Innovations in Information and Communication Science and Technology
Third Postgraduate Consortium International Workshop
E. Cooper, G.A. Kobzev, A.F. Uvarov, and V.V. Kryssanov (Eds.)
IICST 2013: pp. xx-xx.

DO NOT DELETE THIS TEXT-BOX

ADOPTING SCOUTING AND HEURISTICS TO IMPROVE THE AI PERFORMANCE IN STARCRAFT

Wang Z., Nguyen Q.K., Thawonmas R., and Rinaldo F.

Intelligent Computer Entertainment Laboratory, Ritsumeikan University, ruck@ci.ritsumei.ac.jp

ABSTRACT

StarCraft is one of the most famous Real-Time Strategy (RTS) games, and there have been several competitions based on AI bots. In this game, the player often gathers information about their enemy based on a scouting task and from that recognizes and prepares a good strategy to counter the enemy plan. However, these tasks are considered difficult to implement with AI bots. In this paper, heuristic methods are proposed to recognize enemy plans and calculate reasonable attack timing decision. We also propose an effective method of scouting by adopting a potential field. These approaches were applied to a StarCraft AI bot, this bot has won the first place in mixed division of Student StarCraft AI Tournament 2012(SSCAI), the result is also discussed in this paper.

Key words: Real-Time Strategy, Potential Field, Game AI.

1. INTRODUCTION

StarCraft is one the most popular RTS game in the recent decade, it was developed by Blizzard Entertainment in 1998. Like many traditional RTS games, the complicated game play is not only difficult for human players to master, but also provides many challenging problems for AI research. The game starts with a main base and a few workers, players must send workers to gather resources and then produce more workers, build more expansions and train an army to destroy the enemy. However, this process requires not only a lot of skill, but also plenty of expert knowledge. In addition, the uncertainty created by the fog-of-war increases the challenge of game play, which means players also have to deal with imperfect information during game play.

Besides skillful macro-management and micro-control, there are still two other significant factors for winning a game in StarCraft. One is information collection and the other is precise judgment, including correct decision making and right attack timing. In the imperfect environment of game play, the only way to overcome the uncertainty is to keep sending units to gather information about the enemy. However, the scout unit is usually weak and easily becomes the attack target of the enemy, so a mechanism to keep the scout unit alive is necessary. In terms of decision making, however, since it heavily relies on plan recognition and expert bias, it is much more complex for bots to accomplish. Currently, the majority of bots in Starcraft are still depend on script strategies and use the same attack timing. The unchangeable plans usually result in a boring game, or even a loss.
In this paper, in order to solve the problems above, we proposed a scouting mechanism for information collection.

Particularly, we adopt a potential field to keep scout unit alive as long as possible. At the same time, flexible methods are taken to improve efficiency and effectiveness of the scouting task. As far as decision making, a "mental agent" is also implemented to tackle enemy plan recognition as well as the attack timing estimation. Instead of using complicated algorithms, our heuristic method is easier to apply and more effective especially at early stages of the game.

Our bot -IceBot- is a StarCraft Terran bot based on multi-agent architecture. A finite state machine (FSM) is used in this project to control the bot's state transitions during game flow. We submitted our bot to SSCAI 2012, it was ranked 1st in the elimination bracket and 4th in the round robin tournament with over 80% win rate.

The contributions of this paper are as follows:

- Provide an extendible framework for easily creating strategy in StarCraft.
- Provide an effective scouting mechanism to solve information gathering issue in RTS game.
- Provide a heuristic method for recognizing enemy strategy and deciding attack timing.

