and the equation that stated in (2.13) of Chapter 2. In our work, it shows similar phenomenon. The role of the subset with smaller magnitude is more important. Usage count of these subset selections is higher as shown in Figure 7.1, 7.2 and Table 7.1. Most of the subsets with higher magnitude does not occur in the real cases. The reason behind is that the advance unit requires many resources and technology updated. A professional player will only focus on a few advance units and of course, they will not produce balance army with many unit combinations. Therefore, the size of fuzzy measure could be simplified by the subset selection, such as Equation (5.10) and Equation (5.11). It can reduce the training time of GA and CMA-ES. A further study could be done on formulating and designing the simplified fuzzy measure for the real cases.

TABLE 7.1
COMPARISON OF FREQUENCY WITH DIFFERENT MAGNITUDE IN SUBSET

Magnitude	Or-based FI	Choquet Integral
1	23.28571	23.85714
2	7.714286	8.000000
3	4.000000	3.942867
4	2.142857	2.085714
5	0.619408	0.666667
6	0	0

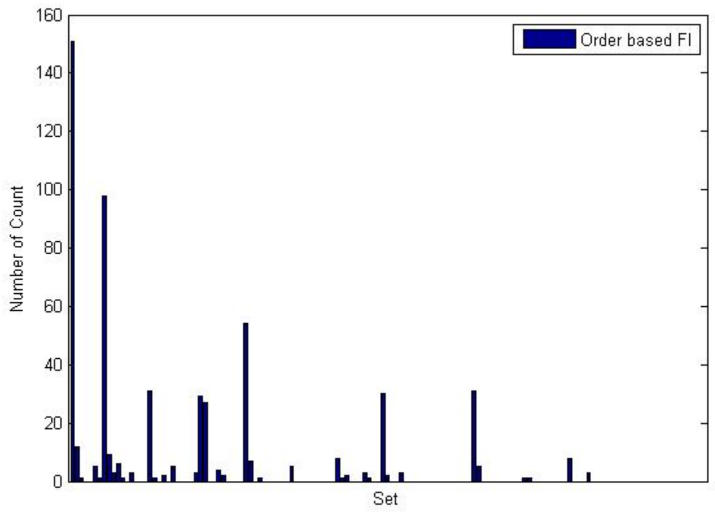


Figure 7.1 Usage count of subset selection for order-based FI in data cluster 1

(X-axis is the set. Statring from the left, they are the set which only contains one element, i.e. , {x_2}, {x_3}…etc. Then they are the sets which contain more elements, i.e., {x_1, x_2}, {x_1, x_3},…, { x_1, x_2, x_3}, etc. The last one on the right is the set which involves the whole set of elements. i.e., { x_1, x_2, …, x_n})

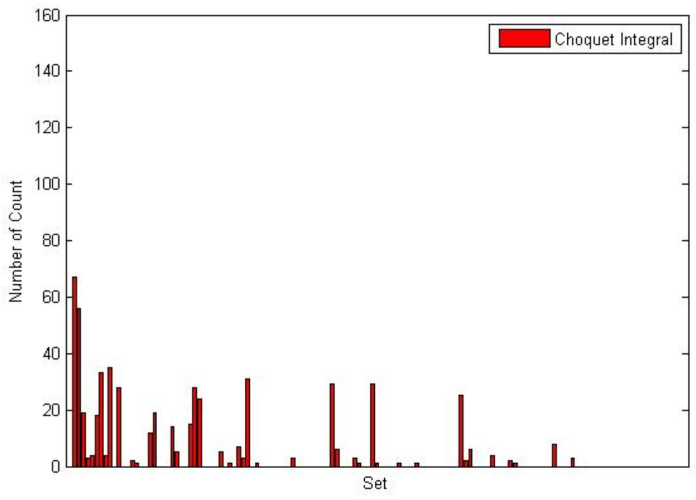


Figure 7.2 Usage count of Subset selection for CI in data cluster 1

Maximum algorithm for priority sorting in fuzzy integral

We have developed an efficient way to evaluate the unit combination. However, it is difficult to find the unit combination that can maximize the combined power. Let $f(x)$ be a function on $[x_1, x_2 ... x_n]$. Fuzzy integral consists of priority sorting. The subset of fuzzy measure, $\mu(x_1),...,\mu(x_1,x_2),...\mu(x_1...x_n)$ in the fuzzy integral will be change based on sorting of $f(x)$. It is a non-linear calculation. Therefore, traditional mathematical equation cannot find a suitabitity of $\left[x_1,x_2...x_n\right]$ to get the maximum value of fuzzy integral, i.e., $\max\left((c)\int fd\mu\right)$. Currently, only some EA methods could be used to find the best fit answer. There is lack of methodologies to search the maximum value in fuzzy integral in an efficient way.

