What did I do Wrong in my MOBA Game?: Mining Patterns Discriminating Deviant Behaviours

Submitted for double blind review
Anonymous authors
An affiliation
Contact email

***Abstract*—The success of electronic sports (eSports), where professional gamers compete in famous leagues and tournaments, brings new challenges to the video game industry. These games must be difficult and challenging for eSports professionals but still enjoyable for amateurs while letting them feel a learning curve: such games require *"a minute to learn, a lifetime to master"*. In this article, we consider *muti-player online battle arena* games (MOBA) and especially DOTA2. In this context, a challenge is to propose data analysis methods and metrics that help players to improve. For that, we design a data-mining method that discovers strategic patterns from historical behavioural traces: Given a model encoding an expected way of playing, these patterns give explanation of behaviours deviating from the norm, and thus also give advices to the player. The method is formally introduced, shown to be customizable to handle several scenarios. We experimented with a dataset of $10,000$ behavioural game traces and show how the discovered patterns can explain the deviant behaviours and advice the player for his next games.**

I. Introduction

The video game industry has been dramatically expending over the last few years , targeting all populations (young, adults, extreme players, ...) and electronic devices (PC, consoles, smartphones, ...). Indeed, over the last decade, the annual turnover generated by the electronic entertainment industry went beyond those of both cinema and music industries, making video games production one of the most lucrative business ever developed. In the meantime, a new scene called electronic sport (eSports) has emerged: the most skilled gamers are hired by professional teams, surrounded by sponsors, and compete in international tournaments [1], widely followed on live streaming platforms such as *Twitch.tv* [2].

The success of eSports brings also new challenges. Although eSport games represent only a tiny proportion of all video games, they gather masses around an editor and its sponsors, thus coming with new and lucrative business models: for example, the *League of legends 2015 World Finals* counted 36 million unique viewers and awarded the winner with a US$ one million prize [3]. From the point of view of the editor and developers, a video game destined to be an eSport cannot be only made for professional players: it requires a fan base of amateurs as for any sport. The fans shall know the game to understand and appreciate the strategic moves, mechanics of professionals (traps, betting [4]) during a game: they shall have played and, of course, enjoyed the game.

As a consequence, several games are played both at the amateur and professional levels and their design is a tedious task for the editors/developers. Indeed, these games must be difficult, balanced and challenging for eSports but still fun and playable for the amateurs, letting them feel a rewarded learning: mastering such games is a long and difficult process, as given by the saying *"A minute to learn, a lifetime to master"*. This dilemma makes the editors regularly patch their games: game play properties are adjusted, some options are removed or added. Indeed, expert players are able to find unexpected and unbalanced ways of playing, and the viewers then reproduce these abusive behaviours in their games. Balancing the game is thus crucial, but it is sometimes not enough given that a player may not know all the strategic possibilities that are offered to him. Introducing leagues in most of the eSports makes players of the same level challenge each other and partially solves the problem. We believe that it is possible to go a step further by analysing the player behaviours (movements and strategic choices) and comparing them to chosen and/or expected behaviours. This could help the player to improve by giving him advices and this is what we develop in this article.

We consider multiplayer online battle arena games (MOBA). We choose DOTA2 because it is a high level eSport strategy game with rich behavioural data containing both, mobility data and strategic choices. Given a set of game logs (historical data), our goal is to discover strategic patterns that can help the player to understand his games and improve. For example, considering all the games of a player in the current season (or patch), we would like to answer to questions such as: What are the particular choices (e.g. build orders, map, team composition, etc) that discriminate victory? What did I do wrong in my game, that is, that deviates from the norm (way of playing in my league, in the current season, etc.) no matter the outcome?

Simple database queries cannot answer these questions: we design a data-mining method that aims to discover patterns composed of strategic choices and that explain strong deviations w.r.t a model. A model can simply gives the game outcome, and we are thus looking for patterns discriminating victory. The model can be also more sophisticated and encode an expected way of playing the game or behaviour: we can score how much each game of a player deviates from the model and mine patterns that explain the deviation and advice the player for his next games. This methodology is also fully customizable to handle several scenarios that we experimented with a dataset of $10,000$ behavioural game traces.

The paper is organized as follows. We present the MOBA games principles and our problem settings in Section II. Basics

pid	Traces	Global information
1	(t_1, -6973,-6428, buy_X),(t_2, -6742,-5290, ab_{A_1}),(t_3, -6440,-750, move) ,(t_4, -4801,5725, move) ,(t_5, -938,563, move), ...	winner=Yes, length=42,...
2	(t_1, -6980,-6440, buy_X),(t_2, -2077,-1550, move),(t_3, -6922,-6185, ab_{A_1}) ,(t_4, -620,-5963, move),(t_5, -6650,-6338, buy_Y), ...	winner=No, length=60,...
3	(t_1, -6928,-6436, buy_X),(t_2, -6050,-902, move),(t_3, -4825,-6130, move) ,(t_4, 10,5962, ab_{A_1}),(t_5, -4900,-6045, move), ...	winner=Yes, length=54,...
4	(t_1, -6806,-6334,buy_X),(t_2, 25,100, move),(t_3, 50,-160, ab_{A_1}) ,(t_4, -702,-5802, move),(t_5, -7105,-6593, buy_Z), ...	winner=No, length=35,...
5	(t_1, -6896,-6450, move),(t_2, -650,-6004, move),(t_3, 4890,-5986, ab_{A_1}) ,(t_4, -598,-5785, move),(t_5, -7445,-6985, ab_{B_2}), ...	winner=Yes, length=38,...

Table I: Simplified example of rough game traces

from pattern mining are then introduced in Section III allowing us to develop our methodology in Section IV. We present our dataset and an experimental validation with several scenarios in Section V. Related work is detailed in Section VI before conclusion.

II. Problem Settings

A. Multiplayer Online Battle Arenas (MOBAs)

MOBA is a specific video game type that mix real-time strategic game and role-play game. We focus on *Defence of the ancient 2* (DOTA2, [5]), but there are several other well-known MOBA games that share the same principles with slight differences (e.g. *Heroes of the storm*, *league of legends*, etc.). This choice is made by data availability and has no impact on the methodological aspects developed hereafter.

A DOTA2 game is played on a map where two teams of five players are battling each other in real time. Each team has to defend their own stronghold and destroy the opponent's one to win. Each player controls a *hero* that he moves on the map, and needs to train, by collecting gold, new items and abilities, and by fighting opposing heroes. Figure 1 displays the initial influence zone of both teams. The red team called *the dire* (resp. blue for *the radiant*), defends their stronghold at the top right corner (resp. bottom left). Three lanes (*top*, *mid*, *bot* in Figure 1 (i)) separate the teams, on which defensive towers are set. The players have well defined roles, depending on the heroes they initially picked and their properties (110 available heroes), but also depending on a few crucial choices during the game. One role consists of defending and extending the influence zone in a specific lane; another is to quickly switch lanes to attack by surprise. Like rugby, positions and moves are really important and can determine the issue of the whole game (defend towers and base, attack players by surprise, make a concerted and synchronized attack on barracks). Knowing that a team only sees controlled map zones, estimating enemies positions and triggering team fights at well-chosen times and in well-chosen areas is key to success.

