Identifying Rush Strategies Employed in StarCraft II Using Support Vector Machines

Teguh Budianto, Hyunwoo Oh, Yi Ding, Zi Long, Takehito Utsuro

Identifying Rush Strategies Employed in StarCraft II Using Support Vector Machines

Teguh Budianto[1], Hyunwoo Oh[2], Yi Ding[1], Zi Long[1], and Takehito Utsuro[1*]

[1] Grad. Sch. Sys. and Inf. Eng., University of Tsukuba, Tsukuba, Japan
`utsuro_@_iit.tsukuba.ac.jp`
[2] Grad. Sch. Interdisc. Inf. Studies, The University of Tokyo, Tokyo, Japan

Abstract. This paper studies the strategies used in StarCraft II, a real-time strategy game (RTS) wherein two sides fight against each other in a battlefield context. We propose an approach which automatically classifies StarCraft II game-log collections into rush and non-rush strategies using a support vector machine (SVM). To achieve this, three types of features are evaluated: (i) the upper bound of variance in time series for the numbers of workers, (ii) the upper bound of the numbers of workers at a specific time, and (iii) the lower bound of the start time for building the second base. Thus, by evaluating these features, we obtain the optimal parameters combinations.

Keywords: Real-Time Strategy Game · StarCraft II · Rush Strategy

1 Introduction

Real-time strategy (RTS) games make up a popular on line computer game genre wherein two sides fight against each other in a battlefield context. The players are required to gather specific resources to develop their combat strengths in the form of advanced buildings, technologies, and armies. Unlike other strategy games such as Go and Chess wherein complete information on the state of play is provided to both players, information in RTS games is limited and rapidly changes as the player's resources respond to various factors. Moreover, RTS game environments are formed from complex and dynamic sets of information depending on the actions taken by the players. These characteristics contribute to the game's difficulty level and prevent the improvement of RTS-based AI technologies. In this paper, the strategies employed in StarCraft II, a well-known RTS game, are analyzed. These strategies form the most important aspect of competing with opponent's playing styles in order to ultimately win the game. Liu et al. [2] investigated the player's styles to predict their future actions. This type of analysis can aid human players to judge opponent's strategies and accordingly decide a defense strategy. An incorrect strategy judgment about an opponent's game plan and the resultant poor choice of action in StarCraft II leads to the selection of an inappropriate defense strategy which harms the player's strength. Thus, strategy identification in RTS games has become a prominent research area, including strategy prediction and strategy modeling.

Table 1. Data set for evaluation

Logs	Number of game logs
Rush strategy	137
Non-rush strategy	616
Total logs	753

Studies focusing on strategy prediction also employ data mining techniques [3] and machine-learning approaches [1]. Improvements in strategy identification contribute to the advancement of RTS-game-based AI in order to discover the most effective strategies that human players can employ depending on their opponent's plan of action.

Considering these factors, this paper examines the strategies used in StarCraft II by classifying them into two main categories: rush and non-rush strategies. A rush strategy involves players that perform sudden attacks on an opponent's base as early as possible in the game. This strategy aims to interfere with the opponent's movements at the early stages of the game. Conversely, in a non-rush strategy, the players mainly focus on development (i.e., producing more workers and upgrading technologies) rather than attacking the opponent base. Therefore, to examine the strategies of the players, we designed a support vector machine (SVM)-based model that automatically classifies StarCraft II game logs into rush and non-rush strategies.

2 Game-Log Features and the Evaluation Procedure

We collected 5,150 one-vs.-one game replays of StarCraft II from `http://www.spawningtool.com`. All replay files were extracted into human-readable logs using a Python library: sc2reader[1]. Our study only focuses on the games between high-level players. This produced 753 game logs wherein each sample comprises a single player's game log, as summarized in Table 1.

We propose several types of features that are closely related to the number of workers each player has in a rush game. In particular, a major difference tends to exist between the number of workers on both sides. Generally, a player not employing rush strategy continues producing a much higher number of workers since the beginning of the game, while a player adopting a rush strategy produces only a moderate number of workers and stops production at a particular time. Thus, by considering these phenomena, we designed features based on the variance between the time-series number of workers and the number of workers at a specific time. We use the information about the number of workers a player has produced up until a certain time. The reason for this is because players cannot have a complete control over the number of workers each player possess at a certain time. In addition, we examine time required for constructing the second base (a building for collecting resources) in order to further design our