Trapping in unit maneuver

Trapping in the RTS game is a special "strategy" looking good in the short term but has bad consequences in the long term for the enemy. The study on traps in RTS is very little and the existing models in literatures regarding the trap as a static obstacle. Miles and Louis [Miles 2004, Louis 2005] applied the genetic algorithm and combined with case-based reasoning is used to the trap avoidance. Naveed [Navved 2011] applied Markov decision process in the trap recognition. Mingliang [Mingliang 2010] was the only one proposed trapping planning. He has improved the A* path finding to find multiple paths for tracking the enemy. However, it is only for well defined and finite path which is not suitable for the dynamic battlefield in RTS game.

Design and recognition of a trap could produce an advance unit maneuver planning. It is more like a human beginning. We could indicate the special cases with high evaluations at shallow episodes and with a low evaluation at the maximum episode, where the evaluation can be measured by the degree of enhancing the own power and/or destroying the opponents' power. The feature and the action of the special cases could be modeled. Currently, potential field only focuses on instance unit planning. By combining with CBR, the theory of potential field could be extended to deal with the time series.

Unit balancing and gameplay design problem in game design

In our work, we considered game unit type as our basic unit for interaction. In fact, there are many skills and properties in each unit type. Interaction also occurs in this level and under study. The gaming companies spend huge resources to fine tune the value of skills and properties. We called it as unit balancing. It is a big issue in game industry. Another big issue is the gameplay design. Nowadays, the investment of game development is huge and some of them are counted in billion US dollar. A good evaluation model or algorithm for gameplay and unit balancing design is still missing.

Interaction in other domain field

Feature interaction is not only in game industry. Another good example is in social network. How the structure ties the users together and what kind of knowledge could be found in the interaction of users are popular research trend. Lots of funding is provided by US Government to investigate the deep belief networks. DBN, fuzzy measure and integral is possible to combine together and provide an efficient interaction extraction.

Reference

[Aha 2005] D. Aha, M. Molineaux, M. Ponsen, "Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game." Lecture notes in Computer Science, 3620: 5, 2005

[Albrecht 1998] D. Albrecht, I. Zukerman, “Bayesian models for keyhole plan recognition in an adventure game” User modeling and user-adapted interaction 8(1): 5-47, 1998

[Alexander 2007] N. Alexander, “Game AI is Dead. Long Live Game AI!”, Intelligent Systems, IEEE 22(1): 9-11, 2007

[Auger 2005] A. Auger, A, N. Hansen, “A Restart CMA Evolution Strategy With Increasing Population Size,” In Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, pp.1769-1776, 2005

[Auger 2010] A. Auger, D. Brockhoff, N. Hansen, “Benchmarking the (1, 4)-CMA-ES with mirrored sampling and sequential selection on the noisy BBOB-2010 testbed”, Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference 2010, ACM, pp. 1625-1631, 2010

[Avery 2010] P. Avery, S. Louis, “Coevolving team tactics for a real-time strategy game,” In Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC-2010), Barcelona, July 18-23, pp. 1-8, 2010

[Barricelli 1957] Nils Aall Barricelli, "Symbiogenetic evolution processes realized by artificial methods", Methodos: 143–182, 1957

[Baum 1966] L. Baum, T. Petrie, “Statistical Inference for Probabilistic Functions of Finite State Markov Chains”, The Annals of Mathematical Statistics 37 (6): 1554–1563, 1966

[Bakkes 2009] S. Bakkes and P. Spronck, “Rapid and Reliable Adaptation of Video Game AI”, IEEE Transactions on Computational Intelligence and AI in Games, 1(2): 93-104, 2009

[Baumgarten 2009] R. Baumgarten, S. Colton, “Combining AI Methods for Learning Bots in a Real-Time Strategy Game” International Journal of Computer Games Technology, 2009

