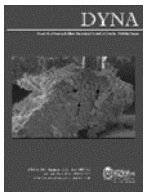
UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS



DYNA
http://dyna.medellin.unal.edu.co/

# Design of a serious emerging games engine based on the optimization algorithm of ant colony

Jose Aguilar, Junior Altamiranda & Francisco Díaz

*CEMISID, Facultad de Ingeniería, Universidad de los Andes, Mérida, Venezuela. aguilar@ula.ve, altamira@ula.ve, francisco.diaz@ula.ve*

Received: January 19th, 2018. Received in revised form: Junio 18th, 2018. Accepted: June 27th, 2018.

**Abstract**
An engine for a serious emergent game (MJSE due its Spanish acronym) must make explicit the possibility of emergence in a serious game, from the coordination of game frames, adapted to the specific educational context where the game is being used. In particular, in previous works have proposed a hierarchical architecture for a MJSE, composed of sub-motors. The main objective of this work is to specify the sub-motors responsible for the emergence process of the serious game, which are based on the algorithm of optimization based on ant colonies (ACO). These sub-motors carry out the management of the set of game frames according to the educational context of interest, in such a way to merge them into a single frame, in order to make the dynamic of the serious game emerge. Additionally, this paper analyzes the behavior of the sub-motors in a case study, with very encouraging results.

*Keywords:* games engine; serious emerging games; optimization algorithm based on ants colonies.

# Diseño de un motor de juegos serios emergentes basado en el algoritmo de optimización de colonia de hormigas

**Resumen**
Un motor de juegos serios emergentes (MJSE) debe hacer explícito la posibilidad de emergencia en un juego serio, a partir del manejo coordinado de tramas de juegos, adaptadas al contexto educativo especifico donde se esté desarrollando el juego. En particular, en anteriores trabajos se ha propuesto una arquitectura jerárquica para el MJSE, constitutiva de submotores. El objetivo principal de este trabajo es especificar los submotores encargados del proceso de emergencia del juego serio, los cuales se basan en el algoritmo de optimización basado en colonias de hormigas (ACO). Dichos submotores realizan la gestión del conjunto de tramas de juego según el contexto-dominio educativo de interés, de tal manera de fusionarlas en una única trama, para hacer de esta forma emerger la dinámica del juego serio. Adicionalmente, en este trabajo se analiza el comportamiento de dichos submotores en un caso de estudio, mostrando resultados muy alentadores como MJSE.

*Palabras claves:* motor de juegos; juegos serios emergentes; algoritmo de optimización basado en colonias de hormigas.

## 1. Introducción

Un motor de juego es un conjunto de programas que permiten la creación, la representación y la ejecución de un juego. Hoy en día, existen una gran variedad de motores para que el usuario pueda crear sus propios juegos. Ahora bien, no se han desarrollado motores de juegos para juegos emergentes, y mucho menos para juegos serios emergentes.

El objetivo específico de este trabajo, es realizar la especificación de un MJSE. En particular, nosotros estamos interesados en MJSEs, cuyo objetivo es diferente al de meramente jugar, tal que puede ser educativo, de entrenamiento, de rehabilitación, entre otros fines, y en particular, cuya dinámica surja en función de lo que va aconteciendo en el contexto [1]. Los juegos serios emergentes parten de dos teorías, la de juegos serios que establece que es un juego que tiene un propósito específico, el cual puede estar relacionado con el aprendizaje, con la comprensión de un tema complejo [2,19-21]; y la de sistemas emergentes, que son sistemas cuyos comportamientos surgen a partir de las interacciones espontáneas de los elementos relativamente simples que los componen, sin leyes explícitas [3,4]. En ese

**How to cite:** Aguilar, J., Altamiranda, J. and Díaz, F., Diseño de un motor de juegos serios emergentes basado en el algoritmo de optimización de colonia de hormigas. DYNA, 85(206), pp. 311-320, September, 2018.

sentido, estos tipos de juegos requieren de un MJSE específico a sus características.

A continuación, comentamos rápidamente los trabajos de mayor interés para nuestra propuesta. En [1] se ha propuesto la arquitectura de un MJSE estructurada en capas, para el cual se hace la propuesta de los submotores encargados del proceso de emergencia en este trabajo. En [5] exponen el modelo conceptual de un motor de juegos basado en capas jerárquicas. En [6] se introduce la noción de juegos emergentes, bajo el enfoque basado en guiones, como un juego en la que su dinámica va surgiendo en función del contexto. En esa investigación se propone un sistema de motores de juegos emergentes denominado EmerGENT, sustentado en autómatas celulares. En [7] muestran un marco para la selección de motores de juegos serios, usando tres criterios para el proceso de selección: 1. Fidelidad audiovisual y funcional, 2. Capacidad de composición y 3. Accesibilidad. Además, realizan dicha comparación en varios motores de juegos (Cry Engine, Source Engine, Unreal y Unity). Finalmente, en [8] se presentan cinco enfoques de narrativas emergentes: crear mundos creíbles, incorporar la experiencia del jugador en el juego, establecer espacios de múltiples opciones de decisión en los estados del juego, permitir la incertidumbre posibilitando crear situaciones inesperadas, posibilitar la co-autoría tal que el jugador es responsable de la historia y la dirección que toma el juego.

En específico, este trabajo propone un MJSE basado en ACO, el cual realiza el manejo de las tramas del juego guiados por el contexto educativo. De esta manera, el MJSE utiliza ACO para hacer emerger la dinámica del juego en función del contexto educativo. El artículo se organiza de la siguiente manera, la siguiente sección presenta los aspectos teóricos bases de este trabajo, seguidamente se presenta la arquitectura general del MJSE, para después detallar el componente que permite la emergencia del juego serio. La sección 3 presenta el diseño del MJSE basado en ACO. A continuación, se presenta un caso de estudio, y se compara el MJSE con otros trabajos similares, para finalmente terminar con las conclusiones.

## 2. Aspectos teóricos

### *2.1. Juegos serios emergentes*

En [2, 7, 9] definen a los Juegos Serios (JS) como juegos diseñados y desarrollados desde un objetivo distinto a la pura diversión. Los JS proveen un ambiente motivador, un contexto de entretenimiento y auto-fortalecimiento, para que los jugadores "aprendan haciendo" a través de sus propios errores, gracias a desafíos adecuados a su nivel de competencia y a una realimentación constante.