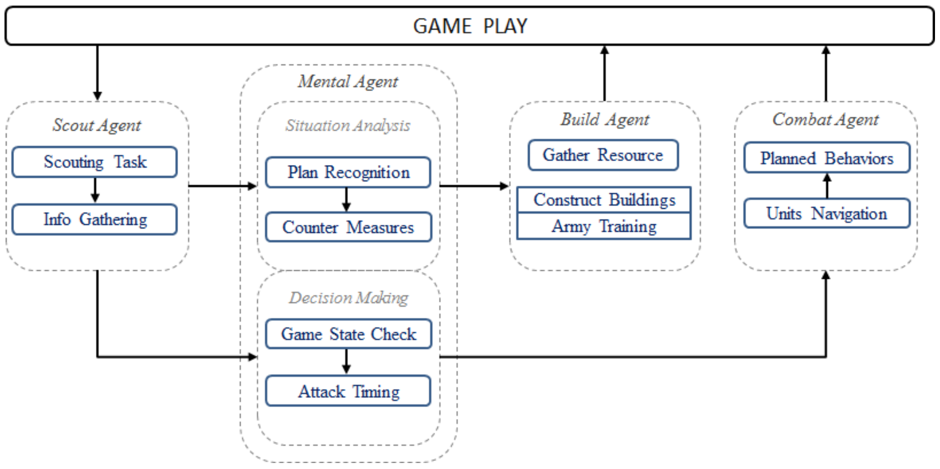

Fig. 1. System framework

2. RELATED WORK

So far, much research has been done on different topics in StarCraft, basically they can be divided into unit navigation and strategy recognition. For unit navigation, research mainly focus on micromanagement in combat and early scouting. Including our previous work on unit micromanagement by adopting a Monte-Carlo method which based on multiple simulation to find the best choice (Wang *et al.*, 2012), a Bayesian model for unit micro control (Synnaeve and Bessiere, 2011), a reinforcement learning for small-scale combat scenario (Wender and Watson, 2012), and a heuristic search is also applied to micromanagement (Churchill *et al.*, 2012). As for strategy prediction, Gabriel et al. proposed a Bayesian model to predict opponents opening by scouting enemy's technology trees and counting building numbers (Synnaeve and Bessiere, 2011), which has advantages to deal with imperfect information in the game. Park et al. applied various machine learning algorithms to predict opponents plans based on the scouted information (Park *et al.*, 2012), they also proposed a heuristic rule to navigate the scout unit. In their work, an algorithm for navigating a scout unit was also proposed using a number of heuristic rules. Besides of collecting information from the game, a data mining approach is also applied for strategy prediction by researching the game replays (Weber and Mateas, 2009).

In this paper, we also proposed the usage of a potential field in unit navigation. However, other than using potential field for micromanagement in combat (Hagelback, 2012), we mainly focus on navigating scouting unit in early game. Moreover, we also proposed a heuristic method for dramatically attack timing determination, which is still a rare but significant research despite the existence of considerable work.

3. METHODOLOGY

3.1 System Framework

We apply a multi-agent architecture to handle complex game play in our project. Each agent is responsible for one of several tasks, such as build task, attack task etc. Specifically, our agents can be functionally divided into four parts, say, build agent, combat agent, scout agent, and mental agent as shown in Figure 1.

- Build Agent: This agent consists of several sub-managers, to accomplish basic tasks such as gather resources, construct buildings, and produce fighters. In order to avoid conflict between tasks, each task binds with a certain priority before being added to the task queue. Finally, through the collaboration between sub agents, tasks will be executed one by one to keep the game-flow going.
- Combat Agent: Responsible for units' behavior control and navigation. Various behaviors will be conducted depend on different game states and types of unit. It is also affected by mental agent on unit navigation.
- Scout Agent: Consist of scout manager and information manager. For collecting and storing raw information of both sides (ours and opponent) in the game. Information gathering is conducted by scout agents, and keeps this knowledge in the information manager. This part will be explained in a later section.
- Mental Agent: This agent is made up of the mental manager and the plan manager. It functions as the brain of all agents, including game flow control, game state recognition and all kinds of decision making. This part is also explained in a later section.

