Learning to Select Actions in StarCraft with Genetic Algorithms

Wei-Lun Hsu
Department of Computer Science
National Chiao Tung University
HsinChu City, Taiwan
wlhsu@nclab.tw

Ying-ping Chen
Department of Computer Science
National Chiao Tung University
HsinChu City, Taiwan
ypchen@cs.nctu.edu.tw

***Abstract*—In numerous different types of games, the real-time strategy (RTS) ones have always been the focus of gaming competitions, and in this regard, StarCraft can arguably be considered a classic real-time strategy game. Currently, most of the artificial intelligence (AI) players for real-time strategy games cannot reach or get close to the same intelligent level of their human opponents. In order to enhance the ability of AI players and hence improve the playability of games, in this study, we make an attempt to develop for StarCraft a mechanism learning to select an appropriate action to take according to the circumstance. Our empirical results show that action selection can be learned by AI players with the optimization capability of genetic algorithms and that cooperation among identical and/or different types of units is observed. The potential future work and possible research directions are discussed. The developed source code and the obtained results are released as open source.**

Keywords—Real-Time Strategy Game; Genetic Algorithm

I. INTRODUCTION

Real-Time Strategy (RTS) games are in a sub-genre of strategy video games in which the game does not progress incrementally in turns. Players in RTS need to gather resources, build structures, research technologies, train units, and defeat units of opponents. Each of these factors is critical for winning an RTS game. This paper presents a study focuses on using an genetic algorithm (GA) to form an RTS game player that can select an appropriate action for each unit according to their respective situation within a skirmish or battle.

Action selection is an important factor for winning a skirmish. When an action is suitable for the current situation, it maximizes the unit's damage to the enemy and/or minimizes the damage from the enemy. If an action is ineffective in the situation, it will lead to failure and hence the loss of skirmish. In one RTS game, several skirmishes may be included, and every skirmish may influence the outcome of the entire game.

StarCraft is a popular RTS game series, and StarCraft: Brood War API (BWAPI) is a widely adopted platform for AI research on RTS games. Researchers can directly execute their AI code in the game with BWAPI. Therefore, trying to harvest the optimization capability of genetic algorithms, we devise a straightforward model for selecting actions and employ a genetic algorithm to optimize the model parameters according to the actual gaming results against the built-in StarCraft computer player. Firstly we design several typical actions and corresponding scenarios to these actions. We then launch actual gaming rounds to examine whether the employed genetic algorithm can successfully adjust the model parameters for our player to select the desired action. More types of units are added into the experiment after the proposed framework is shown effective. In the experiments, cooperation among units, either of the same types or of different types, is observed. Compared with other studies, the main difference of this study is that the actions one unit can take is developed beforehand by human and the model is optimized for selecting one of these actions. The developed source code is released as open source at https://github.com/nclab/starcraft.asga on GitHub.

For the remainder of this paper, Section II describes the background of StarCraft and genetic algorithms. Section III introduces the simulation environment and the representation of action selection for the genetic algorithm. Section IV introduces the three stage of the conducted experiments. Section V provides the experimental results, followed by Section VI which concludes this paper.

II. STARCRAFT AND RELATED WORK

A. StarCraft

StarCraft is a popular RTS game developed by Blizzard Entertainment in 1998 and sold more than 9.5 million copies during 1998 to 2004. As in classic RTS games, StarCraft players have to gather resources, build structures, research technologies, train units, and defeat units of their opponents. Every one of these factors is critical for winning a game. In this paper, we focus on selecting an appropriate action of each unit according to the situation of skirmish. Each unit of the game has some attributes, such as *health*, *damage*, *move speed*, *attack speed*, *turn speed*, etc. Table I shows the detailed properties for **Marine**, **Tank**, and **Wraith** in StarCraft.

B. Genetic algorithms

Genetic algorithms (GAs) are one kind of metaheuristics inspired by the evolutionary process of natural selection. GAs are typically utilized to generate solutions to search or optimization problems. The essence of GAs is the principle of "survival of the fittest," meaning that better individuals have more chances to survive and generate the next generation. In other words, GAs repeat the iteration of selecting good solutions to produce new ones. Figure 1 shows the computational process of a genetic algorithm.

