Knowledge Discovery for Characterizing Team Success or Failure in (A)RTS Games

Pu Yang* and David L. Roberts†
Department of Computer Science
North Carolina State University, Raleigh, North Carolina 27695–8206
Email: *pyang3@ncsu.edu, †robertsd@csc.ncsu.edu

***Abstract*—When doing post-competition analysis in team games, it can be hard to figure out if a team members' character attribute development has been successful directly from game logs. Additionally, it can also be hard to figure out how the performance of one team member affects the performance of another. In this paper, we present a data-driven method for automatically discovering patterns in successful team members' character attribute development in team games. We first represent team members' character attribute development using time series of informative attributes. We then find the thresholds to separate fast and slow attribute growth rates using clustering and linear regression. We create a set of categorical attribute growth rates by comparing against the thresholds. If the growth rate is greater than the threshold it is categorized as fast growth rate; if the growth rate is less than the threshold it is categorized as slow growth rate. After obtaining the set of categorical attribute growth rates, we build a decision tree on the set. Finally, we characterize the patterns of team success in terms of rules which describe team members' character attribute growth rates. We present an evaluation of our methodology on three real games: DotA,[1] Warcraft III,[2] and Starcraft II.[3] A standard machine-learning-style evaluation of the experimental results shows the discovered patterns are highly related to successful team strategies and achieve an average 86% prediction accuracy when testing on new game logs.**

I. Introduction

With the development of eSports, post-competition analysis is an important method for improving players' skills and teams' strategies. Game logs (game replays) are the media for post-competition analysis. The actions players perform in the game can be easily checked via game logs. Therefore, we can easily check the game logs to see which players fail at their tasks and at fulfilling the specific role they play on the team. For example, in a game of *Defense of the Ancients* (a popular action real time strategy game), the Support role is played by "heroes whose purpose are to keep their allies alive. Supports will usually come with skills such as healing spells".[4] By checking the game logs to see whether or not the Supports buy the healing-enhancing items and obtain high-level healing skills, we can know if they fail at their tasks or not. However, the failure of a team strategy may not be the fault of players who fail at their assigned tasks. It may also be the fault of players who fulfill their tasks with superfluous character attribute development. Superfluous character attribute development by a player may lead to insufficient character attribute development of other team members. While it may appear initially that the team's failure was caused by the insufficient development of one member's attributes, the true cause may have actually been the over-consumption of resources by another team member. In this paper we present a knowledge discovery technique that will enable credit assignment for a team's success or failure based on the resource consumption of each of its members; and that will disambiguate between a failure of a player's own accord or the resource over-consumption of another team member.

For example, in DotA there are only three lanes for players to gain experience and gold (resources needed for character attribute development). The characters in a lane share experience and gold. Usually a character who occupies an entire lane has a faster attribute development than characters who share a lane with others. Therefore, if a player's character occupies the entire lane for an unreasonably-long period, it leads to other players' characters having insufficient attribute development. Therefore, the team strategy fails because the other characters can not fulfill their roles.

The imbalanced attribute development of team members' characters can not be easily investigated by checking the game logs since the game environments are highly dynamic. We present a method for discovering patterns in successful team members' character attribute development in team games. We first model character attribute development using time series of attributes. We then find the thresholds to separate fast and slow attribute time series' growth rates by clustering and linear regression. We create a set of categorical attribute growth rates by comparing against the thresholds. If the growth rate is greater than the threshold it is categorized as fast growth rate; if the growth rate is less than the threshold it is categorized as slow growth rate. After obtaining the set of categorical attribute growth rates, we build a decision tree on the set. Finally, we characterize the patterns of team success in terms of rules which describe team members' character attribute growth rates.

To characterize the practicality and accuracy of our method, we tested it on game logs from three commercial games: DotA, Warcraft III, and Starcraft II. A standard machine-learning-style evaluation of the experimental results shows that the team members' character attribute growth rates are highly related to successful team strategies. When testing on new game logs, the patterns of the team success in terms of conjunctions of categorical attribute growth rates can predict the game results (win or lose) with an average of accuracy 86%.

[1] http://www.playdota.com/
[2] http://us.blizzard.com/en-us/games/war3/
[3] http://us.blizzard.com/en-us/games/sc2/
[4] http://www.dota2wiki.com/wiki/Role

II. Background

A. DotA

DotA (Defense of the Ancients) is currently one of the most popular action real-time strategy games. It is a more complex team-based multiplayer game. There are two teams in DotA: the Sentinel and the Scourge, each with five players. Each of the players select one character from a pool of 108 to be their "hero." Each team has an "Ancient," a building that their opponent must destroy to win the game. In DotA, there are three lanes the characters (heroes) can take to obtain experience and gold. The experience and gold in one lane are shared by all the characters in the lane.