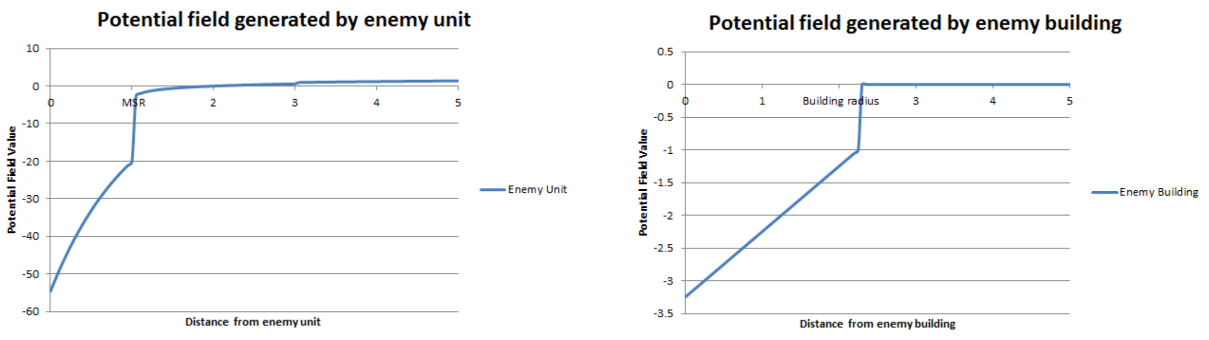

Fig. 2. Examples of potential field generated by enemy units. The left one is a potential field generated by an enemy fighter while the right one is a potential field generated by an enemy building. Here MSR is enemy fighter's maximum attack range. The negative value stands for repulsive force while positive value is attractive force

3.2 Scout Agent

The scout agent is one of the most significant agents in our bot, the flow chart is shown in Figure 3.

3.2.1 Scouting in StarCraft

Scouts are essential in the StarCraft game play, because of the existence of the fog-of-war, which covers the majority of the game map. For each player in the game, only a certain range is visible where his units (army or building) stay. Therefore, without sending reconnaissance units to enemy's territory for gathering information, one will be blind and it will be difficult to come up with a tactical plan to defeat the opponent. One might possibly end the game without knowing the enemy's base position.

The method we propose consists of three steps.

- Find the enemy's main base. In StarCraft, each map has a certain number of start locations, and players will be assigned to any of them randomly. So in the first step, a recon unit just needs to search those locations one by one until it finds the enemy base.
- After finding the enemy base, our scout unit begins to randomly move to the enemy base, and gathering information as much as possible.
- If there are any enemy fighters inside the scout unit's sight range, a potential field will be triggered to keep scout unit alive as long as possible. A repulsion force will be generate on each enemy unit, so that the scout unit will manage to avoid being attacked while keep up its scout behavior.

3.2.2 Applying Potential Field for Scouting

In our implementation when one or more enemy fighters come into therecon unit's sight range, a potential field will be triggered immediately. Thus either attractive force or repulsion force is generated on every unit and terrain to affect recon's moving path.

- Recon Unit Movement: We calculate the potential field values in eight directions from the current position of the scouting unit. The agent then will move along the direction which holds the most attracting force in potential field.
- Enemy Building: We encourage the scout unit to keep distance from constructions to decrease the possibility of being blocked by them. To achieve this, a repulsion force will be generated on the enemy's buildings when the distance between scout unit and the building is smaller than the building's size (Figure 2).
- Enemy Fighter: In terms of enemy fighters and defensive towers, the scout unit should always stay out of their maximum attack range (MSR). In this case, a strong repulsion force is generated inside the enemy shooting range, while the position where is far from enemy will be slightly increased with the distance to encourage the scout unit run away from dangerous (Figure 2).
- Neutral Unit and Terrain: In StarCraft, besides players' units, there also have some neutral units, such as

animals and resource fields. These units are considered as obstacles when the potential field is active. In addition, we also add repulsion force to some special terrain, because they are also possible to block the scout unit when it's moving.

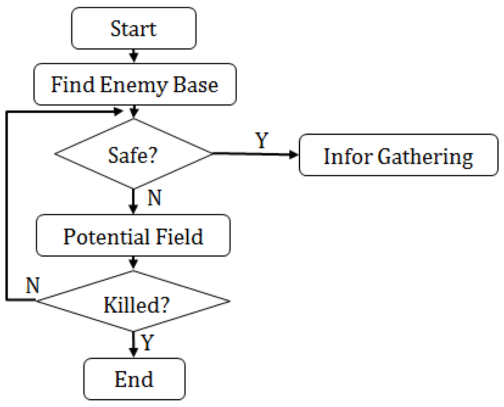

Fig. 3. The scouting procedure

3.3 Mental Agent

Game-flow control and Finite State Machine: FSM is well-known and widely used in many areas for process control as well as modeling application behavior. In our case, FSM is applied to control the whole game process.