TABLE I. PROPERTIES OF THE STARCRAFT UNITS USED IN THIS STUDY

Parameter	*Marine*	*Tank*	*Wraith*	*Purpose*
Hit-points	40	150	120	Hit-points (HP) decrease when unit hit by enemy's unit. Unit dies when Hit-points ≤ 0
Ground Damage	6	30	8	The number of HP that can be subtracted from the ground target's health
Air Damage	6	0	20	The number of HP that can be subtracted from the air target's health
Speed	4	4	6.67	Speed of unit
Range	4	7	5	The maximum distance at which an unit can fire upon the target.
type	ground	ground	air	Type of unit

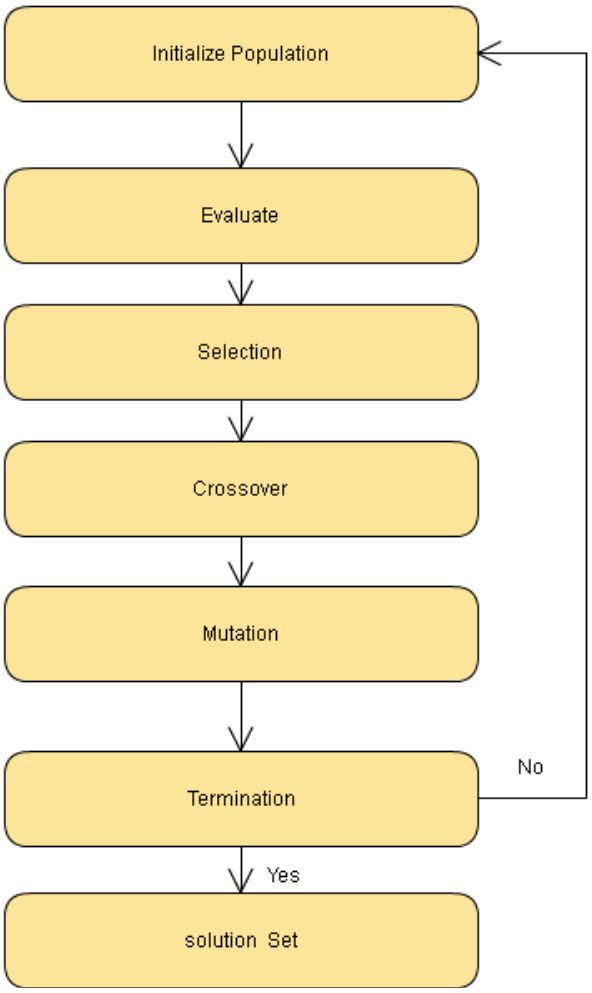


Fig. 1. Computational process of a genetic algorithm

C. Related work

Genetic algorithms have been successfully applied in several real-time strategy game AI studies [1-3]. First of all, there are related studies related to spatial reasoning and unit movement. Liu et. al. used a genetic algorithm to optimize parameters of the reactive control and also used influence maps and potential fields to guide units [4-6].

Ontanón et. al. analyzed eight good AI frameworks in StarCraft AI competitions [7]. They discovered that most AI were designed with three levels of abstraction—strategy (which loosely corresponds to "macro"), tactics, and reactive control (which loosely correspond to "micro") .

Different from that most AI use three levels of abstraction and some focus on the reactive control, we propose in this paper that an action for a unit to take is selected by a learned model from a set of actions designed beforehand by human.

III. METHODOLOGY

A. Actions and scenarios

The first step of action selection is to design three basic, typical actions: *kiting*, *distance first*, and *hit-points first*.

- **Action A—Kiting:** Also known as "hit and run." The unit taking this action attacks enemy and moves backward until next attack. This action is useful where the attack range of the unit is larger than that of enemy units.
- **Action B—Distance first:** The unit attacks the closest enemy. This action is useful to defend and to hold the position.
- **Action C—Hit-points first:** The unit attacks the enemy unit that has the lowest HP. This action is useful to focus on and take out one single enemy unit.