Different heroes have different capabilities and have differing abilities to fill certain roles on their team. All characters (heroes) in DotA can be categorized into four major roles: *Carry*, *Ganker*, *Pusher*, and *Support*. Additionally, each hero has four major attributes: *Agility*, *Damage*, *Intelligence*, and *Strength*. Experience and gold can be used to enhance the four attributes. Attributes increase when upgrading or buying certain items. A Carry is "the hero that a team rallies around late in the game. They are the ones expected to have the highest number of hero kills for their teams. Carries typically lack early game power, but they have strong scaling skills; thus, they are highly dependent on items in order to be successful."[4] Gankers are "heroes with abilities that deliver long duration crowd control (ability that prevent, impede, or otherwise inhibit a Hero from acting) or immense damage early in the game. Their goal is to give the team an early game advantage during the farming phase by killing enemy heroes in their proper lanes."[4] Pushers are "heroes who focus on bringing down towers quickly, thereby acquiring map control. If they succeed, they often shut down the enemy carry by forcing them away from farming. They typically have skills that fortify allied creep waves, summon minions, or deal massive amounts of damage to enemy towers."[4] Supports are "heroes whose purpose are to keep their allies alive and give them opportunities to earn more gold and experience. Supports will usually come with skills such as healing spells or skills that disable enemies. Supports are not dependent on items, and thus, most of their gold will be spent on items such as Animal Courier, Observer Ward, Sentry Ward, and Smoke of Deceit."[4]

Agility, Damage and Strength are all equally important to the Carry. So the Carry is the most resource-hungry member of the team. The Carry always needs to occupy an entire lane by themselves for farming.[4] Intelligence is the most important attribute to the Gankers, Pushers, and Supports. Unlike the resource-hungry Carry, these three roles share the other two lanes. If one of these three roles consumes too many resources related to any attribute other than Intelligence, the Carry will have insufficient attribute development, which generally leads the team losing.

B. Warcraft III and Starcraft II

Warcraft and Starcraft are two popular real-time strategy (RTS) video games released by Blizzard Entertainment. In RTS games, players coordinate and control worker units to gather resources (such as gold and lumber in Warcraft and minerals and gas in Starcraft). With the resources' income, players can purchase or construct additional structures and units to grow their strength. The resources in a game are finite. So, when playing a team game such as a 2-vs-2 game, it is critical to have balanced player military strength development in the team. Although there is no "role" in RTS team games, the attribute growth rates patterns exist in the RTS team games. Military strength can be represented explicitly (as in capacity to inflict damage) or implicitly (as in the quantity of resources possessed). So, In Warcraft team games, each player has four attributes: *Gold*, *Lumber*, *Population*, and *Damage*. In Starcraft team games, each player has four attributes: *Mineral*, *Gas*, *Population*, and *Damage*. Due to the complex game dynamics and team strategies, finding successful team members' attribute development is a difficult task, a skill often taking professional players years to develop.

III. Related Work

To our knowledge this is the first effort to use a knowledge acquisition technique on game log data to obtain descriptions of successful strategies; however, building models of player behavior in general in games is not new. See Smith *et al.* [1] for an extensive survey of player modeling.

Limited work has focused specifically on build order. Kovarsky and Buro [2] is the earliest work introducing the build order optimization problem for real-time strategy games in 2006. They discuss how to deal with object creation and destruction in Planning Domain Definition Language (PDDL), the language used in the automated planning competitions. They apply planning to two problems: how to produce a certain number of units with less time and how to maximize the number of units produced within a predefined time period. The system they build is appropriate to develop build orders in an offline environment. In 2007, Chan *et al.* [3] developed an online planner for build order, focusing on resource collection in the RTS game of Wargus (an open source clone of Warcraft 2). Wargus is simple version of Starcraft because resource collection is simpler and the number of possible actions is small. Chan *et al.* employed means-end analysis scheduling to generate build order plans. The plans generated are not optimal because of the complex nature of the rescheduling problem. However, in some scenarios, they can beat plans generated by human players. Weber and Mateas [4] present a case-based reasoning technique for selecting build orders in the Starcraft RTS game. They apply conceptual neighborhoods to feature vectors in case-based reasoning in imperfect information game environments. Their experimental results show their method outperforms nearest-neighbor retrieval in imperfect information RTS games. As more research was done in this area, Branquinho and Lopes [5] proposed a new approach by combining Means-end analysis with Partial order planning (MeaPop) and Search and Learning A* (SLA*). Their method achieves plans with better plan duration. However, SLA* requires more time for scheduling some plans. Their methods are only being applied to Wargus, because StarCraft requires far more units and is therefore far more complex. Churchill and Buro [6] present heuristics and abstractions to solve build order problems in StarCraft. The heuristic and abstractions reduce the search effort and speed up the search, which produce near optimal plans in real-time. They test their method on an actual game-playing agent and the experimental results show the efficacy by comparing real-time performance with that of professional players.

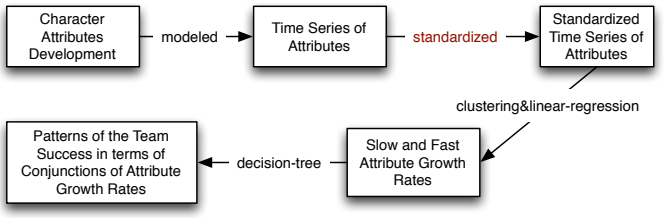

Fig. 1: The complete workflow. Character attribute development is modeled as attribute time series. Then the standardized time series are clustered and linear-regression is used to separate time series into fast and slow attribute growth rates. Finally, the patterns of the team success in terms of conjunctions of categorical attribute growth rates are extracted from the rules created by a decision tree model.

