Proc. of the 4th Annual Asian GAME-ON Conference on Simulation and AI in Computer Games (GAMEON ASIA 2012), Kyoto, Japan, pp. 33-35, Feb. 24-25, 2012.

USING MONTE-CARLO PLANNING FOR MICRO-MANAGEMENT IN STARCRAFT

Wang Zhe, Kien Quang Nguyen, Ruck Thawonmas, and Frank Rinaldo
Intelligent Computer Entertainment Laboratory
Ritsumeikan University

KEYWORDS

RTS games; StarCraft; Monte-Carlo Planning; Micro Management

ABSTRACT

This paper presents an application of Monte-Carlo planning for controlling units in a RTS game StarCraft. We developed an original simulator for applying Monte-Carlo Planning (MCPlan) to solve the problem of random and simultaneous moves in the RTS game. We also apply an $\varepsilon - greedy$ algorithm to model the opponent in a simulation for improving MCPlan's performance. Experimental results are provided at the end of this paper, which shows the potential of MCPlan in this domain.

I. INTRODUCTION

StarCraft is one of the most popular RTS game developed by Blizzard Entertainment. The extremely balanced gameplay and easy access to the game engine not only provides players with multiple options, but also enables it to be an ideal platform to test different AI approaches. So far, although some work has been done on building human-level AI for StarCraft, because of the real-time properties and largely unpredictable random, it is still a challenging game for AI research.

The main issue in this paper focuses on the micro management of StarCraft gameplay. Micro-management is a series of actions that issue commands to each unit of a certain group for maximizing their effectiveness during combat. Good micro management is a significant part of RTS game in which it can bring player advantages in the game, even change the results of a whole match.

There have been a number of previous works that have been done on both unit control issues and the application of MC simulation in RTS games. Unit control problems are usually handled by finite state machines, script or neural networks etc. For example, Weber and Michael used hand authoring ABL behaviors to handle micro-management task in StarCraft (Weber 2011). A Bayesian model is applied to StarCraft for unit control by Gabriel and Pierre (Synnaeve 2011). Monte-Carlo method in RTS games also can be found from previous work. Such as applying MCPlan for high-level planning (Chung 2005), or combining Monte-Carlo method with non-linear value function approximation (VFA) and text recognition technique as a solution for large sequential decision making problems (Branavan 2011). However, these works mainly focus on high-level planning and MC simulation can rarely be found in low-level AI modules such as micro management.

Since MC simulation has an advantage of selecting the best choice without relying on too much expert knowledge, this enables the AI to explore possibilities and perform creatively. Therefore, we try to build an expert AI for micro-management in StarCraft by applying MCPlan and test how it performs.

The contributions of this paper are as follows:

- Implementation of the MCPlan with expert knowledge for micro-management in StarCraft.
- Design of a simulator for complex commercial RTS game combat scenario and a general characterization.

II. METHODOLOGY

Monte Carlo Planning

MCPlan is a mechanism that is based on simulation and does a stochastic sample of possible choices. It determines the best statistical result after multiple roll-outs. It is an effective method to handle random and imperfect information problems with alternating moves, such as chess and poker. A great advantage of MCPlan is the reduction of expert knowledge required, instead of

defining every detail through expert biases; MCPlan relies on an effective evaluation function.

Simulator Design

As one of the most important parts in MC Simulation, the simulator is a model that is used to simulate reality and aid in making predictions. In our case, the simulator is for creating possible game scenarios of the near future. However, unlike high-level strategy, for a micro-management simulator, it has to emulate real-game moves as much as possible as even a small detail could affect the game result greatly.

Generally speaking, three major works in our simulator are as follows:

A) *Character modeling:* Characters should be exactly the same as in real game, including character's hit-points, attack range, special skills etc.

B) *Map modeling:* Map modeling should consider of different terrains, characters' location, and unit overlapping problem.

*C) Enemy behaviour m*odeling: Rather than randomly move, we adopt ε -greedy algorithm to define enemies' movement in simulator. The evaluation function and plans for enemy heuristic are basically the same as MCPlan AI.

III.APPLICATION TO STARCRAFT

We apply our MCPlan algorithm to micro-management part of the game Starcraft by accessing the game engine through BWAPI.

Plan Definition

We try to avoid too complicated a plan in order to give MCPlan enough space for search. But expert bias is still needed. So we mix few complex plans with simple plans. In this way, we not only avoid too much expert knowledge, but also try to improve the effectiveness of plans.

Simulation roll-out

Each roll-out contains two plans: one is fixed and the other one is stochastic. The fixed one is the certain plan that is being chosen for simulation, so it is always be executed first. After that, the program will randomly pick another plan and continue the simulation process. After simulating both plans, the evaluation step begins.

Search Algorithm

Here's our algorithm for searching best plan, namely, UCB1 algorithm.

$$UCB1(i) = R_i + C\sqrt{\frac{\ln N}{N_i}} \qquad (1)$$

In the formula, i is index of each plan, R_i presents the reward that plan i obtained from the evaluation function. C is a predefined constant, N and N_i are the overall number of run times and number of times that plan i has been visited respectively.

