**UNIVERSIDAD DE LOS ANDES**
**FACULTAD DE INGENIERIA**
**DOCTORADO EN CIENCIAS APLICADAS**
**MÉRIDA – VENEZUELA**

# CONCEPCIÓN DE UN MOTOR DE JUEGOS SERIOS EMERGENTES PARA AULAS INTELIGENTES



**Autor:** Ing. M. Sc. Francisco Díaz
**Tutor:** Dr. Jose Aguilar
**Co-Tutor:** Dr. Junior Altamiranda

**Proyecto de Grado presentado ante la ilustre Universidad de los Andes como requisito parcial para optar al Título de Doctor en Ciencias Aplicadas**

Mérida, marzo de 2023

# RESUMEN

El objetivo principal de la presente tesis es realizar el diseño conceptual de un Motor de Juego Serio Emergente (MJSE), el cual permite que los usuarios desarrollen habilidades utilizando videojuegos que se adapten al tema de la clase que se está dando en un Salón de Clases Inteligentes (SaCI). El manuscrito se divide en tres fases: la primera hace explícita la posibilidad de surgimiento en un juego serio a partir del manejo coordinado de la trama o escenas del juego, adaptada al contexto educativo específico donde se está utilizando. Para ello se ha propuesto el sistema adaptativo de trama, este componente se basa en un algoritmo de optimización de colonia de hormigas que cambia las tramas en el juego para seguir el tema deseado en el aula inteligente. La segunda fase propone un sistema adaptativo de parámetros para juegos serios emergentes, que permite la emergencia de propiedades en un videojuego, con el fin de adaptarlo al contexto educativo en el que se está utilizando. En particular, se basa en el uso de algoritmos culturales, que establece un proceso de aprendizaje para modificar sus parámetros, colocando el juego serio al nivel de dificultad que los alumnos puedan jugarlo y aprender de él. En una tercera y última fase se desarrolló un sistema adaptativo de estrategias para juegos serios emergentes, el cual gestiona los movimientos del personaje principal o avatar mediante un sistema clasificador difuso utilizando algoritmos genéticos. En esta tesis, las tres fases son probadas en un SaCI, de tal manera que permita la adaptación del JSE a los estudiantes que lo estén usando en sus procesos de enseñanza-aprendizaje.



**Palabras Claves:** Motor de Juegos Serios Emergentes, Algoritmo de Colonias Hormigas, Algoritmos Cultural, Sistema Clasificador Difuso, Salón de Clases Inteligentes.

# ABSTRACT

The main objective of this thesis is to carry out the conceptual design of an emerging serious game (ESG) engine, which allows users to develop skills using video games that adapt to the theme of the class that is being given in a Smart Classroom (SaCI). This project is divided into three phases: the first makes explicit the possibility of a serious game arising from the coordinated management of the plot or scenes of the game, adapted to the specific educational context where it is being used. This component is based on an ant colony optimization algorithm that changes the plots in the game to follow the desired theme in the smart classroom. The second phase is an adaptive system of parameters for emerging serious games, which allows the emergence of properties in a video game, in order to adapt it to the educational context in which it is being used. In particular, it is based on the use of cultural algorithms, which establishes a learning process to modify its parameters, placing the serious game at a level of difficulty that students can play and learn from it. In a third and final phase, an adaptive strategy system for serious emerging games is developed, which manages the movements of the main character or avatar through a fuzzy classifier system using genetic algorithms. In this thesis, the three phases are tested in a SaCI, in such a way that it allows the adaptation of the ESG to the students that are using it in their teaching-learning processes.



**Key Words:** Serious Emerging Games Engine, Ant Colonies Algorithm, Cultural Algorithms, Fuzzy Classifier System, Smart Classroom.

# TABLA DE CONTENIDO

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# ÍNDICE DE ECUACIONES

# LISTA DE ABREVIATURAS

ACO: Algoritmo de Colonia de Hormigas.
AJSE: Agente Inteligente de Juegos Serios Emergentes.
C: Calidad.
D: Dominio.
FC: Función de Calidad.
FE: Función Evento.
FO: Función Objetivo.
JSE: Juegos Serios Emergentes.
ID: Identificador Único de un Nodo.
IO: Índice de Ocurrencia.
μ: Constante del momento.
m: Momento.
MJSE: Motor de Juegos Serios Emergentes.
MGJSE: Módulo Generador de JSE.
NO: Número de Ocurrencia.
LE: Intervalo de Tiempo, lapso de tiempo que duro el JSE.
LI: Limite Inferiores.
LOM (por sus siglas en inglés de *Learning Object Metadata*): Metadato de Objeto de Aprendizaje.
LS: Limite Superiores.
P: Parámetros.
PA: Puntuación Acumulada, representa la puntuación máxima alcanzada por el jugador.
ROA: Repositorio de Objeto de Aprendizaje.
RA: Recurso de Aprendizaje.
SA: Sistema Académico.
SaCI: Salón de Clases Inteligente.
SAE: Sistema Adaptativo de Estrategia.
SAP: Sistema Adaptativo de Parámetro.
SAS: Sistema Adaptativo de Secuencias.
SAT: Sistema Adaptativo de Trama.
SAV: Sistema Adaptativo de Videojuego.
SCD: Sistema de Clasificador Difuso.
SCORM: Modelo de Referencia de Objetos de Contenido Compartible.
SEV: Sistema Emergente de Videojuego.
STE: Submotor de Trama Emergente.
T: Número de generaciones.
V: Valores.
VLE (por sus siglas en inglés de *Virtual Learning Environment*): Entorno Virtual de Aprendizaje.
XML (por sus siglas en inglés de *eXtensible Markup Language*): Lenguaje Marcado Extensible.

# CAPÍTULO I: GENERALIDADES

## 1.1. INTRODUCCIÓN

Un motor de juego es un conjunto de programas que permiten la creación, la representación y la ejecución de un videojuego. Uno de los objetivos de esta área es desarrollar motores que puedan ser usados por diferentes juegos, adaptándose a los requerimientos específicos de los mismos. Hoy en día, existen una gran variedad de dichos motores [1, 2], para que el usuario pueda crear sus propios juegos.

En particular, nosotros estamos interesados en los juegos serios emergentes (JSEs), definidos como juegos cuyo objetivo es diferente al de meramente 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 [1, 3]. Los JSEs parten de dos teorías, la de *juegos serios*, que establece que un juego tiene un propósito específico, el cual puede estar relacionado con el aprendizaje, con la comprensión de un tema complejo [4]; 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, 5, 6, 7, 8]. En ese sentido, estos tipos de juegos requieren de un nuevo tipo de motor de juego, que llamaremos motor de juego serios emergentes (MJSE), específicos a sus características. Un MJSE se entiende como el conjunto de submotores donde el JSE opera. Así, esta tesis se interesa en definir la conceptualización e implementación de un MJSE.