IV. Methodology

Our knowledge discovery approach for identifying patterns of attribute growth consistent with successful team play involves the following steps (which are represented in Figure 1):

1) A character's development is represented using it's attributes, the values of which evolve over time. The values may, or may not, evolve at regular intervals.
2) The attribute time series are made uniform in length by either up- or down-sampling and they are also normalized.
3) The thresholds to separate fast and slow attribute growth rates are found using clustering and linear regression. A set of categorical attribute growth rates are created by comparing against the thresholds. If the growth rate is greater than the threshold, it is categorized as fast growth rate; if the growth rate is less than the threshold, it is categorized as slow growth rate.
4) After obtaining the set of categorical attribute growth rates, a decision tree on the set is built. The input of the decision tree is the set of categorical attribute growth rates. The output of the decision tree is the rules in terms of conjunctions of categorical attribute growth rates that are predictive of team success. The patterns of the team success are characterized in terms of rules which describe team members' character attribute growth rates.

A. Modeling Character Attribute Development as Time Series

Characters' attributes evolve over time in response to events in the game. These events may occur at irregular intervals, making feature-based modeling difficult. Therefore, we model the development of characters' attributes as time series. These attributes are sampled at (possibly non-uniform) intervals to create time series data. Time series data have a natural temporal ordering which captures the variances in the attribute development. Patterns in the ways these attributes evolve over time form the basis upon which we can draw conclusions.

Note that we are modeling each character based on a number of attributes and that because we are interested in team games, there are multiple characters per team and at least two teams per game. Thus, a single game is actually modeled using a (potentially large) number of time series.

B. Standardizing Time Series

Different games result in time series of varying length and amplitude. Thus, to make the time series comparable between games, we re-sample to make them uniform length and normalize the values between 0 and 1.

To put all of the time series into uniform length we compute the average length of all the time series we have access to. Then we down- or up-sample each of the time series to be that length. We assume that the important information in the time series is contained in the local maxima and minima values. Therefore, when we down- or up-sample the time series, we always keep the local extremal values and interpolate or smooth the values in between. When up-sampling the time series, we interpolate additional values between the extremal values. When down-sampling the time series, we uniformly eliminate values between local extremal values to decrease the length to the average. There are two reasons to compute average instead of cutting to the minimal game length. First, some games are too short. If we cut to the minimal game length, the long games are down-sampled too much and may lose important information. Second, the majority of game lengths are near the average. Therefore, it is reasonable to use average.

Once the time series are of uniform length, we have to normalize their values to account for uncertainty. We normalize the values to be between 0 and 1 by the formula:

$$n(x, S) = \frac{x - min_S}{max_S - min_S} \tag{1}$$

where x is the original value of time series S, max_S is the global maximal value of the time series, and min_S is the global minimal value of the time series. $n(x, S)$ is then the normalized value of x.

C. Labeling Fast or Slow Attribute Growth Rates

In order to discover patterns of the team success in terms of conjunctions of categorical attribute growth rates, we first need to label time series of attributes with their growth rate. In this case, we focus on two growth rates: fast and slow. We use a clustering algorithm to group time series based on their growth rates.

There are many clustering algorithms, including K-means [7], DBSCAN [8], SOM [9], BIRCH [10], and CURE [11]. Among them, K-means partitions n observations into k clusters in which each observation belongs to the cluster with the nearest mean. The nearest mean is represented by a centroid within the cluster. Because of this, for the work described in this paper we use K-means with $k = 2$. We use Euclidean distance between the uniform-length and normalized time series to measure similarity for the K-means clustering algorithm. Each cluster has a centroid that is representative of the attribute growth rates of all time series belonging to the cluster.

After we obtain the centroids of the clusters for each attribute, we use linear regression to find the growth rate

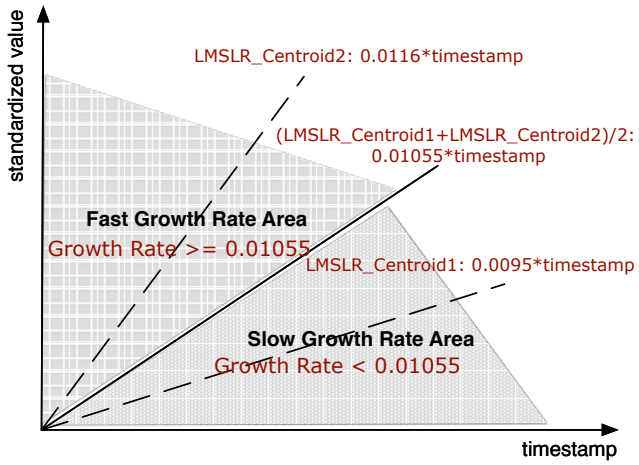

Fig. 2: How to find fast and slow growth rate areas. The two centroids of the two clusters: Centroid1 and Centroid2. The corresponding two LMSLRs are: LMSLR_Centroid1 = 0.0095*timestamp and LMSLR_Centroid2 = 0.0116*timestamp. The solid line is the decision boundary between the fast and slow growth rate areas.

