Proc. of the 2014 IEEE Congress on Evolutionary Computation (CEC 2014), Beijing, China, pp. 9-12, Jul. 6-11, 2014.

Integrating Fuzzy Integral and Heuristic Search for Unit Micromanagement in RTS Games

Tung Nguyen, Kien Nguyen, Ruck Thawonmas
Intelligent Computer Entertainment Laboratory
Ritsumeikan University
Shiga, Japan
ruck@ci.ritsumei.ac.jp

Abstract— **Real-time strategy (RTS) is a sub-genre of strategy video game which typically involves resource gathering, base building, strategy planning, and combat scenarios. With complicated gameplay, vast state and action spaces, RTS games have been proven to be an excellent platform for artificial intelligence research. One of the most challenging problems posed by RTS games is the detailed control of units in combat, i.e., unit micromanagement. In this paper, we present a method of integrating fuzzy integral and fast heuristic search for improving the quality of unit micromanagement in the popular RTS game StarCraft. Experiments are reported at the end of this paper, showing promising results and the potential of the proposed method in this domain.**

Keywords—fuzzy integral; heuristic search; RTS games; StarCraft; unit micromanagement

I. INTRODUCTION

Developing AI systems for RTS games has gained increasing attention in the AI research community in recent years [1]. In a typical RTS game, such as StarCraft, each player starts with a main base and a number of workers. The workers can be used to gather one or more types of resources, and to construct buildings and expansions. The player must spend resources to train his army with the ultimate goal of destroying all units and buildings of the enemy. With fast-paced gameplay and the possibility of simultaneous moves, RTS games have reached a level of complexity unseen in other traditional games like Chess or Go.

In StarCraft, unit micromanagement not only is the key to winning a battle but also decides the result of the whole game. With high quality micromanagement, one side can completely destroy the other side which commands even more or stronger units. Although there have been several AI studies on this domain, bots whose unit behaviors are predefined via static scripts still predominate in AI competitions. However, due to their static nature, they are highly exploitable and can be countered quite easily by using appropriate countermeasures. Therefore, a recent trend in unit micromanagement is using search-based techniques to dynamically control units while still considering the collaboration among them. Some state-of-the-art methods such as Alpha-Beta [2], UCT [3], and Monte Carlo planning [4] have been applied and achieved dominance over script-based techniques.

It is known that the performance of a search-based technique relies a lot on its heuristic evaluation function as it is required in almost all search algorithms. However, in StarCraft, heuristic functions tend to be very generic (like LTD and LTD2 used in [2]) and cannot fully capture the game state, especially when multiple types of units can interact and boost each other. In this paper, we present a fuzzy-based evaluation that can accurately evaluate the game state even when there are several unit types involved. This method when combined with existing heuristic search can result in a better performance than the original algorithms.

II. BACKGROUND

A. Non-additive Properties in RTS Games

There may be dozens of unit types that can interact in different ways in any RTS game. This leads to enormous numbers of possible unit combinations. A combination of two unit types may result in greater or less effectiveness than the sum of their individual impacts. Suppose $\mu(X)$ is the measure of effectiveness provided by unit combination X. In RTS games, it is common that $\mu(X)$ is non-additive, i.e., $\mu(X_1 + X_2) \leq \mu(X_1) + \mu(X_2)$ or $\mu(X_1 + X_2) \geq \mu(X_1) + \mu(X_2)$. For example, a combination of Siege Tank and Vulture is a standard tactic in the popular RTS game StarCraft. Siege Tank (normally used in siege mode) is a powerful unit with massive damage and very long attack range. However, it has a minimum attack range of 2, making it quite weak against melee units. On the other hand, Vulture is an extremely mobile unit which moves quickly enough to be able to "hit and run" against most melee units without risk. Thus, Siege Tank and Vulture are widely used to protect each other. In this case, we can suppose $\mu(\{Tank, Vulture\}) > \mu(\{Tank\}) + \mu(\{Vulture\})$.

B. Fuzzy Measure and Fuzzy Integral

In this sector, we review the concept of fuzzy measure and Choquet integral.