A su vez, los nuevos avances en las tecnologías de la información, en ámbitos como la computación en la nube y ubicua, permiten explotar las herramientas computacionales, con el fin de definir sistemas en un alto nivel de abstracción, como es el caso de los ambientes inteligentes, los cuales permiten adaptar el ambiente de forma dinámica a las necesidades de los usuarios [9, 10, 11].

En específico, este trabajo se interesa en los ambientes inteligentes educativos, y en especial, en las aulas inteligentes, también llamadas SaCI, que redefinen un aula de clases, integrando las tecnologías de computación, comunicación, e inteligencia artificial, entre otras, con el fin de permitir la interacción autónoma entre los componentes de hardware (pizarra inteligente, robots, etc.) y de software (sistemas recomendadores, entornos virtuales de aprendizajes (VLE por sus siglas en inglés de *Virtual Learning Environment*), videojuegos, etc.) en el aula, para mejorar los procesos de enseñanza en ella [11, 12].

En nuestro caso, los JSEs deben ayudar a alcanzar los objetivos deseados en el SaCI, adaptándolos a las necesidades de los estudiantes. Por ello, en este trabajo se pretende desarrollar un MJSE, que se pueda ir adecuando a los requerimientos que vayan surgiendo en el SaCI. Los beneficios que el MJSE aportará a un SaCI son: 1) contribuyen a la inmersión del estudiante con respecto al tema que se está estudiando, 2) ayudan al profesor a hacer más interesante y entretenida la clase, 3) permite el uso de herramientas didácticas en el aula, 4)

se adaptan a los requerimientos que se utilizan en ese instante, 5) el estudiante se lleva el JSE a su casa para practicar y 6) el estudiante desarrolla habilidades a través del juego en función de lo que explica el profesor.

## 1.2. ANTECEDENTES

A continuación, se hace referencia a algunos trabajos cercanos a nuestra propuesta, los cuales permiten conocer las tendencias actuales en esta área.

### 1.2.1. TRABAJOS EN JUEGOS SERIOS

En [13] proponen entornos de aprendizaje inteligente basados en juegos serios, aprovechando las tecnologías de videojuegos comerciales y las técnicas de inteligencia artificial, las cuales son incorporadas en sistemas de tutoría inteligente. Específicamente, trabajan con el videojuego "Crystal Island", siendo un excelente laboratorio para investigar técnicas de inteligencia artificial. Este trabajo también investiga sobre la planificación de narrativas en tiempo real, y sobre modelos para reconocer los estados emocionales en los estudiantes, proponiendo uno basado en Redes Bayesianas Dinámicas.



El artículo [14] analiza el rol de los juegos digitales en el aprendizaje, para estudiantes desde los niveles básicos hasta el nivel de licenciatura. Analizan los juegos serios versus juegos estándares. Los resultados indican que los juegos serios mejoran significativamente el aprendizaje de los estudiantes, y en general, hay procesos de aprendizaje significativo asociados a ellos. El estudio demostró que los efectos varían según las características del juego, en particular, las características visuales y narrativas. En [15] buscan ampliar el conocimiento y la comprensión de la práctica educativa centrada en juegos serios, específicamente, en simulaciones por computadora para estudiar modelos de emprendimiento usando juegos serios.

Por otro lado, en [16] se analiza el uso de juegos serios en el área de la salud, y particularmente, en el campo de enfermedades neurodegenerativas como la enfermedad de Alzheimer. Ellos usan juegos serios para el tratamiento, estimulación y rehabilitación de enfermedades. Además, ese artículo proporciona recomendaciones en términos de opciones ergonómicas para el diseño de juegos serios, para estimular a las personas con enfermedades neurodegenerativas. En Borji & Khaldi [17] explican cómo buscar en internet de manera efectiva juegos serios para estudiantes, a través de un gestor de sistemas de aprendizaje. Usan un esquema de metadatos que describe el juego serio como un recurso de aprendizaje. Este metadato es basado en el estándar IEEE LOM (por sus siglas en inglés de Institute of Electrical

and Electronics Engineers Learning Object Metadata), titulado “SG-LOM” por sus siglas en inglés de Serious Game-LOM, que tiene en cuenta diferentes características de los juegos serios, tales como: título, descripción, contexto y otros.

En [18] se investigan varias aplicaciones de los juegos serios en ambientes inteligentes. En particular, analizan como crear acciones combinando elementos lúdicos con actitudes. La conjunción de ambos elementos permite el diseño de “sistemas persuasivos”, en contextos educativos o de ejercicio físico, integrando a su vez, otros elementos como teléfonos inteligentes, juguetes o arte interactivo. El estudio presentado en [19] recopila sistemáticamente 150 videojuegos que representan el cambio climático, incluidos juegos serios (n = 109) y de entretenimiento (n = 41). El estudio proporciona un mapeo sistemático de juegos existentes que representan el cambio climático, para desarrolladores, diseñadores y educadores interesados. El objetivo de [20] es describir los pasos para el diseño de un juego motivador de rehabilitación de accidentes cerebrovasculares. Se implementa y evalúa un videojuego con sensores Natural User Interface como Leap Motion y Microsoft Kinect 2. Los participantes encontraron las acciones del juego identificables y familiares, similares a sus actividades de la vida diaria. Los resultados indican un gran potencial en la creación de una experiencia inmersiva y motivadora, replicable para otros juegos de rehabilitación.



### 1.2.2. TRABAJOS EN MOTOR DE JUEGO

En [2] proponen un marco para la selección de motores de juegos serios, caracterizando tres elementos para la comparación de los mismos, los cuales son: 1) Calidad audiovisual y funcional, con respecto al aprendizaje y a la inmersión a la que se lleva al estudiante: 2) Capacidad de composición, caracteriza la capacidad de reutilización del contenido creado dentro de un motor juego; 3) Accesibilidad, define la plataforma donde puede desplegarse el juego, considerando aspectos como dispositivos de hardware y software, entre otros. Además, usando esos criterios comparan varios motores de juegos: Cry, Source, Unreal y Unity. El propósito de [21] es presentar y evaluar el desempeño de una herramienta gratuita para inteligencia artificial, denominada framework RAIN, para el motor de juego Unity 3D, cuya finalidad es facilitar el desarrollo de juegos digitales. Diseñada para desarrolladores independientes, simplifica muchas de las partes técnicas para el uso de la inteligencia artificial en personajes no jugables, permitiendo que el desarrollador se enfoque en la elaboración del comportamiento para personajes no jugables.