Por otro lado, en [6] definen a un Juego Emergente (JE) como uno que se va desplegando de manera espontánea, autónoma, y sin leyes explícitas, adecuándose a los jugadores. En [3] proponen el JE "Metrópolis", cuyo funcionalismo parte de la premisa: Las ciudades son auto-gestionadas por decisiones tomadas en conjunto por sus habitantes (jugadores), sin que exista un habitante con un papel más importante. Concluyentemente, en Metrópolis emergen patrones urbanísticos en la ciudad por las decisiones que sus habitantes toman. [4] propone una extensión a Metrópolis con la incorporación de mecanismos emergentes a la dinámica del juego para adaptarla a sus jugadores. Proponen cinco niveles de emergencia en un JE:

1. *Comportamiento:* se generan nuevas tácticas y estrategias en el juego, siguiendo reglas con variantes.
2. *Secuencia:* se crean nuevos escenarios, tramas o temáticas en los juegos, cambiando el ambiente y el contexto del juego.
3. *Final:* patrones que reflejan resultados de los juegos, o continuar el juego de forma infinita (en el caso de los JS no es necesario que termine el juego ni que haya un ganador).
4. *Propiedad:* cambia las características en los objetos.
5. *Modelo de Negocio:* por el surgimiento de mercados y servicios alrededor de los juegos.
6. *Utilidad:* depende para que se va a utilizar el JE.

Finalmente, en un JSE la historia, la dinámica, el guion que va surgiendo, depende del contexto donde se va dando el juego. Así, el JSE posiciona al jugador en un entorno de realimentación de información y motivación al logro, guiado por un objetivo explícito distinto de la pura diversión, para superar desafíos adecuados a su capacidad, y aprender de sus propios errores. En específico, el comportamiento que va dándose en el JSE resulta espontáneo, autónomo, y sin leyes explícitas, adecuándose a los jugadores, a sus entornos, entre otras cosas.

En [5] definen a los motores de juegos como los ambientes computacionales que permiten realizar videojuegos. Pueden ser vistos como programas, librería o *framework*, para el desarrollo de videojuegos. Un motor de videojuegos es el núcleo general que une todas las partes de un juego. Así, los desarrolladores se centran en las mecánicas, las lógicas y las características específicas del juego que está concibiendo.

Por otro lado, en un JSE el MJSE maneja un conjunto de tramas, y las va seleccionando y fusionando en una única trama, según el contexto y el objetivo del juego. Para ello, el MJSE requiere realizar el siguiente conjunto de tareas:

- La búsqueda y selección del conjunto de tramas adecuadas al contexto donde funcionará el JSE,
- La fusión de algunas de las tramas seleccionadas en una única trama, según los objetivos del JS, que se desplegará inicialmente en el videojuego, y la supervisión de la dinámica del desarrollo del JSE, para adecuarlo a la dinámica del contexto donde funciona, a través de la adaptación de su trama.

### *2.2. Arquitectura de MJSE*

Un MJSE debe mantener la relación-lógica entre la trama y el JSE, usando los submotores clásicos de los videojuegos para administrar gráficos e imágenes, controlar el movimiento físico de los objetos, gestionar el sonido, entre otras cosas. Pero ahora también debe integrar otro submotor, que hemos llamado submotor de tramas, para realizar las tareas de gestión de las tramas que componen el JSE. En [1] hemos propuesto una arquitectura general de un MJSE. En esta sección presentaremos brevemente dicha arquitectura (para más detalles ir a [1]).

#### 2.2.1. Arquitectura general del MJSE

La arquitectura del MJSE está basada en capas jerárquicas, en las cuales están los diferentes submotores que la componen (ver Fig. 1).

1. *Núcleo del Motor de Videojuego:* es el elemento central del MJSE, en él se encuentran los seis submotores de base para cualquier videojuego, los cuales son:
   a. Submotor de Gráficos: realiza y maneja los gráficos, imágenes y dibujos primitivos, basados en sus características: texturización, mallas, terrenos, etc.
   b. Submotor Físico: se encarga de realizar los movimientos físicos de los objetos en un ambiente virtual.
   c. Submotor de Sonido: se encarga de gestionar todo lo referente a lo audible: música, audio, ruido, micrófonos, entre otras cosas.
   d. Submotor de Interacción: se encarga de configurar las interacciones dentro y fuera de los videojuegos.
   e. Submotor de Video: se encarga de la unión del sonido y las imágenes en secuencias fílmicas para realizar videoclips, caricaturas, cortes de películas, etc., usadas en el videojuego.
   f. Submotor de Renderización: se encarga de gestionar las imágenes en movimiento, considerando aspectos como: mejorar la iluminación, sombreado y oscuridad, definir sus efectos visuales, entre otros.
2. *Subsistema de Emergencia del Videojuego:* esta capa permite hacer emerger un juego, según los objetivos que se deben cumplir durante el juego o la dinámica que va ocurriendo en el mismo. En esta capa funciona el Submotor de Trama Emergente (STE), el cual interactúa con el jugador/contexto para hacer emerger el JSE, y el submotor de IA, que provee mecanismos requeridos en el proceso de emergencia del JSE.
3. *Subsistema de Adaptación del Videojuego:* esta capa permite adecuar un videojuego sin tener que hacer modificaciones profundas en él, actuando sobre sus características de base. Por ejemplo: puede establecer los valores de renderizados específicos del juego. Esta capa utiliza el SIA para adaptar esos valores de las características en los elementos, el ambiente y los eventos del videojuego.
   Como hemos visto, en estas dos últimas capas operan dos sub-motores:
   a. Submotor de Inteligencia Artificial (SIA): se encarga de introducir comportamientos inteligentes en los personajes y componentes del JSE, para adecuarlo a la dinámica y contexto en el que se está ejecutando. También, permite adaptar las narrativas, y en general los parámetros de los videojuegos, al contexto deseado.
   b. Submotor de Trama Emergente (STE): permite crear las narrativas de los videojuegos al contexto deseado, al objetivo del juego serio. Una vez recolectada la información precisa del contexto, realiza gestión de escenas y eventos, configuración de subtramas/guiones del juego, entre otras cosas.

### 2.2.2. Sub-motor de trama emergente

El STE establece la relación-lógica entre las tramas del JSE. El STE permite gestionar:

- El escenario, los personajes y eventos en la trama, de acuerdo a los requerimientos espacio-tiempo del tema.
- La coherencia y adecuación de la trama, con los requerimientos semánticos del contexto.
- La congruencia entre la actividad deseada a desarrollar y la trama del JSE.

Para ello, el STE posee las siguientes sub-capas:

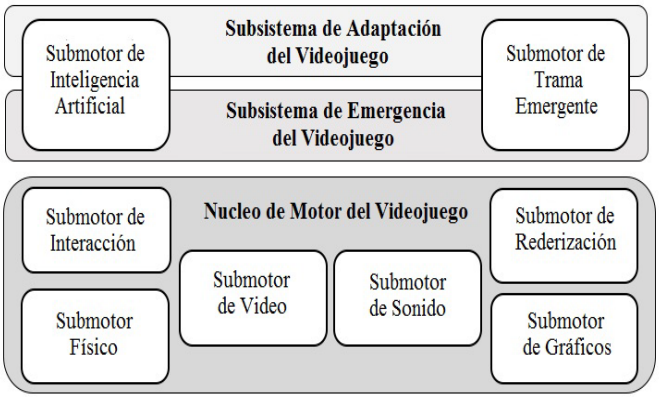



Figura 1: Arquitectura General del MJSE.
Fuente: Elaboración Propia.

