# Evaluación de Modelos de Jugadores mediante Técnicas de Clustering

Fernando Palero, Antonio Gonzalez-Pardo, David Camacho

Departamento de Informática
Universidad Autónoma de Madrid, España.
fernando.palero@inv.uam.es, {antonio.gonzalez,david.camacho}@uam.es
http://aida.ii.uam.es

**Abstract.** Para conseguir captar el interés de los jugadores is importante que los vídeo juegos adapten su dificultad en función de las habilidades del jugador. Ser capaz de evaluar como juegan los usuarios es un componente crucial para modelar el comportamiento de los jugadores en los juegos. Uno de los problemas de crear un modelo de comportamiento de usuario es comprobar si las variable predictoras detectan correctamente las acciones del jugador. En este artículo, estudiaremos el juego de código abierto *Tower Defense* (*Open-source Tower Defence OTD*), basado en un tablero 2D en el que aparecen conjunto de hordas enemigas que deben ser destruidas para evitar que alcancen la salida. Utilizando dicho juego extraeremos en tiempo real la información de las interacciones usuario y los eventos del juego utilizando la técnica de ventana deslizante. Una vez obtenida suficiente información, el modelo se evaluará mediante técnicas de *clustering* (concretamente, *K-Means* y *Spectral Cluster*). Finalmente, estudiaremos la similitud entre las diferentes partidas donde los jugadores han utilizado diferentes estrategias.

**Keywords:** Comportamiento del Jugador, Video Juegos, Ventana Deslizante, K-Means, Tower Defence Game, Spectral Clustering

## 1 Introducción

Hoy en día un gran número de investigaciones, en el ámbito de la informática, están centradas en el desarrollo de inteligencia en los Video Juegos. Diversas técnicas y métodos de diferentes áreas como la Inteligencias Artificial (IA) or la Minería de Datos (MD) han sido aplicadas para analizar los comportamientos de los jugadores [3], para generar inteligencia en los enemigos y/o imitar el comportamiento humano [9], o para validar los niveles de los juegos [5], entre otras cosas. Una de las aplicaciones más conocidas esta relacionada con con el desarrollo de controladores que definan automáticamente el comportamiento real de personajes del juego que no sean jugadores (*Non-Player Character NPC*). Relacionado con este tema, existen muchos trabajos que utilizan juegos famosos para su estudio, tales como, *Ms. Pacman* [4], or *StartCraft* [11].
En la literatura encontramos diferentes aplicaciones que modelan el comportamiento del usuario. Una interesante aplicación es estudiar la interacción de los usuarios con el juego de *Super Mario Bros* [8] para generar automáticamente niveles que ayuden a mejorar la experiencia de usuario. Otras investigaciones [13] proponen metodologías basadas en la selección de atributos y técnicas de

aprendizaje automático para construir modelos que ayuden a incrementar la satisfacción del usuario. En este trabajo, proponemos un caso de estudio relacionando con la validación de modelos de comportamiento de usuarios utilizando el juego de código abierto *Tower Defence*[1] (*OTD*). Este análisis esta basado en un trabajo previo [7] donde 4 estrategias, definidas por las posiciones de las torres, se detectaron utilizando técnicas de visualización. Se asume que cada estrategia corresponde a un comportamiento de usuario.
Con los comportamientos identificados, el siguiente paso es estudiar su evolución a lo largo del tiempo y entonces emplear alguna métrica que nos permita discernir si el modelo de jugador esta bien definido. La métrica utilizada para este fin es la similitud entre estrategias. Para analizar la evolución de estas se ha utilizado un método basado en minería de flujo de datos (*Data Stream Mining*). La minería de flujo de datos [2] es el proceso de analizar secuencias ordenadas de datos en tiempo real. Una técnica comúnmente empleada es la *ventana deslizante* (ver sección 3.1). Finalmente, se estudiará la similitud entre estrategias mediante técnicas de *clustering*, *K-Means* [1] y *Spectral Clustering* [6], y se estudiará cual de las técnica permite hacer un mejor análisis del modelo de comportamiento del jugador.
El resto del artículo se estructura del siguiente modo: la sección 2 presenta una plataforma para la extracción y el análisis del juego *Tower Defence*. La sección 3 describe la metodología utilizada para el análisis de los datos. La sección 4 presenta los resultados experimentales. Finalmente, la sección 5 muestra las conclusiones del trabajo realizado y futuras lineas de trabajo.