[Beume 2008] N, Beume, T. Hein, “Intelligent anti-grouping in real-time strategy games”, IEEE Symposium on Computational Intelligence and Games (CIG 2008), 2008

[Buro 2003] M. Buro, “Real-time strategy games: A new AI research challenge”, International Joint Conference on Artificial Intelligence, vol 18, pp. 1534-1535, 2003

[Buro 2004] M. Buro, “Call for AI research in RTS games”, Proceedings of the AAAI-04Workshop on Challenges in Game AI, pp. 139–142, 2004

[Chamber 2005] C. Chamber, W. Feng, D. Saha, “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, pp 7-12, 2005

[Chen 2000] Ting-Yu Chen, Jih-Chang Wang, and Gwo-Hshiung Tzeng, “Identification of General Fuzzy Measures by Genetic Algorithms Based on Partial Information”, IEEE Transaction on Systems, Man, and Cybernetic—part B, vol. 30, no. 4, 2000

[Choquet 1953] G. Choquet , “Theory of Capacities”, Annales de l'Institut Fourier 5: 131–295, 1953

[Chuen-Tsai 1994] S. Chuen-Tsai, Y.H. Liao, J.Y. Lu, F.M. Zheng, “Genetic algorithm learning in game playing with multiple coaches”, IEEE World Congress on Computational Intelligence, Proceedings of the First IEEE Conference on Evolutionary Computation, 239-243, 1994

[Fraser 1970] A. Fraser; D. Burnell, “Computer Models in Genetics”, New York: McGraw-Hill. ISBN 0-07-021904-4, 1970

[Forbus 2002] K. Forbus,, J. V. Mahoney, “How qualitative spatial reasoning can improve strategy game AIs.” IEEE Intelligent Systems, 17(4): 25-30, 2002

[Genter2011] K. Genter, S. Ontañón, A. Ram, “Learning opponent strategies through first order induction,” In Proceedings of the 2011 Florida Artificial Intelligence Research Society Conference (FLAIRS-2011), Florida, May 18-20, pp. 482-483, 2011

[Gillies 2009] M. Gillies, “Learning Finite-State Machine Controllers From Motion Capture Data”, IEEE Transactions on Computational Intelligence and AI in Games, 1(1): 63-72, 2009

[Hagelback 2008] J. Hagelback and S. Johansson, “Using multi-agent potential fields in real-time strategy games”, Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2, pp 631-638, 2008

[Hansen 2001] N. Hansen, A. Ostermeier, “Completely Derandomized Self-Adaptation in Evolution Strategies”, *Evolutionary Computation*, 9(2), pp. 159-195, 2001

[Hansen 2004] N. Hansen and S. Kern, “Evaluating the CMA evolution strategy on multimodal test functions,” Parallel Problem Solving from Nature (PPSN VIII), pp. 282-291, 2004

[Hansen 2003] N. Hansen, M. Koumoutsakos, “Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation”, Evolutionary Computation, 11(1) pp1-18, 2003

[Hansen 2010] N. Hansen, A. Auger, R. Ros, S. Finck, P. Posik, “Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009”, Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference 2010, ACM, pp. 1689-1696, 2010

[Hebb 1949] D. Hebb, “The organization of behavior: A neuropsychological theory”, Lawrence Erlbaum, 1949

[Hsieh 2008] J.L. Hsieh, and C.T. Sun, “Building a player strategy model by analyzing replays of real-time strategy games”, International Joint Conference on Neural Networks 2008 (IJCNN), 2008

[Hsueh-Min 2009] C. Hsueh-Min, S. Von-Wun, "Planning-Based Narrative Generation in Simulated Game Universes." Computational Intelligence and AI in Games, IEEE Transactions on 1(3): 200-213, 2009

[Ishii 1985] K. Ishii, M. Sugeno, “A model of human evaluation process using fuzzy measure”, Int. J. Man-Machine Studies, 22:19-38, 1985.