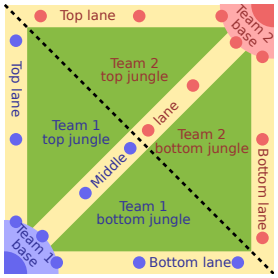


Figure 1: MOBA map

B. MOBA behavioural data

Most eSport games log all actions, events and entities positions into a single file called *replay*. Such a file contains at least all the informations needed for the game engine to replay the game exactly as it was originally. It allows one to exchange replays, observe either his opponents or himself, analyse particular phases and metrics, cast TV shows on live streaming platforms or video sharing websites. Most importantly, fans (sometimes editors) provided replay parsers that can extract all its content mainly for proposing and enhancing community website.[1] This opened exciting doors for data analysts in various domains (AI, data mining, machine learning, etc.) Concerning DOTA2, the kind of available events for each player and information we are interested in are:

- **Positions**: They are expressed in x,y coordinates on the map where (0,0) is the center (Figure 1).
- **Builds**: Several times in a game, a hero can choose a new ability (or upgrade) within a tree. These choices should be carefully made as definitive and inter-dependant. It completely guides the strategy: a wrong build order can lead to poor performing heroes.
- **Items**: Items have specific boosts (strength, defence, magic) and powers and cost an amount of gold depending of their attributes. We know when and where items are bought for each hero.
- **Gold and experience time series**: It gives at any moment the gold and experience earned by the heroes (by eliminating enemy minions and heroes).
- **Global information**: winner, game length, team, steam id of the players, heroes, etc.

Example. Table I gives traces examples: Each row of the table is a series of actions made by one player during one game.

C. Mining patterns explaining deviant behaviours

In these settings, our goal is to discover useful patterns for the user that can help him improve his game. A straightforward way is to consider all traces of a single player, and within, all strategic choices he made. Then, pattern mining techniques allow us to find the largest sets of choices (among an exponential number of possibilities) that were frequently made and appear mostly in losing games. In the next section, we will present the basics of pattern mining and will detail this example.

In the previous paragraph, the target for which we want to find discriminant patterns is known: the outcome of the game. Sometimes it is not the case: one may have won with his team, but did not play as the norm (e.g. other players of the same level). As such, we propose to model the normal behaviour and measure how much a player deviates from the norm. An example, among many others available in [6], is given in Figure 2: it clearly shows mobility norms in MOBA. This gives us a new target: did the player behaved like he should? Then, pattern mining allows us to discover the most discriminant patterns that explain a deviation. In section IV, we describe and illustrate this more sophisticated approach.

[1] See e.g., http://www.dotabuff.com and https://github.com/skadistats/clarity

III. Background on Pattern Mining

In this section, we present through an example the basic notions of pattern mining that will be needed to understand our methodology. First, we introduce the problem of *Frequent itemset mining* [7] before explaining how to discover *Emerging patterns* that discriminate a class label [8].

A. Mining frequent itemsets

Consider a set of transactions, where each transaction is composed of a set of items (in the original problem formulation, a set of items bought in a supermarket [7]). An arbitrary itemset is said to be frequent if the items it contains have been frequently bought together (i.e. more than a user defined threshold). One can easily draw a parallel with MOBA: a transaction could be the set of items bought by a single player in a single game.

Definition 1 (Frequent itemset): Let $\mathcal{I}$ be a set of items. A transaction t is a set of items $t \subseteq \mathcal{I}$. A (transaction) database is a set of transactions $\mathcal{D} = \{t_1, t_2, ..., t_n\}$ with $t_i \subseteq \mathcal{I}, \forall i \in [1, n]$. An itemset X is an arbitrary set of items. Its support is given by the transactions of the database that contain the set of items:

$$supp_{\mathcal{D}}(X) = |\{t | t \in \mathcal{D}, X \subseteq t\}|$$

The frequency of an itemset is the proportion of transactions in the database that contain the itemset:

$$freq_{\mathcal{D}}(X) = \frac{supp_{\mathcal{D}}(X)}{|\mathcal{D}|}$$

The *frequent itemset mining problem* consists in efficiently finding all the frequent itemsets in a transaction database $\mathcal{D}$, that is, with a frequency higher than a minimal frequency threshold θ: all $X \subseteq \mathcal{I}$ such that $freq_{\mathcal{D}}(X) > \sigma$. Equivalently, a minimum threshold on the support size is denoted by min_sup.

Note that several constraints other than minimal frequency can be used, such as minimal size of the itemset, minimal cost of an itemset (in the case where each item has a cost assigned), maximization of a function given by a mathematical model or several condensed representations that reduce the number of patterns [9]. Indeed, as there is an exponential number of itemset ($2^{|\mathcal{I}|}$ in the worst case), a naïve exploration of all itemsets is unachievable: the mathematical properties of constraints can be taken into account for an efficient extraction. Algorithm issues will be discussed in the Section IV.

Example. Consider a transaction database where each transaction consists in the set of items bought by a single player in a single a game. We have for example 5 items: $\mathcal{I} = \{a, b, c, d, e\}$ and an example of transaction database is given in the Table II. We have $supp_{\mathcal{D}}(\{a, b, c\}) = 2$, $supp_{\mathcal{D}}(\{a, b\}) = 3$, $freq_{\mathcal{D}}(\{a, b\}) = 0.6$ and $freq_{\mathcal{D}}(\{a, b, c\}) = 0.2$. If we set the minimal frequency threshold $\sigma = 0.3$, we have that $\{a, c\}$ is frequent while $\{a, b, c\}$ is not a frequent itemset.

t_{id}	transaction
t_1	$\{a, b, c\}$
t_2	$\{a, b, c\}$
t_3	$\{c\}$
t_4	$\{a, b, e\}$
t_5	$\{a, e\}$

Table II: Transaction database

B. Mining itemsets that distinguish a class label

Consider now that each transaction is provided with a label, say positive or negative, for win or lose[2] as depicted in Table III. Patterns that strongly discriminate a label from the other give interesting and comprehensive hypotheses for that label. The main idea is to consider two transactions databases, made of positive and negative examples respectively and then to compute the support of an itemset in both databases separately. Intuitively, if their difference is high, the pattern may be a signature of one specific base. More formally, the mapping

$$class : \mathcal{D} \to \{+, -\}$$

associates to any transaction its class label (positive or negative). Then, we can define the positive and negative bases:

$$\mathcal{D}^+ = \{t | t \in \mathcal{D}, class(t) = +\} \text{ and } \mathcal{D}^- = \mathcal{D} \backslash \mathcal{D}^+$$

Thanks to this partitioning, we can compute a measure that expresses how an itemset discriminates a class label and not another. This notion is the so called *emerging patterns* and the measure is called *growth-rate* [8].

Definition 2 (Normalized growth-rate): Given the databases $\mathcal{D}^+$ and $\mathcal{D}^-$, the normalized growth-rate of a pattern $X \subseteq I$ is given by:

$$\phi(X) = \frac{|supp_{\mathcal{D}^+}(X)| - |supp_{\mathcal{D}^-}(X)|}{|supp_{\mathcal{D}^+}(X)| + |supp_{\mathcal{D}^-}(X)|}$$

where $supp_{\mathcal{D}^+}(X)$ (resp. $supp_{\mathcal{D}^-}(X)$) counts the number of transactions supporting X in the positive database (resp. negative). This means that if $\phi(X) = -1$ (resp. 1) then the itemset X appears only with a negative (resp. positive) label. On the other hand, if $\phi(X) = 0$ then X appears as much in the positive than in the negative database.

