ISSN 2007-9737

Sistema adaptativo para la generación de comportamientos emergentes en juegos serios emergentes

José Aguilar[1,2], Junior Altamiranda[1], Francisco Díaz[2]

[1] Universidad de los Andes, Facultad de Ingeniería, Venezuela

[2] Universidad EAFIT, Colombia

{aguilar, altamira, francisco.diaz}@ula.ve

Resumen: Un sistema de adaptación de videojuegos (SAV) para juegos serios emergentes (JSE), permite comportamientos emergentes en el juego, tales como la aparición de ambientes, eventos, narrativas y personajes, entre otros, con el fin de adaptarse al contexto en el que se esté desarrollando. En artículos anteriores se ha propuesto la arquitectura de un motor para JSE. Además, se ha propuesto un primer subsistema que permite la emergencia de un JSE según los objetivos del entorno, basado en el algoritmo de optimización de colonia de hormigas (ACO). En el presente trabajo, se especifica el segundo componente de dicha arquitectura, el SAV, el cual permite su adaptación dinámica (durante la realización del JSE). El SAV está compuesto por las subcapas de estrategias, secuencias y propiedades, que gestionan cada uno de esos tipos de emergencia posible en un JSE, con la intención de adaptarlo dinámicamente al contexto-dominio donde se está realizando el juego. Además, en este trabajo se analiza el comportamiento de dichas subcapas en un caso de estudio específico, mostrando resultados muy alentadores del SAV en el contexto educativo de un salón de clases inteligente (SaCI).

Palabras claves: Juegos serios emergentes, motor de juegos serios emergentes, sistema de adaptación de videojuegos, salón de clases inteligentes, sistema clasificador difuso, algoritmos culturales, algoritmo de optimización de colonia de hormigas.

Adaptive System for the Generation of Emerging Behaviors in Serious Emerging Games

Abstract: A video game adaptation system (SAV) for serious emergent games (JSE), allows emergent behaviors in the game, such as the appearance of environments, events, narratives and characters, among others, in order to adapt to the context in the one that is developing. In previous articles the architecture of a JSE engine has been proposed. Furthermore, a first subsystem has been proposed that allows the emergence of a JSE according to the objectives of the environment, based on the ant colony optimization algorithm (ACO). In the present work, the second component of said architecture is specified, the SAV, which allows its dynamic adaptation (during the JSE). The SAV is made up of the sub-layers of strategies, sequences and properties, which manage each of these types of possible emergencies in a JSE, with the intention of dynamically adapting it to the context-domain where the game is being played. Furthermore, in this work the behavior of these sublayers is analyzed in a specific case study, showing very encouraging results of SAV in the educational context of an intelligent classroom (SaCI).

Keywords: Serious emerging games, serious emerging game engine, video game adaptation system, smart classroom, fuzzy classifier system, cultural algorithms, ant colony optimization algorithm.

1. Introducción

Los JSE son la fusión de dos teorías, la teoría de juegos serios (JS), la cual establece que un juego tiene un propósito específico que puede estar relacionado con el aprendizaje o con la comprensión de un tema complejo [1]; y la teoría de juegos emergentes (JE), que comprende juegos cuyos comportamientos surgen a partir de las interacciones espontáneas de los elementos

vinculados al juego (tales como los jugadores, los personajes, etc.), sin leyes explícitas [2, 3, 4, 21]. Por ende, el concepto de JSE define juegos cuyo objetivo es diferente al de solamente jugar, el cual puede ser educativo, de entrenamiento, de rehabilitación, entre otros fines, y en particular, cuya dinámica va surgiendo en función de lo que va aconteciendo en el contexto [5].

Un Motor de JSE (MJSE) es un conjunto de programas que permiten la creación, la representación, la ejecución y la adaptación de un JSE [5]. Un MJSE debe poder ser usado por diferentes videojuegos, adaptándose a los requerimientos específicos del entorno. Particularmente, un MJSE está compuesto por un conjunto de submotores. En [5] se ha especificado un MJSE estructurado en capas, el cual, sigue el objetivo específico del JS, usando reglas sencillas para hacer emerger tácticas complejas, considerando al jugador (estudiante) y al entorno educativo.

Por otro lado, un SAV [6] es una parte específica del MJSE que se encarga de adaptar dinámicamente el JSE al entorno, con la intención de adecuarlo a sus necesidades y objetivos. En nuestro caso, el SAV posibilita el comportamiento emergente del JSE, tal que las diferentes formas de emergencia se puedan dar en el juego para su adaptación al entorno. El SAV está compuesto de subcapas, que gestionan diferentes formas de emergencia en un JSE.

A su vez, para diseñar el SAV se requieren de un conjunto de formalismos que le permitan generar los respectivos comportamientos emergentes en el JSE. Los comportamientos emergentes más importantes en un JSE son los de estrategias, secuencias y propiedades, los cuales le permiten adaptarse al contexto-dominio.

En este trabajo se especifican los tres componentes del SAV para el MJSE definido en [5], que le permiten soportar los comportamientos emergentes más importantes en un JSE. Para el comportamiento emergente de estrategias se propone diseñar un modelo genérico basado en reglas que las definan, y usar un sistema clasificador difuso (SCD) [7, 8] para instanciarlas en el JSE, adaptarlas a sus objetivos, y hacer emerger nuevas adecuadas al contexto del JSE. Por otro lado, para el comportamiento emergente de secuencias en un JSE se propone usar el mismo esquema propuesto en [6] para la emergencia inicial del JSE, el cual está basado en el algoritmo ACO; pero esta vez, su uso será para hacer emerger nuevas escenas en el actual JSE adecuadas a su contexto, según lo establecido en [22].

De igual forma, para hacer emerger propiedades se parte de los parámetros del JSE, los cuales son adaptados al entorno donde se realiza el JSE utilizando algoritmos culturales (AC) [9]. Así, en los tres casos, las emergencias que se producen responden a la adaptación del JSE al contexto.

A continuación, comentamos algunos trabajos de interés para nuestra propuesta. En cuanto a trabajos vinculados a permitir la emergencia en un juego, en [3] plantean un JE denominado Metrópolis, que parte de la premisa de que las ciudades se pueden auto-gestionar a partir de las decisiones tomadas en conjunto por sus habitantes (jugadores), sin que ninguno de ellos sea más importante que otro.

En ese juego emergen patrones urbanísticos en la ciudad. Recientemente, Metrópolis se ha extendido en [4] para permitir la emergencia de propiedades, debido a la adecuación de sus parámetros a los jugadores.

Por otro lado, en [10] se analiza la emergencia narrativa en un videojuego, utilizando diversos principios de la narratología, que ayudan en la emergencia del juego. En [11], los autores analizan la relación entre los elementos de las narrativas y las acciones en los juegos. Denominan “Disonancia Ludonarrativa” al desacuerdo entre la “Narrativa Acoplada”, que viene en la programación previa del juego, y la “Narrativa Emergente”, creada por el jugador a partir de las decisiones tomadas durante el juego. A partir de allí, construyen nuevos escenarios en el juego que aproximen ambas narrativas.

En el ámbito del uso de técnicas inteligentes en videojuegos, en [12] presentan un análisis de las técnicas de inteligencia artificial más utilizadas en los videojuegos, y particularmente, en juegos del tipo de simuladores. Entre las técnicas que señalan en dicho trabajo están los sistemas de clasificación de aprendizaje (LSC, por sus siglas en inglés), las redes neuronales, y los algoritmos genéticos, entre otros.

En [13] se utiliza la robótica de rehabilitación con el videojuego ReHabGame, para pacientes con trastornos neuromusculares.

El videojuego utiliza una interfaz basada en lógica difusa, para permitir al jugador controlar un avatar a través de un Kinect Xbox, el brazalete Myo y el pedal del timón. En [14] proponen una versión difusa de un juego gráfico que se centra en los objetivos de accesibilidad, denominado juegos de accesibilidad difusa (FRG). En un FRG, el objetivo del jugador es maximizar su valor de verdad al alcanzar un conjunto de objetivos dado, mientras que un otro jugador apunta a lo contrario.

El objetivo principal de este artículo es presentar los mecanismos que permitan la emergencia durante el uso de juegos serios en el SAV, basados en técnicas inteligentes, para el MJSE propuesto en [5, 6]. El artículo se organiza de la siguiente manera, en la sección 2 se describen los aspectos teóricos bases de este trabajo, tales como la arquitectura general del MJSE, los tipos de emergencia en un JSE, y las técnicas inteligentes usadas en este trabajo.

La sección 3 presenta el diseño del SAV para el MJSE propuesto en [5]. A continuación, se presenta un caso de estudio, y se compara el MJSE con otros trabajos similares, para finalmente, presentar las conclusiones.

2. Aspectos teóricos

2.1. Tipos de comportamientos emergentes en JSE

Los tipos de emergencia que se pueden dar en un JSE son los siguientes [4]:

- **Estrategias:** se generan nuevas logísticas (serie de acciones encaminadas hacia un fin determinado) y tácticas (procedimiento o método que se siguen para ejecutar algo), siguiendo las normas, leyes y reglas del videojuego. Estas emergencias no han sido diseñadas, creadas, ni predefinidas por el diseñador del juego, por ejemplo, la emergencia de estrategias de golpes, tácticas de combos de ataque, etc. en videojuegos de combate, es un ejemplo de lo anterior.
- **Secuencia:** se crean nuevas tramas (orden cronológico de diversos acontecimientos presentados a un jugador) o temáticas (contexto de su desarrollo) en los juegos, lo que puede implicar cambiar el ambiente del juego, los eventos que aparecen en su dinámica, entre otras cosas. Por ejemplo: cambio de escenarios o de época en juegos tipo "Los Sims".
- **Propiedad:** cambia las características y capacidades en los objetos, lo que puede conllevar a nuevos escenarios, personajes, etc. Eso puede implicar el cambio de normas, leyes y reglas en el videojuego, por ejemplo: jugar en sentido de las agujas del reloj en el dominó.
- **Final:** determina cuando debe terminar el videojuego, cambiando aspectos en el mismo. Algunas cosas que podrían definirse en este tipo de emergencia son: hacer emerger vidas infinitas, o finalizar el juego cuando se alcance un objetivo, entre otras cosas. Por ejemplo, en [4], al aparecer ciertos patrones de interés se podría dar por terminado el juego. En particular, en "Metrópolis", al aparecer patrones urbanísticos que determinan cuando se agota el espacio urbano, se daría por concluido el juego.
- **Modelo de Negocio:** según [4], tiene que ver con el surgimiento de modelos de servicios alrededor de los juegos. Por ejemplo, en algunos juegos aparece un sistema de comercio para comprar e intercambiar personajes, herramientas, entre otras cosas, como en "Top Gear" la compra de cauchos, de motor, etc.
- **Utilidad:** hace emerger como se va a utilizar el JSE, en función del contexto o la narrativa del ambiente donde se esté usando. Por ejemplo: "Era Mitológica" puede ser utilizado para explicar hechos históricos, geográficos o religiosos.