[Jack 2006] Yi Jack, Y. and J. Teo, “An Empirical Comparison of Non-adaptive, Adaptive and Self-Adaptive Co-evolution for Evolving Artificial Neural Network Game Players”, IEEE Conference on Cybernetics and Intelligent Systems, 2006

[John 2009] H. Johan. “A Multiagent Potential Field-Based Bot for Real-Time Strategy Games”, International Journal of Computer Games Technology, 2009

[Johansson 2008] J. Hagelback, S. Johansson, “Demonstration of multi-agent potential fields in real-time strategy games,” International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, 2008

[Jones 2001] H. Jones, M. Snyder, “ Supervisory control of multiple robots based on a real-time strategy game interaction paradigm”, IEEE International Conference on Systems Man and Cybernetics, vol 1, pp 383-338, 2001

[Kabanza 2010] F. Kabanza, P. Bellefeuille, F. Bisson, “Opponent behavior recognition for real-time strategy games,” In Proceedings of the 2010 Conference on Artificial Intelligence (AAAI-2010), Georgia, July 11-15, pp. 29-36, 2010

[Keaveney 2011]D. Keaveney and C. O'Riordan, “Evolving Coordination for Real-Time Strategy Games.” IEEE Transactions on Computational Intelligence and AI in Games, 3(2): 155-167, 2011

[Khatib 1986] O. Khabit, “Real time obstacle avoidance for manipulation and mobile robots”, Int. J Robotics Res, vol 5.1, pp.90-98, 1986

[Kuenzer 2001] A. Kuenzer and C. Schlick, “An empirical study of dynamic bayesian networks for user modeling”, Proceeding of the UM’2001 Workshop on Machine Learning for User Modeling, 2001

[Kwon 2000] S. Kwon, M. Sugeno, “A hierarchical subjective evaluation model using non-monotonic fuzzy measures and the Choquet integral”, Fuzzy Measures and Integrals—Theory and Applications: 375–391, 2000

[Liang 2005] J. Liang, P. Suganthan, K. Deb, “Novel composition test functions for numerical global optimization”, In Proceedings of IEEE Swarm Intelligence Symposium (SIS 2005), 2005

[Louis 2005] S. Louis, C. Miles, “Playing to learn: case-injected genetic algorithms for learning to play computer games”, IEEE Transactions on Evolutionary Computation, 9(6): 669-681, 2005

[Lucas 2009] S. M. Lucas, “Computational Intelligence and AI in Games”, IEEE transaction on Computational Intelligence and AI in Games, VOL. 1, 1-3, 2009

[Mehta 2009] M. Mehta, and A. Ram, “Runtime Behavior Adaptation for Real-Time Interactive Games”, IEEE Transactions on Computational Intelligence and AI in Games, 1(3): 187-199, 2009

[Michio 1994] Murofushi Michio, T. and M. Machida, “Non-monotonic fuzzy measures and the Choquet integral”, Fuzzy Sets and Systems 64(1): 73-86, 1994

[Miles 2004] C. Miles, S.J. Louis, R. Drewes, “Trap avoidance in strategic computer game playing with case injected genetic algorithms”, Lecture Notes in Computer Science, 3102, pp. 1365-1376, 2004.

[Miles 2004] C. Miles, S.J. Louis, N. Cole, “ Learning to play like a human: Case injected genetic algorithms for strategic computer gaming”, In Proceedings of the 2004 International Congress on Evolutionary Computation (CEC-2004), Portland, June 19-23, pp. 1441-1448. 2004.

[Miles 2006] C. Miles, S. J. Louis, “Co-evolving real-time strategy game playing influence map trees with genetic algorithms”, Proceedings of the International Congress on Evolutionary Computation, Portland, Oregon, 2006

[Mingliang 2010] X. Mingliang, P. Zhigeng, “Moving-Target Pursuit Algorithm Using Improved Tracking Strategy”, IEEE Transactions on Computational Intelligence and AI in Games, 2(1): 27-39, 2010

[Minsky 1954] M.L. Minsky, “Theory of neural-analog reinforcement systems and its application to the brain-model problem”, Princeton University, 1954

[Montaner 2003] M. Montaner, “A taxonomy of recommender agents on the internet”, Artificial intelligence review 19(4): 285-330, 2003

[Murofushi 2000] T. Murofushi and M. Sugeno, “The Choquet integral in multiattribute decision making”, Fuzzy measures and integrals: theory and applications, pp 333-47, 2000

[Murofushi 2000] T. Murofushi, M. Sugeno, “The Choquet integral in multiattribute decision making”, Fuzzy measures and integrals: theory and applications: 333-47, 2000