## 2 *Framework* basado en *Tower Defence*

Está sección presenta la arquitectura del *framework* basado en *OTD* (ver Figura 1), el cual, ha sido desarrollado para estudiar el comportamiento de los jugadores. El framework está dividido en dos grandes partes, la primera parte está formada por los módulos que se ejecutan cuando el jugador inicia una partida, se ejecutan en tiempo real. Esta primera parte está compuesto por 4 módulos diferentes. El Módulo Generador de Oleadas (*Wave Generator Module WGM*), se encarga de generar un número fijo de hordas compuestas por un número variable de enemigos en cada oleada. El Módulo de Recuperación de Datos (*Recovery Data Modul DRM*) permite automáticamente extraer datos e interacciones del usuario del juego. Finalmente, el Módulo de Inteligencia Computacional (*Computational Intelligence Module CIM*) analiza y devuelve las distribuciones de las partidas. Aplicando técnicas de visualización (histogramas) en las distribuciones se determinan las estrategias. La segunda parte del *framework* se ejecuta cuando no hay usuarios jugando y se compone por el Módulo de Validación del Modelo (*Model Validation Model MVM*). *MVM* analiza las partidas de los jugadores y devuelve los atributos que determinan el comportamiento de los usuarios.
*WGM* determina el tamao de la horda en las oleadas. La ecuación 1 se utiliza para definir el número de enemigos de cada tipo que se generarán en la partida. Donde $N$ representa el número de enemigos, $\beta_T$ es el porcentaje de enemigos de cada tipo $T$, $W$ representa en número de la oleada actual y $\alpha$ es el factor de crecimiento para la generación de enemigos. Para $\alpha = 0$, el número de enemigos en la horda

[1] Disponible en: http://sourceforge.net/projects/otd/

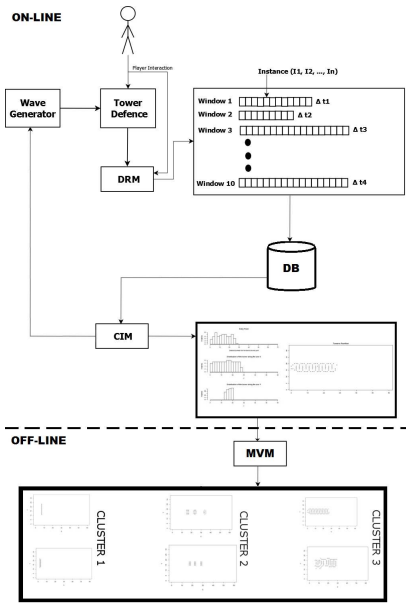


**Fig. 1.** Arquitectura del Framework basado en OTD

es constante en cada oleada. Si $\alpha = 1$, la cantidad de enemigos crece linealmente y, finalmente, si $\alpha = 2$ el crecimiento de la horda tiene un factor de crecimiento cuadrático.

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

*DRM* extrae datos del juego *Towe Defence*, los cuales se clasifican en dos categorias: Datos del Juego (*Game Data GD*) and Datos de Interacciones (*Interplay Data ID*). *GD* proporciona información del entorno, está información está predefinida por las características preestablecidas del juego (por ejemplo: el número de oleadas, el tipo de los enemigos, el tamao de la horda, etc). Por otra parte, existen dos tipo de *ID*: Interacciones del jugador (*Player Interaction IP*) e instantáneas del juego (*Game Snapshot GS*). PI representa las acciones del jugadores y los eventos del juego durante su ejecución. Mientras que GS proporciona información detallada sobre el estado del juego en cada instante de tiempo. Ambos tipos de datos (*PI* y *GS*) se obtienen de la partida periódicamente cada segundo. La información recogida es almacenada en la base de datos y será recuperada posteriormente en otros módulo para ser analizada.

*CIM* es el responsable de llevar a cabo el análisis de los datos recogidos por *DRM*. El módulo realiza 2 procesos básicamente: el primer proceso recupera de la base de datos la información de las ventanas de tiempo almacenadas, las cuales, contienen información sobre las torres. El segundo proceso calcula la distribución de las coordenadas X, Y y la distancia euclidea de las torres al punto de entrada.

Finalmente, *MVM* es responsable de llevar a cabo la validación del modelo de comportamiento del jugador. En la primera fase, las distribuciones calculadas en *CIM* son normalizadas y etiquetadas, y los atributos que permiten discriminar las estrategias se obtienen de los métodos no supervisados (*K-Means* y *Spectral*

*Clustering*). Esta etiqueta es usada para identificar el *cluster* donde las instancias han sido asignadas. Entonces, en una segunda fase, se estudia la similitud entre partidas (ver sección 3.3).