Example. We consider the same example as before, but we add as class attribute the outcome of the game, see Table III. The positive examples (resp. negative) correspond to a win (resp. loss). We have: $\phi(\{a\}) = (2-2)/(2+2) = 0$, $\phi(\{a, b\}) = (2-1)/(2+1) = 0.33$, $\phi(\{a, b, c\}) = (2-0)/(2+0) = 1$ and $\phi(\{e\}) = (0-2)/(0+2) = -1$. Consequently, choosing a, b and c can be interesting for a player as it discriminates victory and as it was played relatively often ($freq_{\mathcal{D}^+}(\{a, b, c\}) = 66.66\%$, $freq_{\mathcal{D}}(\{a, b, c\}) = 20\%$). In this scenario, we are interested in itemset with highest support and growth-rate.

t_{id}	$class(t_{id})$
t_1	$+$
t_2	$+$
t_3	$+$
t_4	$-$
t_5	$-$

Table III: Label of each transaction

C. Handling other types of patterns

We introduced (frequent) (discriminant) itemsets, the simplest kind of patterns. The pattern domain, called *language* $\mathcal{L}$, can also vary: instead of sets, we could have sequences of sets, vector of numbers, trees, graphs, etc. The choice of a pattern domain to encode the original data depends on the goals of the study and the required expressiveness. In all cases, the aim is to compute the theory $Th(\mathcal{D}, \mathcal{L}, \mathcal{C}) = \{\omega \in \mathcal{L} | \mathcal{C}(\omega, \mathcal{D}) = true\}$ where $\mathcal{D}$ is the dataset, $\mathcal{L}$ is the pattern language, and $\mathcal{C}$ is a conjunction of constraints to be checked by all patterns ω could be extracted (minimal frequency or growth-rate, etc) [10].

[2]Multi-labelled data, for example with the played hero as class label, can also be handled directly but are omitted here for sake of simplicity.

Figure 2: Early game positions in *League of Legends* [6]

IV. Method

In this section, we present our methodology for discovering patterns that explain deviant behaviours. First we detail the main idea and notions through an example. Each step of the approach is then formalized and specified in the following.

A. Running example

Our example starts from the assumption that hero positioning is key in MOBA such as *DOTA2* and *League of legends*. Indeed, players with a particular hero and/or role shall, during the first phase of the game (10 to 20 minutes) cover specific areas or lanes on the map. Deviating from this expected behaviour will result in less resources (gold and experience) but can sometimes be worth if short (for engaging and terminating an enemy hero).

This assumption was actually illustrated through several examples from *League of Legends* in a New York Times article: we invite the reader to visit the corresponding webpage [6] while an example is reproduced in Figure 2. It gives all recorded positions of any hero during the first phase of $10,000$ games. Another context leads to a different norm as detailed on their article: mid-game, specific hero or hero role, etc.

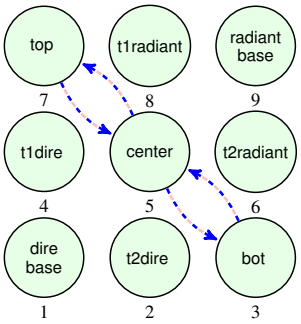
Accordingly, our goal here is to model a chosen behavioural norm in term of mobility. The model can be used then to measure how a player respects the model in a game. Finally, in presence of massive historical data, we can mine player descriptions (patterns) that strongly characterize deviations, that is, find hypotheses or explanations of deviations.

There are thus three key elements in our formalization:

- **Reference model**: We formalize expected movements as a graph where vertices are *points of interest* (POIs) on the map (barracks, towers, lanes, ...) and edges are weighted by their importance. Models can thus differ from their POIs and weights. A very basic example of such graph is given in Figure 3: The map is divided into nine equi-size zones (the POIs) and weights are computed from a set of reference games, e.g. the 20 first minutes of all games of the current season.

pid	Trajectory a	Description	Description	Outlier Score
1	$\langle 1,4,7,5,7,5,7\rangle$	$\{buy_X, buy_Y\}$	$\{ab_{A_1}, ab_{B_2}\}$	0.33
2	$\langle 1,2,3,5,3,5,3\rangle$	$\{buy_X, buy_Y\}$	$\{ab_{A_1}, ab_{B_2}\}$	0.33
3	$\langle 1,5,7,5,7,5\rangle$	$\{buy_X\}$	$\{ab_{A_1}, ab_{B_2}\}$	0.40
4	$\langle 1,2,3,5,3,6,3\rangle$	$\{buy_X, buy_Z\}$	$\{ab_{A_1}, ab_{C_2}\}$	0.66
5	$\langle 1,2,3,5,6,3\rangle$	$\{buy_Z\}$	$\{ab_{A_1}, ab_{C_2}\}$	0.80

Table IV: Contextualized trajectories: descriptions involves items bought (X, Y, Z) and abilities (A, B) taken at level 1 or 2.



poi_id	x,y	label
1	-7500,-7000	dire base
2	-600,-6000	t2dire
3	5000,-6000	bot
4	-6100,-850	t1dire
5	0,0	center
6	6200,-1600	t2adiant
7	-4700,6000	top
8	0,6000	t1radiant
9	7500,7000	radiant base

Figure 3: Simple expert model for DOTA2. Only edges with an important weight are displayed.

- **Player trajectory**: Consider now a set of game traces to analyse. It can be the same set used to compute the model or another. For each, we build a so called *trajectory* that consists in the sequence of visited POIs. This transformation is illustrated from Table I to Table IV. For each generated sequence, we will compute an anomaly score that reflects its deviation from the reference model.
- **Player description**: Each trace to analyse can be described in various ways with contextual information, strategic choices of the player, etc. It is in that description space that we wish to find patterns that explain deviations from the reference model. In Table IV, each trace is augmented with the different items bought by the player and the chosen abilities (build).

The model can be built manually by an expert or from a reference game. It can also be constructed from a set of traces $\mathcal{T}_m$, e.g. all traces of hero *Invoker* of the current season (this is explained later). The traces $\mathcal{T}$ to be analysed, that is, to be compared to the model, can be the same or a different one, e.g. the traces of a given player that wants to improve his *Invoker* game play. To make things simple, we consider a single model in our formalization, but actually there is one model per team (*dire*, *radiant*).

Mining patterns describing deviant behaviours. Once the model is built and the mobility traces generated (sequences of POI), a score is computed for each trace: it reflects the deviation. Then, in the space of all possible descriptions, we look for the patterns that mostly appear deviant traces. In our example, we have a sequence $\langle 1,2,3,5,6,3\rangle$ that strongly deviates from the model: only the edge $(3,5)$ exists in the graph G. The description of this trace contains $\{buy_Z\}$, an interesting pattern: it also appears in trace 4 that also strongly deviates. This is the kind of patterns we are looking for.

B. From rough player traces to contextualized trajectories

A rough game trace of a player is a sequence of records (or events) that describe his actions, states and positions, as well as some global information such as the game length, the winning team, etc. Events have different attributes given their type, along with a time stamp and the coordinates of the player on the MOBA map. Examples are given in Table I.

Definition 3 (Player trace): Let $A = \{A_1, ..., A_n\}$ be a set attributes, where each attribute is either numerical or categorical (that is, takes either numbers or symbols as values). A record $r \in R$ is a n-uplet $r = (a_1, ..., a_n)$ with $a_i \in dom(A_i)$. The sequence $t = \langle r_1, ..., r_k \rangle$ with $r_i \in R$ is a player trace. A collection of player traces is denoted by $\mathcal{T}$. Global attributes are functions $f_i : \mathcal{T} \rightarrow Dom$ where Dom depends of the nature of the attribute, e.g., $outcome : \mathcal{T} \rightarrow \{win, lost\}$

As mentioned before, player traces are considered on two axes: the way the player moves in a set of POIs (called the *trajectory*) and particular actions or properties that describe the player (called the *description*).

