Online Gamers Classification Using K-means

Fernando Palero, Cristian Ramirez-Atencia, and David Camacho

Abstract. In order to achieve flow and increase player retention, it is important that games difficulty matches player skills. Being able to evaluate how people play a game is a crucial component for detecting gamers strategies in video-games. One of the main problems in player strategy detection is whether attributes selected to define strategies correctly detect the actions of the player. In this paper, we will study a Real Time Strategy (RTS) game. In RTS the participants make use of units and structures to secure areas of a map and/or destroy the opponents resources. We will extract real-time information about the players strategies at several gameplays through a Web Platform. After gathering enough information, the model will be evaluated in terms of unsupervised learning (concretely, K-Means). Finally, we will study the similitude between several gameplays where players use different strategies.

Keywords: Player Strategies, Video Games, Sliding Windows, K-Means, Real Time Strategy Game.

1 Introduction

Nowadays a wide number of Computer Science researchers are focused on the develop of intelligence Video Games [1]. Several techniques and methods from areas such as Artificial Intelligence (AI) or Data Mining (DM) have been applied to analyse the gamers behaviours [5], to generate intelligent enemies [13], or to imitate the human behaviour [10], among others. Maybe, one of the most known applications

Fernando Palero · Cristian Ramirez-Atencia · David Camacho
Computer Science Department
Universidad Autónoma de Madrid, Spain
e-mail: {fernando.palero,cristian.ramirez}@inv.uam.es,
david.camacho@uam.es
http://aida.ii.uam.es

D. Camacho et al. (eds.), *Intelligent Distributed Computing VIII,*
Studies in Computational Intelligence 570, DOI: 10.1007/978-3-319-10422-5_22

is related to the development of controllers to automatically define real behaviour of Non-Player Characters (NPC). In this topic, there are several works focused on really famous games such as *Ms. PacMan* [6], or *Starcraft* [12]. Other works have been focused on automatic validation levels by finding the different paths that reach the exit [7].

In the literature we find different applications of gamers strategies detection. One interesting application is to study how the gamers interact with the Super Mario Bros game [9] to automatically generate game levels which will enhance player experience. Other researches [15] have proposed a methodology based on feature selection and preference machine learning for constructing models to increase the player satisfaction. In this paper, we present a case study related to the validation of the attributes that model the players strategies based on an *RTS* game - for this research a Tower Defence game, which is a subgenre of RTS game, has been used. This analysis is based on previous works [8] where 4 player strategies, based on unit position distributions, were detected using visualization techniques.

With the strategies identified, the next step is to study their evolution over time and then employ some metric to evaluate whether the attributes modelling them are well defined or not. The metric adopted in this paper to detect these strategies is the similitude between them. To analyse how the player strategies evolve, it is necessary to use a method based on data stream mining. Data stream mining [4] is the process of analysing ordered sequences of data in real-time. A commonly employed technique is sliding windows technique (Section 3.1). Finally, to study the similitude between strategies, a clustering technique (K-Means) is applied [2].

The rest of the paper is structured as follows: section 2 presents a platform for game data extraction and analysis for the *RTS* Game considered. Section 3 describes the methodology used in the data analysis. Section 4 presents some experimental results. Finally, Section 5 shows the conclusions of this research and future lines of work.

2 Web Platform Architecture

This section presents a platform architecture based on a *RTS* game (see Figure 1) that has been developed to study gamers strategies. The platform has been designed using four different modules. The Adaptive Horde Module (*AHM*) is the responsible for generating a fix number of variable hordes of enemies in each wave. The Collector Module (*CM*) allows to automatically extract data from the game, and to gather the interaction from the users. The Strategy Detection Module (*SDM*) analyses the state of the gameplay at the beginning of a new horde and returns a suitable counter-strategy. Finally, the Attribute Validation Module (*AVM*) analyses and returns the distributions of the gameplays. With these distributions, visualization techniques (histograms) have been used to determine the strategies.