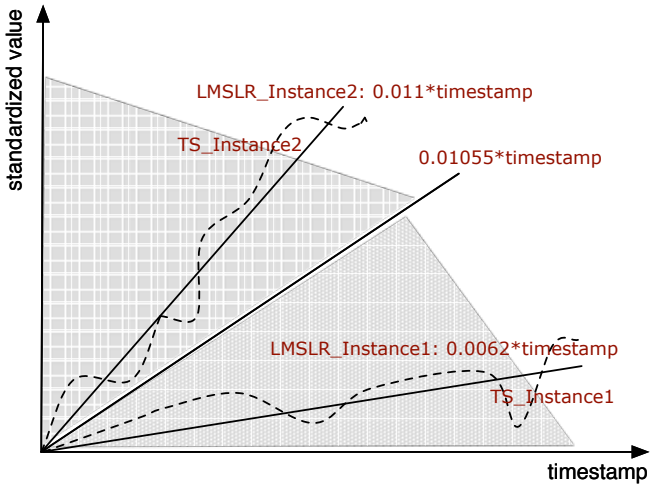

Fig. 3: How to label time series with fast or slow growth rate. Two instances of the time series: TS_Instance1 and TS_Instance2 (dotted curves). The corresponding two LMSLRs (bottom and top solid lines) are: LMSLR_Instance1 = 0.0062*timestamp and LMSLR_Instance2 = 0.011*timestamp. TS_Instance1 is labeled with slow attribute growth rate. TS_Instance2 is labeled with fast attribute growth rate.

values of cluster centroids. For the regression, the independent variable is timestamp and the dependent variable is the corresponding value of the centroid. We use Least Median Squared Linear Regression (LMSLR) [12] to obtain the underlying linear formula of each centroid time series. The reason we choose LMSLR is that it always gives us one stable solution. Note the generated centroid of the cluster is also a standardized time series.

For example, in Figure 2, the two centroids of the two clusters are: Centroid1 and Centroid2. The corresponding two LMSLRs are: LMSLR_Centroid1 = 0.0095*timestamp and LMSLR_Centroid2 = 0.0116*timestamp. Because we are primarily interested in the rate at which characters' attributes change over time as a predictor of their team role fulfillment, we omit the intercept part of the LMSLR linear model and focus on the slope. So, the growth rate of Centroid1 is 0.0095 and the growth rate of Centroid2 is 0.0116. We then compute the decision boundary between fast and slow growing time series by taking the average of the two slopes. In this case, the decision boundary is 0.01055. So, a growth rate $>= 0.01055$ is in the fast growth rate area; a growth rate < 0.01055 is in the slow growth rate area. Once the decision boundary has been identified, each of the time series that fall below the decision boundary are labeled as slow growing and those above are labeled as fast growing. This process is depicted graphically in Figure 3. The simple scheme of averaging the centroids' slopes works to create a decision boundary for k=2; however, the same basic principle of constructing decision boundaries between neighboring cluster centroids could apply to arbitrary numbers of clusters.

D. Discovering Patterns of the Team Success in terms of Conjunctions of Categorical Attribute Growth Rates

After finding the thresholds to separate fast and slow attribute time series' growth rates by clustering and linear regression and creating the set of categorical attribute growth rates, we build a decision tree on the set.

The decision tree builds a classifier with a tree structure from the instances in the set. The tree leaves represent a team win or loss. The tree branches represent conjunctions of categorical attribute growth rates that lead to those wins or loses. The decision tree algorithm we used is C4.5 [13]. The C4.5 algorithm uses "information gain" [14] as the splitting criterion for splitting the branch. At each splitting, the decision tree algorithm chooses the categorical attribute growth rate providing the maximum reduction in uncertainty about the team win or loss. So, the categorical attribute growth rate at the root of the tree is the one with the maximum information gain, and is therefore the best predictor. The categorical attribute growth rate used at the second level of the tree is the next best predictor given the value of the first [15].

After we build a decision tree model, tracing a path from the root to the leaves enables us to obtain the rules that are predictive of team success. Therefore, we can characterize the patterns of team success in terms of rules which describe team members' character attribute growth rates.

The C4.5 decision tree algorithm outputs a tree with many nodes, and therefore has a lot of rules; however, some of the branches do not represent enough examples to be generalizable. Therefore, we have two criteria for choosing the rules: confidence and support. Confidence is the percentage of games represented by the node in the decision tree that result in a win for one of the teams. Support is the number of games represented by the node in the decision tree. The higher the confidence, the more accurate the rule is. The higher the support, the more general the rules is. A rule is created by tracing the path of the decision tree from the root to a leaf which is above the thresholds for confidence and support.

V. EXPERIMENTS

We tested our approach on game logs from three commercial games: DotA, Warcraft III, and Starcraft II. Moreover, we did a machine-learning-style evaluation to validate that the patterns of team success in terms of rules which describe team members' character attribute growth rates achieve 86% prediction accuracy on average when testing on new game logs.

Here we will report the results of experiments using this technique on the three games listed above. DotA, being a five-on-five team game presents the most complexity and has more subtle strategies than the other two games. Therefore, we will devote a deeper analysis to DotA than the other games to demonstrate the subtle information our method is capable of capturing. Results from the other two games will demonstrate the generalizability of this approach.