Definition 4 (Contextualized trajectory): Let $t \in \mathcal{T}$ be a player trace. The trajectory of t is a sequence of POIs $trajectory(t) = \langle v_1, ..., v_n \rangle$ where $v_i \in V$ and V is a set of POIs. The description of t is a set of items $description(t) \subseteq \mathcal{I}$, that are chosen/computed from the player trace. As such, a pair $(trajectory(t), description(t)), \forall t \in \mathcal{T}$ can be understood as a contextualized trajectory.

Example. Player traces given in Table I have been transformed into contextualized trajectories in Table IV using the POIs introduced in the Figure 3. For sake of readability, descriptions are separated in two columns: one giving the objects bought by the player, the second giving the abilities he chose at a specific experience level.

C. Building a reference behavioural model

The reference behavioural model can be understood as an expert model. Experts models are ideas, hypotheses, a priori knowledge of the context. In the most general case, an expert model is a function that returns a score for a given input to quantify an aspect of the data (utility, outlier score, or a domain specific score). For example, the risk of soil erosion is assessed by a formula built by domain experts [11].

In the case of MOBA games like DOTA2 with a fixed map and specific POIs (corners, towers, bases,etc.), it is intuitive to express the knowledge by a graph: a set of interesting places and the transitions between them. Our objective is to have a function that takes a reference model and a player trace as input and then quantifies how the *mobility assumption* is verified (introduced in the Section IV.A). Each node of the graph is a location (POI) and each edge is a possible transition between one node to another, an example of a simple reference model is given in Figure 3 and defined below.

Definition 5 (Behavioural reference model): A behavioural reference model is a graph $G = (V, E)$ with V a set of nodes (which represents locations) and $E \subseteq V \times V$ a set of edges (which represents possible transitions between the locations).

The reference model can be built manually by a game expert or automatically. In our case, we infer the reference model from a set of player traces $\mathcal{T}_m$ that are chosen w.r.t. the data analysis goals. For example, one may select all the player traces of a specific hero during a season, but also all those from a particular professional player instead. Then, we chose a set of POIs V (9 zones in our simple example). After, for each player trace $t \in \mathcal{T}_m$, we compute the trajectory $trajectory(t)$, i.e., the sequence of transitions between POIs (forbidding self loops). From the set of trajectories $\{trajectory(t), \forall t \in \mathcal{T}_m\}$, we compute the distribution of each direct POI transitions. For example, if we consider the single trajectory $\langle 1, 4, 7, 5, 7, 5, 7 \rangle$, we count the follow transitions: the values $(1, 4) \rightarrow 1$,$(4, 7) \rightarrow 1$,$(7, 5) \rightarrow 2$ and $(5, 7) \rightarrow 2$. Actually, these counts give edges weights of the reference model. We choose a *cut off* threshold to remove edges of the graph with too low importance, e.g. whose weight is below the average of the distribution.

D. Measuring the deviation of a trajectory to the model

Once the model is given, each trace to be analysed is confronted to the model. This is done using a function that takes as input a trace t, the expert model $G = (V, E)$ and quantifies the outlierness of t. For that, we build a matrix M where each cell $M(v_i, v_j)$ is the score of outlierness of a single transition (v_i, v_j). A simple way of defining this score by tanking into account the *mobility assumption* is to assign $M(v_i, v_j) = 1$ if the edge belongs to the model (i.e. $(v_i, v_j) \in E$) and 0 otherwise. There are of course several other ways to build this matrix. Using this matrix, the function to compute the outlier score for a player trace t is denoted by $\mu(t, M) \rightarrow [0, 1]$ returning 0 if the trace fully stick to the model and 1 if totally abnormal.

Definition 6 (Outlier Score For a Player Trace): Given a trace t, and M a weight matrix,

$$\mu(t, M) = \frac{\sum_{i=0}^{i=|trajectory(t)|-1} M(t_i, t_{i+1})}{|trajectory(t)| - 1} \quad (1)$$

$|.|$ counts the number of POIs of the trajectory (its size).

Example. Consider the model $G = (V, E)$ given in Figure 3. If we take the sequence $a_1 = \langle 1, 4, 7, 5, 7, 5, 7 \rangle$, we have $M(7, 5) = 0$ because we have an arc $(7, 5)$ and $M(1, 4) = 1$ because $(1, 4)$ do not exist in the set of edges E. Then, the scores for players traces are: $\mu(\langle 1, 4, 7, 5, 7, 5, 7 \rangle, M) = \frac{1+1+0+0+0+0}{7-1} = 2/6 = 0.33$ and $\mu(\langle 1, 2, 3, 5, 6, 3 \rangle, M) = \frac{1+1+0+1+1}{6-1} = 4/5 = 0.80$; the second sequence is thus more anomalous that the first sequence. At this step, we can compute the outlier score for each trajectory (see Table IV). Note that we consider a model for each team separately (*radiant* and *dire*): a player trace is confronted only to the model of its own team.

E. Describing Deviant Behaviours with Discriminant Patterns

With this outlier scoring measure, we can split the original set of traces $\mathcal{T}$ into two parts, positive and negative examples. We will now consider their descriptions in which our goal is to discover discriminant patterns.

Definition 7 (Positive and Negative Traces): Consider an outlier scoring measure μ, a set of player traces $\mathcal{T}$ and a

threshold $\theta \in [0,1]$ called *outlier threshold*, the positive and negative transaction databases are given by:

$$\mathcal{T}^+ = \{description(t) | t \in \mathcal{T}, \mu(a, M) \leq \theta\}$$
$$\mathcal{T}^- = \{description(t) | t \in \mathcal{T}, \mu(a, M) > \theta\}$$

Mining patterns describing deviant behaviours. Once the positive and negative transaction databases are built, we can mine frequent itemsets as defined in Section III. An itemset will thus be a trace description (a subset of $\mathcal{I}$) that frequently occurs in $\mathcal{T}$ and provided with a measure ϕ that tells us if the description appears in traces that stick or not to the model.

Example. Let us introduce the notation $d(t) = description(t), \forall t \in \mathcal{T}$. If we set $\theta = 0.5$ we have two databases of traces $\mathcal{T}^+ = \{d(t_1), d(t_2), d(t_3)\}$ and $\mathcal{T}^- = \{d(t_4), d(t_5)\}$. Let us set $min\_sup = 2$, we have frequent patterns like $X_1 = \{buy_X\}$,$X_2 = \{buy_Z\}$,$X_3 = \{buy_X, buy_Y\}$ with respectively $supp(X_1) = 4$,$supp(X_2) = 2$,$supp(X_3) = 2$. We compute the normalized growth-rate for each pattern: $\phi(\{buy_X\}) = (3-1)/(3+1) = 0.5$, $\phi(\{buy_Z\}) = (0-2)/(0+2) = -1$,$\phi(\{buy_X, buy_Y\}) = (2-0)/(2+0) = 1$. As we can see, $\{buy_X, buy_Y\}$ have a perfect score of 1 it describe only positive traces. In the other hand, $\{buy_Z\}$ have the reverse perfect score of -1 and describe the negative traces. Other examples are $\phi(\{ab_{A_1}, ab_{B_2}\}) = (2-0)/(2-0) = 1$ and $\phi(\{ab_{A_1}, ab_{B_1}\}) = (0-2)/(0+2) = -1$. We can say that players who deviate from the model bought the object Z and took ability A at level 1 and ability C at level 2. The other players took the ability B at level 2 and bought object Y.