[1] https://github.com/GraylinKim/sc2reader

Table 2. Features of a game log x

Features	Variables of x	
Upper bound of the variance of time-series number of workers $f_{vw}(x; u_0, d_0, e_0) = (x.f^v_{vw} \leq u_0) \wedge (x.f^d_{vw} = d_0) \wedge (x.f^e_{vw} = e_0)$	$x.f^v_{vw}$	Variance of x
	$x.f^d_{vw}$	Time for calculating variance [s]
	$x.f^e_{vw}$	End time of calculating variance [s]
Upper bound of number of workers at a specific time $f_{nw}(x; t_0, n_0) = (x.f^t_{nw} = t_0) \wedge (x.f^n_{nw} \leq n_0)$	$x.f^t_{nw}$	Specific time [s]
	$x.f^n_{nw}$	Number of workers
Lower bound of the start time of second base construction $f_b(x; t_0) = (x.f^t_b \geq t_0)$	$x.f^t_b$	Start time of second base construction [s]

features. We observe the differences in the timing of construction of the second bases of both players. Players employing a rush strategy do not necessarily build their second base as early as possible; however, it is expected that these players build their second base before a certain time. Based on the above observation, we propose three types of features: f_{vw}, f_{nw}, and f_b (Table 2).

In our proposed feature-based design, the parameter combinations of the three feature functions were examined to determine the optimal parameter combinations. For f_{vw}, the parameter combinations were examined by changing v_0 from 0 to 2, d_0 from 60 to 300, and e_0 from 240 to 360. For f_{nw}, the parameter combinations were examined by changing t_0 from 300 to 600 and n_0 from 25 to 40. Finally, for f_b, the parameter combination was examined by changing t_0 from 60 to 360. We first divided our data set into 10 subsets of equal size to perform 10-fold cross validation. From the training data of each fold, the optimal parameters combinations of each three feature functions f_{vw}, f_{nw}, and f_b were identified from the combinations possessing maximum recall, precision, and f-measure. Based on this procedure, each feature function generated three optimal parameter combinations in total resulting in nine optimal parameter combinations for each fold. Next, we created and implemented a feature vector constructed from these nine features. Each features represents a parameter of a set of optimal parameter combinations. Our design eventually resulted in 10 different sets of optimal parameter combinations, which we used to train the SVM classifier.

3 Evaluation of Results

We used the confidence in SVM to calculate the performance of each fold of our approach using recall and precision. Further, we generated the average performance curve of our approach, as shown in Figure 1. The curves use 11 points plotted from 0 to 100 in order to display the average performance. We generalized the recall value of each fold to the closest position among these 11 points. Each of the three baseline curves was produced by removing each set of optimal parameter combinations of the three feature functions from the evaluation.

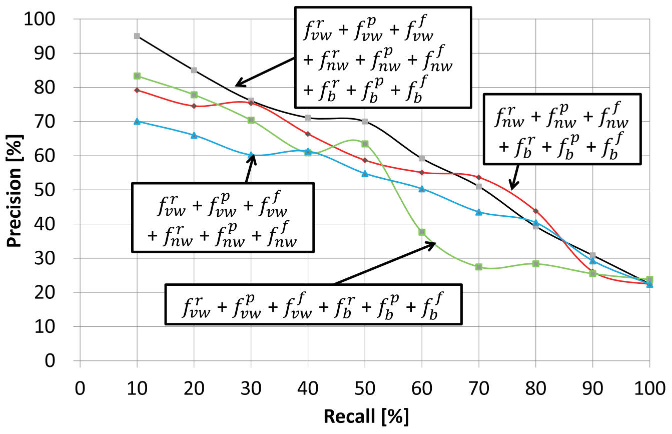

Fig. 1. Recall and precision curve of the overall design

Removing f_{nw} from the evaluation degraded its baseline performance. Probably, this occurred because the differences between the number of workers at a specific time has a significant impact on a rush game. Considering the overall performance, the F-score of the proposed design shows the highest value among all three baselines. Thus, considering the above, we found that there is a significant correlation between the optimal parameter combinations of the three features functions. This indicates that the proposed design incorporating all the combinations worked better than the design considering each combination separately. Therefore, the proposed design could possibly be effective in identifying the use of rush strategies in RTS game-logs-collections.

4 Conclusions

This study proposed a method to identify the rush and non-rush strategies employed by players from RTS game logs. We examined nine optimal parameter combinations of the three feature functions f_{vw}, f_{nw}, and f_b. These combinations along with an SVM were used as the basis for constructing the features that could accurately identify a player's use of a rush strategy.

References

1. H. Park et al. Prediction of early stage opponents strategy for StarCraft AI using scouting and machine learning. In *Proc. WASA*, pages 7–12, 2012.
2. S. Liu et al. Player identification from RTS game replays. In *Proc. the 28th CATA*, pages 313–317, 2013.
3. B. G. Weber and M. Mateas. A data mining approach to strategy prediction. In *Proc. the 5th CIG*, pages 140–147, 2009.