Todo lo anterior, surge como producto de un conjunto de reglas que rigen el JSE, que, en nuestro caso, el MJSE debe permitir manipular. Los tres primeros tipos de comportamientos emergentes son conocidos como *emergencia fuerte*, ya que pueden generar cambios profundos en el juego.

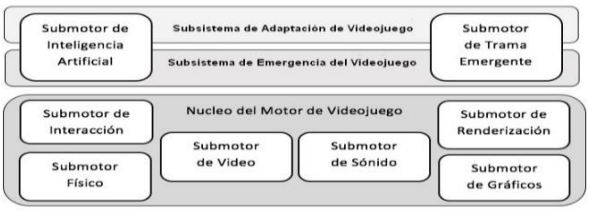

Fig. 1. Arquitectura del motor de juegos serios emergentes

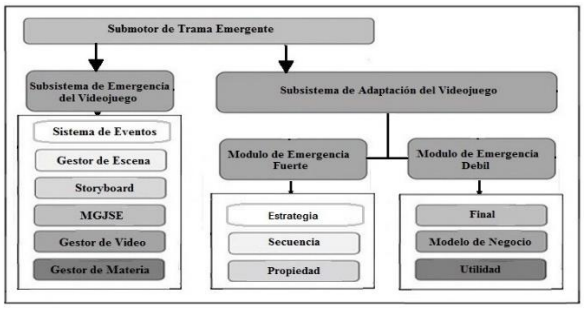

Fig. 2. Submotor de trama emergente

2.2 Arquitectura de un MJSE

La arquitectura del MJSE que se presenta en este trabajo, está basada en [5], la cual fue desarrollará para soportar JSEs en salones inteligentes (SaCI, ver [15, 16, 17] para más detalles de SaCI). El MJSE está basado en capas jerárquicas (ver figura 1), cuyos componentes son:

— **Núcleo del Motor de Videojuego (NMV):** es el elemento central del MJSE, en él se encuentran los seis submotores de base para cualquier videojuego, los cuales son: submotor de gráficos (SG), submotor físico (SF), submotor de sonido (SS), submotor de interacción (SI), submotor de video (SV) y submotor de renderización (SR). Más detalles de esos submotores en [5].

— **Subsistemas de Emergencia del Videojuego (SEV) y de Adaptación del Videojuego (SAV):** los cuales son las capas de la arquitectura MJSE que permiten hacer emerger un JSE. En específico, ambos subsistemas usan los siguientes submotores:

  a. **Submotor de IA (SIA):** se encarga de introducir comportamientos inteligentes en los diferentes componentes del JSE. Para ello, en este componente se despliegan las diferentes técnicas usadas de la IA para permitir la emergencia en un JSE. Un ejemplo de ello es ACO, que lo usa el Submotor de Trama Emergente (STE) cuando es invocado por el SEV, para definir la primera versión del JSE que se comienza a ejecutar.

  b. **Submotor de Trama Emergente (STE):** es el responsable de hacer emerger en el JSE las narrativas y las secuencias de las tramas adaptadas al contexto. Para ello, recolecta la información del contexto, realiza la gestión de escenas y eventos, ensambla subtramas/guiones de diferentes juegos, entre otras cosas. El STE, en el caso del SEV, permite hacer emerger la primera versión del JSE, según los objetivos que se deben cumplir con el JSE. El STE, en el caso del SAV, permite adaptar al JSE durante el desarrollo del mismo incorporando nuevas tramas.

El STE (ver figura 2), cuando es invocado por el SEV, está compuesto por los siguientes componentes (ver [6, 22], para más detalles del STE):

— **Gestor de Materia (GM):** determina la temática que se está tratando en el contexto para, a partir de allí, establecer el objetivo que debe cubrir el JSE

— **Gestor de Videojuegos (GV):** busca en repositorios de ***videojuegos*** (por ejemplo, edugame, advergame, etc.), subtramas o videojuegos para esa temática. Dichas subtramas/videojuegos son definidos como Recursos de Aprendizaje (RA). Para la búsqueda, compara los metadatos de los Ras en los repositorios con el definido por el GM. Si no consigue al menos un videojuego parecido a lo buscado, llama al módulo siguiente.

— **Módulo de Generación de JSE (MGJSE):** es el **responsable** del ensamblaje de un nuevo JSE usando las subtramas provistas por el GV. En un trabajo previo, se ha definido el MGJSE basado en ACO [6]. El MGJSE tiene imbricadas las funciones de los siguientes tres componentes del SEV, para generar inicialmente un JSE.

— **Storyboard (SB):** se encarga de generar los guiones narrativos o subtramas del JSE.

— **Gestor de Escenas (GE):** genera el ambiente, mundo o entorno, requerido por las tramas del JSE.

— **Sistemas de Eventos (SE):** se encarga de generar eventos especializados requeridos por el SB, para generar los comportamientos deseados en el JSE.

Los detalles del STE y SIA cuando son invocados por el SAV (ver figura 1), serán presentados en este trabajo.

2.3 Los sistemas clasificadores difusos (SCD)

Según [7, 8, 13], un sistema clasificador (SC) es un tipo de máquina de aprendizaje que se basa en reglas. Un SCD es un SC cuyas reglas o clasificadores son reglas difusas de la forma Si <condición> Entonces <acción>. De esta manera, la activación de una regla se logra cuando se cumplen las instancias de la <condición> de una regla. El peso de cada regla será un elemento a ser tomado en cuenta cuando se establezca el valor del crédito de las reglas, el cual está basado en el grado de activación de la regla. En específico, la importancia de una regla viene definida por la ecuación (1):

$$S_i(t+1) = S_i(t) + Acti_i(\mathrm{t}) - PS_i + R_i(t), \qquad (1)$$

dónde: *i* es el identificador de la regla; $Acti_i(\mathrm{t})$ es el grado de activación de la regla; $PS_i(\mathrm{t})$ es el pago dado por la activación de la regla; $R_i(t)$ es el pago recibido, definido como $\sum_{j=D} Pr * Act_j(t)$, tal que *D* es el conjunto de reglas que activó la regla *i* en el instante t; y Pr es la rata de pago, dada como un parámetro del SC.

La función de ajuste para la función de pertenencia de un conjunto difuso F, cuando el operador condición es "O", viene dada por [8]:

$$S_F(t+1) = S_F(t) + Acti_i * \mu_{F[x_k]}. \qquad (2)$$

Y cuando el operador condición es "Y", viene dada por [8]:

$$S_F(t+1) = S_F(t) + Acti_i * \frac{1}{\mu_{F[x_k]}}, \qquad (3)$$

dónde: $S_F(t)$es el valor de crédito de la función de pertenencia del conjunto difuso F en el tiempo t; $\mu_{F[x_k]}$es el grado de pertenencia del elemento x_k al conjunto difuso F presente en la <condición>.

La definición de las ecuaciones (2 y 3) permite asignar más crédito al conjunto difuso que influyo más en el nivel de disparo de una regla [7, 8]. El macroalgoritmo de un SCD para ajustar las reglas es:

Inicializar SCD
Repita Mientras *(existan eventos)*
Fusificar (evento)
Activar (SCD, evento)
Actualizar (SCD)

2.4 Algoritmos culturales

Según [9], los algoritmos culturales (AC), desarrollados por Robert G. Reynolds, son un complemento a los algoritmos de computación evolutiva (CE), los cuales han sido exitosos en la resolución de diversos problemas de búsqueda y optimización, en situaciones con poco o ningún conocimiento del dominio. En particular, los AC pueden mejorar su ejecución porque utilizan un conocimiento específico del problema para guiarlos en su resolución.

Están basados en que la evolución cultural puede ser vista como un proceso de herencia en dos niveles: en el nivel micro-evolutivo, que consiste en el material genético heredado por los padres a sus descendientes, y en el nivel macro-evolutivo, que es el conocimiento adquirido culturalmente a través de las generaciones, y que una vez codificado y almacenado, sirve para guiar el comportamiento de una población. En la figura 3 se puede apreciar cada uno de los componentes de los ACs.

En específico, los componentes de un AC son (figura 3):

– **Espacio de la Población:** está formada por individuos. Cada individuo tiene un conjunto de características, a las que les es posible determinar su aptitud. A través del tiempo, tales individuos podrán ser reemplazados por sus descendientes, obtenidos a partir de un conjunto de operadores aplicados a la población. Para ese espacio se definen:

Fig. 3. Estructura de los algoritmos culturales

a. **Función Objetivo:** permite evaluar el desempeño o comportamiento de cada individuo.
b. **Operadores Genéticos:** los cuales permiten la reproducción o modificación de los individuos en el espacio de la población. Los más comunes son los operadores genéticos de cruce y mutación.