1. *Gestor de Materia (GM):* determina la temática que se esta tratando en el contexto, para a partir de allí, establecer el tipo de trama que se debe aplicar en el JSE.
2. *Gestor de Videojuegos (GV):* busca los tipos de videojuegos (edugame, advergame, etc.) para esa temática, usando Repositorios de Objetos de Aprendizaje (ROA). Para esa busqueda, usa los metadatos de los OAs en un ROA y los compara con lo definido por el GM. Si no consigue al menos un videojuego parecido a lo buscado, llama al Módulo de Generación de JSE para generar un videojuego.
3. *Módulo de Generación de JSE (MGJSE):* es el responsable del ensamblaje, de un nuevo JSE, invocando para ello a los siguientes tres módulos de STE. Además, durante el juego, debido a la interacción entre el JSE y su contexto (ambiente, jugadores, etc.), pueden emerger nuevos comportamientos en el JSE por cambios que el MGJSE introduzca en el comportamiento, secuencia, final del juego, propiedad de sus componentes o utilidad del mismo.
4. *Storyboard (SB):* se encarga de generar guiones narrativos para el JSE, usando para ello la información del GM.
5. *Gestión de Escenas (GE):* se encarga de generar el ambiente, mundo o entorno, requeridos por las tramas del JSE.
6. *Sistemas de Eventos (SE):* este módulo se encarga de generar eventos especializados requeridos por el SB, para generar los comportamientos deseados en el JSE.

En específico, el STE se comporta de la siguiente manera (ver Fig. 2):

   a. El GM determina toda la información sobre la temática deseada, según el contexto donde se quiera usar el juego.
   b. El GV seleccionada de ROAs los JSs que sean más compatibles al uso deseado. Para ello, compara los metadatos de los JSs en el ROA, descritos bajo algunos de los estándares (LOM/SCORM) [10], con lo determinado por GM.
   c. El MGJSE determina "SI" se consiguen JSs que cubran la temática deseada; en ese caso los adecua y los envía al entorno donde serán usados (por ejemplo, un entorno virtual de aprendizaje, o VLE por sus siglas en inglés). De lo contrario, el MGJSE utiliza un algoritmo para hacer emerger un JSE compatible con el tema deseado. En este trabajo, se usa un algoritmo ACO para ello.

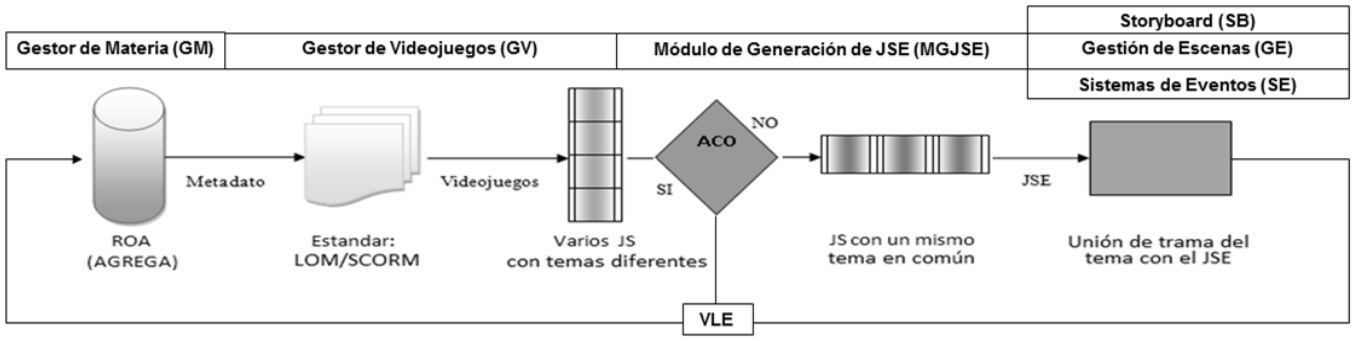



Figura 2. Componentes del STE.
Fuente: Elaboración Propia.

d. El GV seleccionada de ROAs los JSs que sean más compatibles al uso deseado. Para ello, compara los metadatos de los JSs en el ROA, descritos bajo algunos de los estándares (LOM/SCORM) [10], con lo determinado por GM.
e. El MGJSE determina "SI" se consiguen JSs que cubran la temática deseada; en ese caso los adecua y los envía al entorno donde serán usados (por ejemplo, un entorno virtual de aprendizaje, o VLE por sus siglas en inglés). De lo contrario, el MGJSE utiliza un algoritmo para hacer emerger un JSE compatible con el tema deseado. En este trabajo, se usa un algoritmo ACO para ello.
f. La solución final propuesta por el MGJSE, compuesta por uno o varios JSs, articula de manera natural sus bloques narrativos (SB), bajo los escenarios especificados en dichos JSs (GE), y los eventos definidos en ellos (SE).
g. El MJSE supervisa la dinámica del JSE bajo ejecución, para irlo adecuando al contexto de manera emergente, haciendo emerger nuevos comportamientos y secuencias (llama de nuevo a ACO), actualizando propiedades en los objetos, el final o su utilidad, en función de la misma.

## 3. STE basada en ACO

### *3.1. Especificación General de ACO*

ACO es un tipo de meta heurística basada en población, el cual está inspirados en la conducta de las colonias de hormigas reales cuando buscan comida, para resolver problemas de optimización combinatoria [11,12]. En nuestro caso, ACO crea agentes hormigas, y cada una construirá un JSE para un mismo objetivo (temática), usando diferentes tramas. En particular, un videojuego seleccionado de un repositorio de juegos lo denominaremos subtrama (trama seleccionada), mientras que una trama de un JSE es la unión/fusión de una o más subtramas. ACO está compuesto por:

1. *Espacio de solución:* o espacio que recorrerán las hormigas para obtener soluciones. Es un grafo compuesto por nodos que describen las subtramas seleccionadas desde diferentes repositorios, y arcos que establecen las relaciones de dependencia entre ellas, cuando existen. También, el arco es usado por las hormigas para establecer la secuencia lógica entre las subtramas, para general la trama final del JSE, usando para ello feromonas.
2. *Hormigas:* caminan en el grafo de sub-tramas (tramas seleccionadas).
3. *Solución:* las sub-tramas (nodos) son marcadas por una feromona en el grafo, igual que los arcos que los interconectan. Cuando converge el algoritmo ACO, las sub-tramas son seleccionadas según si su feromona pasa un umbral, igual que los arcos que salen de ellas, los cuales establecen la secuencia lógica de ejecución de las subtramas.
4. *Feromonas:* hay dos tipos, una para las sub-tramas (nodos) y otro para los arcos entre las subtramas. La feromona define lo deseable de la subtrama y de los arcos que las interconectan, para pertenecer a la solución final.
5. *Función Feromona* actualiza cada tipo de feromona en función de la calidad del JSE propuesto.
6. *Función Heurística:* define la decisión heurística que toma una hormiga, al estar en una sub-trama (nodo), con respecto a que otra sub-trama (nodo) debe continuar a visitar después de ella.