Algorithmic details. For mining frequent patterns, we use an efficient implementation of CHARM [12]. It extracts frequent closed itemsets. An itemset is closed if it has no superset with exactly the same support. It is well-known in data mining that closed itemsets form a lossless condensed representation of frequent itemsets and that they maximize the growth-rate. For each pattern, we also compute a $\mathcal{X}^2$ score (also maximized by closed itemsets) that allows one to measure how the distribution of the support of the itemset in the positive and negative bases is expected or not (introduced in [13]). Actually, in our experiments, we generally only consider the $\mathcal{X}^2$ score as statistically sound (even when we use the term *growth rate*).

V. Experimental study

This section reports an experimental study of the proposed approach. Firstly, we introduce the dataset that we were able to collect. Secondly, we show how different models can be built and to what extent. It follows a quantitative study that proves the feasibility and scalability of the approach. Finally, several scenarios support the discovery of interesting patterns and their actionability. All the experiments were carried out on an Intel Core i7 CPU 2.50 Ghz machine with 16 GB RAM. All code is written in JAVA and we used the CHARM implementation from the SPMF framework [14].

A. Dataset

Any action performed during a game is stored afterwards in a file (replay), allowing to re-watch the game at any time. We were kindly provided with a collection of $9,193$ replays from the dotabank.com website. This is the largest collection of replays we were able to get that contains information. Replay parsing tools are freely available, we used the *Skadistats Clarity 2* parser[3].

We only considered traces from non-anonymous players (known *steam id*), hence we count a bit less than 10 traces per game in average, for a total of $90,366$ traces. We have a total of $77,112$ different players hence skewed distributions of heroes played and $(playerID, hero)$ pairs (given in Figure 5 left and middle in the first row). These distributions shall influence the choice of scenarios. Indeed, the heroes played the most ($\geq 2,000$ times) are *Mirana*, *Phantom Assassin*, *Invoker* and *Pudge*. On the other hand, the most frequent $(playerID, hero)$ pairs are: 135 times (76561197996901530, *Invoker*), 56 times (76561198024940145, *Lycan*), 55 times (76561198024940145, *Furion*), 51 times (76561198028452006, *Techies*). As we can see, we have the more game traces with *Invoker* $2,090$ in total, and 135 traces for one unique player: this is why we choose to focus on these game traces in what follows.

B. Reference models

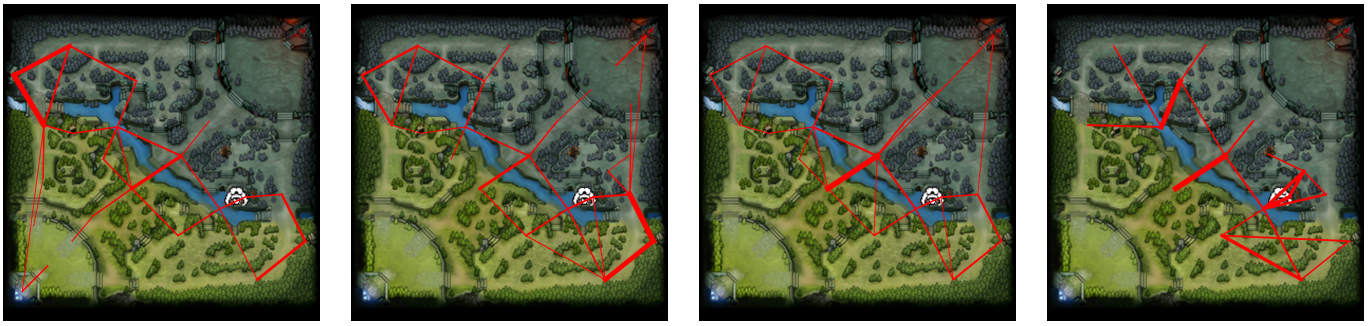
As introduced in Section IV.A, positioning in MOBA is key and determines our reference models. The website [3] presents a study which aggregates $10,000$ games of *League Of Legends*. Three phases (early, middle and late game) give different positions on the map. The study shows also different roles of the heroes: *carry* and *support* go to the bottom lane, *mage* to the center, *tank/melee* at the middle lane, and *jungler* on the whole map. We focus on the early game period as the positioning is crucial at this step.

We constructed several reference models that we present in Figure 4. For each, we consider 33 well known POIs in DOTA2 (bases, shops, center of the map, towers, ...). Models are built for some of the most played heroes in our replay collection. For example, *Invoker* was played $2,089$ times. After computing the POI trajectories, we obtain the graph with edges having a minimum (resp. maximum) weight of 1 (resp. $50,332$) and an average of 731 for dire and 867 for radiant. We use these thresholds to drop non important edges in the graph and present the model for *Invoker* (*radiant* team) in Figure 4 (iv). Note that the size of each edge is proportional to the weight. We follow the same process to present models for *Mirana* 4 (i) and (ii), and Pudge 4 (iii). It can be observed that hero *Mirana* focuses her moves in top and bottom lanes (depending of her team). *Pudge* (in the *dire* team) mainly plays in the mid lane. Finally, *Invoker* (*radiant* team) is a *jungler* thus more present in the top and bottom jungles.

C. Quantitative experiments

Here we give evidence of the computational feasibility of our approach. For this matter, we build the largest transaction database as possible from our initial set of game traces that makes sense for an analysis: (i) The traces $\mathcal{T}_m$ to compute the reference model are those played with the *Invoker hero* (we consider the full game and $|\mathcal{T}_m| = 2,090$), (ii) The traces $\mathcal{T}$ to be analysed are those of the most active player with *Invoker* ($|\mathcal{T}| = 135$), (iii) The set of items $\mathcal{I}$ is composed of objects that can be bought by players in the game (116

[3]https://github.com/skadistats/clarity



(i) (ii) (iii) (iv)

Figure 4: Reference models for early game of: i) *Mirana* (radiant), (ii) *Mirana* (dire), (iii) *Pudge* (dire), (iv) *Invoker* (radiant)