A. DotA

We collected a total of 2,863 game logs played between 06/21/2010 and 02/14/2012. We used a crawler, NCollector Studio,[5] developed by Calluna Software to obtain the game logs from GosuGamers.[6] GosuGamers is an online community for DotA players covering some of the largest international professional and amateur gaming events. It contains an online database with logs from professional tournaments. The logs contain the information needed to generate the time series of representative attributes. When converting these binary logs to text-based game logs, we can obtain game length, game result, each player's character, the timestamps of each character's upgrading, and the timestamps and amount of gold for each purchased item.

There are 108 characters in DotA. According to tasks they perform in the game, they can be categorized into four major roles: Carry, Ganker, Pusher, and Support. According to strategy recommendations from the official DotA documentation, each team has only one Carry[4], at least one Ganker[4], at least one Pusher[4], and at least one Support[4]. Therefore, we analyzed three team compositions. Since five players control five characters to form a team, the three possible team compositions are:

1) one Carry, two Gankers, one Pusher, and one Support
2) one Carry, one Ganker, two Pushers, and one Support
3) one Carry, one Ganker, one Pusher, and two Supports

Recall that each of the team's five players has their own set of four attributes: agility, damage, intelligence, and strength. We filtered each of the four time series per character to be of uniform length and we normalized the values according the procedure described above. The result was a set of character attribute time series for each character that consisted of 60 time steps.

We applied the K-means clustering algorithm ($K = 2$) and least median squared linear regression to find the thresholds to separate fast and slow attribute time series' growth rates. After creating the set of categorical attribute growth rates by comparing against the thresholds, we built a decision tree on the set.

[5]http://www.calluna-software.com/

[6]http://www.gosugamers.net/dota/replays/

B. DotA Results

To obtain the rules from the decision tree, we used a confidence threshold of 70% and used 250 for the amount of support needed. The thresholds are usually adopted by the data-miners. In the future, we will use algorithms to find the best threshold values. Table I shows the summary of the patterns of the team success in terms of conjunctions of categorical attribute growth rates extracted from the rules created by the decision tree.

Once the decision tree has been constructed, it can be used to classify whether or not an individual player's progress was supportive or disruptive of success overall. Assuming a DotA team has played with one of the three combinations of player roles we examined in this work, they could take our model and rapidly perform a post-competition analysis of their play. If they are using a different combination of roles, they can always rebuild the clusters, decision boundary, labels, and decision tree model using a corpus with examples of the team play dynamics they use.

Due to space constraints, we are unable to discuss all discovered patterns. Here, we will describe two extracted rules in detail as examples of how this approach allows us to describe character performance. For pattern 2: "IF G-Str $<$ 0.01055 and G-Int $>$ 0.01085 THEN team (composed of one Carry, two Gankers, one Pusher, and one Support) wins with 89.1% chance." Gankers do not invest resources to develop their Strength attribute, they invest resources to develop their Intelligence attribute by which they use magic to stop opponents from farming resources and to enhance their teammates' farming, especially resource-hungry Carries. Moreover, they save strength resources which are less important to the Ganker for a team win and provide opportunities (farming lanes) to other teammates. Pattern 7: "IF G-Int $>$ 0.01085 and P-Int $>$ 0.01125 THEN team (composed by one Carry, one Ganker, two Pushers, and one Support) wins with 86.7% chance." The team with two Pushers means the team's strategy focuses on destroying towers as quickly as possible (a "quick-rush" team strategy). The Pusher is the role to take responsibility for destroying (pushing in DotA slang) towers using magic skills. So, a fast growth rate for a Pusher's Intelligence is essential to this team's strategy. However, a Pusher is very vulnerable to other roles like Carry and Ganker. So, in order to achieve the "quick-rush" team strategy, teammates must try their best to protect Pushers. The Ganker must deliver long duration crowd control via magic skills (Intelligence attribute) to save enough time for the Pusher to escape the battlefield. Therefore, a fast growth rate of the Pusher's Intelligence and a fast growth rate of the Ganker's Intelligence are essential to the "quick-rush" team strategy. The fast growth rates of other attributes are not necessary for the "quick-rush" team strategy.

We can also obtain more interesting knowledge by comparing all discovered patterns in Table I.

First, Carry is the only role which doesn't appear in all patterns, although Carry is the role which carries and leads a team to victory. This indicates the DotA game is a highly team-oriented game. Although Carry bears the responsibility for ultimate victory, the outcome highly depends on the attribute growth patterns of the other roles.

Second, Table I shows growth rate patterns of Gankers are

TABLE I: Summary of the patterns of the team success in terms of conjunctions of categorical attribute growth rates for DotA extracted from the rules created by the decision tree. The confidence threshold we used is 70%. The support threshold we used is 250. C, G, P, S means Carry, Ganker, Pusher, Support individually. 1C+2G+1P+1S means team has one Carry, two Gankers, one Pusher, and one Support. Agi, Dam, Int, Str means Agility, Damage, Intelligence, Strength individually. Win means the team wins the game. The numeric value in the IF statement is the decision boundary between fast and slow growth rate areas.