ACO permite un proceso de aprendizaje colectivo entre las hormigas, para configurar el JSE [12, 13, 14]. Así, la solución no es más que una secuencia de sub-tramas tomadas del repositorio de tramas, ordenadas según una secuencia lógica entre ellas. Para seleccionar las tramas candidatas del repositorio de tramas que conformaran los nodos del grafo (sub-tramas), se considera la relación entre el contenido de la trama (definido en sus metadatos) y las características del JSE a desarrollar, basada en aspectos como: tema del juego serio, utilidad de la trama, etc.

### *3.2. Macroalgoritmo ACO para el STE*

El macroalgoritmo a usar es el clásico de ACO [12, 13, 14]. En específico, en nuestro caso consiste de una fase de inicialización de parámetros, un proceso iterativo sobre dos componentes hasta que el sistema converja, y la construcción de la solución final. Dicho macroalgoritmo se muestra a continuación:

Inicializar parámetros, feromonas y grafo
**Repita Mientras** (no se cumpla condición de terminación)
ConstruirSolucionesporHormigas
ActualizarFeromona
ConstruirSolucionFinal

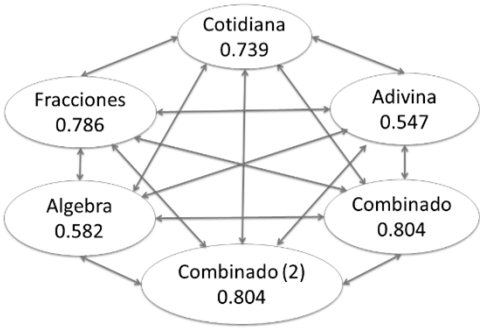



Figura 3: Grafo Teórico.
Fuente: Elaboración Propia


Figura 4: Nodo del grafo.
Fuente: Elaboración Propia.

a. Creación del grafo teórico de recorrido de las Hormigas (*Inicializar grafo*): El grafo teórico es definido como G = (N, E), donde N es un conjunto de nodos que representan las sub-tramas (JSs seleccionados por GV de ROAs), y E un conjunto de arcos que conectan todos los nodos de N (ver Fig. 3). Además, se establece una función de peso $d_{ij}$ para determinar el peso de un arco (i, j) ∈ E, tal que es 1 si existe una relación de dependencia secuencial entre dos nodos ($JS_i$, $JS_j$, ∈ N), y 0 en caso contrario. Eso implica que $d_{ij}\neq 0$ cuando entre dos nodos hay una relación de dependencia entre ellos ver Fig. 3.

Los nodos del grafo guardan la información de la subtrama que representan (ID del nodo), pero adicionalmente almacenan el nivel de similitud entre el nodo y la temática tratada (basado en su metadato, y calculado por GV), y un nivel de feromona actualizado por las hormigas que transitan a través de ellos, que indica la deseabilidad de dicho nodo (ver Fig. 4).

b. Construcción de la solución por parte de las hormigas (*ConstruirSolucionesporHormigas*): En esta fase, algunas consideraciones se realizan:
   - Se define el número de individuos que integran la colonia.
   - El número de individuos de la colonia es invariable.
   - Cada hormiga inicialmente se coloca de modo aleatorio en el grafo para iniciar su recorrido.

Cada hormiga ejecuta una función heurística (o de transición) desde el nodo actual donde se encuentra, para determinar el próximo nodo que puede visitar que no haya previamente visitado. Esta función es definida como la probabilidad de visitar desde el nodo r a cada uno de sus nodos contiguos *s*, en función del nivel de feromona "γ(r,s)" del arco entre el nodo i y cada nodo s, y el índice de similitud "η(s)" de cada nodo *s* con respecto a la temática buscada, lo cual es calculado para todos los nodos aun no visitados por la hormiga k ($J_r^k$). Eso se expresa en la siguiente ecuación:

$$P_{(r,s)}^{k} = \frac{\gamma_{(r,s)} \cdot \eta_{(s)}}{\Sigma_{u \in J_r^k} \gamma_{(r,u)} \cdot \eta_{(s)}} \text{ Si } s \in J_r^k \qquad (1)$$

Particularmente, esta expresión se usa cuando no se quiere parar la construcción de una solución. Es decir, la hormiga puede culminar en cualquier momento la construcción de una solución, o continuar paseando por los nodos hasta recorrer a todos, basado en el siguiente algoritmo:

$$Recorrido\ Hormiga\ k = \begin{cases} parar, & num_{aleatorio} > umbral\_parar \\ continuar\ constr.JSE\ basado\ ec.1, & caso\ contrario \end{cases}$$

### *3.3. Actualización de los feromonas*

En nuestro caso hay dos feromonas, una para los arcos y otra para los nodos. Ambas son actualizadas al final de cada iteración. En ese sentido, cada hormiga actualizará la feromona de cada arista y cada nodo que visita. Para ello, se determina un índice de la calidad del JSE propuesto por cada hormiga, el cual será usado durante el proceso de actualización. Ese índice de calidad es calculado a partir del nivel de similitud del JSE propuesto por cada hormiga con respecto al tema deseado (definido por GV). El nivel de similitud del JSE propuesto por una hormiga k ($\eta^k$) no es más que la agregación de los índices de similitud "$\eta(s)^p$", para p=1… P, de las P subtramas que componen el JSE.

De esta manera, la feromona de cada nodo y arco se actualiza usándose las ecuaciones genéricas siguientes:

$$\gamma_{(r,s)} = \gamma_{(r,s)} + \Delta\gamma_{(r,s)}$$
$$\gamma_{(r)} = \gamma_{(r)} + \Delta\gamma_{(r)} \qquad (2)$$

Donde, $\Delta\gamma_{(r,s),}$ y $\Delta\gamma_{(r),}$ son el incremento de las feromonas, que vienen dadas por la sumatoria de las cantidades de feromona dejadas por las M hormigas en el arco (r, s) o nodo (r):

ID del nodo: JS
Índice de similitud "$\eta_{(s)}$"
Nivel de feromona "$\gamma_{(r)}$"

$$\gamma_{(r,s)} = \frac{\Sigma_{k=1}^{M} \Delta\gamma_{(r,s)}}{M}$$
$$\gamma_{(r)} = \frac{\Sigma_{k=1}^{M} \Delta\gamma_{(r)}}{M} \qquad (3)$$

y tanto $\Delta\gamma_{(r)}^{k}$ $\Delta\gamma_{(r)}^{k}$ como $\Delta\gamma_{(r,s)}^{k}$ son iguales a $\eta^k$