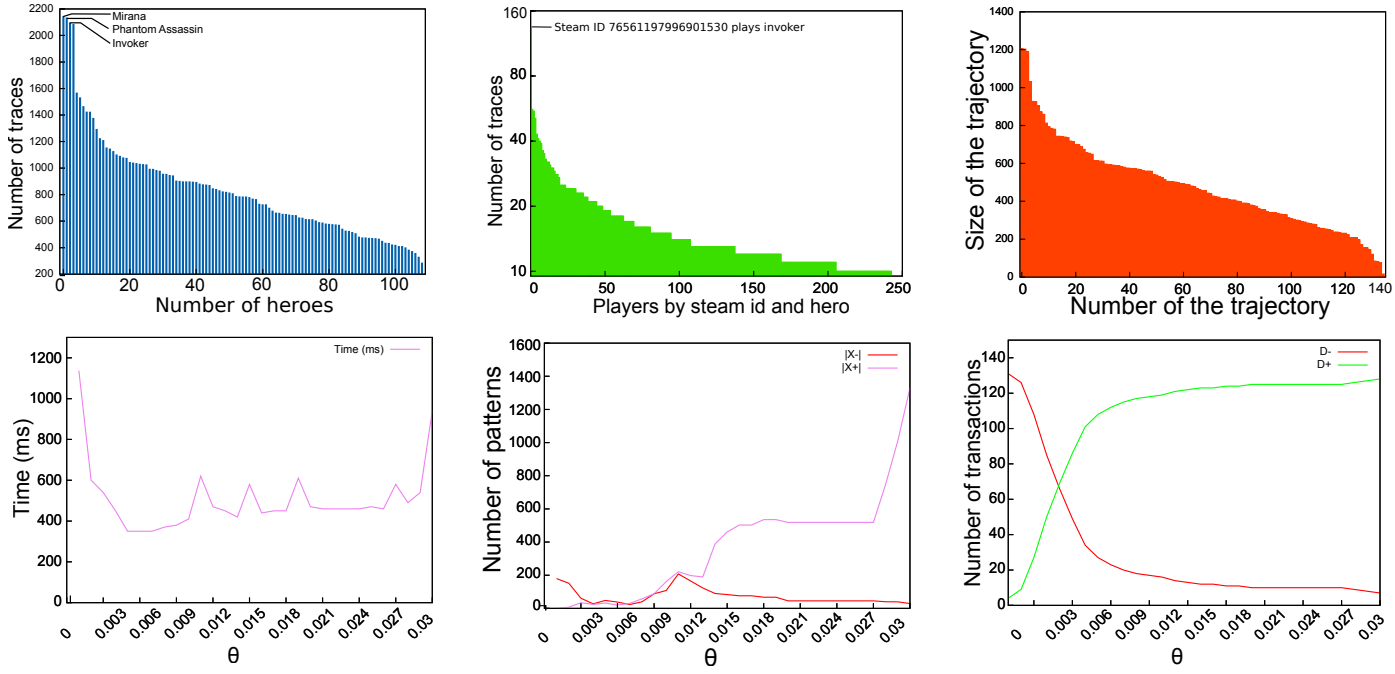


Figure 5: In the first row, several distributions on the DOTA2 dataset (traces by heroes, then by heroes/players, and sizes of behavior part). In the second row, we have several measures in terms of θ: time execution in milliseconds, number of patterns, and number of transactions.

objects) and pairs $(skill, level)$ that tells at which level a skill was chosen (68 possibilities). Finally, traces $\mathcal{T}$ are transformed into contextualized trajectories: the trajectory sizes range from 16 to $1,205$ with an average size of 476 POIs. The distribution is given in Figure 5 (right).

To run the mining algorithm, we need to set several parameters: the minimum frequency σ (set to 1%), the graph weight cut off (set to the average weight) and the θ parameter that allows to split the transaction database into positive $\mathcal{T}^+$ and negative $\mathcal{T}^-$ examples. We report the number of transactions in both positive and negative databases, as well as the run time and number of patterns, for several values of θ. It follows that run times are very low, but the number of patterns can be sometimes too important to be analysed by a Human. Our goal is to get few but accurate negative patterns: We choose $\theta = 0.008$. Negative patterns, denoted by X^- are those with a negative normalized discriminant measure. Figure 10 plots the resulting patterns, using a normalized $\mathcal{X}^2$ measure: Interestingly most of the patterns cover normal behaviours while a small proportion has a measure below -0.5. Note that we only reported pattern mining run times. Parsing each replay and turning into an appropriate formal can take more than one second. This is however done a single time, that is why we did not take it into account when reporting run times.

With the replay collection we possess, our methodology has no computational issues. Editors that own massive replay collections can use parralelized versions of pattern mining algorithms in presence of scaliability issues [15], [16].

D. Qualitative experiments

In DOTA2, each hero has a specific kind of game play. To win, one has to take choices, in a some order, which differs

from a hero to another. Objects to buy give powers: some objects are better for some strategies than others. The same applies when choosing abilities. At each level of experience, a choice has to be made. This choice influence the strategy of the player and efficiency of his hero. So called *build orders* are even shared and discussed on several community websites such as *Dotabuff.com*. Our purpose here is to provide patterns that inform the player about his choices and their capacity to discriminate the outcome of the game (first scenario) and its mobility behaviour (second scenario).

1) Scenario 1 : Patterns that discriminate the game outcome: In this scenario, we label player traces with positive or negative classes depending of $outcome(t)$ function. If $outcome(t) = \{win\}$ then $class(t) = \{+\}$ and $outcome(t) = \{lost\}$ then $class(t) = \{-\}$. In these settings, we seek to answer our first problem: what are the choices (bought objects and skills), that discriminate a victory? a loss?

We consider the games traces of player 76561197996901530 with his hero *Invoker* (the most present in our replay collection). In the 135 player traces, 69 games are lost and 66 are won, which is balanced. Each player trace is described by the objects only bought ($\mathcal{I}$ is the set of all possible objects). This leads to a database of 135 transactions on which we apply the CHARM algorithm with a minimal frequency threshold $\sigma = 1\%$. 390 patterns are extracted in a negligible time. Figure 6 gives for each pattern its frequency in the whole database and growth-rate ϕ. Recall that the closer to -1 is ϕ, the more discriminant for defeat, and the closer to 1 the more discriminant for victory. We can observe patterns with extreme ϕ values (1 and -1), that is, observed only in victorious or lost game. However, their frequency is very low, actually almost never observed. In contrast, the most balanced patterns appear more often (in 60% of the games). Note that this observation was made in a similar scenario studying balance issues in *StarCraft 2* [17]. The table V presents the five patterns that appear only in lost games. *Dotabuff* gives the list of objects purchased during the last year[4]. The patterns have the objects *staff_of_wizardry*,*blade_mail*,*healing_salve* that are not very frequent in the list of purchased objects by *Invoker*. This allows to validate that these five patterns are indeed not common, probably because of their low winning chances.

We conduct a second experiment by taking into account the skills chosen at each level. $\mathcal{I}$ is thus composed of pairs $(hero\_level, ability\_level)$. With the same parameters, we extract 177 itemsets. We select one of the patterns having the lowest growth-rate. It appears in 4 games of the player. We chose to compare it with skills choices statistics from Dotabuff in Figure 7. Each row denotes a spell of *Invoker* while the columns denotes the hero levels. A cell thus counts the number of known games in which a player chose to improve a skill a at given level. The cells in green represent the discovered patterns: it sticks quite well common behaviour in levels 4,7,8,17 and to a less extent 10. As such, it seems that considering skill choices and bought items may be not enough. One may follow well known build orders, but badly move on the map. Moreover, results depends not only of the choices of a single player but also of the team. MOBA games are based on team cooperation and positions are very important on the map. In the next scenario, we use a reference model that capture these moves to improve quality of patterns.

[4]http://www.dotabuff.com/heroes/invoker/items?date=year

#	X	$supp(X)$	$\phi(X)$
1	$\{tpscroll, force\_staff, blade\_mail\}$	3	-1.0
2	$\{tpscroll, staff\_of\_wizardry, blade\_mail\}$	3	-1.0
3	$\{tpscroll, healing\_salve, gloves, power\_treads\}$	3	-1.0
4	$\{boots, tpscroll, healing\_salve, blade\_mail\}$	3	-1.0
5	$\{tango, tpscroll, force\_staff, blade\_mail\}$	2	-1.0

Table V: Pattern discriminating defeat (Scenario 1)

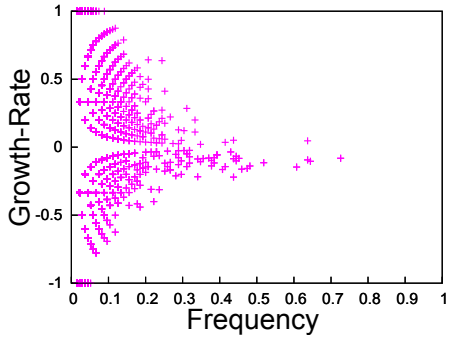


Figure 6: Pattern frequencies and growth-rates (Scenario 1)

2) Scenario 2 : Patterns of traces that deviate/respect a reference model: In this scenario, we label each player trace with the help of the reference model in the early game illustrated in Figures 4. As we are studying games of the hero *Invoker*, we use its model for *dire* (resp. radiant) when labelling a trace played as *dire* (resp. *radiant*). Our goal is to describe sets of players traces that deviate the most regarding to these models. After labelling each trace, we select $\theta = 0.008$ in order to split the positive and negative examples (as explained in Section V.C.). Then, we do the same two experiments as in the first scenario, that is, describing traces with (i) bought objects, (ii) with skill choices. In both cases, we have $|\mathcal{T}^-| = 27$ and $|\mathcal{T}^+| = 108$ while the model was computed on $\mathcal{T}_m = 2,090$ traces of other players.