AHM receives from *SDM* the parameters necessary to generate the enemies in the different hordes that would appear at each wave. Equation 1 applies the received

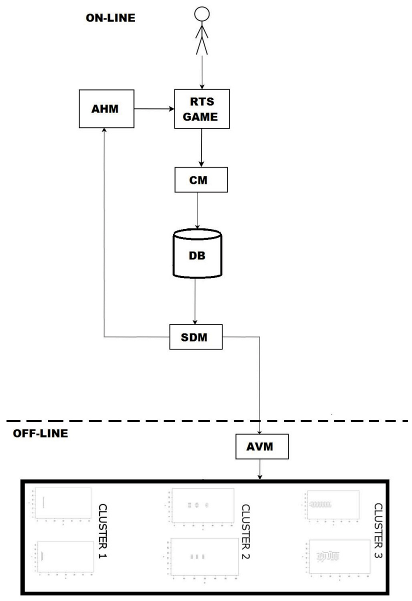

Fig. 1 Framework Architecture based on the RTS game platform

parameters to generate the quantity of different enemies types in the hordes within a gameplay. N represents the number of enemies, β_T the percentage of enemies of type T, W is the current wave and α is a growing factor for the enemies generation. For $\alpha = 0$, the number of enemies in the horde is constant in each wave. If $\alpha = 1$, the amount of enemies grows linearly and, finally, if $\alpha = 2$ the horde has a quadratic growth factor.

$$\#Enemies_T = \beta_T N W^{\alpha} \tag{1}$$

CM extracts data from the RTS game, which in this approach is related to the Unit Data (*UD*). *UD* provides information about the units (in this case towers) based on their features (i.e. position, the unit type, its strength, etc). All this gathered information is then passed to a database module to be stored, and will be recovered in other modules to be analysed.

SDM is responsible for carrying out the analysis of the data gathered by *CM*. It works based on two basic processes: the first process recovers the units information from a data window stored in the database, and the second one calculates both the distributions of X, Y and the euclidean distances from the units to the entry point.

Finally, *AVM*, which works off-line, is responsible for carrying out the validation of the attributes. In a first phase, the distributions that have been calculated in *SDM* are normalized and labelled, and the attributes that help to identify the strategy are obtained from the unsupervised method K-means (see section 3.2). These labels will be used to identify the cluster where the instances have been assigned. Then, in a second phase, the similitude between gameplays strategies is studied (see section 3.3).

3 Description of the Data Analysis Procedure

Three main techniques have been used to achieve the players strategies analysed. The first one is based on a sliding-window technique that is used to gather data and create instances of features. The second one is based on K-means clustering to group distributions by labels. The last technique studies the similitude between the distributions. This section describes these methods.

3.1 Sliding-Window Technique

The most popular approach to deal with data stream involves the use of sliding windows [14]. Sliding-Windows provides a way to divide the data stream into a set of examples to analyse. The procedure for using sliding windows for data stream mining is shown in Algorithm 3. The input of the algorithm is the samples set from the RTS Game. One sample corresponds to one window, and the size of the window is dynamic and changes according to the life time of the wave. The life of a wave is defined as the time between the apparition of its first horde and the disappearance of its last horde. With the size of the windows defined, in each iteration of the algorithm a new window is analysed and the distribution of coordinates X, the distribution of coordinates Y and the distribution of euclidean distances from the units to the exit are returned.

Algorithm 3. This algorithm is an adaptation of [4]

```
  Parameter: $\mathcal{S}$: a data stream of example $\mathcal{W}$: window of examples
  Result: $\mathcal{C}$: the distribution of the coordinate $X$, the distribution of the coordinate $Y$ and
          the distribution of the euclidean distance from the units to the exit from the
          window $\mathcal{W}$
1 Initialize window $\mathcal{W}$
2 forall the example $x_i \in \mathcal{S}$ do
3     $\mathcal{W} \leftarrow \mathcal{W} \cup \{x_i\}$
4     build $\mathcal{C}$ using $\mathcal{W}$
5 end
```

3.2 K-Means Clustering

K-means algorithm is used to partition the input data set into k partitions. However, K-Means algorithm has two problems. The first one, in contrast to other algorithms, is that K-Means cannot be used with arbitrary distance functions or on non-numerical data. And the second one, K-Means algorithm cannot guarantee finding the best space partition. To solve the first problem we use the euclidean distance and transform all dataset to numerical data. For the second, we execute the algorithm using always the same *'k'* several times (see Algorithm 4) and then we choose the best result returned. For this selection, we use two metrics: intra-cluster distance and inter-cluster distance. Intra-cluster [11] distance measures (equation 2) the average of the distances between the points and its respective cluster centroids. In the equation 2 we can see the intra-cluster metre, where N is the number of instances of data extracted from the game, K is the number of clusters, and z_i is the centroid of cluster C_i.

$$intra(x, z_i) = \frac{1}{N} \sum_{i=1}^{K} \sum_{x \in C_i} |0x - z_i|0^2 \tag{2}$$

$$inter(z_i, z_j) = min|0z_i - z_j|0^2; i = 1, 2, ..., K - 1; j = i + 1, ..., K; \tag{3}$$