Although there are no coercive rules of StarCraft game play, still there are some patterns that need to follow. For instance, players usually produce workers and construct basic buildings before training fighters. Thus, our bot always follows a fixed opening in the early stage of game. And after that, the transition between states heavily relies on scout agent and mental agent. A scout agent provides significant raw data, while a mental agent fusing those data and then triggers next state.

Plan recognition: Enemy plan recognition is a critical part for subsequent decision make, without it a bot can only follow a simple script, and easily lose the game to the changeable opponents. However, suppose we already successfully finish the scout task, it is still difficult to fusing lots of raw data to predict enemy's plan. In this case, a heuristic technique is applied in our bot.

- Recognize from feature unit: Although there are various strategies in StarCraft game play depending on the player's style and selected race, each strategy has some required constructions or fighters in general. Especially in the early stage of the game, since the unit number is much fewer than in the later game, this method is more efficient than a complicated algorithm. Therefore, we directly related a symbol building with a kind of strategy. That means, once the mental agent finds such kind of units or a certain combination of units in raw data, it will recognize the enemy's intention quickly and pick the solution from predefined plans database.
- Infer from economy estimation: In StarCraft, reserving too much resources is not encouraged. This is because reserve resources will delay the attack timing or even lose the game. So players should convert the resources to the military force as soon as possible in order to gain advantages. Therefore, based on this unwritten rule, an experienced player is capable to guess opponent's plan without seeing any feature unit.

We also applied this heuristic in our bot as a backup plan. Suppose our recon unit has already scouted the enemy base successfully in the early game, but we barely found any feature unit that is related to any kind of strategy. In this case, the mental agent will try to estimate the enemy's current reserved resources with the following function:

$$RR = TR - \sum_{i=1}^{N} P_i \, . \tag{1}$$

In this formula, *RR* is the reserved resources of the enemy, *TR* is the total resources that enemy has collected so far and P_i is the price of each individual unit of enemy we have discovered. However, since it's impossible for us to know how much resources that enemy has ever collected precisely, we just simply take ours as a reference because there is no big disparity between two players on the economy condition in the early game.

After calculating the enemy's reserved resources, we compare it with a threshold *B*. The threshold *B* here can be a simple constant or a function related to the game time. Assuming that opponent never reserves resources, the condition $RR > B$ is satisfied, this means the enemy is hiding some information from us (such as construct feature buildings somewhere else), the mental agent will take a defensive plan immediately to prevent any possible unaware attack.

Attack timing estimation: Determine the right timing to launch an attack is also a challenging task for bots. So far, most of the bots still decide attack timing simply based on fixed game time or military size. However, the disadvantages of fixed attack timing is obvious. First of all, it is easy for the opponent to predict and allows the opponent to defeat the strategy after a few games. Secondly, one may lose many good chances to defeat his opponent before the fixed attack timing comes. Hence a mechanism of dynamically decide attack timing is necessary.

Fighting value comparison: The most direct way to decide attack timing is to compare the number of fighters both sides. However, due to the various types of fighters, we should not simply calculate the military size. In this case, we propose a heuristic way to calculate fighting value for both sides by function (2):

$$FV(f) = \sum_{i \in N_f} DPF_i \cdot HP_i \, , \tag{2}$$

where the *FV* (*f*) and N_f represent the fighting value and the set of units of force f respectively. DPF_i refer to damage per frame of fighter unit *i* and HP_i is the hit-point. With this function, we do not just judge the military force by number, but it is actually based on the "value" of each fighter. Specially, for DPF_i, we can calculate it based on the following function:

$$DPF_i = \frac{damage_i}{wcd_i} \, . \tag{3}$$

In this function, the *DPF* value of each fighter *i* is the damage of it divide to its weapon cool down time wcd_i, which also called attack rate. However, knowing the fighting value of both players is still far from enough, because of the imperfect information in the game play. So in order to solve this problem, we also take the dead units number into consideration as a human player usually do. For that, we recalculate the fighting value of each force based on the followed function:

$$\begin{aligned} RFV(f) &= \frac{FV(f)}{N_{dead}^{f}} \qquad if\ N_{dead}^{f} > 0, \\ RFV(f) &= FV(f) \qquad otherwise \ , \end{aligned} \tag{4}$$

where *RFV* (*f*) and N_{dead}^{f} are the recalculated fighting value and the number of dead units of force f respectively. If the N_{dead}^{f} of either side equal to 0, then the original *FV* (*f*) will be kept. Thus, we determine the attack timing when the following condition is satisfied:

$$RFV(our) > C \cdot RFV(emeny) \, . \tag{5}$$

Here *our* and *enemy* represent our force and enemy force respectively. *C* is a coefficient constant value between two military forces. It is also a heuristic value for navigating our army's attack target. So, if the value of *C* is high, which means our military force is much more powerful than enemy, then our army will aim at destroy enemy's main base directly. Otherwise we should attack from enemy's furthest expansion, because the resistance of a expansion is usually weaker than main base. Therefore we can not only to get rid of the risk of losing our army due to the underestimate enemy, but also have a chance to weaken enemy's economy.

4. PERFORMANCE EVALUATION

4.1 Module Evaluation

Our implementation uses BWAPI (BWAPI, 2012) to retrieve information and control StarCraft. Besides that, we also use two other add-ons for easier pre-processing and managing information retrieved from StarCraft. One is BWTA (BWTA, 2012), a popular terrain analyzer in StarCraft, the other one is BWSAL (BWSAL, 2012), a framework that already provides several useful functions for BWAPI. Because of a time restrain, our Build Order Manager is still heavily based on the original Build Order Manager from BWASAL, though it is still not very good and lacks functionality for cancelling training units and buildings.

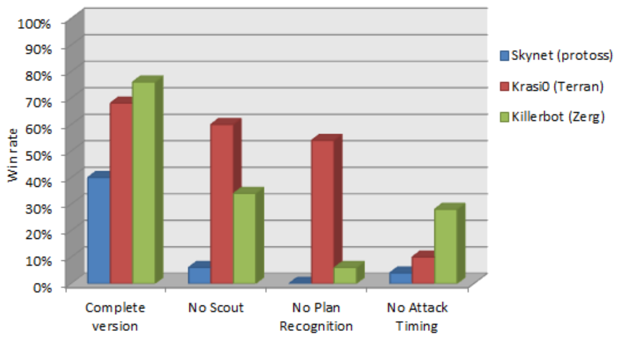

Fig. 4. The win rate of our bot against three example bots with different races from BWAPI Ladder

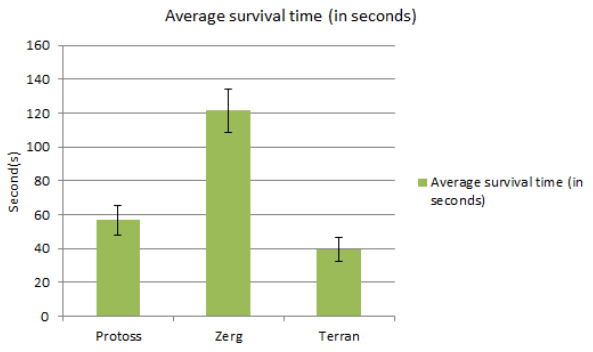

Fig. 5. The survival time of the scout unit

In the experiments, we first evaluated the performance of our rules, after that, we tried to evaluate the usefulness of some agent in our framework. To do this, we compared the performance of our bot against three strong and well-known bots from the BWAPI Ladder (a place where best StarCraft bots in the world play against each other):

- Skynet_2.01: Protoss bot, currently ranked 3rd in BWAPI Ladder, first place in CIG StarCraft 2011 and 2012 competition. This is a very aggressive bot with a good strategy in economy developing and unit controlling.
- Krasi0_2.19: currently ranked 11th,it is the highest ranked Terran bot in BWAPI Ladder so far.
- Killerbot_3: Zerg bot, currently ranked 1st in BWAPI Ladder. This one and another deviation of it (Killerbot2), both are Zerg bots, now currently ranked as first and second bot in the Ladder.