Team Compositions	Win Confidence	Patterns of the Team Success in terms of Conjunctions of Categorical Attribute Growth Rates
1C+2G+1P+1S	75.8%	1. IF G-Str $<$ 0.01055 THEN Win
	89.1%	2. IF G-Str $<$ 0.01055 and G-Int $>$ 0.01085 THEN Win
	84.7%	3. IF G-Str $<$ 0.01055, G-Int $<$ 0.01085, and S-Int $<$ 0.0103 THEN Win
	84.9%	4. IF G-Str $<$ 0.01055 and G-Int $>$ 0.01085 THEN Win
	76.2%	5. IF G-Str $<$ 0.01055, G-Int $<$ 0.01085, and G-Dam $>$ 0.00645 THEN Win
1C+1G+2P+1S	78.3%	6. IF G-Int $>$ 0.01085 THEN Win
	86.7%	7. IF G-Int $>$ 0.01085 and P-Int $>$ 0.01125 THEN Win
	72.8%	8. IF G-Int $>$ 0.01085 and G-Str $>$ 0.01055 THEN Win
	77.4%	9. IF G-Int $>$ 0.01085, P-Int $>$ 0.01125, and P-Str $<$ 0.0126 THEN Win
	81.0%	10. IF G-Int $>$ 0.01085, P-Int $>$ 0.01125, P-Str $<$ 0.0126, and S-Int $<$ 0.0103 THEN Win
1C+1G+1P+2S	85.1%	11. IF S-Int $<$ 0.0103 THEN Win
	75.3%	12. IF S-Int $<$ 0.0103 and G-Int $>$ 0.01085 THEN Win
	87.8%	13. IF S-Int $<$ 0.0103, G-Int $>$ 0.01085, and P-Str $<$ 0.0126 THEN Win

highly associated with a team's game results. The reason is that Gankers are heroes with abilities that deliver long duration crowd control or immense damage early in the game. Their goal is to give the team an early game advantage during the farming phase by killing enemy heroes in their proper lanes. Their main role is to stop opponents from farming resources and to provide a good environment for teammates to farm as quickly as possible. Since Gankers mainly depend on magic skills (the Intelligence attribute) to play well, it is unsurprising that the Ganker's Intelligence attributes occurs in 12 out of 13 optimal growing patterns in Table I.

Third, team strategies can be reflected by the patterns. For example, if a team's composition is one Carry, two Gankers, one Pusher, and one Support, the attribute growth rate patterns (patterns 1 to 5) mention 11 roles and 10 of the 11 in those patterns are Gankers. So, the team strategy mainly lies in the Gankers' growth rate patterns. The team strategy is to let Gankers kill as many enemy heroes as possible and gain an early game advantage during the farming phase. If the Ganker is successful, the team's Carry will gain a significant advantage while the opponent's Carry will be suppressed. If a team composition is one Carry, one Ganker, two Pushers, and one Support, the attribute growth rate patterns (patterns 6 to 10) mention 12 roles and 5 of the 12 roles in the patterns are Pushers. The team strategy mainly focuses on Pushers' growth rate patterns. The team strategy is to let Pushers bring down towers quickly and shut down the enemy Carries by forcing them away from farming. If a team composition is one Carry, one Ganker, one Pusher, and two Supports, the attribute growth rate patterns (patterns 11 to 13) mention 6 roles and 3 of the 6 roles are Supports. The team strategy is to let Supports keep their allies alive and give them opportunities to earn more gold and experience.

We performed 10-fold cross-validation to validate the accuracy of our model. The results are presented in Table II. Because DotA is an adversarial game, this is a binary classification problem: team Sentinel wins or loses (which is the same as a Scourge win). This arbitrary choice didn't affect the accuracy. If the Sentinel team wins it is a true positive (TP).

TABLE II: Summary of results of 10-fold cross-validation evaluation metrics across three team compositions in DotA. C means Carry; G means Ganker; P means Pusher; S means Support. 1C+2G+1P+1S means team has one Carry, two Gankers, one Pusher, and one Support. Classification accuracy (CA), Sensitivity (Sens), Specificity (Spec).

Team Compositions	CA	Sens	Spec
1C+2G+1P+1S	0.8322	0.8564	0.8037
1C+1G+2P+1S	0.8290	0.8532	0.7980
1C+1G+1P+2S	0.8438	0.9055	0.7510

If the Scourge team wins it is a true negative (TN). Table II shows all values are above 0.75 for all team compositions. The average accuracy is 83.5%.

C. Warcraft III

We collected a total of 2,325 2-vs-2 game logs from a Warcraft III replays website.[7] There are four races a player can choose: Human, Night-elf, Orc, and Undead. So, there are $4*4 = 16$ possible team compositions for a 2-player-team. We represented each player as four attribute time series. The four attributes we used are Gold, Lumber, Population, and Damage. The average game length is 32 timesteps. Therefore, we up- or down-sampled the attribute time series to be 32 samples long using the procedure described above. As before, we applied the K-means clustering algorithm, found the decision boundary between fast and slow attribute growth rates, created the set of categorical attribute growth rates, and constructed the decision tree model.

D. Warcraft III Results

To obtain the rules from the decision tree, we used confidence threshold 70% and 250 for the amount of support needed. Due to space constraints, we are unable to list all

[7] http://w3g.replays.net/

TABLE III: Summary of the patterns of the team success in terms of conjunctions of categorical attribute growth rates for Warcraft III extracted from the rules created by the decision tree. The confidence threshold we used is 70%. The support threshold we used is 250. H, N, O, U means Human, Night-elf, Orc, Undead individually. Gol, Lum, Pop, Dam means Gold, Lumber, Population, Damage individually. 1H+1O means team has one Human race and one Orc race. Win means the team wins the game. The numeric value in IF statement is the decision boundary between fast and slow growth rate areas.