Inter-cluster distance (equation 3) measures the distance between cluster centres. To choose the result that makes a good partition of the data space, it is necessary to minimize the intra-cluster distance and maximize the inter-cluster [3] distance measure. The aim is to minimize the validity measure (equation 4).

$$validity(x, z_i, z_j) = \frac{intra(x, z_i)}{inter(z_i, z_j)} \tag{4}$$

Algorithm 4. Algorithm to choose the best K-Means partitioning

```
  Parameter: $\mathcal{W}$: window of examples
  Result: $\mathcal{C}$: data labelled in the window $\mathcal{W}$
1 Initialize window $\mathcal{W}$
2 $k$=4
3 $vecValidity \leftarrow []$
4 for $j = 1$ to 10 do
5     $labels \leftarrow KMeans(k, \mathcal{W})$
6     $validity \leftarrow CalculateValidity(labels, \mathcal{W})$
7     $vecValidity(i) \leftarrow validity$
8 end
9 $vecK(k) \leftarrow min(vecValidity)$
```

3.3 Distribution Similitude

The different $\mathcal{W}$ that compose the strategies are labelled by groups with K-Means to study the similitude between distributions. Equation 5 represents the similitude between two distributions D_1 and D_2. In this equation, w_i represents the wave number, and $D_1(w_i)$ and $D_2(w_i)$ indicate the group of the distribution in w_i. The similitude is calculated dividing the number of the label coincidences from the distributions ($D_1(w_i)$ and $D_2(w_i)$) between the number of waves.

$$Similitude(D_1, D_2) = \frac{\#\{i \epsilon \{1 \dots \#Waves\} | D_1(w_i) = D_2(w_i)\}}{\#Waves} \quad (5)$$

4 Experimental Results

We have studied the similitude between the strategies detected in the previous work [8] (*Zigzag*, *Horizontal*, *Grouped* and *Vertical* distributions). In this new approach, we are interested in calculating the similitude between strategies. For this purpose, we have previously analysed and labelled the gameplays dataset according to the strategies identified. With this experiment we determine if the features selected to distinguish the strategies are adequate.

4.1 Distributions Similitude Comparison

Sliding-window technique has been used to extract the ten wave distribution of a gameplay. The distributions are grouped by labels that are assigned by K-Means algorithm. We choose a $k = 4$ for K-Means that corresponds to the number of strategies found in the previous work. Finally, we use these labelled groups to study the similitude between strategies distributions, using equation 5 for this purpose. With these groups we have obtained a table of similitude (table 1).

Table 1 Similitude between distributions, where Z is the Zigzag distribution, G is the Grouped distribution, H is the Horizontal distribution and V is the Vertical distribution.

Game ID		1	2	3	4	5	6	7	8	9	10	11	12
	Distribution	Z	Z	Z	G	G	G	H	H	H	V	V	V
1	Z	1	**0,6**	**0,1**	0,1	0	0,5	0,6	**0,5**	0,5	0,5	**0,5**	0,5
2	Z	0,6	1	0	0,2	0	0,4	0,4	0,4	0,4	0,4	0,4	0,4
3	Z	0,1	0	1	0,1	0,1	0,1	0	0	0,1	0	0	0,1
4	G	0,1	**0,2**	0,1	1	**0,7**	0,3	**0,2**	0,2	0,3	0,1	0,1	**0,3**
5	G	0	0	0,1	0,7	1	0,1	0	0	0,1	0	0	0,1
6	G	0,5	0,4	0,1	0,3	0,1	1	0,9	0,9	1	0,8	0,8	1
7	H	0,6	0,4	0	0,2	0	**0,9**	**1**	**0,9**	**0,9**	**0,8**	**0,8**	**0,9**
8	H	0,5	0,4	0	0,2	0	0,9	0,9	1	0,9	0,8	0,8	0,9
9	H	0,5	0,4	0,1	0,3	0,1	1	0,9	0,9	1	0,8	0,8	1
10	V	0,5	0,4	0	0,1	0	0,8	0,8	0,8	0,8	1	1	0,8
11	V	0,5	0,4	0	0,1	0	0,8	0,8	0,8	0,8	1	1	0,8
12	V	0,5	0,4	0,1	0,3	0,1	1	0,9	0,9	1	0,8	0,8	1

In table 1, it is appreciated that similitude between different gampeplays using Zigzag distribution is low. The same happens with Grouped distributions. This means that these kind of strategies are difficult to identify. Moreover, different distributions could be included inside this strategy, so it is necessary to take more samples. On the other hand, the similitude between gameplays using Horizontal or Vertical distributions is high. This implies that these strategies are well identified by the employed attributes (units positions distributions).