With these settings, and for the case (i) only 39 patterns have a negative growth rate. The support and growth rate of all patterns is given in Figure 10. The most discriminant for deviant behaviours are given in the figure 8. We can see that more informations about objects are extracted than before. The objects $robe$, $sage\_mask$, $healing\_salve$ and $staff\_of\_wizardry$ are all rare purchased items with *Invoker*. For the case (ii), we extracted 91 patterns that describe deviant behaviours. We show one of them in Figure 9. As we can see, the pattern is very informative for the user. He can see his good choices and the mistakes he makes at specific levels. In this build, he makes mistakes at level 9, by choosing the ability 3 instead of 2, at level 12 with the ability 2 instead of 4, at level 14 and at level 16 where a choice his made by only 4.4% of players. Then, we can see that our method discriminate better the bad and infrequent moves.

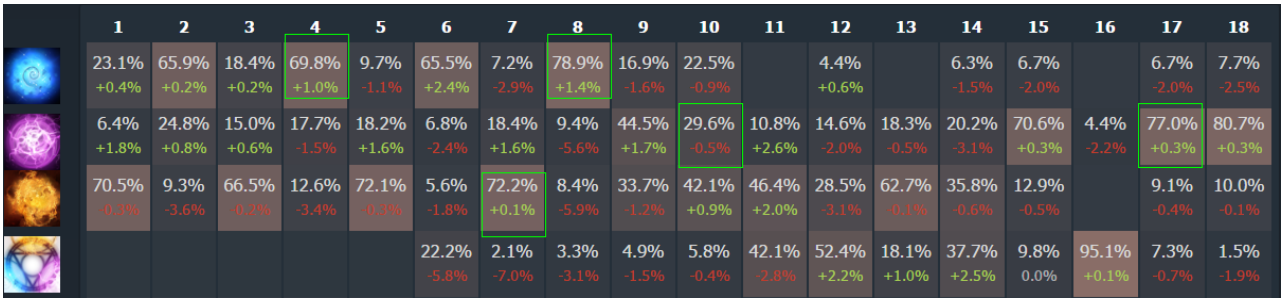


Figure 7: Comparing an itemset of skill choices with Dotabuff statistics (Scenario 1)

#	X	$supp_{\mathcal{D}}(X)$	$\phi(X)$
1	$\{tpscroll, phase\_boots, tango, robe\}$	3	-1.0
2	$\{tpscroll, phase\_boots, tango, sage\_mask\}$	3	-1.0
3	$\{tpscroll, boots, sage\_mask, force\_staff, healing\_salve\}$	5	-1.0
4	$\{tpscroll, phase\_boots, force\_staff, point\_booster, dust, scythe\_of\_vyse\}$	3	-1.0
5	$\{tpscroll, boots, sage\_mask, force\_staff, healing\_salve, staff\_of\_wizardry\}$	3	-1.0

Figure 8: Patterns discriminating deviant trajectories.

VI. Related Work

The analysis of behavioural video game data is not new. It seems to have been for long mainly a test bed for artificial intelligence techniques. For example, the real-time strategy game *StarCraft* has attracted much attention for the design of intelligent agents and even serves as test bed for AI to compete trough tournaments [18]. The problem of building an agent able to beat a Human is so hard that it is now an objective for both Facebook AI research and Google Deep Mind.

With the advent of massively multiplayer online played games, the game industry got interested in analysing these massive sets of historical data by means of visualization, machine learning and data mining techniques. This is one of the many facets of *video game analytics* aiming to enhance user experience and extending game lifetime [19]. There are several relevant tasks that demand massive data. Identifying imbalanced strategies in *StarCraft 2* thanks to pattern mining was recently studied [17]. It is also possible to predict who is playing thanks to keystrokes used by *Starcraft2* players [20] and thus to recognize banned players with a new identity [21]. Without being exhaustive, we can also denote the tasks of detecting unexpected situations and bugs [19] cheaters [22]; improving match making systems [23], [24] designing interactive player advice systems [25]; understanding when to surrender in a MOBA [26], etc. Massive datasets can also be used in eSport analytics, e.g. [27], [28]. Our methodology is useful mainly for the task of understanding player behaviours, and addressed either to a player or to the game editor. Several works considered the problem of player advising (recommendation) for MOBA and RTS games. Silva et al. present an approach to help novice players of *League of Legends* [29]. The system tracks the player actions and gives tips in real-time, e.g., when the hero's health is too low, his positioning is wrong, ... They use domain knowledge from expert players to build a decision tree that is used in real-time. Chunha et al. purpose an advisory system to help players in a RTS game [25]. They formalize expert guides and also use them for helping the player to learn the game more rapidly. In both cases, the systems rely on expert knowledge *a priori* and manually designed, which can be a tedious task. Our approach can use either an existing model or derive it automatically from selected game traces. It is however not designed to help the player in real time, but afterwards, when analysing his games (or those of others).

Anomaly detection is a well-known task in data-mining (as surveyed in [30]). The task consists to detect objects which strongly differ from others w.r.t. an outlier measure. There are a lot of applications, e.g. fraud detection in many domains. Contextualized anomaly detection [31] [32] [33] still considers a set of objects in which outliers are searched for, but also in which context the objects are outliers: a normal object may be an outlier in some subspace of the data. Our approach differs as we are neither looking for outliers or contextualized outliers, but contextual information (or hypotheses) that mostly characterize objects that deviate from a norm.

VII. Conclusion

With the recent developments of eSports, the video game industry faces new challenges for developing games attractive for both professional and amateur players. Gamers are in need of data analytic tools that highlight their strengths and weaknesses and help them during a long learning process. For that matter, discriminant pattern mining techniques provide intelligible explanations for several kinds of targets. We developed in this article a global methodology that enable to output strategic patterns explaining victory and explaining deviations from a reference behaviour that can be customized for various scenarios. Several improvements can be made at each step of the methodology to get more accurate results, although the pillar of the approach would not change. Indeed, we mainly use two thresholds: one as graph cut off, and a second to split the traces into positive and negative examples. We think that it possible to adapt the itemset mining algorithm to directly consider the POIs trajectories when computing the growth-rate. We focused on mining itemsets, but other pattern languages can be considered, e.g. sequential patterns. It remains also to achieve a deeper experimental validation on several heroes, players, description encodings, ... Another issue concerns the lack of data: it is hard to find a sufficient number of replays involving the same players and player/hero pairs. Finally, we plan to provide all the presented material online and a website that allow players to test the methodology.