We choose these as the opponents to measure our performance against all three races in StarCraft. Also, to evaluate the performance of each agent in our framework, we disabled it and pitted the modified AI against those bots from BWAPI Ladder again. The agents we tried to measure the usefulness are:

- Scouting agent: No more scout behavior after knowing enemy's base position.
- Plan recognition manager in mental agent: Without plan recognition part in mental agent, the bot only follows scripted plans.
- Attack timing decision in mental agent: Without attack timing decision manager in mental agent, bot always follow a fixed attack timing. The timing in the experiment was set when our supply used reached or over 100.

Each experiment was run with 50 games, the win rates are shown in Figure 4. Figure 4 shows that our bot is weak against Skynet, but it defeated both strong Terran and Zerg bots. Skynet took full use of the flexibility of Protoss, and the excellent macro control on economy and the micro control on army helped it seizes the advantage from the early game, which result in our loss even if fully understanding its plan. Killerbot is also an aggressive bot with a nice economy management as well as military control. However, the fixed strategy makes it defendable even if it ranks top on the BWAPI Ladder. In terms of Krasi0, it seemed weak against early rush strategy, except when it showed impressive performance and lots of intelligence during the game play.

Also, from Figure 4 we can easily see the importance of each module. Against Protoss or Zerg, the win rates sinks drastically when a module is removed from complete version. However, against Krasi0, the win rate still stay high either without scout agent or plan recognition. The reason why is because our bot always take a rush strategy in the early game when against Terran opponent, which results in the game being over in a short time, so there was no room for either agent to function itself.

There has been a huge difference between disable either the scout agent or the plan recognition manager. For the former, even though the scout task is terminated in the early game, plan recognition heuristic still works when any of our fighters discover enemy's feature units, therefore the counter measures are still taken even though it might be late. On the contrary, if the plan recognition manager is disabled, there will be no reaction for our bot even if we already scouted successfully.

In terms of attack timing manager, despite of the timing it provided not reasonable every time, it still significantly increase the win rate of our bot and shorten the game time. Obviously, the win rate decrease rapidly when the manager is disabled. Particularly, when our bot took an early rush strategy versus krasi0, the bot missed several attack timings and ended up with a loss eventually. Nevertheless, sometimes the fixed attack timing was coincidentally reasonable, therefore we still defeated the opponent occasionally.

4.2 Potential Field Evaluation

We evaluated the performance of potential fields in our bot by playing 10 games with StarCraft build-in AI from each race respectively. The reason why we didn't choose human player as experiment subject is that the survive time of recon unit heavily depend on the opponent's micro-control skill. Since the skills of players vary from each other, it is hard to evaluate the real performance.

After the scout unit spots the first enemy fighter in the enemy's territory, we recorded the alive time until it was destroyed by enemy, see the result in Figure 5.

The result shows the survival time of the scout unit varies from different races. The scout unit was able to run from enemy's attack successfully at the beginning, but with the increasement of the enemy fighters, the scout unit could not avoid destruction soon after, especially when facing several long range attack enemies (such as Terran Marine). Compare with the high-efficiency Terran, Zerg AI seemed have difficulty to destroy our scout unit, the reason is build-in Zerg always produce unmovable defensive buildings first, which gives the scout unit plenty space and time to survive. Although the survive time is long enough for a bot to finish the scout task, it is still not comparable to a skillful human player.

4.3 Competition

The first StarCraft AI competition was at the AIIDE 2010 (AAAI conference on Artificial Intelligence and Interactive Digital Entertainment). At the event, there were more than 26 teams registered. Our bot first joined the StarCraft AI competition on CIG 2012 (IEEE Conference on Computational Intelligence and Games), as our first trial, our bot only placed 9th out of 10 teams. From CIG 2012, we improve our bot a lot although it still a heavily rule based system in conjunction with potential field and heuristic methods.

This time, we sent our bot to SSCAI 2012 (Student StarCraft AI Tournament) which was the second year of SSCAI tournament and had 52 participants from all over the world, including strong bots from the BWAPI Ladder (BWAPI Ladder, 2012) and team developed bots. The tournament was divided into two divisions:

- Student division: Round Robin tournament, where everybody played one game with each other, and ranked by their score (3 points for winning a game, 1 point for a tie and 0 point for losing a game).
- Mixed division: Elimination bracket of 8 best bots.