Definition 1: A fuzzy measure on a measurable space (X, F) is a real-valued set function $\mu: F \rightarrow \mathbb{R}$ satisfying:

1) $\mu(\emptyset) = 0$,
2) $\mu(A) \leq \mu(B)$ whenever $A \in F, B \in F, A \subseteq B$.

Definition 2: A non-monotonic fuzzy measure on (X, F) is a real-valued set function $\mu: F \rightarrow \mathbb{R}$ satisfying $\mu(\emptyset) = 0$.

Non-monotonic fuzzy measure matches well with non-additive properties in RTS games and can be trained by machine learning methods.

After defining a set of fuzzy measures, we can use fuzzy integral to calculate the expected utility of an uncertain event. As a classical fuzzy integral, Choquet integral has been successfully applied in many areas such as statistical mechanics and potential theory. Its definition is as follows.

Definition 3: Let μ be a fuzzy measure on X. The discrete Choquet integral of a function $f: X \to \mathbb{R}^+$ with respect to μ is defined by

$$\int f(x) \circ \mu(X) = \sum_{i=1}^{n} \Big(\big(f(x_i) - f(x_{i-1})\big) \times \mu(\{x | f(x) \geq f(x_i)\}) \Big) \quad (1)$$

where $0 = f(x_0) \leq f(x_1) \leq f(x_2) \leq \cdots \leq f(x_n)$.

III. RELATED WORK

So far, there has been only a limited number of AI research incorporating fuzzy measure and fuzzy integral into RTS game agents by Y.J. Li et al. [5]. They applied different fuzzy integrals to solve the unit selection problem. The key to winning in most RTS games is to build up a strong army with appropriate unit types which can gain massive destroy power against enemy army. Different unit combinations give different effectiveness, therefore, to estimate the power of each unit combination becomes one of the most essential tasks in the game. Due to feature interactions existing among different unit types, the effectiveness of a unit combination cannot be simply calculated by using weighted average. They therefore tried to apply fuzzy integrals for that calculation and proposed three new fuzzy integrals and compared with the classical Choquet integral. The details of those fuzzy integrals are given as below. Note that X is a unit combination, x is a unit type, $f(x)$ is defined as the proportion of x and the fuzzy integral returns the effectiveness of that combination.

A. Max-based Fuzzy Integral

$$\int f(x) \circ \mu(X) = \sum_{i=1}^{n} \Big(f(x_i) \times max\big(\mu(S_i)\big) \Big) \quad (2)$$

where $x_i \in S_i$, n is the number of unit types.

Max-based fuzzy integral is developed with the policy of winner takes all. It considers only the most powerful unit combination which involves the current unit type.

B. Mean-based Fuzzy Integral

$$\int f(x) \circ \mu(X) = \sum_{i=1}^{n} \left(f(x_i) \times \frac{1}{m_i} \sum_{j=1}^{m_i} \mu\big(S_{ij}\big) \right) \quad (3)$$

where $x_i \in S_{ij}$, n is the number of unit types, m_i is the number of sets which include x_i.

Mean-based fuzzy integral is calculated considering all the interactions that are related to each unit type. All the fuzzy measures which involve the current unit type will be selected and the average value is computed.

C. Order-based Fuzzy Integral

$$\int f(x) \circ \mu(X) = \sum_{i=1}^{n} \big(f(x_i) \times \mu(\{x | f(x) \leq f(x_i)\}) \big) \quad (4)$$

where n is the number of unit types, $f(x_1) \geq f(x_2) \geq \cdots \geq f(x_n) > 0$.

Order-based fuzzy integral takes into account the unit production sequence in RTS games. Data analysis shows that advanced units often dominate the proportion in the army and thus should be considered as having more cooperation with other units. Each unit type will calculate the interaction with the one having less production than it, and the largest $f(x)$ is combined with the fuzzy measure of all unit types.