– **Espacio de Creencias:** es donde se almacenan los conocimientos que han adquirido los individuos en generaciones anteriores. La información contenida en este espacio debe ser accesible a cualquier individuo, quien puede utilizarla para modificar su comportamiento. En ese espacio existen las siguientes categorías de conocimiento:

a. **Situacional:** toda la información del contexto: eventos y sus importancias, mejores/peores soluciones obtenidas en el pasado, combinaciones de valores ideales para una situación dada, etc.
b. **Normativo:** son comportamientos ideales, rangos de valores idóneos de cada una de las variables, o las gamas de comportamientos aceptables.
c. **Dominio:** conocimiento de los objetos del dominio, de las relaciones entre ellos, de los objetivos del dominio, entre otros conocimientos. Por ejemplo, almacena el conocimiento sobre el contexto donde se aplica el juego JSE.
d. **Histórico:** son patrones temporales, comportamientos históricos a resaltar, etc.
e. **Topográfico:** son patrones espaciales, características del espacio de soluciones, caminos de interés, etc.

– **Protocolo Comunicación:** las funciones de aceptación e influencia son los protocolos que permiten la interacción entre el espacio de creencias y el de la población. Estas funciones actualizan ambos conocimientos, de la siguiente manera:

a. **Función aceptación:** incorpora las experiencias individuales de un grupo selecto de individuos, que se obtiene de entre toda la población, para actualizar el conocimiento en el espacio de creencias.
b. **Función influencia:** determina como el conocimiento del espacio de creencia influye en los individuos de la población. Ejerce cierta presión, para que los individuos resultantes de la variación se acerquen a los comportamientos deseables, y se alejen de los indeseables, según la información almacenada en el espacio de creencias. El macroalgoritmo del AC es:

Inicio
t = 0 // Generar población inicial
Iniciar el espacio de creencias
Iniciar la población (posibles soluciones)
Repetir
Evaluar los individuos en la población
Reproducir los individuos usando los operadores genéticos y la función de influencia
Actualizar el espacio de creencia usando la función de aceptación
Seleccionar una nueva generación de individuos
Hasta (haber alcanzado la condición de finalizar)
Fin

2.5 Algoritmo de colonia de hormigas (ACO)

ACO es un tipo de metaheurística basada en una población, el cual está inspirado en la conducta de las colonias de hormigas reales cuando buscan comida, para resolver problemas de optimización combinatoria [6, 22]. ACO está compuesto por:

— **Espacio de solución:** es un grafo o espacio que recorrerán las hormigas para obtener soluciones.
— **Hormigas:** caminan en el grafo.

ISSN 2007-9737

— **Solución:** los nodos son marcados por una feromona en el grafo, igual que los arcos que los interconectan. Cuando converge el algoritmo ACO, los nodos son seleccionados según si su feromona pasa un umbral, igual que los arcos que salen de ellos, como parte de la solución final.
— **Feromonas:** define lo deseable de los nodos y de los arcos que los interconectan, para pertenecer a la solución final.
— **Función Feromona:** actualiza cada tipo de feromona, en función de la calidad de la solución propuesta.
— **Función Heurística:** define la decisión heurística que toma una hormiga, al estar en un nodo, con respecto a que otro nodo debe continuar a visitar después de él.

El macroalgoritmo clásico de ACO [2, 6] se muestra a continuación:

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

3. Subsistema de adaptación del videojuego

La capa SAV (ver figura 1) permite que en un JSE se generen comportamientos emergentes durante el juego, actuando sobre sus características de base. En particular, esta capa permite la adaptación de las características desus elementos, la emergencia de nuevas estrategias, secuencias de tramas, ambientes y eventos, en el videojuego. Para realizar esas tareas, se apoyará en el SIA y STE. En específico, los tipos de emergencia fuerte [4] de la sección II.A que soportará el SAV, serán definidos en esta sección.

3.1. Mecanismo para hacer emerger estrategias

Permite la emergencia generando nuevas variantes tácticas y estrategias en el juego, sin dejar de seguir las normas, leyes y reglas del mismo. El punto relevante, en este caso, es definir como se modelarán las tácticas y estrategias.

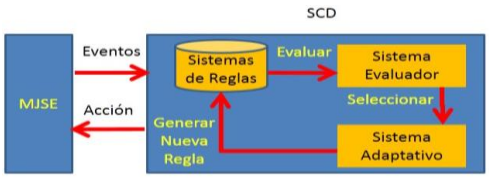

Fig. 4. Modelo de SCD

En nuestro caso, ellas serán definidas por reglas, en las cuales en el antecedente de la regla se establece que debe suceder (eventos que deben ocurrir), y en el consecuente las acciones que deben ocurrir en el juego dado esos eventos. También, es fundamental definir como se adaptarán las reglas, que en nuestro caso será usando SCD. En la figura 4 se presenta el modelo de SCD a usar, y su comportamiento en SAV es como sigue:

1. Cada uno de los submotores de MJSE genera eventos.
2. El Sistemas de Reglas (antecedente) recibe los eventos que van ocurriendo en el juego, los fusifica, y determina que reglas se activan.
3. El Sistema Evaluador (consecuente), según los eventos recibidos, evalúa el grado de activación de cada regla en el SCD, y determina la salida inferida del SCD.
4. El Sistema Adaptativo va adaptando las reglas, en función de sus comportamientos durante el juego (grado de activación). Aquellas más efectivas (más usadas), permiten alcanzar el objetivo del juego, por lo que perduran más tiempo, y son usadas por este sistema para generar nuevas reglas.

A continuación, se definen todos los elementos que permitirán caracterizar las tácticas y estrategias, e implementar el SCD, para permitir la emergencia de comportamientos en un juego:

a. Variables Difusas y Conjuntos Difusos: tanto los eventos de cada uno de los submotores, como las acciones en el JSE, serán definidas por las siguientes variables y conjuntos difusos:

i. Contexto JSE (CJ): representan variables que definen el tema del JSE [8] (CJx para [x=0,1,2,3…n]). Por ejemplo, en un JSE en el contexto educativo, suponiendo una clase

de matemática, las variables difusas podrían ser: suma, resta, multiplicación o división de números reales. Todas tendrían los mismos conjuntos difusos, los cuales serían: Ninguna (N), Escasa (E), Mediana (M) y Alta (A).

ii. **Evento Físico (EF):** representa eventos que ocurren en el juego (EF_x para [x=0,1,2,3…m]). Por ejemplo, un carro moviéndose, un personaje saltando, un futbolista jugando, el avatar tropieza una pelota, etc. Los conjuntos difusos considerados son: Verdadero (V), Más o menos (MM) y Falso (F).

iii. **Evento Acústico (EA):** representa tipos de eventos auditivos, como: cantar, gritar, etc. (EA_x donde [x=0,1,2,3…p]). Así, representa cada tipo de evento auditivo. Por ejemplo: el sonido de un rayo, el ruido al partirse un vaso, etc. Los conjuntos difusos considerados son: V, MM y F.

iv. **Evento Cámara (EC):** eventos que ordenan el movimiento, tanto en posición como en rotación, de la cámara que muestra el juego al jugador (EC_x para [x=0,1,2,3…r]). Por ejemplo: mirar hacia arriba o hacia abajo. El conjunto difuso considerado para estas variables son: N, E, M y A.

v. **Evento de Video (EV):** es cuando en el videojuego se utiliza una animación, efecto especial o video. Por ejemplo: cuando aparece un efecto especial de larga duración, la presentación en videoclip, etc. (EV_x, donde [x=0,1,2,3…q] representa los tipos de eventos videos). Los conjuntos difusos considerados son: V, MM y F.

vi. **Acción de Movimiento (AM):** se encarga de pedir al MJSE que aplique un movimiento sobre al avatar (AM_x para [x=0,1,2,3…s]). Sus conjuntos difusos son: N, E, M y A.

vii. **Acción de Destreza (AD):** son teclas especiales del videojuego que varían según el tipo de habilidad a permitir: salto, agachar, abrir, cerrar, agarrar, soltar, etc. o cualquier otra actividad que este contemplada en el JSE (AD_x para [x=0,1,2,3…t]). Normalmente, son scripts básicos predeterminado. Por ejemplo: disparar en Contar Strike, saltar en Mario Bros, patear en FIFA. Los conjuntos difusos considerados son: N, E, M y A.

viii. **Acción de Avanzada (AA):** define una acción que es disparada por un algoritmo de inteligencia artificial después que haya una solución (AA_x [x=0,1,2,3…v]). Por ejemplo, después de utilizar los arboles de comportamiento (BT) en Halo 2, o min-max en ajedrez, etc. El conjunto difuso considerado para estas variables son: N, E, M y A.

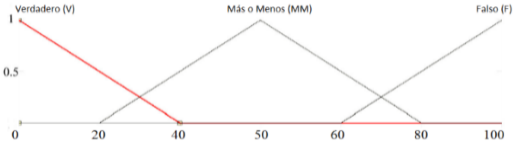

Fig. 5. Funciones de pertenencia de la variable difusa

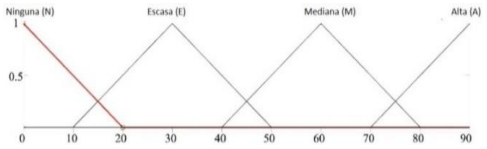

Fig. 6. Funciones de pertenencia de la variable difusa

b. Funciones de Pertenencia: a continuación, se definen las funciones de pertenencia de cada uno de los conjuntos difusos asociados a cada variable difusa. Se propone, de manera general, el uso de funciones de pertinencia del tipo trapezoidal.

Las variables difusas EF, EA y EV, pueden estar caracterizada por la función de pertenencia mostrada en la figura 5. El universo de discurso [0%, 100%] determina el grado de ocurrencia de ese evento en un momento dado.