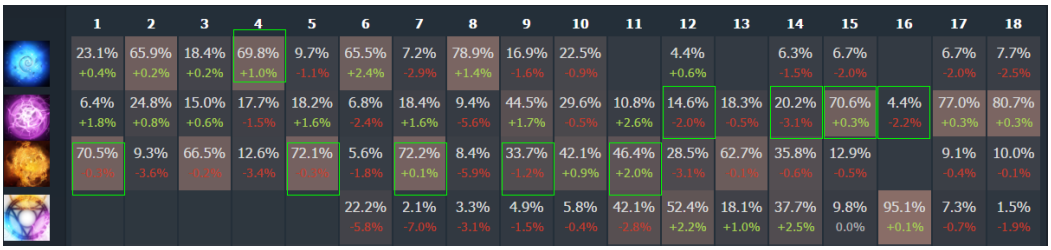


Figure 9: Comparing an itemset of skill choices with Dotabuff statistics (Scenario 2)

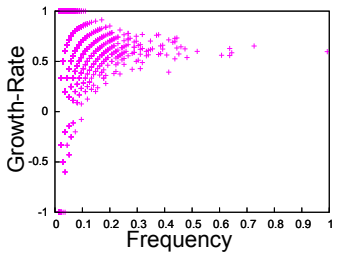


Figure 10: Pattern frequencies and growth-rates (Scenario 2)

REFERENCES

[1] T. L. Taylor, *Raising the Stakes:E-Sports and the Professionalization of Computer Gaming*. MIT Press, 2012.

[2] M. Kaytoue, A. Silva, L. Cerf, W. M. Jr., and C. Raïssi, "Watch me playing, i am a professional: a first study on video game live streaming," in *Proceedings of the 21st World Wide Web Conference, WWW 2012*, 2012, pp. 1181–1188.

[3] "Worlds 2015 viewership," http://www.lolesports.com/en_US/articles/worlds-2015-viewership, accessed: 2016-06-06.

[4] G. Cheung and J. Huang, "Starcraft from the stands: understanding the game spectator," in *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011*, 2011, pp. 763–772.

[5] "Dota2, wikipedia," https://en.wikipedia.org/wiki/Dota_2, accessed: 2016-06-06.

[6] "Watch 10,000 league of legends games in 30 seconds," http://www.nytimes.com/interactive/2014/10/10/technology/league-of-legends-graphic.html?_r=0, accessed: 2016-06-08.

[7] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, 1994, pp. 487–499.

[8] G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences," in *SIGKDD'99*, 1999, pp. 43–52.

[9] J. Boulicaut and B. Jeudy, "Constraint-based data mining," in *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, O. Maimon and L. Rokach, Eds. Springer, 2010, pp. 339–354.

[10] H. Mannila and H. Toivonen, "Levelwise search and borders of theories in knowledge discovery," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 241–258, 1997.

[11] F. Flouvat, J. Sanhes, C. Pasquier, N. Selmaoui, and J.-F. Boulicaut, "Improving pattern discovery relevancy by deriving constraints from expert models," in *ECAI'14*, Aug. 2014, pp. 327–332.

[12] M. Zaki and C.-J. Hsiao, "Efficient algorithms for mining closed itemsets and their lattice structure," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 462–478, April 2005.

[13] T. Guns, "Declarative pattern mining using constraint programming." *Constraints*, vol. 20, no. 4, pp. 492–493, 2015.

[14] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu, and V. S. Tseng, "SPMF: a java open-source pattern mining library," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3389–3393, 2014.

[15] B. Negrevergne, A. Termier, M.-C. Rousset, and J.-F. Mehaut, "ParaMiner: a Generic Pattern Mining Algorithm for Multi-Core Architectures," *Journal of Data Mining and Knowledge Discovery (DMKD)*, 2013.

[16] S. Moens, E. Aksehirli, and B. Goethals, "Frequent itemset mining for big data," in *Big Data, 2013 IEEE International Conference on*, Oct 2013, pp. 111–118.

[17] G. Bosc, P. Tan, J. F. Boulicaut, C. Raissi, and M. Kaytoue, "A pattern mining approach to study strategy balance in rts games," *IEEE Transactions on Computational Intelligence and AI in Games*, 2015.

[18] S. Ontañón, 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 Trans. Comput. Intellig. and AI in Games*, vol. 5, no. 4, pp. 293–311, 2013.

[19] A. Von Eschen, "Machine learning and data mining in call of duty," in *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, ser. LNCS 8724. Springer, 2014, an Industrial Invited Talk.

[20] E. Q. Yan, J. Huang, and G. K. Cheung, "Masters of control: Behavioral patterns of simultaneous unit group manipulation in starcraft 2," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI 2015)*, B. Begole, J. Kim, K. Inkpen, and W. Woo, Eds. ACM, 2015, pp. 3711–3720.

[21] O. Cavadenti, V. Codocedo, J. Boulicaut, and M. Kaytoue, "When cyberathletes conceal their game: Clustering confusion matrices to identify avatar aliases," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA 2015)*, 2015, pp. 1–10.

[22] M. A. Ahmad, B. Keegan, J. Srivastava, D. Williams, and N. S. Contractor, "Mining for gold farmers: Automatic detection of deviant players in mmogs," in *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering, CSE 2009*, 2009, pp. 340–345.

[23] M. Véron, O. Marin, and S. Monnet, "Matchmaking in multi-player on-line games: Studying user traces to improve the user experience," in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, ser. NOSSDAV '14. New York, NY, USA: ACM, 2014, pp. 7:7–7:12.

[24] A. Buchan and J. Taylor, "A qualitative exploration of factors affecting group cohesion and team play in multiplayer online battle arenas (mobas)," *The Computer Games Journal*, pp. 1–25, 2016.

[25] R. L. D. F. Cunha, M. C. Machado, and L. Chaimowicz, "Rtsmate: Towards an advice system for rts games," *Computers in Entertainment*, vol. 12, no. 1, pp. 1:1–1:20, Feb. 2015.

[26] M. Claypool, J. Decelle, G. Hall, and L. O'Donnell, "Surrender at 20? matchmaking in league of legends," in *IEEE Games Entertainment Media Conference (GEM)*, 2015, pp. 1–4.

[27] F. Rioult, J.-P. Metivier, B. Helleu, N. Scelles, and C. Durand, "Mining Tracks of Competitive Video Games," in *AASRI Conference on Sports Engineering and Computer Science (SECS 2014)*, Londres, United Kingdom, 2014.

[28] T. M. Matthias Schubert, Anders Drachen, "Esports analytics through encounter detection," in *Proceedings of the MIT Sloan Sports Analytics Conference 2016*, M. Sloan, Ed., 2016.

[29] V. do Nascimento Silva and L. Chaimowicz, "A tutor agent for moba games," in *SBGames*, 2015.

[30] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[31] F. Angiulli, F. Fassetti, and L. Palopoli, "Detecting outlying properties of exceptional objects," *ACM Trans. Database Syst.*, vol. 34, no. 1, pp. 7:1–7:62, Apr. 2009.

[32] G. Tang, J. Bailey, J. Pei, and G. Dong, "Mining multidimensional contextual outliers from categorical relational data," in *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, ser. SSDBM. New York, NY, USA: ACM, 2013, pp. 43:1–43:4.

[33] L. Duan, G. Tang, J. Pei, J. Bailey, A. Campbell, and C. Tang, "Mining outlying aspects on numeric data," *Data Mining and Knowledge Discovery*, vol. 29, no. 5, pp. 1116–1151, Sep. 2015.