All these fuzzy integrals have been proven to give better results than the classical Choquet integral when applied to RTS games. However, because their research focuses on strategy planning (i.e., the main purpose is to find the most powerful unit combination to produce), the function $f(x)$ is defined without considering the unit number of each types and the properties of individual units (damage, cooldown, current hit points). As an example, suppose that there are three different unit combinations as follows: 1) 5 Protoss Zealots and 5 Protoss Dragoons, each unit having 50 hit points; 2) 5 Protoss Zealots and 5 Protoss Dragoons, each unit having 100 hit points; 3) 10 Protoss Zealots and 10 Protoss Dragoons, each unit having 100 hit points. It can easily be seen that the fuzzy integrals mentioned above return the same value for all three combinations. Thus, this prompted us to come up with another definition of $f(x)$ when we apply those fuzzy integrals to our problem—unit micromanagement.

IV. METHODOLOGY

Our main idea is to improve the quality of evaluation functions used in existing search algorithms by applying fuzzy measure and fuzzy integral to estimate the value of a game state. In our problem, we define that value as the difference between the power of the player's army and the enemy's army, where the power of one's army is estimated by calculating the following fuzzy integral:

$$Army\ power = \int f(x) \circ \mu(X) \quad (5)$$

Here $\mu(X)$ is the fuzzy measure of the corresponding unit combination and can be considered as its contribution rate for the total power. $f(x)$ is a unit statistic function defined by

$$f(x) = \frac{1}{N} \sum_{i=1}^{n} \frac{hp(u_i)}{max\_hp(u_i)} \quad (6)$$

where hp denotes hit points, x is a unit type, u_i is a unit whose type is x, n is the unit number of type x, and N is the maximum number of units that a player can have in the game. Note that when all units have the maximum number of hit points, the value of $f(x)$ is equal to the proportion of unit type.

A. Data Collection and Analysis

We selected StarCraft: Brood War (SC: BW) as our research platform. It is a military science fiction, real-time strategy game released by Blizzard Entertainment in 1998. As of May 2007, SC: BW has sold almost ten million copies and became one of the most popular video games of all time. For AI researchers, SC: BW is also an ideal test bed for AI algorithms thanks to the BWAPI (Brood War API [6])'s comprehensive interface into the game engine.

300 replays of professional one-versus-one SC: BW games are collected from the Internet. Due to time constraints, we decided to focus only on Protoss vs. Protoss match-up and games that involve 8 most used unit types. After analyzing the replays, we obtained the data of 940 battles. Before and after each battle, the unit statistics and the score of both players were recorded. Those scores were given by the game system for destroying enemy units and used to learn the fuzzy measure in our research.

B. Learning Fuzzy Measure by Genetic Algorithm

There is a relation between the score a player obtains in each battle, the power of his army and the result of the battle. Data analysis shows that the higher the score, the more powerful the army and the higher chance of winning. Thus, we can consider the estimated power of the army as the expected value of the score. This leads to the problem of finding the fuzzy measure that "best" fits the collected data. In this paper, we use a genetic algorithm approach to solve this problem.

Each chromosome is made up of $2^n - 1$ fuzzy measures corresponding to all possible unit combinations, i.e. $\mu(\{x_1\})$, $\mu(\{x_2\})$,..., $\mu(\{x_1, x_2\})$,..., $\mu(\{x_1, x_2, \dots, x_n\})$ where x_1 , x_2,..., x_n represent n kinds of unit types. In our problem, $n = 8$ and real-valued encoding is used, which results in the length of a chromosome being 255. The fitness calculation of a chromosome is described as follows:

1. Extract the fuzzy measure from that chromosome.
2. Extract real scores and values of the unit statistic function from the replays, normalize those real scores between 0 and 1.
3. Calculate the estimated scores by using fuzzy integral as (5).
4. Calculate the root mean square error.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(score_{est} - score_{real})^2} \qquad (7)$$

5. Calculate the fitness.

$$Fitness = \frac{1}{1 + RMSE} \qquad (8)$$

In our GA, a mixture of roulette wheel selection and elitist selection was used to construct a new population. Population size, the number of generations, crossover rate, and mutation rate were set to 1000, 500, 0.75, 0.05 respectively. The result of the learning process is illustrated in Fig. 1. Shown are the best fitnesses obtained by using different fuzzy integrals: max-based fuzzy integral, mean-based fuzzy integral, and order-based fuzzy integral. As can be seen, order-based fuzzy integral showed the best performance among three methods, which is the same result as obtained by Y.J. Li et al. in [5]. Thus, only order-based fuzzy integral and the corresponding fuzzy measure will be used in our evaluation below.