Las variables difusas CJ, AD, EC, AM, AD y AA se caracterizan por la función de pertenencia mostrada en la figura 6. El universo de discurso es [0%, 100%], por las mismas razones anteriores.

c. Reglas de Control Genéricas: a continuación, se presentan un grupo pequeño de reglas asociadas con las variables difusas, que en otro artículo futuro se extenderá, las cuales se dividen según el tipo de estrategia de juego:

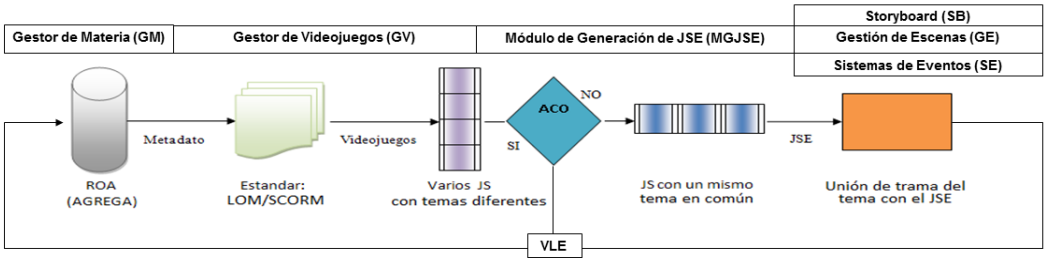

Fig. 7. Componentes del STE

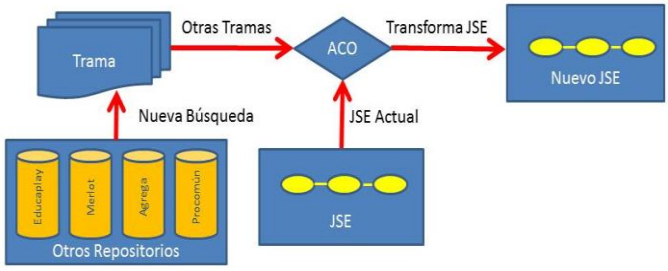

Fig. 8. Modelo de cambio de los escenarios

i. JSE de agilidad: son juegos de saltos y poderes, como por ejemplo, Mario Bros, Donkey Kong y otros. Un ejemplo de sus posibles reglas serían:

Si <EF=hueco> entonces <AM=saltar>
i <EF=enemigo> entonces <AD=disparar>

ii. JSE de velocidad: son juegos de manejo de algún vehículo, como por ejemplo: carro, moto, avión, barco, bicicleta, etc. Un ejemplo de sus posibles reglas sería:

Si <EF=chocar>y <EA=gritar>entonces <AA=proteger>

Si <EF=chocar>o <EF=parar> entonces <AM=parar>

Existen muchos más grupos de reglas de estrategias, vinculadas a JSE de lucha, rompecabezas, entre otros.

d. Comportamiento de las reglas difusas (instancias): a continuación, se dan ejemplos de algunas de las instancias de las reglas de control genéricas, que podrían definir las estrategia y táctica en un JSE dado:

Si EF=hueco es V y EF=enemigo es V entonces AM=saltar es A y AD=disparar es A

Esta regla indica que, si un evento físico es un hueco y otro es un enemigo, entonces se realizan las acciones saltar y disparar al mismo tiempo.

Si (EF=chocar es V o EA=grita es MM) entonces AA=proteger es A.

En esta regla se establece que, si ocurre un evento físico de chocar y más o menos un evento auditivo, entonces se invoca a una técnica de inteligencia artificial que permita establecer la estrategia de protegerse.

3.2. Mecanismo para hacer Emerger Secuencias

Esta emergencia se caracteriza por crear nuevas tramas, o cambiar el orden en las actuales, en los juegos. Este tipo de emergencia utiliza el algoritmo ACO y el sistema de búsqueda en repositorio (ver figura 7).

En particular, para esta emergencia se requiere determinar la brecha entre lo que está aconteciendo en el JSE y lo requerido cuando se diseñó el JSE. Con esa información, se determinan las necesidades aun por cubrir por el JSE, y basado en ello, se busca en los repositorios de tramas las nuevas que puedan ser de interés.

Con ellas, se procede a invocar al algoritmo ACO, el cual usa como base el esquema actual de tramas del JSE y las nuevas tramas conseguidas, para intentar transformarlo, o proponer uno completamente nuevo (ver figura 8).

ACO está compuesto en la emergencia de secuencia por:

— **Espacio de solución:** 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. Pero además, el grafo incluye un subgrafo que representa el actual JSE en ejecución.

— **Hormigas:** caminan en el grafo de subtramas, para seleccionar las escenas.

— **Solución:** es la ruta del grafo final que posee más feromona en los arcos y nodos, tal que los nodos representan las subtramas nuevas del JSE

— **Feromonas:** hay dos tipos, una para las subtramas (nodos) y otra para los arcos entre las subtramas.

— **Función Feromona:** actualiza cada tipo de feromona en función de la calidad del nuevo JSE.

— **Función Heurística:** define cual subtrama (nodo) se debe tomar a medida que se va construyendo la solución. Para el uso de ACO en el proceso de emergencia de secuencia, se debe considerar lo siguiente:

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 figura 3). Además, se incluye el subgrafo que representa al actual JSE en ejecución. Por otro lado, 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.

b. Construcción de la solución por parte de las hormigas (Construir Soluciones por Hormigas): En esta fase, algunas consideraciones se realizan:

i. Se define el número de hormigas que integran la colonia.

ii. Cada hormiga inicialmente se coloca de modo aleatorio en el grafo para iniciar su recorrido, y determina un JSE (una solución).

Cada hormiga ejecuta una función heurística (o de transición) desde el nodo actual donde se encuentra, para determinar el próximo nodo que visita 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 r 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:

$$\mathrm{P}_{(r,s)}^{k} = \frac{\gamma_{(r,s)} \cdot \eta_{(s)}}{\Sigma_{\mathrm{u}\,\epsilon\,\mathrm{J}_r^k} \gamma_{(r,u)} \cdot \eta_{(u)}} \text{ Si } s \,\epsilon\, J_r^k \,. \tag{4}$$

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 la siguiente regla:

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

Ahora bien, en el grafo inicialmente se ponderan los nodos de las nuevas tramas y sus arcos de interconexión, de acuerdo a sus similitudes con los objetivos no cubiertos por el actual JSE, de tal manera de darles una posibilidad de ser seleccionadas por las hormigas, y poder competir así con los nodos y arcos del subgrafo que representa el actual JSE.

c. 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 (recorrido de cada

hormiga), según cómo se definió en [6, 22]. En ese sentido, cada hormiga actualiza 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 a los objetivos no cubiertos del tema deseado (definido por GV). El nivel de similitud del JSE propuesto por una hormiga k (η^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:

$$\gamma_{(r,s)} = \gamma_{(r,s)} + \Delta\gamma_{(r,s)}, \quad \gamma_{(r)} = \gamma_{(r)} + \Delta\gamma_{(r)}, \tag{5}$$

donde, Δγ (r, s) y Δγ (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):

$$\gamma_{(r)} = \frac{\Sigma_{k=1}^{M}\Delta\gamma_{(r)}}{M} \qquad \gamma_{(r,s)} = \frac{\Sigma_{k=1}^{M}\Delta\gamma_{(r,s)}}{M} \tag{6}$$

En general, tanto $\Delta\gamma_{(r)}^{k}$ como $\Delta\gamma_{(r,s)}^{k}$ son iguales a η^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:

$$\gamma_{(r,s)} = (1-\rho) + \Delta\gamma_{(r,s)}, \quad \gamma_{(r)} = (1-\rho) + \Delta\gamma_{(r)}, \tag{7}$$

donde, ρ ∈ (0,1] es el coeficiente de evaporación de feromona. Este proceso se realiza repetidamente, hasta que la colonia converge en un grupo de soluciones (JSE).

d. Construcción de la solución final (Construir Solución Final): una vez que la colonia concluye su trabajo, se debe pasar a construir la solución final, es decir, la nueva versión del 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 nuevo JSE propuesto). Este procedimiento es muy parecido al definido en [6].

Tabla 1. Estructura interna de un individuo en el Espacio de Población

JSE_i	P_1	P_2	P_j	FO_i

3.3. Mecanismo para hacer Emerger Propiedades

Esta emergencia cambia las características en los objetos, normas, leyes y reglas que tiene el videojuego, basado en un proceso adaptativo de las propiedades emergentes. Para ello, se considera los siguientes aspectos:

- ¿Cuáles elementos del JSE se pueden parametrizar, tal que dichos valores se puedan adecuar al contexto?
- ¿Cómo desde la interacción de los jugadores, se pueden establecer mecanismos de aprendizaje de esos parámetros?

En este caso, se propone utilizar AC de la siguiente manera:

1. **Espacio de Población:** en nuestro caso, los individuos son las múltiples ejecuciones de un JSE dado por grupos de jugadores, los cuales pueden o no ser los mismos cada vez. En específico, un individuo es definido por los valores de los diferentes parámetros que se usaron en ese JSE. En la tabla 1, JSE_i representa el individuo *i*, y P_j los valores de los j parámetros usados en ese juego la *i* vez que se jugó.

Por otro lado, para generar nuevos individuos se usa el operador de cruce [20]; pero además, un operador de mutación guiado por el conocimiento en el espacio de creencia. Este operador es muy importante, porque es el que explota el conocimiento cultural.

2. **Función Objetivo:** permite evaluar el

Tabla 2. Representación del Conocimiento Situacional

P_i	Vj	IO_j	C_j

Tabla 3. Representación del Conocimiento Normativo

P^1		P^2		P^u	
LI	LS	LI	LS	LI	LS
$_1$	$_1$	$_2$	$_2$	$_u$	$_u$

Tabla 4. Representación del Conocimiento Dominio

D	F (	P^1	P^2	P^u
	i	valor ideal	valor ideal	valor ideal

Tabla 5. Representación del Conocimiento Histórico

$Evento_k$	P^1	P^2	P^u	FE_k

desempeño del JSE, según el objetivo para el cual se diseñó el mismo. En nuestro caso, el mejor individuo será aquel que maximice esa función objetivo, definida por la Ec. (8):

$$FO = \Sigma_{i=1}^{n} (a * PA_i - b * LE_i)/\mathrm{n} \quad (8)$$

donde, *a* y *b* son constantes definidas por el usuario, que permiten normalizar las unidades en la función. El número de ejecuciones de ese individuo con esos parámetros en el JSE es *n,* la cantidad de puntuación acumulada (suponiendo que se asocia a si se alcanzan los objetivos del JS) en el JSE cada vez *es* PA_i y el lapso de tiempo que duro el JSE cada vez es LE_i, el cual puede estar basado en el número de rondas, tiempo, entre otros, para determinar quién es más rápido o lo hace en menos movimientos.