The second step is to design three scenarios corresponding to the three aforementioned actions. Figure 2 shows the snapshots of three scenarios.

- **Scenario A:** Ally Tank has a longer attack range than enemy Marine. Action A performs efficiently.
- **Scenario B:** Enemy Tank at the bottom left corner is in the attack range of ally Tank. Action B lets ally Tank attack the closest enemy and is a good decision.
- **Scenario C:** All units focus on one enemy is better than that each unit gets involved in a one-to-one battle. Action C should be taken in this scenario.

The goal to design these scenarios is mainly to examine the effectiveness of the proposed learning framework described in the following sections. If the model, after trained with the actual gaming results, provides the selection of action A in scenario A, B in B, and C in C, the proposal can be, to a certain extent, considered effective. Interested readers can refer to the videos at http://nclab.github.io/starcraft.asga to examine the typical outcomes of different actions in different scenarios.

B. Chromosome representation

We encode the chromosome into 15 integers. The detailed information on representation is shown in Table II. When our game engine receives an individual, the first 12 integers are interpreted as the threshold values to individually determine whether or not one action should be taken under the circumstance. It is possible that more than one actions are considered valid in a given situation. The last 3 integers are a permutation of the three actions indicating the priority to select.

C. Fitness evaluation

In evaluation, skirmishes in scenarios are run in StarCraft. An individual will be evaluated three times in one scenario. After all of the scenarios are finished, we can use the gaming results provided by StarCraft to compute the fitness value. The fitness value (F) is computed as

$$F = S_a - S_e - T \quad (1)$$

Scenario A designed for action A (kiting)

Scenario B designed for action B (distance first)

Scenario C designed for action C (hit-points first)

Fig. 2. Snapshots of the three scenarios. Red units are the ally

S_a is the score of ally. The more defeated enemy units, the higher S_a. S_e is the score of enemy. The more defeated ally units, the higher S_e. T is the duration of the game round, and the maximum game time is 60 seconds. The scale of S_a and S_e is larger that of T, because defeating the enemy and avoiding damage are more important than finishing the fight quickly. Hence, if all enemy units are defeated without ally casualties in a short skirmish, very high fitness will be obtained. After all games for one individual are finished, we compute the final fitness value. The final fitness value is simply the average of the fitness of each scenario.

D. Genetic algorithms

We set the size of population to 40 and run the genetic algorithm for 90 generations. Rank-based wheel selection is used to select chromosomes for crossover. The size of a rank is 10, and the probability of four ranks are 0.52, 0.27, 0.14 and 0.06. Individuals with the first ten high fitness are rank 1. The probability of crossover is 1, and the probability of mutation is 0.01. All offspring compose the population in next generation.

TABLE II. CHROMOSOME REPRESENTATION

Variable	*Type*	*Description*
R_a	Integer	Range of unit minus range of closest enemy in action A
E_a	Integer	Sum of enemy count in unit range in action A
HP_a	Integer	Weakest enemy HP divided by unit damage in action A
ED_a	Integer	The number of enemy units minus the number of ally units in action A
R_b	Integer	Range of unit minus range of closest enemy in action B
E_b	Integer	Sum of enemy count in unit range in action B
HP_b	Integer	Weakest enemy HP divided by unit damage in action B
ED_b	Integer	The number of enemy units minus the number of ally units in action B
R_c	Integer	Range of unit minus range of closest enemy in action C
E_c	Integer	Sum of enemy count in unit range in action C
HP_c	Integer	Weakest enemy HP divided by unit damage in action C
ED_c	Integer	The number of enemy units minus the number of ally units in action C
O_1	Integer	The action is the first priority
O_2	Integer	The action is the second priority
O_3	Integer	The action is the third priority