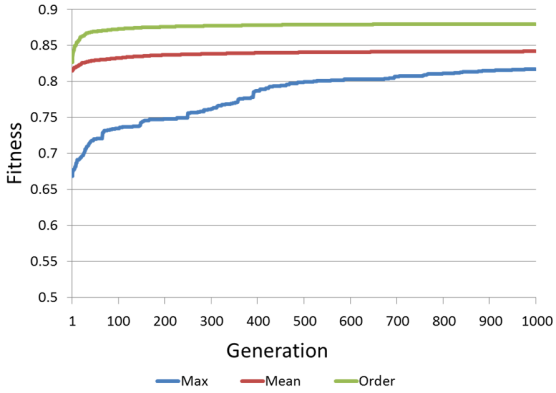

Fig. 1. Best fitnesses at each generation obtained by different fuzzy integrals. The best value is 0.88.

V. EVALUATION

Experiments were carried out on the simulator SparCraft [7] to compare the performance of the proposed method and other evaluation methods often used in this field. SparCraft is an open source StarCraft combat simulation package with a high level of accuracy. Researchers can easily implement new algorithms and integrate them into this simulator and use it as a test bed for AI research. Two state-of-the-art search algorithms (Alpha-Beta and UCT) and a novel greedy search algorithm called Portfolio Greedy Search are already implemented and will be used in our experiments. Portfolio Greedy Search does not perform any recursive tree search, but instead relies on heuristic evaluations using deterministic playouts at the root node. D. Churchill and M. Buro showed that it can outperform both Alpha-Beta and UCT for large StarCraft combat scenarios. For further details we refer the reader to [3].

A. Experiments

Those search algorithms implemented in SparCraft evaluate a game state by performing deterministic script-based game playouts. By assuming both players use the same scripted policy and performing a playout, it is able to estimate which player has an advantage at a given state. After a playout reaches its terminal condition, the following evaluation formula is called.

$$LTD2(s) = \sum_{u \in U_1} \sqrt{hp(u)} \times dpf(u) - \sum_{u \in U_2} \sqrt{hp(u)} \times dpf(u) \qquad (9)$$

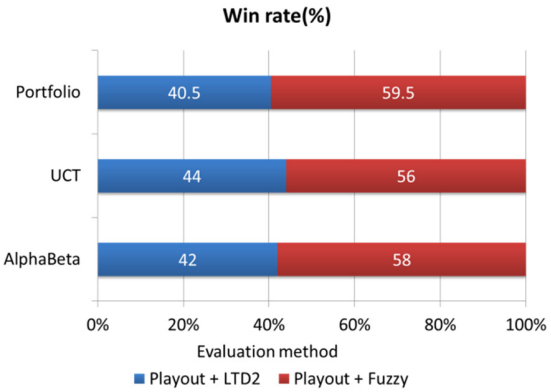

Fig. 2. Average win rates of the proposed evaluation method against the existing method when applied in different search algorithms.

Here hp denotes hit points, dpf denotes damage per frame, U_1, U_2 are set of units controlled by player 1 and player 2 respectively. It has been shown in [2] that the combination of playouts and LTD2 formula gives much better performance than using only LTD2 formula. In our experiments, we replaced LTD2 formula by fuzzy-based evaluation to evaluate the results of the playouts, and compared the performance of the new combination with the original one.

Each experiment consisted of 4 combat scenarios, in which two players control similar armies of 24 Protoss units. In order to mimic real battles in the game, the number of unit types was set to vary from 1 to 4 and the number of units of each type was generated randomly. At the beginning of each battle, two forces are placed separately but symmetrically around the X-axis of the map.

B. *Results*

50 games were played for each combat scenario, giving 200 total games for each pair of opponents. The performance of the proposed method against the existing one is presented in Fig. 2. It can be seen that the win rate of fuzzy-based evaluation exceeded 50% in all combat scenarios, i.e., it helped improve those search algorithms mentioned above. The most significant difference is obtained when integrating the proposed method into Portfolio Greedy Search. This must be because Portfolio Greedy Search does not perform any recursive tree search and relies most on the goodness of heuristic evaluations. Besides, as shown in Fig. 3, the proposed method tends to achieve better results in combat scenarios in which more unit types are involved and interact with each other.