3. **Espacio de Creencias:** En el espacio de creencias se tienen cinco categorías desconocimiento, pero en este trabajo solo se usan cuatro:

   a. **Conocimiento Situacional:** contiene ejemplos de éxitos de los JSE, pero en particular, los valores (V_j) de los parámetros (P_i) en dichos juegos, con sus índices de ocurrencia (IO_j) a través de los diferentes juegos. Además, la calidad promedio de ese valor de parámetro (C_j), determinada como el promedio de la calidad de los JSEs donde se usó ese parámetro (ver tabla 2).

   b. **Conocimiento Normativo:** define los rangos de valores idóneos de cada uno de los parámetros del JSE. En la tabla 3, LI y LS son los límites inferiores y superiores de cada parámetro P_i.

   c. **Conocimiento de Dominio:** en nuestro caso, indica los contextos donde se puede aplicar el JSE, con los valores de parámetro adecuados para cada uno de ellos. En la tabla 4, D_i representa el conjunto de parámetros (P_j) de un JSE en el dominio *i*, y la función de calidad (FC_i) es calculada como el promedio de calidad de las veces que se ha usado el JSE en ese dominio.

   d. **Conocimiento Histórico:** son patrones temporales del comportamiento de los JSEs. Es un conocimiento que explota el comportamiento histórico del juego, en nuestro caso, basado en eventos de interés que debe incluir el JSE. Para ello, se establece un vector del tipo evento, que establece que valores ideales deben tener los parámetros P_j para que el evento k ocurra, y la función FE_k establece la calidad de esos valores para generar ese evento (tabla 5).

4. **Protocolo Comunicación:** dictan las reglas para hacer interactuar a la población de JSEs con el espacio de creencias.

   a. **Función de Aceptación:** según Reynolds [9, 19], con un 20% de los individuos de la población, es suficiente para nutrir con sus experiencias el espacio de creencias. Este trabajo sigue ese criterio al usar las funciones de aceptación. Además, las funciones de aceptación actualiza los diferentes valores de calidad en los diferentes tipos de conocimiento: en el conocimiento situacional, los valores de C_j e IO_j. $\forall_j$=1…u; en el normativo, los valores de LI^j y LS^j, $\forall$j=1…u; en el conocimiento de dominio, se puede actualizar porque hay un nuevo dominio D_s, o porque los valores ideales de los parámetros han cambiado, o porque FC_s ha cambiado

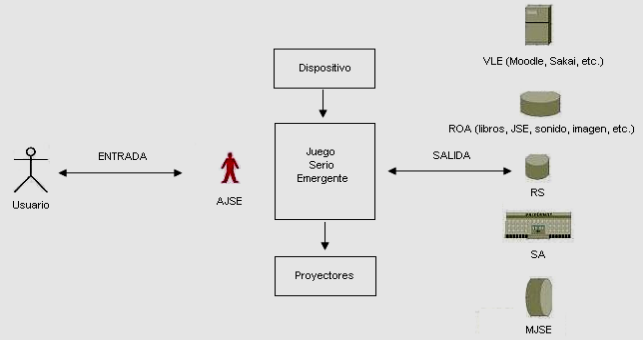

Fig. 9. Modelo conceptual de un AJSE en un SaCI. (Fuente [18])

Fig. 10. 3 videojuegos de fracciones utilizando domino (Fuente [6])

para los actuales valores ideales. Finalmente, en el conocimiento histórico, pueden cambiar los valores ideales de los parámetros, o el valor FE_k para los actuales valores ideales. La manera de actualizar en cada caso será diferente.

En el caso del *conocimiento situacional*, se actualiza de la siguiente manera: IOj es actualizado mediante la siguiente ecuación:

$$IOj = NOj + IOj, \quad (9)$$

donde, NOj es el nuevo número de ocurrencia del valor Vj en la actual generación. A su vez, Cj, también es necesario actualizar, para eso se emplea la siguiente ecuación:

$$C_j = C_j * \bar{m} + \bar{C_j} * m, \quad (10)$$

donde, $\bar{m}$ es el complemento del momento, es decir, (1 - m), y $\bar{C_j}$ es el promedio del valor C_j de todos los individuos dentro del 20% aceptados proveniente de la población actual con el valor de Vj. Finalmente, m es el momento, que viene dado por la ecuación:

$$m = \mu / t. \quad (11)$$

Tal que μ es una constante de momento entre 0 y 1, y t es el número de generaciones (t = 1, 2, 3). De esta manera, cada vez que llega una nueva experiencia de la población, C_i se actualizan.

En el caso del *conocimiento normativo*, los valores de LIj y LSj se actualizan de manera similar, según la siguiente ecuación:

$$Lac(P^u) = [(Lv * \bar{m} + \bar{P} * m) / 2], \quad (12)$$

donde, $Lac\ (P^u)$ es el límite actual (ya sea LI o LS) del parámetro P^u, Lv es el límite anterior, $\bar{P}$es el promedio del valor del límite de todos los individuos, dentro del 20% aceptados proveniente de la población actual.

De esta manera, cada vez que llega una nueva experiencia de la población, los límites de cada parámetro se actualizan.

En el *conocimiento de dominio*, al aparecer un nuevo dominio *i* (D*i*), se actualiza la matriz de la tabla 4 con una nueva fila. Si los valores ideales para un dominio ya existente *i* han cambiado, se sustituyen en la tabla, y se coloca su valor de FC_i. En particular, los valores ideales para un dominio i son los que maximizan la FO de la Ec. (8). Por otro lado, si solo se debe actualizar el valor de la función calidad (FC*i)* de los actuales valores ideales, se debe usar la siguiente ecuación:

$$FCi = FCi * \bar{m} + \overline{FC_i} * m\,, \quad (13)$$

donde, FC_i es la función de calidad actual; $\overline{FC_i}$ es el promedio del valor FC_i de todos los individuos dentro del 20% aceptados proveniente de la población actual en el dominio Di.

Finalmente, en el *conocimiento histórico*, si los valores ideales de los parámetros Pi para un evento k deben cambiar, simplemente se sustituyen en la tabla 5. En este caso, los valores ideales para un evento k son los que maximizan la FO de la Ec. (8). Ahora bien, si solo se debe actualizar el valor de FEi de los actuales valores ideales, se debe usar la siguiente ecuación:

$$FE_i = FE_i * \bar{m} + \overline{FE_i} * m\,, \quad (14)$$

donde, $\overline{FE_i}$ es el promedio del valor FE_i de todos los individuos dentro de los 20% aceptados proveniente de la población actual con el evento k.

b. **Función de Influencia:** se establece usar el conocimiento almacenado en el espacio de creencias para realizar operaciones de mutación guiada en el espacio poblacional. El *conocimiento situacional* permite una mutación directa a los mejores valores Vj de un parámetro Pi dado; el *conocimiento normativo* permite reajustar los rangos de un parámetro Pi; el de *dominio* usa los valores ideales de P_j para el dominio en el que se quiere diseñar el JSE, y el *histórico* usa los valores ideales de P_j, según el evento k que se desee que aparezca.

4. Caso de estudio

4.1 Contexto de aplicación: SaCI

Se considera un Salón de Clases Inteligente (SaCI) como el propuesto en [15, 16], tal que en el aula inteligente todos sus componentes son modelados usando el paradigma de Sistemas Multiagentes (SMA), el cual caracteriza a sus dispositivos de hardware (pizarra inteligente, laptop, tableta, smartphone, etc.) y de software

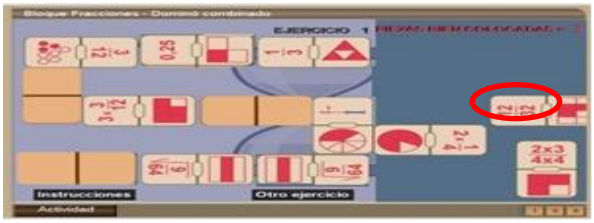

Fig. 11. El videojuego de domino “Combinado”

Tabla 6. Estado de Agente Difuso

	Variables	Valores posibles
Entradas		
Evento	CJ	fracciones
	EF	movimientos de los números (dominós)
Salida		
Acciones	AA	realizar un cálculo
	AM	moverse

Tabla 7. Ejemplo de Reglas Genéricas

N°	Regla
R1	Si <CJ=matemáticas> entonces <AA=calcular>
R2	Si <AA=calcular> entonces <EF=dato>
R3	Si < EF =dato>entonces <AM=mover>

(Entorno Virtual de Aprendizaje, Sistema Académico, etc.) como agentes.

SaCI utiliza un middleware reflexivo autonómico para entornos de aprendizaje inteligente en la nube, llamado AmICL, propuesto en [15, 16, 17]. Este middleware reflexivo autonómico está basado en SMA, y posibilita el despliegue de las diferentes comunidades de agentes de SaCI.

En particular, uno de esos agentes es el Agente Juego Serios Emergentes (AJSE), el cual gestiona los JSE de forma autónoma. El AJSE, una vez recolectada la información del entorno de SaCI (perfil de los estudiantes, objetivos del actual proceso de aprendizaje, etc.), adapta el JSE a SaCI, llamando al GM.