Shafi & Abbass [22] presentan una revisión sobre tecnología avanzada en el uso de sistema de clasificación de aprendizaje en la concepción de juegos. Particularmente, estudian 129 artículos en los cuales se usan técnica de inteligencia artificial como: redes neuronales,

sistema clasificador de aprendizaje, algoritmos genéticos, entre otros. En específico, analizan la arquitectura flexible de un motor de juego basado en ellas. En [23] proponen un framework para diseñar y desarrollar videojuegos, el cual se basa en la descomposición de un motor de Juegos, lo que permite la reutilización de sus submotores. Los submotores en que se descompone el motor de juego son los sistemas de renderizado gráfico, de detección de colisiones, audio, y de gestión de los objetos de un juego y de las reglas.

Finalmente, en [24] se diseña un motor de juego basado en eventos, que permite seguir la dinámica del juego a través de eventos. Además, transmite comandos a la lógica del juego, enviando y recibiendo mensajes según los eventos recibidos, usando para ello un sistema de eventos para videojuegos. También proponen un motor de inteligencia artificial que se comunica con el sistema de eventos, que es coordinado a través de un agente inteligente que controla el juego usando esos dos sistemas. Finalmente, en el trabajo [25] analizan y evalúan los motores de juego más modernos y sus principios claves. Ese trabajo destaca las recomendaciones y conclusiones fundamentales sobre algunos tipos importantes de motores de juego cuando se utilizan para implementar juegos serios para la educación. Concluyen que la gamificación en el aprendizaje y la educación tienen un atractivo masivo entre los alumnos.



### 1.2.3. TRABAJOS EN JUEGOS EMERGENTES

El artículo [6] analiza el uso de la narrativa emergente en los juegos digitales. El artículo destaca diversos principios de la narratología que pueden ayudar en la concepción de mecánicas de juegos electrónicos, para fomentan alta narrativa emergente. Por otro lado, en [3] proponen el juego emergente denominado Metrópolis, que parte de la premisa que las ciudades son auto-gestionadas por decisiones tomadas en conjunto por sus habitantes (jugadores), sin que exista un habitante con un papel más importante. En Metrópolis, emergen patrones urbanísticos en la ciudad por las decisiones que sus habitantes toman. Además, recientemente, en [8] se propone una extensión a Metrópolis, introduciendo mecanismos emergentes a la dinámica del juego, para que se adapte a sus jugadores.

En [26] se analiza la relación entre el juego, la mecánica y la narrativa emergente. Así, ellos investigan la relación entre el diseño de juegos digitales y la narrativa que surge durante el juego. Ellos concluyen que la forma en que se desarrollan los mecanismos, conduce a un mayor o menor grado de influencia de las acciones del jugador en la narrativa del juego. Los autores de [27] analizan la relación entre los elementos de narrativas y las acciones en los juegos. Denominan “Disonancia Ludonarrativa” al desacuerdo entre la “Narrativa Lineal” 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. Para comprender mejor la separación entre

ellas, definen los siguientes conceptos: narrativa lineal, narrativa emergente, suspensión de la incredulidad, círculo mágico y contratos de Hocking.

Chauvin et al. [28] proponen un sistema para el desarrollo de narrativas para juegos emergentes. En particular, presentan cinco enfoques de narrativas emergentes: crear mundos que sean creíbles, incorporar la experiencia del jugador dentro del juego, establecer espacios de múltiples opciones de decisión en los estados del juego, permitir la incertidumbre posibilitando crear situaciones inesperadas, y posibilitar la co-autoría tal que el jugador sea responsable de la historia y la dirección que toma el juego. Los autores de [29] presentan una metodología de aprendizaje basado en desafíos en un programa destinado al desarrollo de software para el ecosistema tecnológico de Apple. El sistema de diseño de experiencias educativas fomenta narrativas emergentes sin guion para la educación experiencial. Para ello, clasifica los componentes para diseñar una experiencia educativa que permita que la progresión del aprendizaje sea impulsada afectivamente por los alumnos. Al centrarse en establecer parámetros y dar a los alumnos autonomía como coautores, el modelo describe mecanismos que permiten que surjan narrativas poderosas y espontáneas basadas en una motivación intrínseca.

El objetivo de Figueira et al. [30] es presentar BinG, un framework para recopilar información de procedencia de una sesión de juego y así comprender las relaciones subyacentes que se encuentran entre los diferentes elementos presentados en el juego (como ítems y enemigos). Esta información de procedencia se usa luego para producir nuevos parámetros para equilibrar el juego de acuerdo con las habilidades del jugador actual. BinG permite el uso de diferentes modelos de equilibrio, que el desarrollador puede activar cuando los desee, para producir nuevos factores de equilibrio que se ajustarán en el juego.



### 1.2.4. TRABAJOS CON TÉCNICAS DE INTELIGENCIA ARTIFICIAL EN VIDEOJUEGOS

Con respecto a los algoritmos de enjambres, en [31] investigan el uso del algoritmo de optimización de abejas para crear grupos de agentes autónomos cooperativos en juegos de estrategias de batallas. En [32] se presenta un juego cooperativo para diseñar una red. El modelo de juego busca que los jugadores cooperen para maximizar el beneficio. Dado que los jugadores pueden diseñar diferentes redes, la cooperación permite diseñar la red óptima que satisfaga a todos los jugadores y minimice el costo. Para eso proceso cooperativo se usa el Modelo de Optimización Basada en Colonia de Hormigas (ACO, por sus siglas en inglés Ant Colony Optimization).

Tregel et al. [33] usan los enfoques Beam-ACO y Ant-Q en juegos basados en ubicación. Particularmente, ellos son usados en el análisis y la planificación de rutas, ya que en el juego se proponen una gran cantidad de ubicaciones deseables para un jugador en cuestión. Ellos proponen un sistema para crear de forma dinámica una ruta personalizada para los jugadores, basada en datos de juegos anteriores. Estas rutas se pueden personalizar completamente en lo que respecta a su tiempo y ubicación, así como a la meta deseada del jugador para el juego, lo que les permite, por ejemplo, maximizar los lugares visitados. En el trabajo [34] se usó ACO como método para encontrar la mejor manera de completar el juego Sudoku. El resultado de esta investigación muestra que ACO funciona un 89 % mejor que los métodos tradicionales como el algoritmo Backtracking. Khalid et al. [35] tienen como objetivo empoderar a los personajes no jugables con rasgos inteligentes. Se mezcla ACO con un enfoque basado en algoritmos genéticos para el juego de disparos en primera persona. Los personajes no jugables usan ACO para explotar y optimizar su estrategia actual, y un algoritmo genético para compartir y explorar estrategias entre los personajes no jugables.

En cuanto a algoritmos culturales y evolutivos, Waris y Reynolds [36] aplican un algoritmo cultural, como mecanismo colaborativo para distribuir el conocimiento de un juego en una red de población. El rendimiento de este mecanismo se compara con otros enfoques. Los resultados preliminares sugieren que este enfoque es mejor. En [37], Yang presenta un algoritmo evolutivo inspirado en la teoría de juegos para optimización global (GameEA). Se proporciona una formulación para estimar las expectativas de pago, que es un mecanismo para convertir a un jugador en un tomador de decisiones racional.