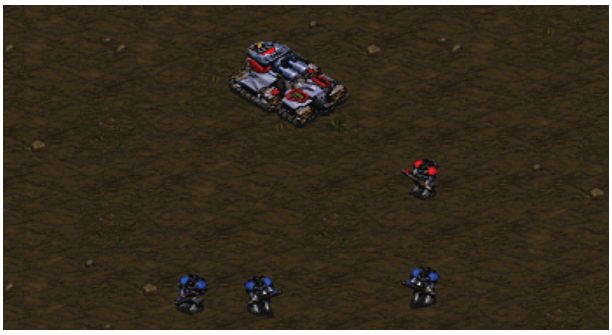
IV. THREE STAGES OF EXPERIMENTS

We conduct three stages of experiments with the aforementioned methods. Each experiment is using the genetic algorithm to find the best threshold values of selection conditions. The first stage determines the feasibility of our proposed framework for action selection. The second stage focuses on the cooperation among units. In the final stage, we observe that cooperation between Ground and Air units can be implicitly learned by our proposed framework.

A. Action selection for one unit type

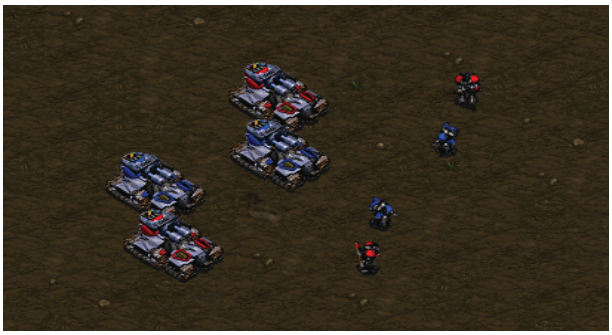
We conduct the first stage of experiments that only one type of units is used in the three scenarios shown in Figure 2 to examine the effectiveness of our proposal. The type of unit chosen in this stage is Tank which has high HP and endures high ground damage. Because Tank has a larger attack range than Marine, it is good at hit and run (action A). Action B and action C are easy for Tank to perform, because Tank can attack the closest enemy or the enemy with the lowest HP.

Only one action can gain high scores in each scenario. If the unit selects action A, good results in scenario A will be obtained. Since individuals are evaluated in all scenarios, only the ones make the unit to select an appropriate action (A for A, B for B, etc.) can be rewarded higher fitness values.



Scenario A designed for action A (kiting)

Scenario B designed for action B (distance first)



Scenario C designed for action C (hit-points first)

Fig. 3. Snapshots of the three scenarios. Two types of units are used

B. Cooperation among units

In RTS games, it is difficult to use units of only one type to fight against all enemies. Therefore, we add a new unit type in the second stage in order to observe if cooperation among units emerges. The skirmish is more complicated with two types of units, and hence we redesign the scenarios as Figure 3.

There are two types of units in these scenarios. Tank is stronger than Marine and has long attack range and large damage point. It can make use of its long range attack in scenario A. Marine has low HP and short attack range while it costs less. It can pick down enemy or fight enemy with Tank.

Cooperation among units is a key point of this experiment. Maybe Marine will be sacrificed to protect Tank. With this experiment, we do find some cooperative behavior, which will be described in the next section.

C. Cooperation between ground and air units

In StarCraft, the existence of air units makes the game even more complicated. Because some ground units without air weapon cannot attack air units. Players have to consider how to use ally air units and how to engage enemy air unit.

We add a type of air units in the final stage of experiments. Because air units have some different attributes, we add new action D, new scenario D, and modify chromosome representation. Action D is *air-first*. It makes the unit to focus on air enemy units. Attacking the enemy air units is the first priority to prevent our ground unit from being defeated by the enemy air units. There are air units, ground units without air weapon, and ground units with air weapon in scenario D. Figure 4 shows the snapshots of these scenarios. The modified representation is shown in Table III. We also add the threshold values for the enemy air unit to all actions on the chromosome.