[Murofushi 2005] T. Murofushi and M. Sugeno, “Fuzzy measures and fuzzy integrals”, Fuzzy measures and integrals: theory and applications: 3–41, 2005

[Naveed 2011] M. Naveed, A. Crampton, D. Kitchin, “ Real-time path planning using a simulation-based Markov decision process”, In Proceedings of the 2011 SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, England, December 13-15, 2011

[Ontanon 2007] S. Ontanon and K. Mishra, “Case-Based Planning and Execution for Real-Time Strategy Games”, Lecture notes in Computer Science, 4626L:164, 2007

[Peter 1999] J. Peters, L. Han and S. Ramanna, “The Choquet integral in a rough software cost decision system”, Fuzzy Measures and Integrals: Theory and Applications, Studies in fuzziness and soft computing, pp 392, 1999

[Peters 1999] J. Peters, L. Han, S. Ramanna, “The Choquet integral in a rough software cost decision system”, Fuzzy Measures and Integrals: Theory and Applications, Studies in fuzziness and soft computing: 392, 1999

[Preuss 2010] M. Preuss, N. Beume, Holger Danielsiek, T. Hein, B. Naujoks, N. Piatkowski, R. Stüer, A. Thom, and S. Wessing, “Towards Intelligent Team Composition and Maneuvering in Real-Time Strategy Games.” IEEE Transactions on Computational Intelligence and AI in Games, 2(2): 82-98, 2010

[Ponsen 2004] M. Ponsen, “Improving adaptive game AI with evolutionary learning”, Delft University of Technology, 2004

[Ponsen 2005] M. Ponsen, H. Muz-Avila, “Automatically acquiring domain knowledge for adaptive game AI using evolutionary learning”, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press, 2005

[Ponsen 2007] M. Ponsen, P. Spronck, “Knowledge acquisition for adaptive game AI”, Sci. Comput. Program. 67(1): 59-75, 2007

[Ranganathan 2003] A. Ranganathan and R. Campbell, “A middleware for context-aware agents in ubiquitous computing environments”, Springer-Verlag New York, Inc. New York, NY, USA, 2003

[Reynolds 2005] R. Reynolds, Z. Kobti, “Unraveling ancient mysteries: reimagining the past using evolutionary computation in a complex gaming environment”, IEEE Transactions on Evolutionary Computation 9(6): 707-720, 2005

[Sankar 2004] K. Sankar, Simon C.K. Shiu, “Foundations of Soft Case-Based Reasoning”, Wiley-interscience, 2004

[Schank 1977] R. Schank and R. Abelson, “Scripts, Plans, Goals and Understanding”, Erlbaum, Hillsdale, New Jersey, US, 1977

[Schiaffino 2000] S. Schiaffino and A. Amandi, “User profiling with case-based reasoning and bayesian networks”, Proceeding of the International Joint Conference, 7th Ibero-American Conference, 15th Brazilian Symposium on AI, IBERAMIA-SBIA 2000, Open Discussion Track Proceedings on AI, 2000

[Sharma 2007] M. Sharma, M. Holmes, “Transfer learning in real-time strategy games using hybrid CBR/RL”, Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, pp 1041 -1046, 2007

[Shir 2006] O. Shir and T. Back , “Niche radius adaptation in the cma-es niching algorithm”, Parallel Problem Solving from Nature (PPSN IX), pp. 142-151, 2006

[Sugeno 1974] M. Sugeno, “Theory of fuzzy integrals and its applications. Ph.D. thesis”, Tokyo Institute of Technology, Tokyo, Japan, 1974

[Sugeno 1985] K. Ishii and M. Sugeno, “A model of human evaluation process using fuzzy measure”, Int. J. Man-Machine Studies, 22:19-38, 1985

[Szita 2009] I. Szita, M. Ponsen, “Effective and Diverse Adaptive Game AI”, IEEE Transactions on Computational Intelligence and AI in Games, 1(1): 16-27, 2009

[Wang 1996] Jia Wang, Zhangyuan Wang, “Using Neural Networks to Determine Sugeno Measure by Statistics”, International Journal of Neural Network, Vol.10, No.1, pp. 183-195, 1996

[Wang 1997] Z. Wang, K.S. Leung, J. Wang, “Genetic algorithms used for determining nonadditive set functions in information fusion”, Proc. IFSA'97, vol. 1, pp. 518-521, 1997