# 3 Descripción del procedimiento de análisis de los datos

Principalmente se utilizan 3 técnica para analizar las estrategias de los jugadores. En la primera se utiliza la técnica de la ventana deslizante para dividir el flujo de datos en un conjunto de muestras que puedan ser analizadas. La segunda consiste en utilizar el *clustering* para agrupar las distribuciones mediante etiquetas. La última técnica estudia la similitud entre distribuciones. A continuación en esta sección describiremos estos métodos.

## 3.1 Técnica de la Ventana Deslizante

El método más popular para trabajar con flujos de datos es la ventana deslizante [12]. Esta metodología permite dividir el flujo de datos en muestras que se pueden analizar. El procedimiento empleado para obtener las muestras se muestra en el algoritmo 1. La entrada del algoritmo es el conjunto de muestras extraídas del juego *Tower Defence*. Una muestra corresponde a una ventana, el tamao de la ventana es dinámico y cambia de acuerdo al tiempo al duración de una oleada. La duración de una oleada viene determinada por la aparición de la primera horda y la desaparición de la última. Con el tamao de la ventana definido, en cada iteración del algoritmo una nueva ventana es analizada. De este análisis se obtiene las distribuciones de las coordenadas $X$, $Y$ y la distribución de las distancias euclideas de las torres al punto de entrada.

**Algorithm 1:** Este algoritmo es una adaptación de [2]

**Parameter**: $\mathcal{S}$: el flujo de datos $\mathcal{W}$: muestras extraídas en cada ventana
**Result**: $C$: la distribución de la coordenada $X$, la distribución de la coordenada $Y$ y al distribución de la distancia euclidea de las torres al punto de entrada de cada $\mathcal{W}$

```
1 Inicializamos la ventana W
2 forall the muestra x_i ∈ S do
3 |   W ← W ∪ {x_i}
4 |   Construir C usuando W
5 end
```

## 3.2 Algoritmos de Clustering

En este trabajo se han utilizado 2 algoritmos de *clustering*: *Spectral Clustering* [6] y *K-Means*.*Spectral Clustering* hace referencia a una clase de técnicas en las que se utiliza los auto-valores de la matriz de similitud, generada a partir de los datos que se quieren dividir, para particionar datos en grupos disjuntos. Donde los datos de un mismo grupo tiene una gran similitud y los datos fuera de este tiene baja similitud. En el caso de K-Means nos encontramos con un algoritmo no determinista, es decir, que en cada ejecución obtenemos un resultado diferente. Por

lo que es necesario utilizar métricas que nos permitan determinar cual de las ejecuciones ofrece el mejor resultado. Para ello se ha desarrollado el algoritmo 2. El algoritmo ejecuta diversas veces *K-Means* utilizando siempre la misma *'k'*, para cada resultado devuelto se calcula su *validez* (ver ecuación 4). En la *validez* se calculan las distancias intra-cluster (ver ecuación 2) e *inter-cluster* (ver ecuación 3).

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

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

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

La distancia *intra-cluster* [10] calcula la suma de las distancias medias de todos los puntos a sus respectivos *centroides*. En la fórmula de la métrica tenemos $N$ que es el número de instancias de datos extraídas del juego, $K$ es el número de *clusters* y $z_i$ es el *centroide* del *cluster* $C_i$. En el caso de la distancia *inter-cluster* se mide la distancia entre los centros de los diferentes *cluster* y se escoge la más pequea. Para elegir el resultado que mejor particiona el espacio de datos es necesario minimizar la distancia *intra-cluster* y maximizar la distancia *inter-cluster*. El objetivo es minimizar la medida de *validez*.