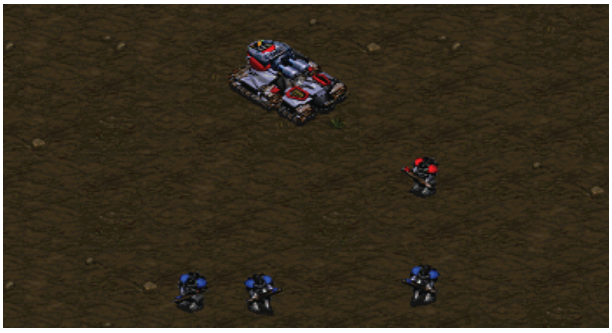
V. Results and Discussion

We use BWAPI to interface with StarCraft in order to evaluate the individuals generated by the adopted genetic algorithm. As described in previous sections, we use the genetic algorithm to optimize the model parameters for action selection of one unit type. Figure 5 shows the average fitness of the genetic algorithm running on three scenarios in the first stage of experiments. We can see that the average fitness value in the initial population is 630 and increases to 1328 at generation 90, which is twice as much as the initial value.

Second, we learn the action selection model for two types of units. Figure 6 shows the average fitness values. We can see that the average fitness value in the initial population is 292 and increases to 756 at generation 90, which is 2.5 times as much as the initial value.

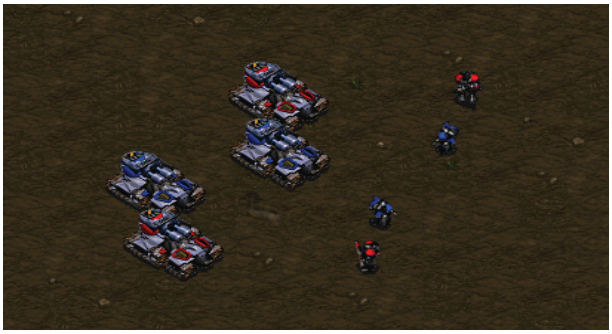
At last, we add air units into the experiment. Figure 7 shows the average fitness values. We can see that the average fitness value in the initial population is -610, which means that almost no skirmish is won by the ally. However, the average fitness value increases to 1140 at generation 90.

Based on the series of experiments, we observe some interesting behavior of units.



Scenario A designed for action A (kiting)

Scenario B designed for action B (distance first)



Scenario C designed for action C (hit-points first)

Scenario D designed for action D (air first)

Fig. 4. Snapshots of the four scenarios. Air units.are added in scenario D

TABLE III. CHROMOSOME

Variable	*Type*	*Description*
R_a	Integer	Range of unit minus range of closest enemy in action A
F_a	Integer	Sum of air enemy HP divided by sum of air damage in action A
HP_a	Integer	Weakest enemy HP divided by unit damage in action A
ED_a	Integer	The number of enemy unit minus the number of ally unit in action A
R_b	Integer	Range of unit minus range of closest enemy in action B
F_b	Integer	Sum of air enemy HP divided by sum of air damage in action B
HP_b	Integer	Weakest enemy HP divided by unit damage in action B
ED_b	Integer	The number of enemy unit minus the number of ally unit in action B
R_c	Integer	Range of unit minus range of closest enemy in action C
F_c	Integer	Sum of air enemy HP divided by sum of air damage in action C
HP_c	Integer	Weakest enemy HP divided by unit damage in action C
ED_c	Integer	The number of enemy unit minus the number of ally unit in action C
R_d	Integer	Range of unit minus range of closest enemy in action D
F_d	Integer	Sum of air enemy HP divided by sum of air damage in action D
HP_d	Integer	Weakest enemy HP divided by unit damage in action D
ED_d	Integer	The number of enemy unit minus the number of ally unit in action D
O_1	Integer	The action is the first priority
O_2	Integer	The action is the second priority
O_3	Integer	The action is the third priority
O_4	Integer	The action is the fourth priority

A. *Changing actions*

We observe the first changing action in Scenario B of the one unit type experiment. Figure 8 shows the process of the best GA individual at generation 90. The trained model selects the starting action that Tank attacks the enemy bottom left Tank (action B). Next, Tank changes to action A, attacking enemy Marine with moving to the upper right direction (action A). Finally, Tank defeats the enemy Tank (action B).

Video: http://nclab.github.io/starcraft.asga/Section-V-A.avi

B. *Cooperation among units*

Using two types of units in the experiment, we can discover the cooperation from scenario B. Figure 9 shows process of the best GA individual at generation 90. The trained model selects the starting action that Marines and Tank attack the enemy Tank (action B). Marines and the enemy Tank are then defeated. Tank changes to action A, attacking the enemy Marine, and finally wins by its longer attacking range.