La actualización de las feromonas también tiene un proceso de evaporación de las mismas, la cual se ejecuta sobre todos los

nodos y arcos en el grafo al final de cada iteración, según la siguiente función:

$$\gamma_{(r,s)} = (1-\rho) + \Delta\,\gamma_{(r,s)} \qquad \gamma_{(r)} = (1-\rho) + \Delta\,\gamma_{(r)} \qquad (4)$$

Donde, $\rho \in (0,1]$ es el coeficiente de evaporación de feromona. En general, este proceso se realiza repetidamente hasta que la colonia converge en un grupo de soluciones (JSEs).

c. Construcción de la solución final *(ConstruirSolucionFinal):* Una vez que la colonia concluye su trabajo, se debe pasar a construir la solución final, es decir, el JSE que propondrá ACO. Para ello, se hace un recorrido sobre todos los nodos del grafo, seleccionándose los nodos con mayor valor de feromona, y los arcos que saldrán de cada uno de ellos serán seleccionados según un valor de feromona también (será el que tenga el valor mayor), tal que se garantice que todos los nodos (subtramas) conformen un camino (esa será la secuencia lógica del JSE propuesto).

### *3.4. Comparación entre las subtramas y el tema deseado*

El nivel de similitud entre el nodo y la temática tratada, calculado por GV, consiste en comparar el metadato de la subtrama con respecto a la información del curso establecida por GM. Eso determina el interés/parecido de dicha subtrama con respecto al curso. Para realizar esa comparación se usará el estándar LOM [15], y en particular, los criterios siguientes:

1. *Title:* es el nombre del JS. En este caso, se compara si una o varias palabras del título es o son igual(es) a la temática que se está buscando. Por ejemplo: si se está buscando "Fracciones", se compara con el título del JS "Domino de Fracciones". Se da una puntuación en la siguiente escala: igual (1), parecida (0,9-0,8), sinónimos (0,7-0,6), algunas palabras (0,5-0,4), o pocas palabras (≤0,3).
2. *Language:* es el lenguaje para el que fue hecho el JS. En este caso, se compara el idioma deseado con el del JS, según la escala: mismo idioma (1), idiomas derivados (0,9-0,8), idiomas parecidos (0,7-0,6), escritura parecida (0,5-0,4), poco en común (≤0,3).
3. *Description:* contiene una descripción del contenido del JS. Por ejemplo: "juego de dominó interactivo que relaciona sus piezas según las diferentes representaciones fraccionarias gráficas o numéricas"). En este caso, se compara si una o varias palabras de la descripción es o son igual(es) a la que están buscando, usando la siguiente escala: igual (1), parecida (0,9-0,8), sinónimo (0,7-0,6), algunas palabras (0,5-0,4), o pocas palabras (≤0,3).
4. *Keyword:* palabras claves que representan el tema principal del JS, por ejemplo: fracciones, domino, cálculo y probabilidad. En este caso, se compara usando la misma escala de los ítems 1 y 3.
5. *Coverage:* describe la zona geográfica o región en la que es aplicable el JS. Se usa la siguiente escala: universal (1); además, cuando es nacional, regional, estadal y local, y coincide con lo que se busca, también es 1. De lo contrario es 0.
6. *Format:* identifica el software necesario para usar el JS. Si es compatible con lo que se busca su valor de comparación es 1, de lo contrario es 0.
7. *TypicalAgeRange:* edad intelectual del destinatario típico del JS, por ejemplo: "7-9", "0-5", "15", ">18", "adecuado para niños mayores de 7 años", "adults only". Si es compatible con lo que se busca su valor de comparación es 1, si es cercano 0.5, de lo contrario es 0.
8. *Dificulty:* este elemento describe lo difícil que resulta el JS para los destinatarios típicos, por ejemplo: escala muy alta, alta, media, baja o muy baja. En este caso, se compara usando la misma escala del ítem 7.
9. *Duration:* Tiempo aproximado o típico que necesitan para asimilar el JS los destinatarios objetivo. En este caso, se compara usando la misma escala del ítem 7.
10. *InteractivityLevel: e*l grado de interactividad que caracteriza a este JS. Se mide como muy alto, alto, medio, bajo, y muy bajo. Si es compatible con lo que se busca su valor de comparación es 1, si es cercana 0.5, de lo contrario es 0.
11. *SemanticDensity:* La densidad semántica de un JS puede ser estimada en función de su tamaño, ámbito o – en el caso de recursos auto-regulados tales como audio y vídeo duración. La densidad semántica de un JS es independiente de su dificultad. Se mide como: muy alto, alto, medio, bajo y muy bajo. En este caso, se compara usando la misma escala del ítem 10.
12. *IntendedEndUserRole:* El usuario(s) principal(es) para el que ha sido diseñado este JS, el cual puede ser un profesor, el autor, un aprendiz, entre otros. En este caso, se compara usando la misma escala del ítem 10.
13. *Context:* El entorno principal para el que fue diseñado el JS, utilizando para ello los siguientes valores: escuela, nivel educativo, entre otros. En este caso, se compara usando la misma escala del ítem 10.
14. *CognitiveProcess:* es el tipo específico del proceso cognitivo en el JS, y según el estándar OED:1989 pueden ser los siguientes valores: ejercicio, simulación, cuestionario, diapositiva, texto narrativo, examen, experimento, planteamiento de problema, autoevaluación, conferencia, entre otros. En este caso, se compara usando la misma escala del ítem 10.
15. *Cost:* Indicación de si el JS requiere pago para su uso, y puede ser si(0)/no(1). Si es compatible con lo que se busca su valor de comparación es 1, de lo contrario es 0.

Si no es especificado algunos de los ítems anteriores, su valor de comparación por defecto es 1. Una vez que se tienen los valores de esos criterios del estándar LOM, tanto del tema deseado (lo define GM) como de los JS recuperados de los ROAs, GV compara criterio por criterio para cada objeto recuperado y el tema deseado, para establecer el nivel de similitud entre ellos. Cada criterio donde coinciden aumenta el nivel de similitud entre ellos, es decir, se suman los criterios similares para calcular la similitud de una subtrama con respecto al tema deseado, y se divide entre el numero de criterios usados en la evaluación (en nuestro caso, 15).

## 4. Caso de estudio

A continuación, vamos a mostrar un ejemplo de cómo funcionaría el MJSE propuesto. Se parte de la hipótesis que se está en una clase de matemáticas, y se requiere definir un JSE

con el objetivo de explicar y resolver problemas con fracciones. Los datos del curso son definidos en la tabla 1. Además, los datos de los estudiantes y su contexto educativo son mostrados en la tabla 2.

Toda esa contextualización lo haría GM. Por otro lado, se usó el repositorio de aprendizaje https://goo.gl/otiH78, el cual usa los metadatos en formato SCORM [10] .