**Algorithm 2:** Algoritmo para elegir la mejor partición para K-Means

**Parameter**: $\mathcal{W}$: ventana de muestras
**Result**: $C$: muestras de la ventana etiquetadas $\mathcal{W}$

```
1 Inicializamos la ventana W
2 k=4
3 vecValidity ← []
4 for j = 1 to 10 do
5 |   labels ← KMeans(k,W)
6 |   validity ← CalculateValidity(labels,W)
7 |   vecValidity(i) ← validity
8 end
9 vecK(k) ← min(vecValidity)
```

### 3.3 Similitud

Las diferentes $\mathcal{W}$ que componen las estrategias son etiquetadas por grupos con los algoritmos de *clustering* para estudiar la similitud entre dos estrategias (ver ecuación 5) $D_1$ y $D_2$. En esta ecuación $w_i$ representa el número de oleada y $D_1(w_i)$ y $D_2(w_i)$ indican el grupo al que pertenece la estrategia en $w_i$. La similitud es calculada dividiendo el número de las etiquetas en las que coinciden ambas estrategias ($D_1(w_i)$ y $D_2(w_i)$) en las diferente oleadas por el número de oleadas.

$$Similitud(D_1, D_2) = \frac{\#\{i \epsilon \{1 \ldots \#Waves\} | D_1(w_i) = D_2(w_i)\}}{\#Waves} \tag{5}$$

## 4 Experimentación

En este trabajo hemos estudiado la similitud entre las estrategias detectadas en el trabajo previo [7] (*Zigzag*, *Horizontal*, *Agrupada* y *Vertical*). Por este motivo, previamente se han analizado y etiquetado las diferentes partidas de acuerdo a las estrategias identificadas. Con este experimento se quiere determinar si las características utilizadas para distinguir las estrategias son adecuadas y además comprobar que algoritmo de *clustering* nos permite agrupar mejor las partidas por estrategias.

### 4.1 Comparación de similitud

La técnica de ventana deslizante ha sido utilizada para extraer 10 distribuciones de una partida. Las distribuciones han sido agrupadas por etiquetas asignadas por los diferentes algoritmos de *clustering*, *K-Means* y *Spectral Clustering*. Para ambos algoritmos se ha utilizado un valor de $k = 4$ que corresponde con el número de estrategias encontradas en el trabajo previo. Finalmente, hemos utilizado los grupos etiquetados para estudiar la similitud entre las partidas, utilizando la ecuación 5 para este propósito. Con estos grupos hemos obtenido 2 tablas similitud. La correspondiente a *K-Means* y a *Spectral Clustering*

| ID Juego | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Distribución | 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 |

**Table 1.** Similitud entre partidas usando K-Means, donde Z es indica que se ha utilizado estrategia en Zigzag , G corresponde a la Agrupada, H corresponde a la Horizontal y V corresponde a la Vertical.

La tabla 1 nos muestra la similitud que se obtiene al analizar los datos usando *K-Means*. Como se puede observar en la tabla la similitud entre partidas que utilizan la estrategia Zigzag es baja, lo mismo sucede con la estrategia Agrupada. Por otro lado, las estrategias Horizontales y Verticales si presentan una alta similitud entre partidas las han utilizado.

Por otro lado, si comparamos la similitud entre partidas donde se han aplicado diferentes estrategias podemos observar 2 cosas. La primera, las estrategias Lineales (Vertical y Horizontal) tienen similitudes muy altas. Esto indica que se

están identificando como una sola. Y la segunda, que las estrategias No Lineales (Zigzag y Agrupadas) no se discriminan correctamente, esto se debe a que este tipo de estrategias se pueden reproducir utilizando diferentes distribuciones de torres.

| ID Juego | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Distribución | Z | Z | Z | G | G | G | H | H | H | V | V | V |
| 1 | Z | 1,0 | 0,5 | 1,0 | **0,4** | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| 2 | Z | 0,5 | 1,0 | 0,5 | **0,5** | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| 3 | Z | 1,0 | 0,5 | 1,0 | **0,4** | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| 4 | G | **0,4** | **0,5** | **0,4** | 1,0 | 0,0 | 0,0 | 0,0 | 0,0 | **0,5** | 0,0 | 0,0 | **0,5** |
| 5 | G | 0,0 | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | 0,0 | 0,0 | 0,0 | **1,0** | **1,0** | 0,0 |
| 6 | G | 0,0 | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | 0,0 | 0,0 | 0,0 | **1,0** | **1,0** | 0,0 |
| 7 | H | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | **0,0** | 0,0 | 0,0 | 0,0 |
| 8 | H | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | **0,0** | 0,0 | 0,0 | 0,0 |
| 9 | H | 0,0 | 0,0 | 0,0 | **0,5** | 0,0 | 0,0 | **0,0** | **0,0** | 1,0 | 0,0 | 0,0 | **1,0** |
| 10 | V | 0,0 | 0,0 | 0,0 | 0,0 | **1,0** | **1,0** | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | **0,0** |
| 11 | V | 0,0 | 0,0 | 0,0 | 0,0 | **1,0** | **1,0** | 0,0 | 0,0 | 0,0 | 1,0 | 1,0 | **0,0** |
| 12 | V | 0,0 | 0,0 | 0,0 | **0,5** | 0,0 | 0,0 | 0,0 | 0,0 | **1,0** | **0,0** | **0,0** | 1,0 |

**Table 2.** Similitud entre partidas usando Spectral Clustering, donde Z es indica que se ha utilizado estrategia en Zigzag , G corresponde a la Agrupada, H corresponde a la Horizontal y V corresponde a la Vertical.