Video: http://nclab.github.io/starcraft.asga/Section-V-B.avi

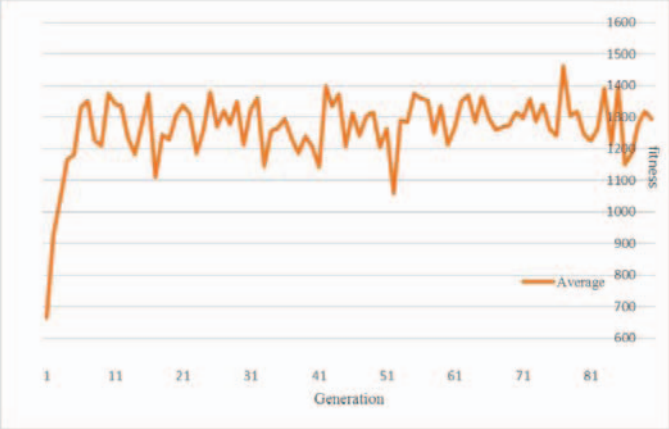


Fig. 5. Average fitness of GA with one type of units

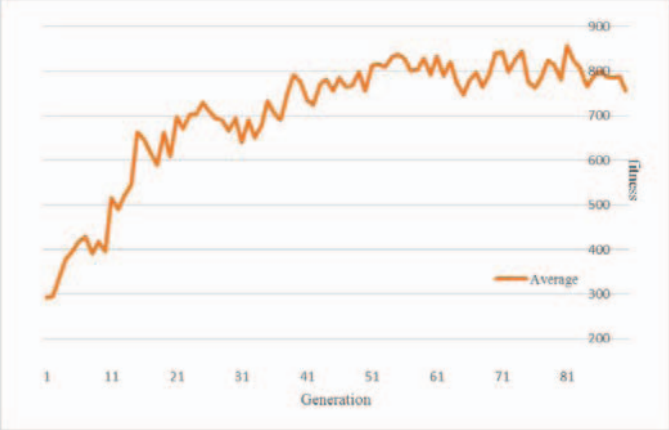


Fig. 6. Average fitness of GA with two types of units

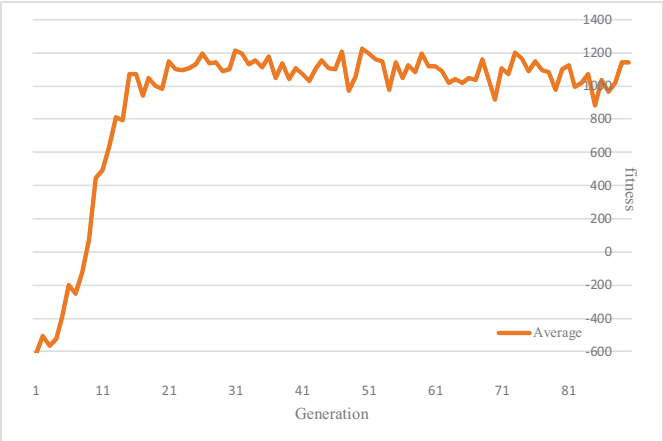
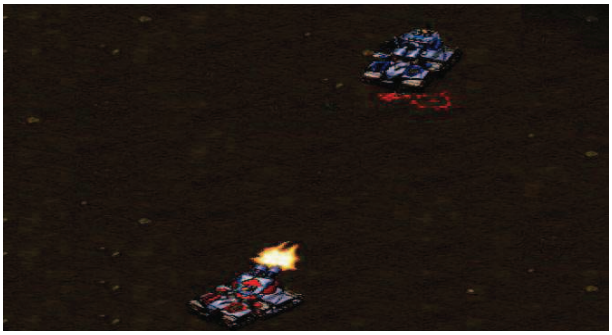