En el ámbito de lógica difusa, en [38] se describe 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 una Xbox Kinect, la pulsera Myo y el pedal de timón. Específicamente, el sistema utiliza reglas difusas del tipo de Mamdani que definen las estrategias de rehabilitación, las cuales se adaptarán de acuerdo al desempeño de los jugadores en terapia y nivel de habilidad física. En [39] describen un juego con comportamientos estocásticos, donde utiliza un sistema de transición difusa (FTS) para generar estrategias. El objetivo de un jugador es maximizar su valor para lograr tareas mientras el oponente busca lo contrario.

## 1.3. PLANTEAMIENTO DEL PROBLEMA

A pesar de la evidencia de la eficacia de los juegos serios en el ámbito educativo [17], algunos autores han encontrado obstáculos para su uso, como por ejemplo en [8], donde señalan que "… los contenidos de los juegos muchas veces no responden a las necesidades de las asignaturas, o son hechos para un contexto y objetivo preciso". Por otro lado, en [6, 3] explican que un juego emergente tiene un carácter autonómico, cualidad operativa que le permite tomar en cuenta el desempeño del jugador y entorno en el que opera. Según [7, 8, 3], las formas de emergencia en un juego emergente pueden ser de seis tipos, las cuales pueden ser: estrategia, secuencia, propiedad, final, modelo de negocio y utilidad (en la sección 2.2.3 se darán detalles de las mismas). Ahora bien, los actuales motores de juego no permiten los diferentes tipos de emergencia, las cuales deben ser diseñadas de manera explícita por el diseñador del juego.

Por otro lado, según [12], un ambiente inteligente se basa en una plataforma tecnológica que controla en forma automática su funcionamiento, para adecuarse a las necesidades de sus usuarios. En [40], se sugiere que "el uso racional de estos recursos tecnológicos, suele requerir que los programas de computación involucrados tengan un grado de flexibilidad y autonomía, que difícilmente puedan ser logrados mediante las técnicas de diseño y programación clásicas". Lo anterior significa que estos sistemas deberían ser lo suficientemente "inteligentes", como para actuar en forma adecuada en situaciones cambiantes, imprevistas y complejas. Lo expuesto anteriormente se logra en mayor o menor grado, a través del paradigma de sistemas multiagentes, definido como una aplicación computacional donde los agentes cooperan o compiten con otros para lograr metas colectivas o individuales [9].



Esta investigación se enfoca en el diseño de un ambiente inteligente para entornos de aprendizaje, que explote los JSEs presentes en el ambiente para mejorar las experiencias de los usuarios en sus procesos de aprendizaje. En particular, se trabajará en el SaCI [9], el cual requiere de juegos serios que se puedan adaptar a los requerimientos que vayan surgiendo en el SaCI, y en especial, que permita la emergencia de nuevas dinámicas en los juegos serios, adecuadas a las temáticas que se estén impartiendo en el SaCI.

Para ello, en este trabajo se propone el desarrollo de un MJSE, y la especificación de un agente inteligente de JSE (AJSE), para permitir la integración y adecuación autonómica de los JSEs en las dinámicas de aprendizaje de un SaCI. En ese sentido, el MJSE y el AJSE posibilitarán la adaptación de los JSEs en el SaCI, a partir de la emergencia de nuevos temas, reglas, estrategias, elementos, entre otros, en el JSE, de acuerdo a las necesidades de un SaCI. De esa manera, un JSE es visto como un objeto más del SaCI, que deberá interactuar con los otros componentes en el SaCI (pizarra inteligente, VLE, etc.), para aprender y adaptarse, de tal manera de coadyuvar a mejorar los procesos de aprendizaje que ocurren en los SaCI.

Por otro lado, el MJSE será usado por el AJSE en un SaCI que posea dispositivos tecnológicos como: Pc, laptop, celulares, tabletas, video beam o pizarra inteligente, donde el

profesor elige el tema que se va a dar en clases. A continuación, el AJSE selecciona de uno o varios repositorios los videojuegos con mayor puntaje compatibles con la clase, y usa el MJSE para ajustar el JSE al nivel de los estudiantes. Posteriormente, el MJSE va adecuando el JSE a los niveles adecuados a la evolución en su proceso de aprendizaje de los estudiantes, para ir ayudando en la explicación del tema que se está dando en el aula. De esta manera, el MJSE permite ir adaptando al JSE para que exista una mayor inmersión del estudiante y logre desarrollar habilidades en función del tema que se está explicando en el SaCI.

## 1.4. OBJETIVOS

### 1.4.1. OBJETIVO GENERAL

Especificar un Motor de Juegos Serios Emergentes para un SaCI.

### 1.4.2. OBJETIVOS ESPECÍFICOS

1. Caracterizar el uso de los juegos serios y juegos emergentes en un SaCI.
2. Caracterizar las dinámicas emergentes en los juegos serios en el contexto educativo.
3. Especificar la arquitectura del motor de juegos para JSE en un SaCI.
4. Desarrollar un prototipo del motor de juegos y realizar pruebas sobre un SaCI.



## 1.5. ALCANCE

Al finalizar la tesis, se cuenta con un diseño conceptual de un motor de juego para JSE en un SaCI [41, 42]. Además, se crea una arquitectura de software que permite la integración del JSE en el SaCI. A su vez, se desarrolla un prototipo de esa arquitectura que soporta al uso de JSE en un SaCI. En particular, para el desarrollo del prototipo se especifica e implementa de manera detallada el MJSE, y se especificará el AJSE para el SaCI, a través del cual se hará la integración del JSE en un SaCI, el cual invocará al MJSE para permitir la adaptación de los JSEs a las necesidades del SaCI. Para ello, se desarrollarán tres sistemas: sistema adaptativo de trama (SAT), sistema adaptativo de parámetros (SAP) y sistema adaptativo de estrategias (SAE), que son parte del sistema adaptativo de videojuego (SAV) dentro del MJSE [42]. Específicamente, en esta tesis se definen:

1. El componente del MJSE denominado SAT, que permite la actualización de un juego serio de acuerdo a la temática que se está impartiendo en un SaCI. Este componente se basa en ACO [32, 33, 34, 43] que cambia las tramas en el juego para seguir el tema deseado en el SaCI. Así, este componente gestiona un conjunto de tramas de juego que pueden ser de interés en un dominio de contexto educativo, para modificar dinámicamente un juego

serio inicialmente concebido a una asignatura enseñada en el aula inteligente, con el fin de adecuarlo al contexto pedagógico actual.