The basic view of our MC simulation is as follows:

1) Loading predefined plans one by one to the simulator.
2) Simulate each plan (both fixed one and random one), evaluate the whole roll-out and reward the plan.
3) Choose the plan with the highest evaluation based on UCB1 function, then simulate and reward it again.
4) Repeat step 3.
5) Choose the plan with best average result for the AI player in a real game.

Evaluation function

An evaluation function is used for measuring the effectiveness of each plan in different situations. Our evaluation function is designed based on individual units and basically contains 4 aspects as seen below:

$$Q = \omega_1 HP + \omega_2 DM + \omega_3 SP + \omega_4 EG \qquad (2)$$

In this function, four elements are individual unit's hit-point, damage to enemy, move speed and remained energy respectively. And we manually set value for the four weights ω based on expert bias.

IV.EXPERIMENTS

We designed three experiments to test MCPlan AI's by comparing with other subjects. Each experiment is a different combat scenario of Terran against Zerg. Experiments ran on PCs with 3.20GHz CPU and 4G RAM.

Experimental Scenarios

We did all the experiments in a small map with all plain terrain and limited space (game map size 16×16). Two forces were placed in short distance but can't attack each other at the beginning of game in order to make sure all units get involved in combat as soon as possible.

The game techniques and skills that be used are : U-238 Shells, Stim packs for Terran Marine, healing for Terran Medic and Metabolic Boost for Zerg Zergling. For more detail of units and skills, we refer reader to blizzard official website.

***Experiment 1*:** Four Terran Marines against six Zerg Zerglings.

***Experiment 2*:** Three Terran Marines aginst 1 Zerg Lurker.

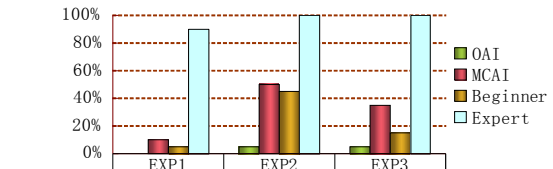
***Experiment 3*:** Five Terran Marines and one Terran medic aginst six Zerg Zergling and 1 Zerg Lurker.

In order to highlight the effectiveness of micro management, unbalanced military forces were distributed to two sides. That is, Zerg force has advantages in all experiments and is always controlled by the original AI of StarCraft. We then applied different subjects to control the Terran force against the Zerg. They are the original AI of StarCraft, MCPlan AI, and various-level human players respectively. Moreover, the difficulty of the experiments are descending (E1>E2>E3), whereas the complexity of scenarios are ascending (E1<E2<E3).

We ran each experiment 20 times for each subject, and human players were allowed to get familiar with each experiment by 5 trials and then compared their performance by wins rate.

V. EXPERIMENT RESULTS

The experiment results are shown in Figure 1. The results in this figure show the win rates of different subjects. It shows that the original AI can hardly win a game in all experiments under disadvantage situation and even failed all games in experiment 1. The results also suggest that MCPlan AI has potential to overcome unbalanced number of units to defeat weaker AI, and has better performance than beginner players, but still far from expert human player whose wins rate over 90% on the average.



	EXP1	EXP2	EXP3
OAI	0%	5%	5%
MCAI	10%	50%	35%
Beginner	5%	45%	15%
Expert	90%	100%	100%

Figure 1: Win rate of subjects in three experiments

**OAI: Original AI of StarCraft *MCAI: Monte-Carlo Planning AI*

VI. CONCLUSION AND FUTURE WORK

This paper has presented a preliminary work on solving the problem of micro-management in RTS games. We have described a mechanism of applying MCPlan to the game Starcraft. After we analyzed and identified the domain, we successfully developed a simulator for the game and tested our method. Our work includes plans design, game process simulation and evaluation function creation. From our experiments, we got initial results which indicated a promising potential of MCPlan in this domain, but still not high enough for real play.

For future work, we intend to improve simulator which requires more accurate game simulation and more efficient evaluation function. We also try to seek a better way to improve simulation speed, which is an essential problem in Monte-Carlo simulation.

REFERENCES

B. Weber, M. Mateas, and A. Jhala, "Building Human-Level AI for Real-Time Strategy Games", in *AAAI Fall Symposium on Advances in Cognitive Systems* (*ACS 2011*).

M. Chung, M. Buro, and J. Schaeffer, "Monte Carlo Planning in RTS Games", in *IEEE Symposium on Computational Intelligence and Games* (*CIG*), 2005.

S.R.K. Branavan, D. Silver, R. Barzilay, "Non-Linear Monte-Carlo Search in Civilization II," *in IJCAI* 2011.

G. Synnaeve and P. Bessiere, "A Bayesian Model for RTS Units Control Applied to StarCraft," in *IEEE Conference on Computational Intelligence and Games* (*CIG*), 2011.