Para ello, el AJSE interactúa con los agentes de SaCI: Gestor del Repositorio de Objeto de

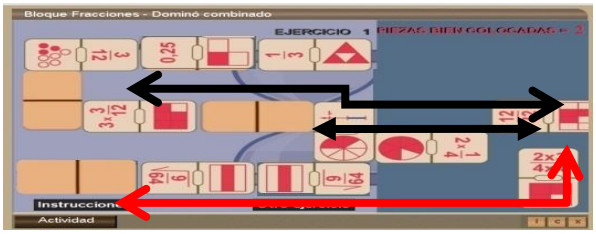

Fig. 12. Comienzo

Fig. 13. Coloca Domino (12/32 | 3/8)

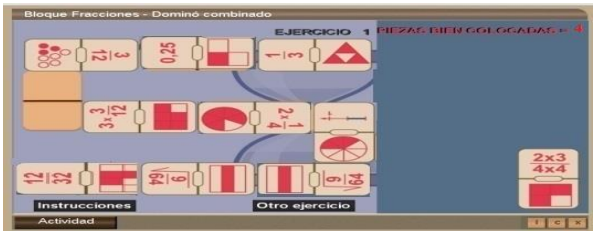

Fig. 14. Colocación del Domino 1/6 | 1/2

Fig. 15. Coloca Domino 2x3/4x4 | 1/4

Fig. 16. Cinco JSE seleccionados de Eduplay

Aprendizaje (ROA), Sistema Académico (SA), Sistema Recomendador (RS) de Recursos Educacionales y el Entorno Virtual de Aprendizaje (VLE, por sus siglas en inglés) (ver figura 9) [18].

4.2 Contexto dinámico pedagógico para analizar el comportamiento del SAV

Una vez en el SaCI, 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. En [8], se muestra cómo se genera inicialmente el JSE usando los componentes del MJSE, para explicar la clase de fracciones. En particular, se usa el repositorio de objetos de aprendizaje agrega (ver, http://www.proyectoagrega.es/), y se seleccionan tres videojuegos del domino en fracciones compatible con el tema de aprendizaje (ver figura 10).

Se supone que inicialmente el SEV propone como JSE inicial el videojuego domino "Combinado" (ver figura 11):

El videojuego de domino "Combinado" consiste en que cada uno de los dominós tiene dos zonas opuestas entre sí, ya sea izquierda (arriba) o derecha (abajo). Estas zonas van a tener una figura, fracción, porcentaje, etc., que va a dar un resultado de fracciones matemáticas, el cual debe ser igual al espacio vacío adyacente a alguna pieza de domino. Por ejemplo, en la figura 11 el círculo rojo señala la zona izquierda del domino que tiene como valor 12/32, este debe ser colocado en una zona vacía que tenga el mismo resultado, que es $\sqrt{9/64}$ = 0,375, que se encuentra en el espacio vacío abajo a la izquierda.

A continuación, se explica cómo SaCI usa el SAV para cada tipo de emergencia:

Adecuación por Emergencia Fuerte de Estrategia:

a. **Eventos y Acciones:** mediante el SCD se establecen los diferentes tipos de eventos y acciones que pueden suceder en el juego (ver tabla 6), con sus respectivos valores. Por ejemplo, en el caso de CJ, el contexto educativo es utilizar fracciones de las matemáticas. En el caso de EF, los movimientos de los números (en este caso, los dominós). En el caso de las acciones, las

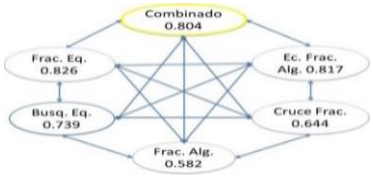

Fig. 17. Grafo del JSE inicial con los 5 JSE de Eduplay

Tabla 8. Listados de pruebas difusas para el domino12/32│3/8

EF		Dato= 12/32│3/8	AA
A	**1**	(3/8=12/32)=F y (3/8=3x(3/12))=F	Falso
	2	(**3/8=3/8)=V** y (12/32=3x(3/12))=F	Falso
B	**1**	(1/6=12/32)=F y (3/8=1/2)=F	Falso
	2	(1/6=3/8)=F y (12/32=1/2)=F	Falso
C	**1**	**(null=12/32)=MM y (3/8=$\sqrt{9/64}$)=V**	**Verdadero**
	2	(null=3/8)=MM y (12/32=$\sqrt{9/64}$)=V	Verdadero

Tabla 9. Listados de pruebas difusa para 1/6│1/2

EF		Dato= 1/6│1/2	AA
D	**1**	(1/6=2x1/4)=F y (1/6=1/2)=F	Falso
	2	(1/6=1/6)=V y (2x1/4=1/2)=V	Verdadero
E	**1**	(3/8=1/6)=F y (2x1/4=3x3/12)=F	Falso
	2	(3/8=2x1/4)=F y (1/6=3x3/12)=F	Falso

Tabla 10. Listados de pruebas para 2x3/4x4│1/4

E		Dato= 2x3/4x4│1/4	AA
J	1	(03/8=2x3/4x4)=V y (1/4=3x(3/12))=V	Verdadero
	2	(3/8=2x3/4x4)=F y (1/4=3x(3/12))=F	Falso

Tabla11. Nueva Regla Generada por el SCD

N°	Regla
R4	Si CJ=matemáticas es A y AA=calcular es A entonces EF=dato es V y AM=mover es A

posibles son realizar un cálculo AA o moverse AM.

b. Estrategias: se establecen los tipos de reglas para el contexto del JSE (según CJ). Como es resolver fracciones matemáticas, se deben establecer reglas genéricas alrededor de esa temática. La tabla 7 muestra algunos ejemplos de dichas reglas. La regla 1 establece que por el dominio, entonces se debe realizar una acción de calcular. La regla 2 indica que después de calcular, entonces ocurre el evento EF=dato. Finalmente, la regla 3 indica que al llegar ese evento, entonces ocurre la acción AM=mover ocurre.

c. Ejecución del JSE: a continuación, se muestra parte del desarrollo del JSE:

1er paso: se activa la regla R1, tal que la primera acción es AA=calcular, la cual puede ser: sumar, restar, dividir, etc.

2do paso: se activa la regla R2, que es AA=calcular el primer domino de arriba hacia abajo, con el dato=12/32│3/8, comparando con los datos de los 3 espacios vacíos que se observan en la figura 12.

En este caso, se generan los EFs= EF_{a1}, EF_{a2}, EF_{b1}, EF_{b2}, EF_{c1}, EF_{c2} y la información generada se usa de diferentes maneras; por ejemplo: la zona izquierda (12/32) del domino se compara con la figura (3/8) de la parte de arriba del espacio vacío que señala la flecha a, y (3/8) de la zona derecha del domino se compara con 3x(3/12) de la zona de abajo del espacio vacío de la flecha a, dando como resultado que es falso=F, como se muestra en la tabla 8, y así sucesivamente para cada domino y espacio vacío (fila de la tabla 8).

3er paso: en la tabla 8, generada por R2, se observa que existen 6 posibles soluciones para el domino con el dato =12/32│3/8, pero solo 2 validas (c1 o c2). Con las validas, se dispara la regla R3, para mover el dato =12/32│3/8 según lo permitido por EF_{c1} o EF_{c2}, con lo cual, revisa todas las opciones, colocando el domino en 2 posiciones posibles 12/32│3/8 o viceversa 3/8│12/32, que es el mismo resultado de la zona izquierda del domino colocado en la mesa $\sqrt{9/64}$ = 12/32 = 3/8 = 0,375, tanto en el espacio derecho como izquierdo y null (vacío), dando como resultado más o menos = MM, para luego moverse como indica la flecha roja en la figura 13 (se escogió

Tabla 12. Metadatos

LOM	Tema Deseado	Puntuación
Title	Fracciones Equivalentes (VLE)	1
Language	es (SA)	1
Description	fracciones (VLE)	0.91
Keyword	fracciones (VLE)	1
Coverage	universal (SA)	1
Format	javascript, html 5 o flash (ROA)	1
typicalAgeRange	15 (SA)	0,42
Difficulty	very high (SA)	1
Duration	30 minutos (VLE)	0,43
interactivityLevel	very high	1
semanticDensity	very high (SA)	0,53
intendedEndUserRole	learner(SA)	1
Context	schoolmate (SA)	1
cognitiveProcess	practise (SA)	1
Total		**12,39/15=0,826**

Tabla 13. Población inicial

Ind	P_1	P_2	P_3	P_4	**FO**
JSE_1	5	5	10	10	FO_1
JSE_2	2	1	2	8	FO_2
JSE_3	4	5	10	6	FO_3

una de las dos soluciones al azar).

Siguientes Pasos: se evalúan nuevas iteraciones que aparecen en el juego, debido a los movimientos en el mismo. Por ejemplo, después del último movimiento, la regla R2 genera la tabla 9.

Como se observa en la tabla 9, existen 4 posibles soluciones (EF_{d1}, EF_{d2}, EF_{e1} y EF_{e2}), con una sola valida EF_{d2}.

Posteriormente, se dispara la regla R3, para realizar el siguiente movimiento según EF_{d2} , donde la flecha roja indica hacia donde se movió la pieza, quedando como se observa en la figura 14.

En la siguiente iteración, de nuevo se vuelve a activar R2, generando la tabla 10. Como se observa en la tabla 10, existen 2 posibles soluciones (EF_{j1}, EF_{j2}) con una sola valida EF_{j1}.

Al final, se realiza el último movimiento según lo establecido por EF_{j1}, moviéndose en la pieza en dirección que apunta la flecha roja, en la figura 15.

d. Actualización de la regla: basado en las reglas R1, R2 y R3, el SCD puede producir un proceso de generación de nuevas reglas (estrategias). Por ejemplo, la tabla 11 muestra una posible nueva regla R4 generada con ellas.

Una vez actualizada las reglas, se iniciaría un nuevo ciclo (iteración) del SCD para validar las estrategias definidas en el JSE.

Adecuación por Emergencia Fuerte de Secuencia

Si se da el caso de que el JSE de domino "combinado" no cumple con las expectativas del profesor que está explicando la clase en el SaCI, entonces se podría utilizar la emergencia fuerte por secuencia. Para ello, se busca en otros repositorios de aprendizaje, nuevas tramas o JSE, que permiten explicar de mejor forma las fracciones en la clase de matemáticas del SaCI. Debido a todo esto, se realizan los siguientes pasos:

a. Seleccionar nuevos repositorios: en este caso de estudio se escoge (http://www.educaplay.com), eligiendo varios JSE que pueden ser utilizados para el tema, como se observa en la figura 16.

b. Búsqueda de nuevas tramas: se extraen los metadatos de los JSE o tramas seleccionadas, como se observa en la tabla 12, en la cual se da un ejemplo de un metadato para una de las tramas.

c. Generación del nuevo JSE: se construye primero un subgrafo que representa las tramas del JSE inicial "Combinado", al cual se le agregan las nuevas tramas de los 5 JSE nuevos de eduplay de la figura 16, para conformar el grafo que se usa para construir un nuevo JSE (figura 17).