2. El componente denominado SAP, que permite la emergencia de propiedades en un juego, con el fin de adaptarlo al nivel de los estudiantes en el contexto educativo en el que se está utilizando. En particular, el SAP se basa en el uso de algoritmos culturales [36, 37, 44], que establece un proceso de aprendizaje para modificar los parámetros de un JSE en función del nivel de los alumnos que están en clases.
3. El componente denominado SAE, el cual maneja la emergencia de estrategias. SAE se usa en un SaCI de tal manera que permita controlar en tiempo real en un JSE sus estrategias (por ejemplo, los movimientos de los personajes principales (avatares)), por intermedio de reglas usando un sistema de clasificador difuso (SCD) [38, 39, 45], para responder a las eventualidades generadas en el videojuego.
4. La especificación de un AJSE [41] que permite la integración y adecuación autonómica de los JSEs en la dinámica de aprendizaje de un SaCI. En este sentido, el AJSE invoca al MJSE para posibilitar el uso sincronizado de los componentes del SAV (SAT, SAP y SAE) [42] en los momentos apropiados de la clase, para ayudar al profesor a explicar un tema dado utilizando JSEs en el SaCI. Para ello, el AJSE interactúa con los otros agentes en el SaCI (pizarra inteligente, celular, tableta, VLE, etc.).



## 1.6. ORGANIZACIÓN DE LA TESIS

El Capítulo II comprende los aspectos teóricos necesarios para la comprensión del trabajo desarrollado en la tesis. Los temas principales que aborda este capítulo son: juegos serios y sus conceptos vecinos (por ejemplo, videojuegos y motores juegos), sistemas y juegos emergentes, y las técnicas inteligentes usadas en este trabajo (algoritmo de colonia de hormigas, algoritmos culturales y sistema de clasificador difuso).

En el Capítulo III se define el diseño del sistema. Se proporciona la definición del JSE, y la arquitectura del MJSE. Posteriormente, se detallan los SAT, SAP y SAE, especificando los componentes da cada uno de ellos y los algoritmos utilizados. Finalmente, se explica cómo integrar un MJSE en un SaCI a través de un agente JSE.

En el Capítulo IV se hacen las pruebas de entonación, experimentos y análisis de resultados. Contiene los casos de estudios considerados en la tesis para verificar el funcionamiento de los sistemas adaptativos del MJSE. Los casos de estudios permiten determinar la eficacia y eficiencia del sistema propuesto.

El Capítulo V contiene las conclusiones generales de la tesis y los trabajos futuros que podrían ser realizados, para continuar investigando en este campo del conocimiento.

# CAPÍTULO II: MARCO TEÓRICO

El presente capítulo expone el marco teórico de base de nuestro trabajo, vinculado a juegos serios, motor de juego, juegos emergentes y técnicas inteligentes para el desarrollo de videojuegos. Este capítulo es la base para el diseño del MJSE adaptado a un SaCI propuesto en este trabajo.

## 2.1. JUEGOS SERIOS

A continuación, se muestran los conceptos importantes alrededor de los juegos serios.

### 2.1.1. VIDEOJUEGOS

Un videojuego es un juego electrónico en el que interactúan una o más personas, por intermedio de un controlador, con un dispositivo manejador de imágenes [22]. La figura 2.1 presenta la evolución de los videojuegos más populares comerciales (árbol genealógico de los videojuegos más destacados).



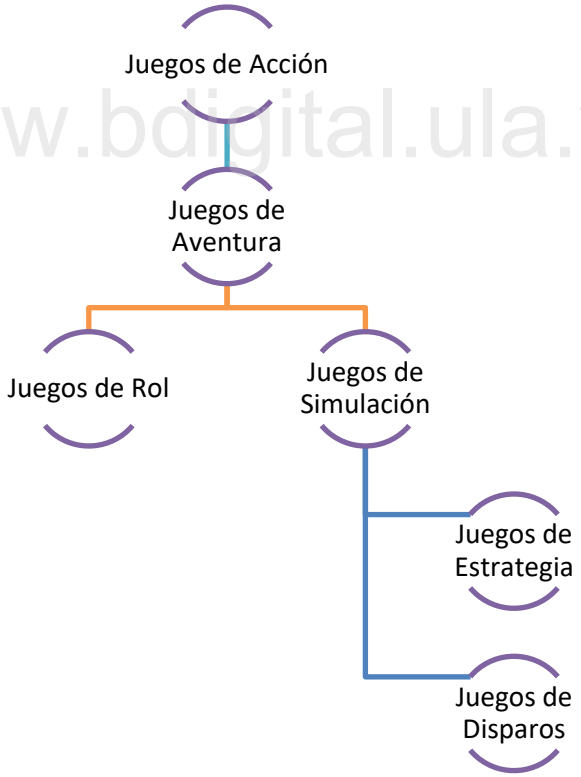



**Figura 2.1.** Árbol genealógico de los videojuegos. Fuente [22]

A continuación, explicaremos las ramas del árbol genealógico:

1. Los Juegos de Acción: son los más conocidos por eso están en la base del árbol genealógico, de allí se originan casi todos los tipos de videojuegos. Es definido por un jugador humano que generalmente controla y maniobra a un personaje en un entorno dado con desafíos basados en reflejos [22]. Algunos de los famosos sub-géneros de juegos de acción incluyen juegos de árcade (máquinas recreativas diseñadas para uso exclusivo de videojuegos, por ejemplo: Pong, Pacman, etc.); plataformas (se caracterizan por tener que caminar, correr o saltar sobre una serie de acantilados como, por ejemplo: Mario, Megaman, Sonic, etc.), y pelea (son juegos de combate cuerpo a cuerpo, como por ejemplo: Street Fighter, Mortal Kombat, etc.).
2. Los Juegos de Aventura: son caracterizados por la indagación, la investigación, la solución de acertijos, la interacción con los actores del videojuego. Se enfoca en tramas, en vez de desafíos basados en reflejos, como ocurre en los juegos de acción, lo cual hace emocionante jugarlo, enganchando a los jugadores en su narrativa. Abarcan una gran variedad de tramas de literatura y películas, tales como: fantasía, ciencia ficción, misterio, terror, y comedia como, por ejemplo: Resident Evil, Assessine's Creed y God of War.
3. Los Juegos de Rol o RPG: son caracterizados por una trama fuerte y un conjunto de reglas, tal que uno o más jugadores asumen un papel o un personaje en la trama, y realizan acciones que coinciden con ella [22]. Final Fantasy y Fallout son ejemplos de este tipo de juegos.
4. Los Juegos de Simulación: intentan modelar algunos aspectos de situaciones del mundo real, tales como el comercio o el tráfico [22]. Estos juegos tienen numerosos usos en áreas como la educación y política como, por ejemplo: los Sims sirven en la planificación familiar, y Sim City puede ser usado para definir políticas urbanísticas. Otros ejemplos son los simuladores de vehículos, que imitan lo que ocurre en la vida real con los autos, motos, aviones, etc.; un ejemplo es Flight Simulator, un simulador de vuelo.
5. Los Juegos de Estrategia: son juegos en que las acciones de los jugadores suelen estar relacionadas con la táctica, logística y planificación de recursos. Para ello, la inteligencia y las habilidades técnicas del jugador son necesarias en un medio ambiente cambiante [22]. Star Craft y War Craft son ejemplos. Los juegos de estrategias basado en turnos, es otra variante de juegos de estrategia, por ejemplo: Medieval, Civilization, etc.
6. Los Juegos de Disparos: son juegos de acción donde se utiliza un arma o pistola para ir pasando niveles en el juego. Los más famosos son Call of Duty y Medalla de Honor. También están los juegos de disparos y táctica, donde se combinan el uso de armas y la planificación táctica, por ejemplo en SWAT.