Team Compositions	Win Confidence	Patterns of the Team Success in terms of Conjunctions of Categorical Attribute Growth Rates
1H+1O	75.1%	1. IF H-Pop $<$ 0.0317 THEN Win
	88.7%	2. IF H-Pop $<$ 0.0317 and O-Gol $>$ 0.0249 THEN Win
	94.1%	3. IF H-Pop $<$ 0.0317, O-Gol $>$ 0.0249, and H-Dam $>$ 0.0403 THEN Win
	96.2%	4. IF H-Pop $<$ 0.0317, O-Gol $>$ 0.0249, and O-Pop $<$ 0.0645 THEN Win

discovered patterns for all 16 team compositions. So, we use 1H+1O team composition as an example. The top four patterns are listed in Table III. From it, we can conclude Human's population is critical to the team. The Human's Population attribute growth rate should not be greater than 0.0317. In that case, the team has a 75.1% chance to win. The Orc's gold is second-most critical to the team. When the Orc's Gold attribute growth rate is greater than 0.0249, the team increases its chance to win by 13%. Furthermore, when the team also makes the Human's Damage growth rate greater than 0.0403 or the Orc's Population growth rate less than 0.0645, the team's chance of winning increases by 19% or 21% respectively.

We performed 10-fold cross-validation to validate the reliablity of our patterns of growth rates. Because Warcraft III is an adversarial game, this is a binary classification problem: team win or loss. If a team wins it is a true positive (TP). If the other team wins it is a true negative (TN). The average accuracy is 87%. The sensitivity is 0.88; the specificity is 0.91; the AUC is 0.89.

In order to compare, we additionally collected 3,564 1-vs-1 game logs from the same Warcraft III replays website and repeated the above procedures. We found no rules above the 70% confidence and 250 support thresholds. Therefore, the patterns of the team success in terms of conjunctions of categorical attribute growth rates are common in team games and are not common in non-team games. The reason is that in non-team games the attribute growth rates (resources obtainment) are more free than in team games. In non-team games different individual strategies have different attribute growth rates.

E. Starcraft II

We collected a total of 1,847 2-vs-2 game logs from three Starcraft II replay websites.[8] There are three races a player can choose: Protoss, Terran, and Zerg. Therefore, there are $3*3 = 9$ team compositions for a 2-player-team. We represented each player as four attribute time series. The four attributes we used are Minerals, Gas, Population, and Damage. Since different armor reduces different damage, we use raw damage value. The average game length is 29 timesteps. Therefore, we up- or down-sampled the attribute time series to be 29 samples long using the procedure described above. As before, we applied the K-means clustering algorithm, found the decision boundary between fast and slow attribute growth rates, created the set of categorical attribute growth rates, and constructed the decision tree model.

[8]http://www.gosugamers.net/starcraft2/replays/, http://www.sc2win.com/, http://www.gamereplays.org/starcraft2

F. Starcraft II Results

To obtain the rules from the decision tree, we used confidence threshold 70% and 200 for the amount of support needed. Due to space constraints, we are unable to list all the discovered patterns for all nine team compositions. So, we use 1T+1Z team composition as a representative example. The top four patterns are listed in Table IV. In 1T+1Z team composition, the Zerg's Population is important which is consistent to most Zerg's tactics. Since Zerg's units are relatively cheaper than Protoss's and Terran's, the Zerg's tactics usually involve a large amount of units. When the Zerg's Population attribute growth rate is greater than 0.029, the team has a 73.3% chance to win. Since a large amount of units consume both population and minerals, the team's chance of winning increases to 86% when the Zerg's Minerals attribute growth rate is greater than 0.0582. By comparing pattern 2 and 3, the team's win chance goes up to 94.7% when the Terran's Damage attribute growth rate is greater than 0.0403. Interestingly, if the Terran's Minerals attribute growth rate is greater than 0.0582 and the Terran's Population growth rate is less than 0.0629, the team still can have a 94.4% chance of winning. The reason is likely that when the Terran increases harvesting minerals quickly but maintains a slow population growth, they are able to invest resources in high-tech. With the Zerg's population advantage and the Terran's technology advantage, the team can achieve an overall advantage over their opponent.

We performed 10-fold cross-validation to validate the reliability of our patterns of growth rates. Because Starcraft II is also an adversarial game, this is also a binary classification problem: a team wins or loses. If a team wins it is a true positive (TP). If the other team wins it is a true negative (TN). The average accuracy is 86%. The sensitivity is 0.90; the specificity is 0.83; the AUC is 0.85.

In order to compare, we also collected 2,450 1-vs-1 game logs from the same three Starcraft II replay websites and performed the above procedures. We also found no rules above the 70% confidence and 200 support thresholds. Therefore, we also can draw the same conclusion as in Warcraft III that patterns of the team success in terms of conjunctions of categorical attribute growth rates are common in team games and not in non-team games.