La tabla 2 nos muestra la similitud obtenida utilizando el algoritmo de *Spectral Cluster*. Se puede observar que con este algoritmo la similitud entre partidas donde se ha utilizado la misma estrategia es bastante alta. Aun así, cuando comparamos partidas que han utilizado diferentes estrategias, vemos que la estrategia Agrupada se confunde con las demás estrategias. Esto se debe a que las estrategias Zigzag, Horizontal y Vertical son casos particulares de dicha estrategia.
De este experimento se extraen 2 conclusiones. La primera, el algoritmo de Spectral Clustering agrupa mejor las estrategias que K-Means. Y la segunda, es necesario utilizar más atributos para definir las estrategias, no es suficiente con las posiciones de las torres.

## 5 Conclusiones y Trabajo Futuro

Este trabajo aporta una evaluación del comportamiento de los jugadores en los juegos. Para conseguir este propósito, se ha diseado un *framework* basado en *OTD*. Para ello, se ha empleado el modelo creado en un trabajo previamente, el cual utiliza las distribuciones de las posiciones de las torres. El experimento llevado a cabo trata de determinar si los atributos utilizados para el modelo de comportamiento de usuarios son suficientemente descriptivos. Por este motivo, hemos utilizado algoritmos de *clustering* (*K-Means* y *Spectral Clustering*) para agrupar las estrategias. Más tarde estos grupos se utilizan para calcular la similitud entre partidas.
De la similitud obtenida por ambos métodos de *clustering* hemos obtenido las siguientes conclusiones. La similitud obtenida mediante *K-Means* nos indica que

K-Means solo distingue entre estrategias Lineales (*Horizontal* o *Vertical*)y No Lineales (*Grouped* o *Zigzag*). En el caso de *Spectral Clustering* se obtiene muy buena similitud entre estrategias empleadas en diferentes partidas. Esto indica que este algoritmo distingue mejor las estrategias, aun así la estrategia *Agrupada* se confunde con las demás estrategias. Por lo que se pude concluir que el modelo utilizado para detectar las estrategias no bueno. Y por otra parte, el algoritmo de Spectral Clustering discrimina mejor las estrategias.
En trabajos futuros, será necesario utilizar más atributos para mejorar la clasificación de estrategias y estudiar más métodos no supervisados para determinar cual hace mejor partición de los datos.

## References


1. K. Alsabti, S. Ranka, and V. Singh. An efficient k-means clustering algorithm. *Association for the Advancement of Arti?cial Intelligence*, 1997.
2. D. Brzezinski. Mining data streams with concept drift. Master's thesis, Poznan University of Technology, 2010.
3. R. Dey and C. Child. Ql-bt: Enhancing behaviour tree design and implementation with q-learning. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE, 2013.
4. D. J. Gagne and C. B. Congdon. Fright: A flexible rule-based intelligent ghost team for ms. pac-man. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 273–280. IEEE, 2012.
5. A. Gonzalez-Prdo, F. Palero, and D. Camacho. An empirical study on collective intelligence algorithms for vide games problem-solving. *Computing and Informatics*, In press, 2014.
6. U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec. 2007.
7. F. Palero, A. Gonzalez-Prdo, and D. Camacho. Simple gamer interaction analysis through tower defence games. Submited, 2014.
8. C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling player experience in super mario bros. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 132–139. IEEE, 2009.
9. M. Polceanu. Mirrorbot: Using human-inspired mirroring behavior to pass a turing test. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE, 2013.
10. S. Ray and R. H. Turi. 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*, pages 137–143, 1999.
11. G. Synnaeve and P. Bessiere. A bayesian model for rts units control applied to starcraft. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 190–196. IEEE, 2011.
12. H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 226–235, New York, NY, USA, 2003. ACM.
13. G. N. Yannakakis and J. Hallam. Feature selection for capturing the experience of fun. *Proceedings of the AIIDE*, 7:37–42, 2007.