### 2.1.2. CONCEPTUALIZACION DE LOS JUEGOS SERIOS

El cofundador de Games for Health Project y Serious Games Initiative, Ben Sawyer [4], definió el término juegos serios como cualquier uso significativo de juegos computarizados cuya principal directiva no es el entretenimiento. Los juegos serios 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. A continuación, presentaremos varios aspectos relacionados a los juegos serios.

Según [46], debido a la gran variedad de juegos existentes, es difícil establecer criterios específicos que determinen una clasificación clara de los juegos serios. Existen varias tipologías de juegos, considerando diferentes criterios. A continuación, presentaremos dos de ellas:

1. Basados en Estándares de Metadata: según [17], una categorización puede ser basada en los metadatos que se usen para caracterizar los juegos serios, los cuales permiten compartir, reutilizar e indexar los juegos serios. Dos ejemplos de metadatos son:
   *a.* **Metadatos de Objetos de Aprendizaje (LOM):** es un conjunto de metadatos para describir los recursos de enseñanza y aprendizaje, diseñado en 2002 por el comité IEEE LTSC LOM [17]. La contribución de LOM es la definición de 60 campos organizado en nueve categorías (1. General, 2. Circulo de Vida, 3. Meta-Metadata, 4. Técnica, 5. Educación, 6. Reglas, 7. Relación, 8. Anotación y 9. Clasificación).
   b. **Modelo de Referencia de Objetos de Contenido Compartible (SCORM):** es una colección de estándares y especificaciones para compartir objetos de aprendizajes en la web. También, se define cómo el contenido que se puede empaquetar en archivos transferibles, llamado "formato de intercambio de paquetes" [17]. Es basado en XML (por sus siglas en inglés, eXtensible Markup Language).



2. Basada en el Modelo G/P/S: es una clasificación general basada en tres aspectos (ver figura 2.2) [17]:
   a. **Gameplay (Jugabilidad):** se refiere a cómo puede ser utilizado el juego.
   b. **Purpose (Propósito):** se refiere al propósito final del juego, además del entretenimiento.
   c. **Scope (Ámbito):** se refiere a quien es dirigido los juegos serios: tipo de mercado, público quién lo usa, etc.


**Figura 2.2**. Representación del modelo G/P/S. Fuente: [17]

El modelo G/P/S proporciona una visión general de cómo cada juego se juega, para qué propósito, y quien es el destinatario. El uso de esa Información permite navegar rápidamente por una amplia gama de juegos serios, para elegir los que son relevantes para una necesidad dada.

### 2.1.3. FUNDAMENTOS DE LOS MOTORES DE JUEGOS SERIOS

A continuación, explicaremos algunas metodologías para el diseño de un juego serio, posteriormente, presentaremos una arquitectura base de un motor de juegos serios (los programas que se necesitan para crear juegos serios).

#### 2.1.3.1. DISEÑO DE JUEGO SERIO

Según [47], en la figura 2.3 se puede ver el proceso de diseño de un juego serio, el cual tiene cinco etapas:

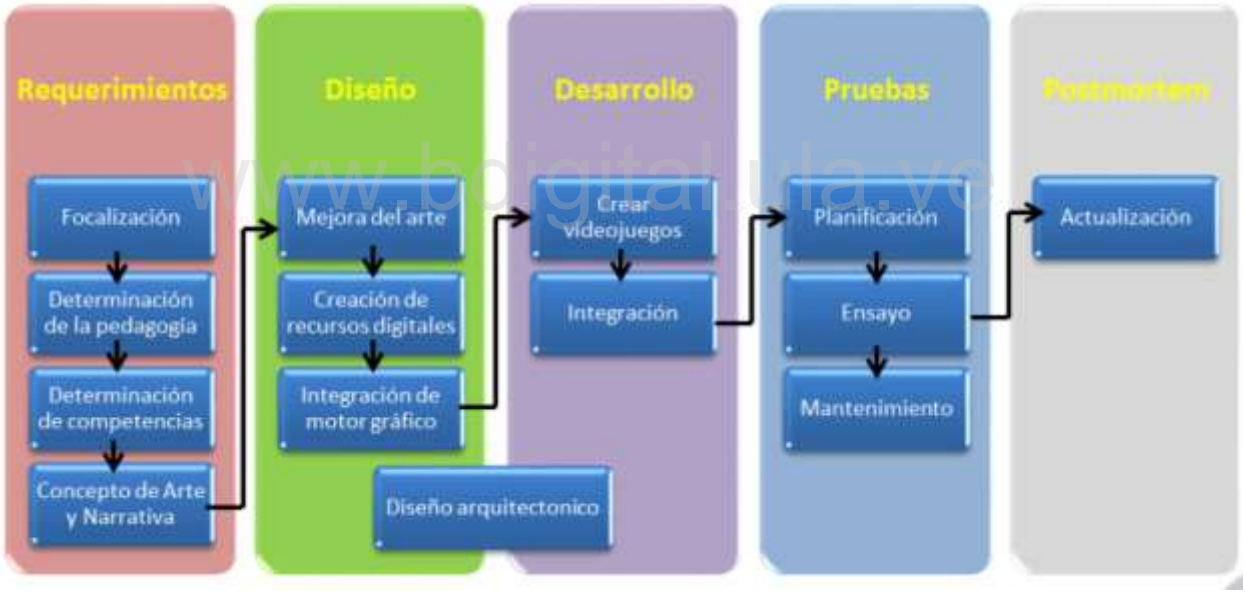



**Figura 2.3**. Proceso de diseño del juego serio. Fuente [47]

Este proceso de diseño de juegos serios se basa en los paradigmas tradicionales de ingeniería de software [47]. A continuación, se explican brevemente cada etapa del proceso:

1. Requerimientos: el objetivo es establecer metas que cubran el juego. Específicamente, establecer los mecanismos pedagógicos a través del cual el conocimiento será transferido a los estudiantes.
2. Diseño: su objetivo es crear los recursos digitales que necesita el videojuego. Estos recursos digitales incluyen: ilustraciones 2D, modelos 3D, mapas, objetos, materiales, superficies, sonidos y música.