Con la información provista por GM, GV usa los metadatos SCORM de cada objeto de aprendizaje en AGREGA para hacer la comparación con el tema deseado. La Tabla 3 da un ejemplo de esa comparación entre un juego denominado "Domino de Fracciones" y el tema al que se le desea dar apoyo usando JSE (clase de matemáticas sobre problemas con fracciones), y el valor final de esa comparación.

Una vez que se realiza la comparación de todos los JS en los ROA con el tema deseado, el MJSE ejecuta el siguiente algoritmo:

Si ($\exists!_i$ Similitud (metadata $RA_i$, Tema_Deseado) >= 0.850) entonces

Envía $RA_i$ con esa puntuación al VLE

Si (($\exists_i$ 0.4<=Similitud ($metadata_i$, Tema_Deseado)<=0.850) entonces

Invoca ACO

Tabla 1.
Datos del curso

| Datos Básicos | | Planificación | |
|---|---|---|---|
| ***a.*** | ***Materia:*** *Matemática* | ***a.*** | ***Tipo de Actividad:*** *Evaluación* |
| ***b.*** | ***Modulo:*** *Número Racional* | *b.* | ***Explicación:*** *5 minutos* |
| *c.* | ***Tema:*** *Fracciones* | ***c.*** | ***Uso del Juego:*** *30 minutos* |
| *d.* | ***Clase:*** *45 Minutos* | ***d.*** | ***Preguntas:*** *5 minutos* |
| *e.* | ***OAs:*** *JSE* | ***e.*** | ***Finalización:*** *5 minutos* |

Fuente: Elaboración Propia

Tabla 2.
Datos de los estudiantes y del centro educativo

| Datos Básicos | Datos Básicos SA |
|---|---|
| **a.** ***Idioma:*** *Español.* | **a. Nro. de Estudiantes:** 30 inscritos |
| **b.** ***Lugar:*** *Timotes - Venezuela* | **b. Promedio de Exámenes:** 1er parcial: 10 – 2do parcial: 15 |
| **c.** ***Institución:*** *Canónigo Uzcátegui* | **c. Asistencia de Estudiantes:** 27 asisten |
| **d.** ***Carrera:*** *Ingeniería de Sistemas* | **d. Promedio total o SP** = 13 pts. |
| **e.** ***Pensum:*** *9° año* | **e. Nro. de Clases:** 20 clases de 30 |
| **f.** ***Semestre****: 2do Lapso* | **f. Porcentaje de Asistencia:**86,5 % |
| **g.** ***Asignatura:*** *Matemática de 9° año* | |

Fuente: Elaboración Propia

Es decir, el MJSE determina si hay algún JS que cumple con el tema deseado, de lo contrario, si consigue algunos más o menos similares, intenta hacer emerger un JS para el tema deseado usando ACO.

En nuestro caso, supongamos que ningún juego supera el umbral de similitud de 0.85, pero los 6 juegos de la Fig. 5 cumplen el valor de similitud de estar entre 0.85 y 0.4. Esos 6 juegos serán los usados por ACO. Con ellos se construye el grafo que él usará (serán sus nodos, ver Fig. 6).

Ahora se puede iniciar la construcción de soluciones por parte de las hormigas. Cada hormiga se coloca inicialmente aleatoriamente en cualquier nodo contenido en el grafo (ver Fig. 6), y se toma la decisión del nodo a ser visitado utilizando el modelo de Monte Carlo basado en la ecuación (1).

Supongamos que, en ese caso, la hormiga decide trasladarse al nodo "Combinado" usando la ec. (1), tal como se observa en la Fig. 7. Continuando con el proceso, de nuevo debe decidir donde moverse, y la hormiga decide ahora moverse al nodo "Combinado (2)" (ver Fig. 8). El puede continuar con la construcción de la solución o detenerse en cualquier nomento. En este caso, suponemos que culmina cuando revisa por última vez al nodo "Adivina" (ver Fig. 9).

Tabla 3.
Ejemplo de Comparación

| LOM | Tema Deseado | Recurso de Aprendizaje | Puntuación |
|---|---|---|---|
| title | fracciones (VLE) | Dominó de **fracciones** | 1 |
| language | es (SA) | es | 1 |
| description | fracciones (VLE) | Varios juegos de dominó interactivos que relacionan sus piezas a partir de diferentes representaciones **fraccionarias** gráficas o numéricas. | 0.9 |
| keyword | fracciones (VLE) | **fracciones** equivalentes | 1 |
| coverage | universal (SA) | universal | 1 |
| format | javascript, html 5 o flash (ROA) | application/javascript | 1 |
| typicalAgeRange | 15 (SA) | 10 | 0,4 |
| difficulty | very high (SA) | medium | 0,5 |
| duration | 30 minutos (VLE) | PT0H10M0S | 0,4 |
| interactivityLevel | very high | very high | 1 |
| semanticDensity | very high (SA) | medium | 0,5 |
| intendedEndUserRole | learner (SA) | learner | 1 |
| context | schoolmate (SA) | schoolmate | 1 |
| cognitiveProcess | practise (SA) | practise | 1 |
| cost | no (ROA) | no | 1 |
| | **Total** | | **11,8/15=0,786** |

Fuente: Elaboración Propia

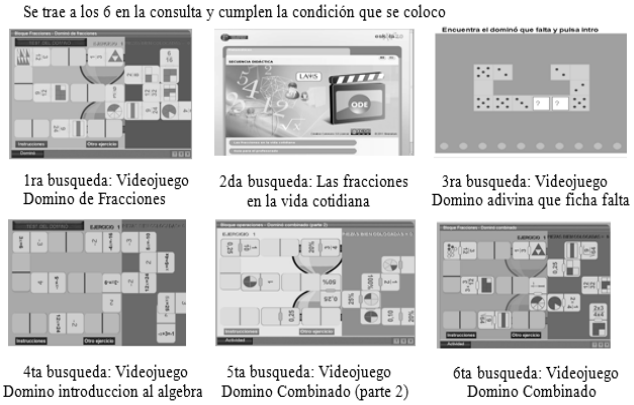



Figura 5. Videojuegos que usará ACO
Fuente: Elaboración Propia

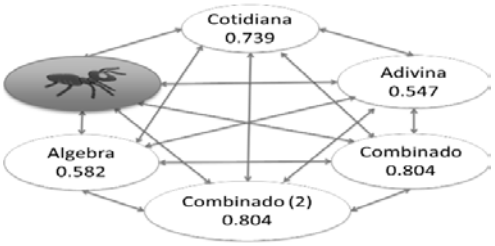



Figura 6. Elije aleatoriamente como nodo inicial "Fracciones".
Fuente: Elaboración Propia

En ese momento se inicia la actualización de los feromonas, lo cual es realizado usando la información de calidad de las propuestas de JSE de todas las hormigas de esa iteración (ver seccion 4.c, sobre el proceso de actualización). Al terminar la actualización, vuelve a comenzar cíclicamente el algoritmo con una nueva generación (iteración).