Although there were only few fixed strategies set in our bot, by applying heuristic methods to predict enemy strategies based on partial information gathered from scouting, our bot has revealed its advantages in this competition. For instance, it had taken a defensive strategy when played against KillerBot in the semifinal, who is a strong Zerg bot by applying priority state machine and highly ranked in the BWAPI Ladder, is considered as the

best counter measure to an aggressive bot because lots of bots could not survive its powerful attack, see Figure 6. And in the final game, our bot had earned a neat victory by a prompt response to opponents early rush and seized a
perfect timing to strike back. As the final result, our bot has won the first place in the mixed division and has ranked 4th in the student division, see Figure 7 and Figure 8.

Fig. 6. Our bot(White Terran) play against KillerBot(Green Zerg) in the semifinal of SSCAI 2012, our bot gained lots of benefits by taking a defensive strategy

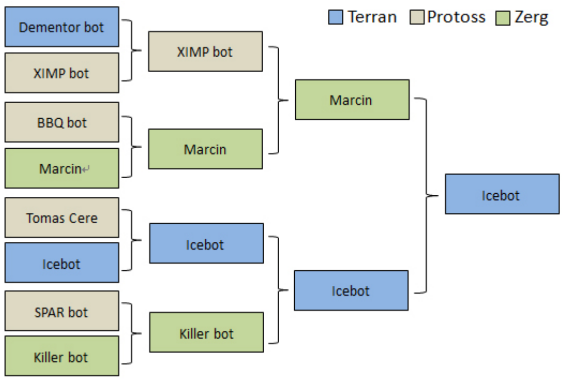

Fig. 7. The elimination bracket for top eight bots

5. CONCLUSION AND DISCUSSION

In this paper, we have described the architecture of our bot and show the significance of scouting, enemy plan recognition and attack timing decision in the RTS game StarCraft. The approach we applied to the scout task is a potential field, which can be successfully used in RTS game as a mechanism for information gathering. In addition, we also proposed heuristic methods on opponent's plan recognition and attack timing determination. The experiment results showed that our methods greatly improved the performance of AI bot. However, some shortages are still can be found in the current work. For example, in the potential field part, the scout unit was occasionally stuck and confused when facing several enemy units. And as far as attack timing determination is concerned, an incorrect estimate of the enemy's fighting force leads to wrong attack timing still happens every now and then. The reason why is because the module relies on the scouting result too much, which means an inaccurate information might result in our army completely annihilated by enemy.

In future work, we will focus on improving the performance of scouting unit, try to extend its alive time. And meanwhile add a prediction module to improve the timing decision, as well as reduce the reliance on the scout agent. Additionally, we plan to apply some machine learn algorithm to collaborate with current heuristic method for enemy plan recognition.