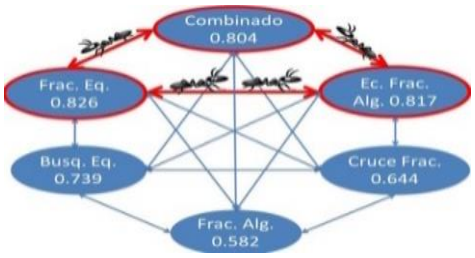

Fig. 18. Nuevo JSE

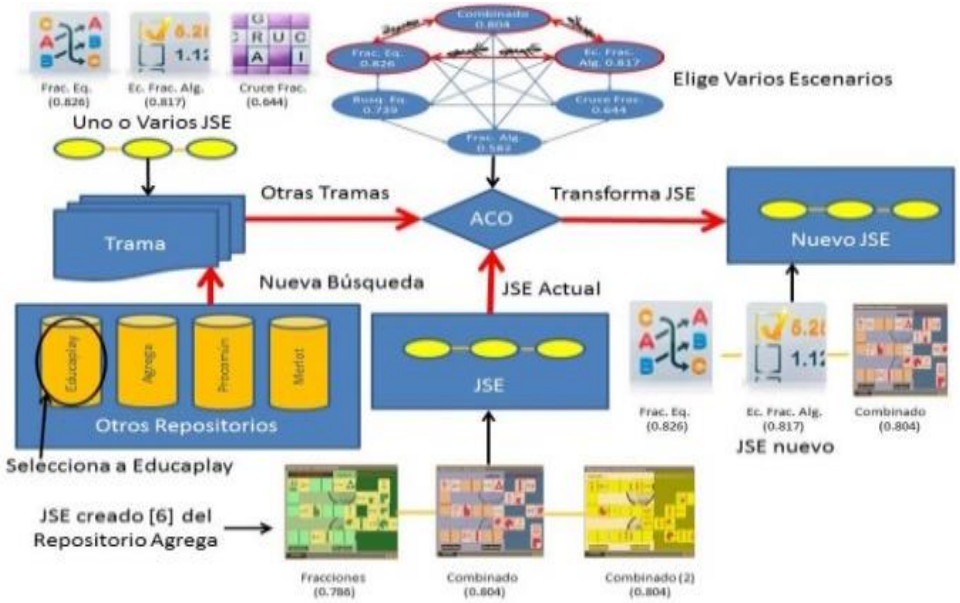

Fig. 19. Síntesis de los pasos del JSE

En ese grafo, se interconectan todos los nodos de las tramas, con las subtramas del JSE inicial, para formar el grafo que ACO usa en su proceso.

A partir del grafo inicial, empieza a actuar el algoritmo ACO (ver figura 18). El proceso que se sigue es el mismo explicado en [6, 22].

La figura 18 muestra el JSE nuevo generado, el cual está conformado por tres subtramas. Una subtrama es “Fracciones Equivalentes”, la cual posee 0.826 de puntuación de metadatos, le sigue “Ecuación con Fracciones Algebraicas” con 0.817 y el Domino “Combinado” con 0.804 se observa que sigue utilizando este domino por su alta puntuación.

La figura 19 sintetiza los pasos seguidos el proceso de emergencia por secuencia, antes narrados.

Adecuación por Emergencia Fuerte de Propiedad:

Determina los valores idóneos de los parámetros del JSE. Para determinar el SAV sigue las siguientes fases:

a. **Población inicial:** son las diferentes ejecuciones de este JSE, dado por diferentes grupos de estudiantes en el SaCI. Cada ejecución del juego se realiza con un conjunto de valores específicos de los parámetros del JSE. A partir de las ejecuciones del JSE, se optimizan sus parámetros. Para este JSE, sus parámetros son: P1= fichas iniciales que tiene el jugador, P2=fichas iniciales en el tablero, P3= penalización por mala jugada, y P4= cuantas rondas se juegan en total. La tabla 13

muestra la población inicial, suponiendo 3 estudiantes que jugaron. Las figuras 20 y 21 muestran un momento del juego, para JSE_1 y JSE_3.

b. Función Objetivo: para dar un ejemplo del cálculo de esta función, se emplea la Ec. (8) en uno de los individuos de la tabla 13. En este caso en particular, se supone a = 0.1 y b = 1 Además, se supone que los estudiantes jugaron 1 vez el JSE, y las puntuaciones obtenidas por cada una fueron: PA_1=2386, PA_2= 2663 y PA_3= 2556, con una duración en cada juego en tiempo de LE_1 = 80, LE_2 = 90, y LE_3 = 70 segundos. Ahora se puede calcular el FO*i* de cada uno. Por ejemplo, para el primer individuo sería:

$$FO = 2386 * .01 - 80 = 158.6$$

Esto mismo se hace para el resto de individuos. Ese individuo en particular tendrá la estructura mostrada en la tabla 14.

c. Espacio de Creencias: en este ejemplo, solo se utilizan dos conocimientos, el situacional y el normativo. La tabla 15 muestra el *conocimiento situacional* después que los 3 estudiantes jugaron, Vj son los valores con los que se ha usado hasta ahora el parámetro j en esos juegos realizados, con sus respectivos índice de ocurrencias (IO_j) y calidad del mismo (C_j).

Por otro lado, la tabla 16 muestra el *conocimiento normativo* después que los 3 estudiantes jugaron.

d. Protocolo Comunicación: los JSE actualizan el conocimiento en el espacio de creencia de la siguiente forma:

i. **Función de Aceptación:** los valores de IOj, Cj, LI y LS, de las tablas 15 y 16, se van actualizando a través de las generaciones, usando las Ecs 9, 10, 11 y 12, respectivamente. En este sentido, a medida que vayan jugando los estudiantes, esos conocimientos se van actualizando usando esas ecuaciones.

ii. **Función de Influencia:** se aplica la mutación dirigida por usar el conocimiento en el espa3cio de creencia en la población. La mutación se aplica a la población con cierta probabilidad, y usa aleatoriamente uno de los dos conocimientos definidos previamente.

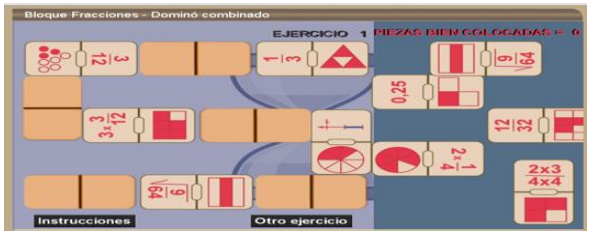

Fig. 20. Domino "Combinado" JSE_1

Fig. 21. Domino "Combinado" JSE_3

Tabla 14. Ejemplo de Individuo

P_1	P_2	P_3	P_4	**FO**
5	5	10	10	158,6

Tabla 15. Conocimiento Situacional para P3

Vj	**IOj**	**Cj**
10	2	170,6
2	1	176,3

Tabla 16. Conocimiento Normativo del JSE bajo estudio

P1		**P2**		**P3**		**P4**	
1	5	1	8	2	100	5	100

Ahora bien, el AC usa también otros operadores genéticos en la fase de reproducción de nuevos individuos en la población, éstos son el operador de cruce y de mutación clásico.

A continuación, presentamos en la tabla 17, una posible evolución del AC en el tiempo, con los 3 jugadores que repiten varias veces e JSE, para terminar de aprender las fracciones en el curso de SaCI.

Supongamos que se cruzan los individuos del circulo amarillo (JSE1 y JSE2) de la Tabla anterior.

Tabla 17. Evolución en el tiempo

#Iters	Población (3 individuos)					
	JSE	**P_1**	**P_2**	**P_3**	**P_4**	**FO**
1ra	1	5	5	10	10	158,6
	2	2	1	2	8	176,3
	3	4	5	5	6	185,6
n_ava	1	5	1	2	10	145,9
	2	2	5	5	8	156,6
	3	6	5	10	4	92,5

Espacio de Creencias Conocimiento Situacional

	Pj	**Vj**	**IOj**	**Cj**
1ra	1	5	1	158,6
		2	1	176,3
		4	1	185,6
	2	1	1	176,3
		5	2	170,6
	3	10	2	170,6
		2	1	176,3
	4	10	1	158,6
		8	1	176,3
		6	1	185,6
n-ava	1	1	5	181,4
		2	4	124,2
		3	6	151,1
		4	4	143,5
		5	5	101,6
		6	4	122,1
		7	2	94,5

Conocimiento Normativo

	P1		**P2**		**P3**		**P4**	
1ra	1	5	1	8	2	100	5	100
n_ava	1	7	4	10	3	75	3	60

Si se supone que el punto de cruce es la mitad, en la siguiente iteración se producen dos nuevos individuos (instancias del JSE, ver Tabla 18).

Así va evolucionando el juego, y para cada nuevo juego realizado por los estudiantes con las nuevas inicializaciones de parámetros en el JSE, se actualiza el conocimiento en el espacio de creencias, hasta un momento en que convergen esos valores (dejan de cambiar. En ese momento, se han obtenidos los valores y rangos de los parámetros con los que los estudiantes aprenden mejor las fracciones usando ese JSE.

Por ejemplo, si suponemos que es a la iteración n-ava, el mejor valor de P1 será 1, y su rango entre 1 y 7, y así para el resto de los parámetros.

Tabla 18. Nuevos Individuos

Ind	P_1	P_2	P_3	P_4
JSE_2	2	1	10	6
JSE_3	4	5	2	8

Tabla 19. Comparación con otros trabajos recientes

Comportamiento Emergente	[3,4,21]	[13]	[14]	[23]	[24]	Presente Trabajo
Estratégico		X	X			X
Secuencia				X	X	X
Propiedad	X	X				X

5. Comparación con otras propuestas

En esta sección, se compara nuestro trabajo con otros. La comparación que se propone realizar es basada en los tipos de comportamientos emergentes que permiten los MJSE (ver tabla 19).