3. Desarrollo: el objetivo es crear el juego, e integrar todos sus elementos a través de menús, opciones, etc.
4. Prueba: su objetivo es experimentar con el videojuego en los siguientes aspectos: técnico, absorción de conocimiento, usabilidad, y utilidad.
5. Postmortem: el objetivo es analizar toda la información de procesos y productos recopilada durante el desarrollo. Esto permite mejorar futuros desarrollos por intermedio de la mejora continua (actualización).

Otra metodología para la producción de juegos serios ha sido definida en [46], la cual se divide en tres etapas:

1. Análisis Contextual: consiste en la organización del proceso de desarrollo de los juegos serios. Previamente al diseño, los objetivos han de estar claramente expresados, se debe definir bien el público, y el plan de diseño se debe claramente acotar.
   a. **Objetivos***:* simbolizan lo que se desea lograr con el juego serio. No definen el Juego Serio, pero sí lo delimitan.
   b. **Público:** influye en el diseño del producto, como su entorno. Variables como el género, el sexo, la edad, la educación, el nivel social, las motivaciones e intereses, el nivel de conocimiento del alumnado, etc., influirán en las especificaciones finales del juego serio.
   c. **Plan de Diseño:** es la última premisa a tener en cuenta antes de comenzar con el diseño del juego serio. El proyecto de creación de un juego depende de los recursos, el presupuesto y el calendario definido; y cuanto antes se delimiten y se organicen, mejor se podrá predecir las consecuencias y limitaciones.



2. Desarrollo Metodológico: esta tarea realiza el diseño del juego serio, siguiendo lo especificado y partiendo de las premisas de la fase anterior, y comprende:
   a. **Definir el estilo visual:** incluyendo el uso de colores.
   b. **Plantear y definir los elementos:** estructurales de cada pantalla, como el fondo, las ventanas, etc.
   c. **Crear los elementos de control:** como botones, enlaces, etc.
   d. **Integrar elementos:** como las imágenes y textos del juego serio.
   e. ***Desarrollar* personajes:** animados e inanimados del juego serio, entre otras cosas.

3. Evaluación: una vez elaborado el juego serio, se debe validar su funcionalidad. En esta etapa se realizan pruebas del juego, reexaminando las presunciones proyectadas sobre el mismo. Se valida la interfaz, si son alcanzados los objetivos del juego serio, la actividad lúdica que permite, entre otras cosas. Según [15], sugiere que los Juegos Serios se evalúan utilizando tres criterios de evaluación:
   a. **Fidelidad:** se refiere a la cantidad de realismo.
   b. **Verificación:** se enfoca en asegurar si el juego serio está funcionando según lo previsto.
   c. **Validez:** su objetivo es garantizar que el juego serio este diseñado para modelar correctamente los procesos que existen en la realidad.

### 2.1.3.2. ARQUITECTURA DE UN MOTOR DE JUEGO SERIO

A continuación, profundizamos en la arquitectura de un motor de juegos serios, ya que nos interesa como objetivo principal de nuestra propuesta. En [48] definen a los motores de Juegos Serios como los ambientes computacionales que permiten realizar Juegos Serios. 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 están concibiendo. La arquitectura de un motor de Juegos Serios clásicamente se especifica en capas (ver figura 2.4):

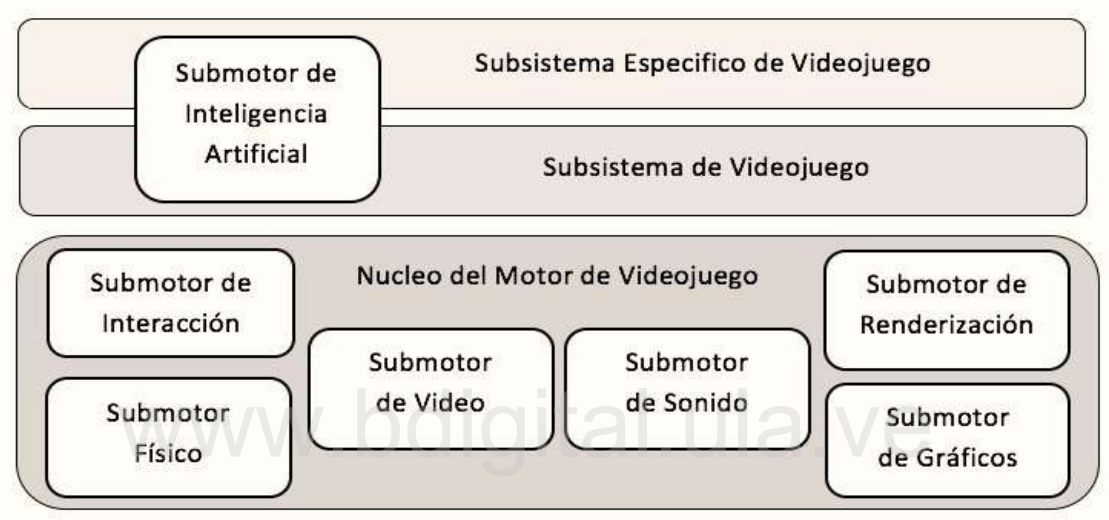



**Figura 2.4.** Arquitectura de motor de juego

1. Núcleo del Motor de Videojuego: es el elemento central del motor, 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, como son: interfaz de usuario, controladores de entrada y salidas, redes de comunicación, etc.
   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., para ser usadas en el videojuego.
   f. **Submotor de Renderización:** se encarga de gestionar las imágenes en movimiento, para darles buena calidad gráfica al videojuego, considerando aspectos como: iluminación, sombreado y oscuridad, entre otros.

2. Subsistema de Videojuego: según [48], incluye todos aquellos módulos relacionados con el funcionamiento interno del juego, es decir, agrupa tanto las características del entorno virtual como la de los distintos personajes, donde se definen las reglas que gobiernan el entorno virtual en el que se realiza el juego serio como, por ejemplo: la necesidad de obtener una victoria de un rival antes de enfrentarse a otro de más nivel. Permite, también, la definición del funcionamiento del personaje principal o avatar, así como los objetivos que se deben cumplir durante el juego.

3. Subsistema Específico de Videojuego: según [48], integra aquellos módulos responsables de ofrecer las características especiales del videojuego. En función del tipo de juego a desarrollar, en esta capa se situarán específicos módulos como, por ejemplo: los relativos al sistema de cámaras virtuales, mecanismos de inteligencia artificial específicos de los personajes no controlados por el usuario personaje no jugable, sistemas de armas, etc. Estos dos últimos subsistemas usan el siguiente submotor:

   a. **Submotor de Inteligencia Artificial:** se encarga de introducir comportamientos inteligentes en los personajes y componentes del juego, 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.