TABLE IV: Summary of the patterns of the team success in terms of conjunctions of categorical attribute growth rates for Starcraft II extracted from the rules created by the decision tree. The confidence threshold we used is 70%. The support threshold we used is 200. P, T, Z means Protoss, Terran, Zerg individually. Min, Gas, Pop, and Dam means Minerals, Gas, Population, and Damage individually. 1T+1Z means the team has one Terran race and one Zerg race. Win means the team wins the game. The numeric value in IF statement is the decision boundary between fast and slow growth rates areas.

Team Compositions	Win Confidence	Patterns of the Team Success in terms of Conjunctions of Categorical Attribute Growth Rates
1T+1Z	73.3%	1. IF Z-Pop $>$ 0.0290 THEN Win
	86.4%	2. IF Z-Pop $>$ 0.0290 and Z-Min $>$ 0.0582 THEN Win
	94.7%	3. IF Z-Pop $>$ 0.0290, Z-Min $>$ 0.0582, and T-Dam $>$ 0.0403 THEN Win
	94.4%	4. IF Z-Pop $>$ 0.0290, T-Min $>$ 0.0582, and T-Pop $<$ 0.0629 THEN Win

VI. Future Work

There are a number of exciting avenues for future research. First, to determine how successfully we can guide the gameplay of using players with different skill levels (novice, median, expert) using the patterns of successful attribute growth rates. Second, we would like to further validate our rules with professional players to further double-check or filter the patterns in successful team members' character attribute growth rates and create a knowledge base. This knowledge base can be used to guide professional players' training progress and amateur players' learning progress. Third, our method has three free parameters: confidence, support and the number of clusters. One avenue of future research involves using an optimization algorithm, such as a genetic algorithm [16] or randomized hill climbing [17], to determine the best values for these parameters. This way, we can ensure that there is a solid reasoning behind picking a specific threshold value. Lastly, we hope to use the knowledge learned from discovering how effective team members play to set goals for AI game agents that will help them play more successfully.

VII. Conclusion

In this paper, we have introduced an approach for automatically discovering patterns in successful team members' character attribute development in team games. We first model the team members' character attribute development using attribute time series. We then cluster the standardized time series of attributes into two clusters: time series indicative of fast attribute growth rates and time series indicative of slow attribute growth rates. Linear regression is used to find the growth rate values of cluster centroids of both the fast cluster and the slow cluster. Finally, we characterize the patterns of the team success in terms of conjunctions of categorical attribute growth rates by building a decision tree model. The enemy team composition and players performance impact the attribute growth rates. For example, a team of players play differently when they face a different enemy team composition, which is reflected by the game logs. Since our method is based on the game logs, the impact of different enemy team compositions and players performance is considered completely.

In this work we opted to use just two growth rates (fast and slow) for our analysis; however, there is no limitation in the technique that requires it. While the results we got using just two growth rates were highly accurate, it would be interesting for future work to examine the effects of using different numbers of growth rate clusters.

We have shown it is possible to discover the patterns of the team success in terms of conjunctions of categorical attribute growth rates in team games using data only. The only knowledge engineering in our method involves formatting the data properly and contains no value judgements or expert opinions. By moving away from knowledge-based methods, we can make post-competition analysis for players more efficient. With our technique, they can easily investigate which characters do harm to other characters. This is hard to figure out directly from the game logs without our method.

References

[1] H. S. Adam, Lewis and Sullivan, "An Inclusive Taxonomy of Player Modeling," *UCSC-SOE-11-13*, 2011.

[2] A. Kovarsky and M. Buro, "A First Look at Build-order Optimization in Real-Time Strategy Games," in *GameOn Conference*, 2006.

[3] H. Chan, A. Fern, S. Ray, N. Wilson, and C. Ventura, "Online planning for resource production in real-time strategy games," in *ICAPS*, M. S. Boddy, M. Fox, and S. Thiébaux, Eds. AAAI, 2007, pp. 65–72.

[4] B. G. Weber and M. Mateas, "Case-based reasoning for build order in real-time strategy games," in *AIIDE*, C. Darken and G. M. Youngblood, Eds. The AAAI Press, 2009.

[5] A. A. B. Branquinho and C. R. Lopes, "Planning for resource production in real-time strategy games based on partial order planning, search and learning," in *Systems Man and Cybernetics (SMC)*. IEEE, 2010.

[6] D. Churchill and M. Buro, "Build order optimization in starcraft," in *AIIDE*, V. Bulitko and M. O. Riedl, Eds. The AAAI Press, 2011.

[7] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the 5th Berkeley Symp. on Mathematics Statistics and Probability*, L. M. LeCam and J. Neyman, Eds., 1967.

[8] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[9] T. S. H. T. Kohonen and M. R. Schroeder, *Self-Organizing Maps*, 3rd ed. Springer-Verlag, December 2000.

[10] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *Proceedings of International Conference of Management of Data*, June 1996.

[11] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large databases," *ACM SIGMOD Record*, 1998.

[12] P. J. Rousseeuw, "Least median of squares regression," *American Statistical Association Journal*, vol. 79, pp. 871–880, 1984.

[13] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers, 1993.

[14] T. M. Mitchell, *Machine Learning*. The Mc-Graw-Hill Inc., 1997.

[15] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[16] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[17] S. Russell and P. Norvig, *Artificial intelligence: a modern approach (2nd edition)*. Prentice Hall.