En el JE Metrópolis [3, 4, 21], existen ciudades auto-gestionadas, donde aparecen 2 tipos de emergencia: la final y la de propiedad.

Por ejemplo: patrones urbanísticos que indican cuando debe finalizar el juego (emergencia del final), o la modificación de los radios de cobertura de cada tipo de edificación, según las características de los jugadores (emergencia de propiedad).

Ahora, bien, en ese juego no se propone un MJSE, sino simplemente ese juego tiene esas dos emergencias.

En [13] proponen un JS para la rehabilitación física basado en un Kinect de Xbox One, que permite el seguimiento de articulaciones en un pedal del timón. El sistema está construido para la rehabilitación adaptativa, basada en la robótica, cinemática inversa (IK), y reglas difusas del tipo Mamdani's que definen las estrategias de rehabilitación, las cuales se van adecuando de acuerdo al jugador (emergencia de estrategia), y sus variables de salida definen los parámetros del JS, en función del rendimiento de la persona en la terapia, y su nivel de habilidad física (emergencia de propiedad).

En [14] se describen juegos gráficos con incertidumbre estocástica, donde se utiliza sistema de transición difusa (FTS, por sus siglas en inglés) para hacer emerger estrategias. El objetivo de un jugador es maximizar su valor para alcanzar tareas, mientras que el rival apunta a lo contrario.

En [23] se propone un método de aprendizaje reforzado para resolver tareas en escenarios (emergencia de secuencia), compuesto de dos etapas: primero, aprende a mapear videos de múltiples fuentes con una representación en común, siguiendo unos objetivos (es supervisado); segundo, se utiliza un método para la exploración de los escenarios propuestos en el mapeo, para determinar si se imita la trayectoria de un jugador.

Para ello, se construye una función de recompensa que alienta a un agente a imitar el juego de uno o varios humanos.

En [24] se presenta un enfoque novedoso de motor de juego de simulación avanzada. Particularmente, los agentes inteligentes hacen predicciones sobre su escenario para reajustarlos (emergencia de secuencia).

Se hace uso como si fuera un JS, del famoso juego Súper Mario Bros, para probar el enfoque. Se demuestra la predicción de estados futuros en escenarios de ese juego, usando redes neuronales convolucionales (CNN).

Como se ha podido observar, ninguno propone un MJSE, algunos proponen un MJ [24], otros simplemente permiten que se den ciertos tipos de emergencia durante el juego [3, 4, 13, 14, 23]. Además, permiten una o dos emergencias, y no se adaptan a un contexto.

En cambio, el componente SAV del MJSE propuesto en este trabajo, permite la emergencia fuerte a nivel de: estrategias (para lo cual utiliza SCD), secuencias (para lo cual utiliza ACO), y propiedades (para lo cual utiliza AC), lo que permite adecuar el JSE a un SaCI.

6. Conclusiones

En este artículo presentamos el componente SAV para un MJSE, el cual permite la emergencia fuerte, de tal manera de adaptar un JSE al proceso al entorno. El componente SAV fue probado en un aula inteligente, de tal manera de permitir al JSE adecuarse al proceso de enseñanza-aprendizaje que se esté dando en un momento dado en el aula inteligente. Las opciones de adaptación que posibilita el SAV sobre un JSE en un SaCI están dirigidas a la emergencia de estrategias, en el sentido de posibilitar nuevas formas, reglas, etc., de comportamiento en el JSE. Esto es realizado a través del uso de SCDs que manipulan las estrategias de acuerdo a sus usos durante el juego. También, posibilita la emergencia de secuencias, de tal forma de permitir nuevos escenarios durante el juego, utilizando ACO para dicha tarea. Finalmente, permite la emergencia de propiedades, de tal manera de adecuar los parámetros del JSE al contexto estudiantil de SaCI en un momento dado, utilizando para ello los ACs.

Como proyecto futuro se realizarán pruebas del MJSE en diferentes procesos de enseñanza-aprendizaje, y se evaluara su impacto en los mismos usando indicadores pedagógicos que permitan mostrar su efectividad en dichos procesos.

Referencias

1. **Bellotti, F., Berta, R., & de Gloria, A. (2010).** Designing effective serious games: opportunities and challenges for research. *International Journal of Emerging Technologies in Learning*, Vol. 5, pp. 22–35. DOI:10.3991/ijet.v5s3.1500.
2. **Steven, J. (2008).** *Sistemas emergentes: o qué tienen en común hormigas, neuronas, ciudades y software.* Ediciones Turner/Fondo de Cultura Económica, España.
3. **Aguilar, J., Altamiranda-Pérez, J., Díaz-Villarreal, F., Cordero, J., Chávez, D., & Gutiérrez-de-Mesa, J. (2019).** Metropolis: an emerging serious game for the smart city. *DYNA*, Vol. 86, No. 211, pp. 215–224. DOI:10.15446/dyna.v86n211.80864.
4. **Aguilar, J., Altamiranda, J., & Chávez, D. (2016).** Extensiones a metrópolis para una emergencia fuerte. *Revista Venezolana de Computación*, Vol. 3, No. 2, pp. 38–46.
5. **Aguilar, J., Altamiranda, J., Díaz, F., & Mosquera, D. (2016).** Motor de juego serios en ARMAGAeco-c. *Revista Científica UNET*, Vol. 28, No. 2, pp. 100– 110.
6. **Aguilar, J., Altamiranda, J., & Díaz, F. (2018).** Design of a serious emerging games engine based on the optimization algorithm of ant colony. *DYNA*, Vol. 85, No. 206, pp. 311–320. DOI:10.15446/dyna.v85n206.69881.
7. **Aguilar, J., Menolascina, Y., & Rivas, F. (2005).** Compiler design for fuzzy classifier systems. *WSEAS Transactions on Systems*, Vol. 4, No. 4, pp. 262–267.
8. **Aguilar, J. & Cerrada, M. (2000).** Un sistema clasificador difuso para el manejo de fallas. *Revista Técnica de la Facultad de Ingeniería*, Vol. 23, No. 2, pp. 98–108.
9. **Terán, J., Aguilar, J., & Cerrada, M. (2014).** Cultural learning for multi-agent system and its application to fault management. *Conference: IEEE Congress on Evolutionary Computation (CEC)*. DOI:10.1109/CEC.2014.6900438.
10. **Brandão, R. & Neves, A.M. (2014).** Design da ludonarrativa: principios da narratología aplicados ao game design para concepção de mecánicas. *Proc. XIII SBGames,* pp. 112–119.
11. **Bigogno, M., Réda, V., & La-Carreta, M. (2017).** Dissonância ludonarrativa x suspensão da descrença: quando o gameplay desmente a narrativa ou quando o jogador apenas a aceita. *Proc. SBGames*, pp. 1068–1071.
12. **Shafi, K. & Abbass, H.A. (2017).** A survey of learning classifier systems in games. *IEEE Computational intelligence magazine*, Vol. 12, No. 1, pp. 42–55, DOI:10.1109/MCI.2016.2627670.
13. **Esfahlani, H.S., Cirstea, S., Sanaei, A., & Wilson, G. (2017).** An adaptive self-organizing fuzzy logic controller in serious game for motor impairment rehabilitation*. IEEE 26th International Symposium on Industrial Electronics (ISIE´17),* pp. 1311–1318. DOI: 10.1109/ISIE.2017.8001435.
14. **Pan, H., Li, Y., Cao, Y., & Li, D. (2017).** Reachability in fuzzy game graphs. Proceedings *IEEE Transactions on Fuzzy Systems*, Vol. 25, No. 4, pp. 984–972.
15. **Aguilar, J., Valdiviezo, P., Cordero, J., & Sánchez, M., (2015).** Conceptual design of a smart classroom based on multiagent systems. *Proc. International Conference on Artificial Intelligence (ICAI'15),* pp. 471–477.
16. **Sánchez, M., Aguilar, J., Valdiviezo, P., Cordero, J. (2015).** A smart learning environment based on cloud learning. *International Journal of Advanced*

Information Science and Technology (IJAIST), Vol. 39, No. 39, pp. 36–49.

17. **Cordero, J., Sánchez, M., Aguilar, J., & Valdiviezo, P. (2015).** Basic features of a reflective middleware for intelligent learning environment in the cloud (IECL). *Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, pp. 1–6. DOI:10.1109/APCASE.2015.8.
18. **Aguilar, J., Altamiranda, J., Díaz, F., (2020).** Specification of a managing agent of emergent serious games for a smart classroom. *IEEE Latin America Transactions*, Vol. 18, No. 1, pp. 51–58. DOI:10.1109/TLA.2020.9049461.
19. **Jin, X. & Reynolds, R.G. (1999).** Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problem: a cultural algorithm approach. *Proceedings Congress on Evolutionary Computation (CEC´99)*, pp. 1672–1678. DOI:10. 1109/CEC.1999.785475.
20. **Mitchell, M. (2002).** *An introduction to genetic algorithms.* MIT press.
21. **Aguilar, J. (2014).** *Introducción a los Sistemas Emergentes*. Universidad de Los Andes.
22. **Aguilar, J., Altamiranda, J., Diaz, F., Gutiérrez, J., & Pinto, A. (2019).** Sistema adaptativo de tramas para juegos serios emergentes basado en el algoritmo de optimización de colonia de hormigas. *Proceedings XLV Latin American Computer Conference (CLEI´19).*
23. **Aytar, Y., Pfaff, T., Budden, D., Le-Paine, T., Wang, Z., & de-Freitas, N. (2018).** Playing hard exploration games by watching YouTube. *Proceedings 32nd Conference on Neural Information Processing Systems*, pp. 1–15.
24. **Guzdial, M., Li, B., & Ried, M. (2017).** Game engine learning from video. *26th International Joint Conference on Artificial Intelligence* (IJCAI'17), pp. 3707–3713. DOI:10.24963/ijcai.2017/518.

Article received on 31/03/2019; accepted on 08/01/2020.
Corresponding author is José Aguilar.