Fig. 7. Average fitness of GA with three types of units, including air units

Begin state

Starting action B: attacking the closest enemy

Changing to action A: kitting on the Marine



Final state: fighting and defeating the enemy Tank

Fig. 8. Process of the best GA individual at generation 90 with one type

Begin state

Starting action B: attacking the enemy Tank

Changing to action A: kitting on enemy Marines

Final state: winning with the long-range attack

Fig. 9. Process of the best GA individual at generation 90 with two types

C. *Cooperation between ground and air units*

Air units will be unharmed, if the enemy units do not have air weapon. Hence, cooperation between ground and air units is effective. Figure 10 shows the process of the best GA individual at generation 90. In the beginning, Marines attack the enemy Wraith (action D). Tank and Wraith attack enemy Marines, then moving to the bottom left corner (action A). Marines of both sides are defeated. Tank and Wraith attack the enemy Tank (action B). Both players have the same type of units, while the enemy Wraith was hit by Marines at the start. Finally, the ally has two units against one enemy unit.

Video: http://nclab.github.io/starcraft.asga/Section-V-C.avi

VI. Conclusions and Future Work

Our research focused on building an RTS game player that can select an appropriate action of each unit according to the situation of skirmish. We employed a genetic algorithm to search for the best threshold value of selection conditions. After three stages of investigation and experiments, we found that the model learned by using our proposed framework enabled the units not only to select an appropriate action but also to cooperate with other ally units. The source code developed in this study has been released as open source at our repository https://github.com/nclab/starcraft.asga on GitHub.

Potential future research directions include

- **Automatic action generation:** All of the actions in this study are designed by human. If actions can be automatically generated, more different combinations can be explored and investigated.
- **More enemy AI:** The default StarCraft AI was adopted as enemy. In the future, the champion of the StarCraft AI competition may be adopted. Moreover, competition can also be held between the individuals generated by the genetic algorithms.
- **More types of units or races:** Only one race and three types of units were considered in this study. In StarCraft, each race has its distinct characteristics. The attempt may be extended to more types of units and/or other races in StarCraft.

The results of this study show a promising research direction. More efforts may be put into this line of research to further the progress of building AI agents capable of playing with human opponents on equal terms.

Acknowledgments

The work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 105-2221-E-009-157. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

Begin state

Starting action D: attacking the enemy Wraith

Changing to actions A and B

Final state: enemy has no air unit

Fig. 10. Process of the best GA individual at generation 90 with air unit

REFERENCES

[1] N. Othman, J. Decraene, W. Cai, N. Hu, M. Y. H. Low, and A. Gouaillard, "Simulation-based optimization of StarCraft tactical AI through evolutionary computation," in *Proceedings of 2012 IEEE Conference on Computational Intelligence and Games* (*CIG*), 2012, pp. 394-401.

[2] I. Zelinka and L. Sikora, "StarCraft: Brood War—Strategy powered by the SOMA swarm algorithm," in *Proceedings of 2015 IEEE Conference on Computational Intelligence and Games* (*CIG*), 2015, pp. 511-516.

[3] P. Garćıa-Sánchez, A. Tonda, A. M. Mora, G. Squillero, and J. Merelo, "Towards automatic StarCraft strategy generation using genetic programming," in *Proceedings of 2015 IEEE Conference on Computational Intelligence and Games* (*CIG*), 2015, pp. 284-291.

[4] S. Liu, S. J. Louis, and C. Ballinger, "Evolving effective micro behaviors in RTS game," in *Proceedings of 2014 IEEE Conference on Computational Intelligence and Games* (*CIG*), 2014, pp. 1-8.

[5] C. Ballinger and S. Louis, "Comparing heuristic search methods for finding effective real-time strategy game plans," in *Proceedings of 2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (*CISDA*), 2013, pp. 16-22.

[6] S. Liu, S. J. Louis, and M. Nicolescu, "Using CIGAR for finding effective group behaviors in RTS game," in *Proceedings of 2013 IEEE Conference on Computational Intelligence and Games* (*CIG*), 2013, pp. 1-8.

[7] S. Ontanón, 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*, vol. 5, no. 4, pp. 293-311, 2013.