Una vez que el algoritmo converge, se pasa a construir la solución final. En ese caso, se escogen los nodos cuyo feromona pasa cierto umbral, y se conectan entre ellos según los valores más altos de los arcos que los unen. Si suponemos la Fig. 9 como la final, una vez que converge ACO, suponemos que los nodos que superan el umbral son solo 3, y los arcos que los unen (porque son los más altos) son los identificados con color rojo.

Basado en ello, el JSE creado esta compuesto por 3 sub-tramas, organizadas en la siguiente secuencia lógica:

- JS1: Fracciones – Combinado – Combinado (2)
- JS2: Algebra – Cotidiana
- JS3: Adivina – Fracciones

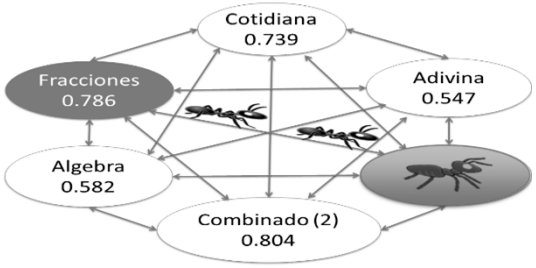



Figura 7. Elije moverse al nodo "Combinado"
Fuente: Elaboración Propia

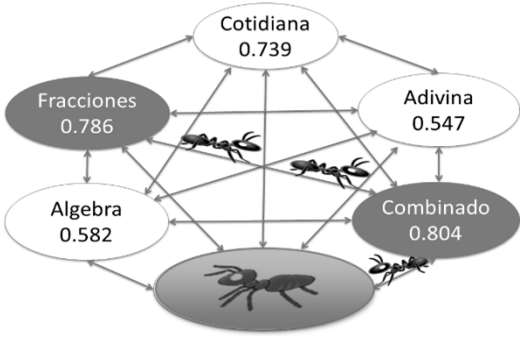



Figura 8. Elije aleatoriamente nodo "Combinado (2)"
Fuente: Elaboración Propia

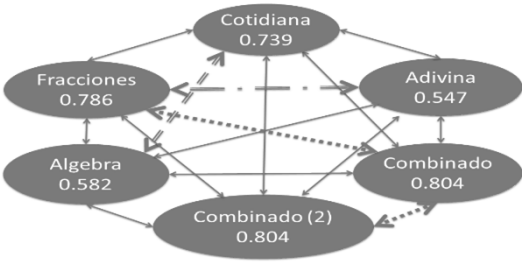



Figura 9. Solución final propuesta de JSE
Fuente: Elaboración Propia

Ese sería el JSE que emerge. Ese juego después sería observado durante su ejecución por el sistema adaptativo del MJSE, para determinar si debe ajustar parámetros, o recrear otro nuevo JSE.

## 5. Comparación con otras propuestas

Las bases teóricas de los JSE han sido introducidas en [6]. Es una investigación basada en mundos de juegos que manipulan la trama emergente, interactuando con el entorno, usando el motor de juegos emergentes (EmerGENT), cambiando el comportamiento del clima, y presentando en el entorno fenómenos como incendios, huracanes, inundaciones y terremotos. En [8] proponen un sistema para el desarrollo de narrativas para juegos emergentes con cinco enfoques: coherencia, agencia, espacio de posibilidad, incertidumbre y co-autoría. En [16] proponen entornos de aprendizaje inteligente basados en JS, se investiga sobre planificación narrativa en tiempo real, y sobre modelos para reconocer los estados emocionales en los estudiantes, proponiendo uno basado en una arquitectura que utiliza una red de decisión dinámica (RDD) y redes bayesianas dinámicas (RDB).

La principal diferencia de nuestra propuesta y los trabajos anteriores es que nuestra propuesta hace emerger un JSE utilizando diferentes subtramas, según las características del contexto donde será usado. Para ello, usa un algoritmo ACO que le permite generar posibles alternativas de JSE, para hacer emerger aquella que cumpla mejor los atributos que caracterizan al tema deseado.

En la Tabla 4 detallamos esa comparación, basado en varios criterios: Objetivo del juego, cual es la capacidad adaptativa que posee, cual MJS usa, la teoría de base que utiliza, y el tipo de emergencia que permite.

Tabla 4.
Comparación con otros trabajos recientes.

| Criterios | [6] | [8] | [16] | Presente Trabajo |
|---|---|---|---|---|
| **Objetivo del Videojuego** | Construcción de mundos | Sistemas de narrativas para JE | Entornos de aprendizaje inteligente basados en JS | Orientado por la temática del contexto |
| **Capacidad de Adaptación** | Se adapta cambiando los elementos del ambiente | Incorpora narrativas emergentes en los juegos | Sistemas de tutoría inteligente | Se adapta al tema del del contexto |
| **Motores para JSE** | Motor de Juegos Emergentes (EmerGEnT) | No usa | Motor de juego multiplatafo rma Unity | MJSE |
| **Teoría de base** | Autómatas Celulares | Narrativas emergentes | RDD y (RDB) | ACO |
| **Tipos de Emergencia** | Comportamie nto, secuencia, final, propiedad, y utilidad | Coherencia, agencia, espacio de posibilidad, incertidum bre y co-autoría. | Comportami ento, secuencia y propiedades | Comportamie nto, secuencia, propiedad, modelo de negocio final y utilidad |

Fuente: Elaboración Propia

Nuestro MJSE es el que permite más tipos de emergencias. En particular, pueden emerger comportamientos en la dinámica del juego, nuevas secuencias en los juegos, características, así como la forma de culminar según el fin del juego. Además, es el único que puede hacer emerger un nuevo JS guiado por el contexto.

## 6. Conclusiones

El uso de JSEs que se adecuen a un contexto dado (por ejemplo, a un aula inteligente), ayudan al proceso que esté ocurriendo en dicho contexto, haciéndolo más emocionante, entretenido y dinámico, motivando así a los actores en ese ambiente. Para ello, se requiere de un MJSE como el propuesto en este trabajo, con en particular un STE. La especificación de esto último es la innovación presentada en este trabajo, hecha a través de un algoritmo ACO.

Nuestra arquitectura permite emerger el JSE, pero, además, adecuarlo en tiempo de ejecución a la dinámica en el contexto. Esa es la principal diferencia de nuestro MJSE con respecto a trabajos anteriores.

ACO permite la emergencia de un JSE, a partir del manejo ordenado y sistemático de un conjunto de subtramas vinculadas a un contexto y dominio dado, para ser fusionadas en una única trama que lo represente. El algoritmo ACO de manera autónoma, selecciona las mejores subtramas que conformarán el JSE.