[Wang 1998] W. Wang, Z. Wang and George J.Klir, “Genetic algorithms for determining fuzzy measures from data”, Journal of Intelligent and Fuzzy Systems 6 171-183, 1998

[Wang 1999] Z. Y. Wang, K. Xu, J. Wang, “Using genetic algorithm to determine non-negative monotone set functions for information fusion in environments with random perturbation”, International Journal of Intelligent System, 14: 949-962, 1999

[Wang 2001] X.Z. Wang, D.S. Yeung, E.C.C. Tsang, “A comparative study on heuristic algorithms for generating fuzzy decision trees,” IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 31(2), pp. 215-226, 2001

[Wang 2006] Z. Wang, Y. He, “A Comparison among Three Neural Networks for Text Classification”, 8th International Conference on Signal Processing, 2006

[Wang 2008] Z. Wang, R. Yang, K. Lee, K. Leung, “The Choquet integral with respect to fuzzy-valued signed efficiency measures”, IEEE International Conference on Fuzzy Systems, 2008

[Wang 2009]X.Z. Wang, C.R. Dong, “Improving generalization of fuzzy if-then rules by maximizing fuzzy entropy,” IEEE Transactions on Fuzzy Systems, 17 (3), pp. 556-567, 2009

[Wang 2010] H. Wang, Y. Gao, X.G. Chen, “RL-DOT: A reinforcement learning NPC team for playing domination games,” IEEE Transactions on Computational Intelligence and AI in Games, 2(1), 17-26, 2010

[Wang 2011]X.Z. Wang, L.C. Dong, J.H. Yan. “Maximum ambiguity based sample selection in fuzzy decision tree induction,” IEEE Transactions on Knowledge and Data Engineering, DOI:10.1109/TKDE.2011.67, 2011

[Weijun 2010] S. Weijun, M. Rui and Y. Chongchong, “ A Study on Soccer Robot Path Planning with Fuzzy Artificial Potential Field”, 2010 International conference on Computing, Control and Industrial Engineering, pp 386, 2010

[Watson 2001] R.Watson and J. Pollack, “Coevolutionary dynamics in a minimal substrate,” Proc. Genetic Evol. Comput. Conf., 2001.

[Wilcoxon 1945] F. Wilcoxon, “Individual comparisons by ranking methods”, Biometrics Bulletin 1 (6): 80–83, 1945

[Xu 2003] K.B. Xu, Z.Y. Wang, P.A. Heng, K.S. Leung, “Classification by nonlinear integral projections,” IEEE Transactions on Fuzzy Systems, 11(2), pp. 187-201, 2003

[Yang 2007]R. Yang, Z.Y. Wang, P.A. Heng, K.S. Leung, “Classification of heterogeneous fuzzy data by Choquet integral with fuzzy-valued integrand. IEEE Transactions on Fuzzy Systems, 15(5), pp. 931-942, 2007

[Yeung 2004] D. Yeung, X. Wang, Eric Tsang, “Handling interaction in fuzzy production rule reasoning”, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34(5): 1979-1987, 2004

[Yeung 2006] SF. Yeung, J. Liu, J. Liu, J. Yan, “Detecting cheaters for multiplayer games: theory, design and implementation”, 3rd IEEE Consumer Communications and Networking Conference, 2006

[Yin 2008] L. Yin and Y. Yin, “An Improved Potential Field Method for Mobile Robot Path Planning in Dynamic Environment”, Proceeding of the 7th World Congress on Interlligent Control and Automation, pp 4847, 2008

[Yu 2006] J. Yu, and J. Teo, “An Empirical Comparison of Non-adaptive, Adaptive and Self-Adaptive Co-evolution for Evolving Artificial Neural Network Game Players”, IEEE Conference on Cybernetics and Intelligent Systems. 1-6, 2006

[Yeung 2006] S. Yeung, J. Lui, “Detecting cheaters for multiplayer games: theory, design and implementation”, 3rd IEEE Consumer Communications and Networking Conference, 2006. CCNC, 2006

[Zadeh 1994] Zadeh, A. Lotfi, “Fuzzy Logic, Neural Networks, and Soft Computing”, Communication of the ACM, March 1994, Vol. 37 No. 3, pages 77-84, 1994