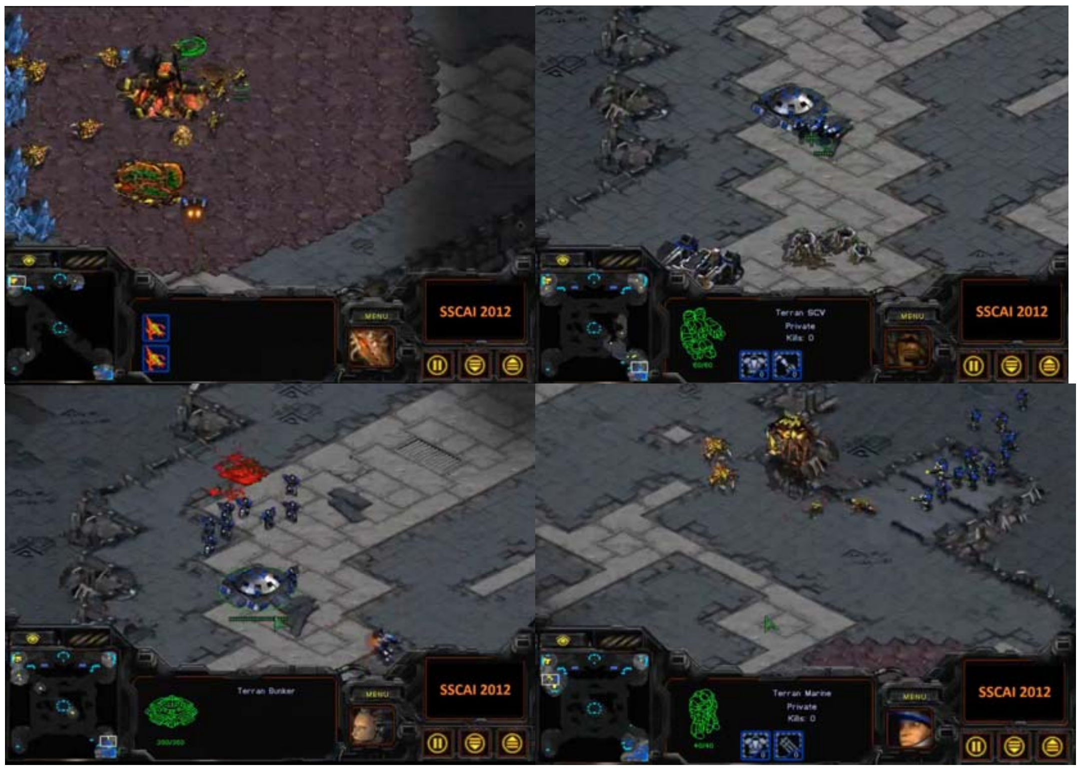

Fig. 8. Our bot (blue Terran) in the final game, an example of our heuristic methods for strategy recognition and attack timing determination(from left to right, up to down). Our bot reacted quickly to enemy's early rush, and seized a right timing to attack after a successful defense

REFERENCES

BWAPI. http://code.google.com/p/bwapi/ (last accessed on December, 2012).

BWAPI Ladder. http://bots-stats.krasi0.com/ (last accessed on December, 2012).

BWSAL. http://code.google.com/p/bwsal/ (last accessed on December, 2012).

BWTA. http://code.google.com/p/bwta/ (last accessed on December, 2012).

Churchill David, Safdine Abdallah, and Buro Michael, (2012). Fast heuristic search for RTS game combat scenarios, In: *Proceedings of the 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pp.112-117.

Hagelback Johan, (2012). Potential-field based navigation in StarCraft, In: *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pp.388-393.

Park Hyunsoo, Cho Ho-Chul, Lee KwangYeol, and Kim Kyung-Joong, (2012). Prediction of Early Stage Opponents Strategy for StarCraft AI using Scouting and Machine Learning, In: *Workshop at SIGGRAPH ASIA (Computer Gaming Track).*

Synnaeve, G. and Bessiere, P. , (2011). Bayesian model for RTS units control applied to StarCraft, In: *2011 IEEE Conference on Computational Intelligence and Games (CIG)*, pp.190-196.

Synnaeve, G. and Bessiere, P., (2011). A Bayesian Model for Opening Prediction in RTS Games with Application to StarCraft, In: *2011 IEEE Conference on Computational Intelligence and Games (CIG)*, pp.281-288.

Wang Zhe, Nguyen Quang Kien, Thawonmas Ruck, and Rinaldo Frank (2012). Monte-Carlo Planning for Unit Control in StarCraft, In: *Proc. of the 1st IEEE Global Conference on Consumer Electronics (GCCE),* pp. 268-269, Tokyo, Japan.

Wang Zhe, Nguyen Quang Kien, Thawonmas Ruck, and Rinaldo Frank (2012). Using Monte-Carlo Planning for

Micro-Management in StarCraft, In: *Proc. of the 4th Annual Asian GAME-ON Conference on Simulation and AI in Computer Games (GAMEON ASIA),* pp. 33-35, Kyoto, Japan.

Weber, B. G. and Mateas, M., (2009). A data mining approach to strategy prediction, In: *IEEE Conference on Computational Intelligence and Games (CIG)*, pp.140-147.

Wender Stefan and Watson Ian, (2012). Applying Reinforcement Learning to Small Scale Combat in the Real-Time Strategy Game Star-Craft:Broodwar, In: *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pp.402-408.