Como trabajo futuro, se probará el prototipo de ACO en diferentes contextos, se desarrollará el *sistema adaptativo de JSE*, que permite monitorear y adecuar el JSE a la dinámica de juego que en un momento se está dando en él. Para ello, se analizarán las diferencias emergencias posibles en un JSE (comportamiento, secuencia, final, propiedad o utilidad), y como gestionarlas desde el MJSE. También, se hará la especificación de un Agente de JSE para un Salón de Clases Inteligentes (SaCI) [17], que realizará la gestión del JSE en un aula inteligente usando nuestro MJSE; y la integración del MJSE en AmICL [18], para especificar el MJSE como un servicio en la nube.

## Referencias


[1] Aguilar, J., Altamiranda, J., Díaz, F. and Mosquera, D., Motor de juego serios en ARMAGAeco-c. Revista UNET. 28, pp. 100-110, 2016.

[2] Bellotti, F., Berta, R. and De-Gloria, A., Designing effective serious games: opportunities and challenges for research, Special Issue: Creative learning with serious games. Intl. Journal of Emerging Technologies in Learning (IJET). 5, pp. 22-35, 2010. DOI: 10.3991/ijet.v5s3.1500

[3] Aguilar, J., Cardozo, J., González, C. and Rengifo, B., Una aproximación a los juegos emergentes, Metrópolis, simulador de ciudades autogestionada, Proceedings of the 47va Conferencia. Latinoamericana de Informática, 2013.

[4] Aguilar, J., Altamiranda, J. and Chávez, D., Extensiones a Metrópolis para una emergencia fuerte. Revista Venezolana de Computación, 3(2), pp. 38-46, 2016.

[5] Vallejo, D. and Martín, C., Desarrollo de videojuegos: Arquitectura del motor de videojuegos (2ª Ed.). Universidad Castilla la Mancha, España, 2013.

[6] Sweetser, P., An emergent approach to game design – Development and play, School of Information Technology and Electrical Engineering, The University of Queensland, Australia, 2006.

[7] Petridis, P., Duenwell, I., Panzoli, D., Arnab, S., Aristidis, P., Hendrix, M. and Freitas, S., Game engines selection framework for high-fedelity serious aplications. IBIMA Publishing, International Journal of Interactive Worlds. UK, 2012. DOI: 10.5171/2012.418638

[8] Chauvin, S., Levieux, G., Donnart, J. and Natkin, S., An out-of-character approach to emergent game narratives, Proceedings of the 9th International Conference on the Foundations of Digital Games, 2014.

[9] Chan, R., Zhang, H. and Tao, X., Serious game design for stroke rehabilitation. International Journal of Information Technology. 23, 2017.

[10] ADL, Advanced Distributed Learning Sharable Content Object Reference Model (ADL-SCORM), 2006.

[11] Aguilar, J., Introducción a los sistemas emergentes, Venezuela, Talleres Gráficos, Universidad de Los Andes, 2014.

[12] Dorigo, M. and Stutz, T., Ant colony optimization: overview and recent advances, In: Handbook of met heuristics, pp. 227-263, 2010. DOI:10.1007/978-1-4419-1665-5_8

[13] Aguilar, J., A general ant colony model to solve combinatorial optimization problems. Revista Colombiana de Computación. 2(1), pp. 7-18, 2001.

[14] Aguilar, J., Velasquez, L. and Pool, M., The combinatorial ant system, applied artificial intelligence. Taylor and Francis. 18(5), pp. 427-446, 2004. DOI: 10.1080/08839510490442067

[15] IEEE LTSCLOM. Draft Standard for Learning Object Metadata. IEEE 1484.12.1, [Consultado, el 2 de Noviembre de 2017], 2002.

[16] James, C., Eun, Y., Seung, Y., Bradford, W., Jonathan, P. and Jennifer, L., Serious games get smart: intelligent game-based learning environments. AI Magazine. pp. 31-49, 2013. DOI: 10.1609/aimag.v34i4.2488

[17] Valdiviezo, P., Cordero, J., Aguilar, J., Sánchez, M., A smart learning environment based on cloud learning. International Journal of Advanced Information Science and Technology. 39(39), pp. 39-52, 2015.

[18] Sanchez, M., Aguilar, J., Cordero, J. and Valdiviezo, P., Basic features of a Reflective Middleware for Intelligent Learning Environment in the Cloud (IECL), Proceedings of the Asia-Pacific Conference on Computer Aided System Engineering, 2015. DOI: 10.1109/APCASE.2015.8

[19] Heins, S. et al., Robotic-assisted serious game for motor and cognitive post-stroke rehabilitation, Proceedings of the 5th International Conference on Serious Games and Applications for Health, 2017. DOI: 10.1109/SeGAH.2017.7939262

[20] Ben-Sadoun, G., Manera, V., Alvarez, J., Sacco, G. and Robert, P., Recommendations for the design of serious games in

neurodegenerative diseases. Frontiers in Aging Neuroscience. 10, 2018. DOI: 10.3389/fnagi.2018.00013

[21] Jaramillo, A., Salvador, L. and Luján, S., A mobile serious games assessment tool for people with motor impairments, Proceedings of the 9th International Conference on Education Technology and Computers, pp. 172-177. DOI: 10.1145/3175536.3175569

**J.L. Aguilar-Castro,** es graduado en Ing. de Sistemas en 1987 en la ULA, MSc. en Ciencias de la Computación en 1991 de la Universite Paul Sabatier-Francia, Dr. en Ciencia de la Computación en 1995 de la Universite Rene Descartes-Francia. PhD en Ciencias de Computación en la Universidad de Houston, USA en 2000. Es profesor titular del Departamento de ULA, miembro de la Academia de Ciencias de Mérida y del Comité Técnico IEEE CIS sobre Redes Neuronales. Ha sido investigador postdoctoral en varias Universidades y publicado más de 500 trabajos en sistemas paralelo/distribuidos, inteligencia computacional, etc.
ORCID: 0000-0003-4194-6882

**J.A. Altamiranda-Pérez,** es graduado en Ing. de Sistemas en 2002, MSc. en Ciencias de la Computación en 2006 y Dr. en Ciencias Aplicadas de la ULA. Venezuela. Realizo pasantías de investigación en la Université de Rennes I en Francia. Realiza trabajos de investigación en el área de minería de datos, computación inteligente, sistemas multiagentes, bioinformática, en el CEMISID de la ULA.
ORCID:0000-0003-0339-3244

**F.J. Díaz-Villarreal,** es graduado de Ing. de Computación en el año 2004 en la UVM, MSc. en Informática Aplicada IUPTJE en el 2010, tutor de la comunidad de aprendizaje masterdrez PROEA de la UPTMKR desde 2014 y estudiante del Doctorado en Ciencias Aplicadas en la ULA desde 2017. Actualmente, es Ingeniero Informático del Servicio Autónomo de Registro Notarias (SAREN).
ORCID: 0000-0002-3859-476X