Looking at the similitude between different strategies, we observe that Zigzag distributions can be confused with other distributions, as the similitude values between two gameplays using Zigzag distributions are similar to those using a Zigzag distribution and other type of distribution. This confirms the aforementioned inclusion of strategies inside Zigzag distributions. For Grouped distributions, the same problem appears.

Finally, comparing the similitude of gameplays using Horizontal and Vertical distributions, we can appreciate that these similitude values are high in all cases.This happens because the metric that we use to generate the features does not consider the orientation of the distribution. We can observe that Vertical distribution is the inverse of Horizontal distribution. In Vertical distribution, we can see that the unit distribution along the axis X is one bin. However, in Horizontal distribution, we have this behaviour in the Y axis.

In conclusion, we find that the attributes based on the units positions distributions is not very effective to identify player strategies. We only can distinguish between linear (*Horizontal* or *Vertical*) distribution or not linear. It is necessary to use more attributes that help to improve this model.

5 Conclusions and Future Work

This work provides an evaluation of the attributes used to identify strategies in gameplays. To achieve this purpose, a framework based on an RTS platform has been designed, and the strategies detected in a previous work based on the units positions distributions have been employed. The experiment carried out tries to determine if the attributes selected to identify the players strategies are sufficiently descriptive. For this purpose, we have used K-means to group strategies. Latter these groups have been used to calculate the similitude between gameplays.

From the similitude study, we have concluded that the attributes employed to detect players strategies have low performance. We have found that the actual attributes identifies linear distributions (*Horizontal* or *Vertical*), but they are not good at discriminating not linear distributions (*Grouped* or *Zigzag*), due to it is necessary to use more attributes.

In future works, it will be necessary to use more features to perform a better classification and study more unsupervised techniques, such as spectral clustering, to determine which do a best data partitioning. Moreover, we could apply Online Learning, so the attributes are updated every time new data is gathered.

References

1. Alayed, H., Frangoudes, F., Neuman, C.: Behavioral-based cheating detection in online first person shooters using machine learning techniques. In: 2013 IEEE Conference on Computational Intelligence in Games (CIG), pp. 1–8. IEEE (2013)
2. Alsabti, K., Ranka, S., Singh, V.: An efficient k-means clustering algorithm. Association for the Advancement of Artificial Intelligence (1997)
3. Bello-Orgaz, G., Menendez, H., Camacho, D.: Adaptive k-means algorithm for overlapped graph clustering. International Journal of Neural Systems 22(05), 1–19 (2012)
4. Brzezinski, D.: Mining Data Streams with Concept Drift. Master's thesis, Poznan University of Technology (2010)
5. Dey, R., Child, C.: QL-BT: Enhancing behaviour tree design and implementation with Q-learning. In: 2013 IEEE Conference on Computational Intelligence in Games (CIG), pp. 1–8. IEEE (2013)
6. Gagne, D.J., Congdon, C.B.: Fright: A flexible rule-based intelligent ghost team for Ms. Pac-Man. In: 2012 IEEE Conference on Computational Intelligence and Games (CIG), pp. 273–280. IEEE (2012)
7. Gonzalez-Pardo, A., Palero, F., Camacho, D.: An empirical study on collective intelligence algorithms for vide games problem-solving. Computing and Informatics (In press, 2014)
8. Palero, F., Gonzalez-Pardo, A., Camacho, D.: Simple Gamer Interaction Analysis through Tower Defence Games (submited, 2014)
9. Pedersen, C., Togelius, J., Yannakakis, G.N.: Modeling player experience in Super Mario Bros. In: IEEE Symposium on Computational Intelligence and Games, CIG 2009, pp. 132–139. IEEE (2009)
10. Polceanu, M.: MirrorBot: Using human-inspired mirroring behavior to pass a Turing test. In: 2013 IEEE Conference on Computational Intelligence in Games (CIG), pp. 1–8. IEEE (2013)
11. Ray, S., Turi, R.H.: Determination of number of clusters in k-means clustering and application in colour image segmentation. In: Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques, pp. 137–143 (1999)
12. Synnaeve, G., Bessiere, P.: A Bayesian model for RTS units control applied to StarCraft. In: 2011 IEEE Conference on Computational Intelligence and Games (CIG), pp. 190–196. IEEE (2011)
13. Traish, J.M., Tulip, J.R.: Towards adaptive online RTS AI with NEAT. In: 2012 IEEE Conference on Computational Intelligence and Games (CIG), pp. 430–437. IEEE (2012)
14. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 226–235. ACM, New York (2003)
15. Yannakakis, G.N., Hallam, J.: Feature selection for capturing the experience of fun. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment, vol. 7, pp. 37–42 (2007)