VI. CONCLUSION

In this paper, we presented a new approach for constructing an evaluation function, essential in most search algorithms when applied to computer games. We learned the fuzzy measure from real game data and used it to calculate the order-based fuzzy integral as estimation for the goodness of a game state. We carried out experiments with various settings to evaluate the proposed method and the results are encouraging. We were successful in improving the existing search methods and achieved a better quality of unit micromanagement.

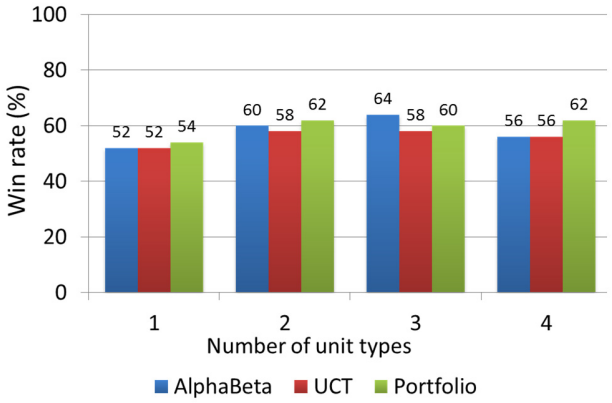

Fig. 3. Detailed win rates of the proposed evaluation method against the existing method when applied in different search algorithms.

For future work, we want to incorporate the search algorithms using fuzzy-based evaluations into our StarCraft bot—ICEbot[1]. We plan to collect more data and perform clustering in order to obtain better fuzzy measures that can handle more scenarios seen in the game. In addition, it is also our intention to combine the proposed method in this paper with one of our previous works that utilizes potential flow for positioning combat units [8], and aim for a more human-like StarCraft agent.

REFERENCES

[1] S. Ontanon, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill and M. Preuss, "A survey of real-time strategy game AI research and competition in StarCraft," IEEE Transactions on Computational Intelligence and AI in Games, Dec.2013, pp. 293–311.

[2] D. Churchill, A. Saffidine and M. Buro, "Fast heuristic search for RTS game combat scenarios," Proc. The Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, California, Oct.2012, pp. 112–117.

[3] D. Churchill and M. Buro, "Portfolio greedy search and simulation for large-scale combat in StarCraft," 2013 IEEE Conference on Computational Intelligence in Games, Niagara Falls, ON, Aug.2013, pp. 1–8.

[4] Z. Wang, K. Nguyen, R. Thawonmas and F. Rinaldo, "Monte-Carlo planning for unit control in StarCraft," 2012 IEEE 1st Global Conference on Consumer Electronics, Tokyo, Oct.2012, pp. 263–264.

[5] Y.J. Li, P.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," 2011 IEEE International Conference on Fuzzy Systems, Taipei, Jun.2011, pp. 170–177.

[6] A. Heinermann et al. BWAPI, An API for interacting with StarCraft: BroodWar [Online]. Available: http://code.google.com/p/bwapi/

[7] D. Churchill. SparCraft, StarCraft Combat Simulation [Online]. Available: http://code.google.com/p/sparcraft/

[8] T. Nguyen, K. Nguyen and R. Thawonmas, "Potential flow for unit positioning during combat in StarCraft," 2013 IEEE 2nd Global Conference on Consumer Electronics, Tokyo, Oct.2013, pp. 10–11.

[9] F. Sailer, M. Buro, and M. Lanctot, "Adversarial planning through strategy simulation," 2007 IEEE Symposium on Computational Intelligence and Games, Honolulu, HI, Apr.2007, pp. 80–87.

[10] Z. Wang and G.J. Klir, Fuzzy Measure Theory. Springer, 1992.

[11] M. Mitchell, An Introduction to Genetic Algorithms. MIT Press, 1998.

[1] ICEbot has participated in Student StarCraft AI Tournament (SSCAI http://sscaitournament.com/) since 2012. It was ranked 1st and 2nd in Mixed division of SSCAI 2012 and SSCAI 2013 respectively.