## 2.2. SISTEMAS EMERGENTES



Según [3, 5], un sistema emergente: "es lo que ocurre cuando un sistema de elementos relativamente simples se organiza espontáneamente y sin leyes explícitas, para dar lugar a comportamiento inteligente". A continuación, explicaremos la emergencia y sus conceptos vecinos:

### 2.2.1. EMERGENCIA

Según [49], se habla de emergencia cuando hay nuevas propiedades, comportamiento, estructura y patrones coherentes a nivel macro, que surgen dinámicamente de las interacciones entre las partes a nivel micro. Estas propiedades, comportamientos, estructuras, patrones, etc., son nuevas con respecto a las partes individuales del sistema. Un clásico indicador de emergencia son los patrones observables en un nivel superior, con características temporales y espaciales específicas.

En [49] se señalan las siguientes características en un proceso emergente:

1. Efecto Micro-Macro: el comportamiento global del sistema (es decir, las propiedades emergentes) es el resultado de las interacciones entre las distintas entidades del sistema.
2. Novedad Radical: el comportamiento global es nuevo con respecto a los comportamientos individuales en el nivel micro, es decir, los individuos en el nivel micro no tienen una representación explícita del comportamiento global.

3. Coherencia: se refiere a una correlación lógica y coherente de las partes.
4. Interacción de las Partes: las propiedades emergentes surgen de las interacciones entre las partes.
5. Dinamismo: las propiedades emergentes surgen como parte de la evolución del sistema en el tiempo. Tal propiedad emergente es un nuevo tipo de comportamiento, que se hace posible en un momento determinado en el tiempo.
6. Control Descentralizado: no hay control central, es decir, ninguna parte del sistema orienta el comportamiento a nivel macro.
7. Relación Bidireccional: del nivel micro al nivel macro, las partes dan lugar a una estructura emergente. En la otra dirección, la estructura emergente influye en sus partes, es decir, las propiedades a alto nivel tienen efectos en el nivel inferior.
8. Robustez y Flexibilidad: la falla o reemplazo de una entidad no causará una falla general a nivel de las propiedades emergentes. Esta flexibilidad hace que las entidades individuales puedan ser reemplazadas, y que la estructura emergente pueda permanecer.

### 2.2.2. CONCEPTOS VECINOS

1. Auto-organización: es un proceso dinámico y adaptable donde los sistemas adquieren y mantienen la estructura por sí mismos, sin control externo. La "estructura" puede ser una estructura espacial, temporal o funcional. Al "no haber un control externo" hay ausencia de dirección, manipulación, interferencia, presiones, o de participación desde fuera del sistema. Un sistema auto-organizado no sólo regula o adapta su comportamiento, sino que crea su propia organización [49].



Algunos autores confunden o asocian la emergencia con la auto-organización, incluso tratándolos a veces como sinónimos, sin embargo, ambos conceptos se complementan. Como se afirma en [49], la esencia de la emergencia es la existencia de un comportamiento global nuevo, diferente del de los elementos constitutivos del sistema, mientras que la esencia de la auto-organización es un comportamiento adaptable, que de forma autónoma adquiere para mantener un orden.

En [49] se señala la importancia e interés reciente de la auto-organización y emergencia para gestionar recursos distribuidos. La Web es un ejemplo, ya que muchas aplicaciones sobre la Web obtienen contribuciones pequeñas, a muy bajo costo, de un gran y diverso grupo de colaboradores, para producir productos y servicios de información. En [50] se señalan las siguientes características de la auto-organización:

a. **Aumento del Orden:** la organización puede ser vista como un aumento en el orden del comportamiento del sistema, que permite al sistema adquirir una estructura espacial, temporal o funcional.
b. **Autonomía:** es la ausencia de control externo. Un sistema debe organizarse sin interferencia del exterior.
c. **Adaptabilidad o Robustez:** con respecto a los Cambios, la robustez es usada en términos de la adaptabilidad en presencia de perturbaciones y cambio. Un sistema

auto-organizado se espera que haga frente a ese cambio y mantenga su organización autónomamente.

d. **Dinamismo:** la auto-organización se trata de un proceso dinámico. Con el tiempo hay un aumento en el orden, es decir, una evolución hacia más orden.

2. Retroalimentación: también referida de forma común como lazo cerrado [51], es un mecanismo por el cual una cierta proporción de la salida de un sistema se redirige a la entrada, como señal de control de su comportamiento. Según [52], existen dos tipos de sistemas principalmente: los no realimentados o de lazo abierto, y los realimentados o de lazo cerrado.

   El lazo cerrado funciona de tal manera que hace que el sistema se realimente, es decir que la salida vuelve a la entrada para que analice la diferencia, y en función de la misma haga los ajustes requeridos. Cualquier entorno que tenga como naturaleza el control de ciertas variables, como por ejemplo la temperatura, la velocidad, o la presión, puede ser realizado usando lazos cerrados. Los sistemas de lazo abierto no comparan la variable controlada con la entrada de referencia. La retroalimentación se divide en:

   **a. Positiva:** promueve la creación de estructuras a través de reglas de comportamiento simples. Por ejemplo: el rastro dejado por las hormigas a través de la feromona, o la danza en el caso de las abejas.

   **b. Negativa:** permite el equilibrio con la retroalimentación positiva, y ayuda a estabilizar el patrón colectivo. Puede tomar la forma de saturación, agotamiento o competición. Por ejemplo: la evaporación de la feromona.

   **c. Bipolar:** puede aumentar o disminuir la señal o actividad de salida. La realimentación bipolar está presente en muchos sistemas naturales y humanos. De hecho, generalmente la realimentación es bipolar, es decir, positiva y negativa según las condiciones medioambientales, como respuesta adaptativa del sistema.

3. Inteligencia Colectiva: es una disciplina clave del área de computación emergente, que define una forma de inteligencia que surge de la colaboración de muchos individuos, generalmente de una misma especie. La Inteligencia Colectiva, también llamada "Inteligencia de enjambre", es un campo de investigación multidisciplinario, que se interesa en los procesos distribuidos (no supervisados) de una organización, para resolver problemas complejos en ellas, como, por ejemplo, los que están presentes en un cierto número de sociedades animales [49]. En la literatura se han propuesto un gran número de sistemas artificiales bio-inspirados para diferentes cosas [51], coordinación de robots, auto-ensamblaje, organización de bases de datos, protección de virus, etc.

   Otros modelos y aplicaciones que se han desarrollado, basados en la inteligencia colectiva, son, por ejemplo: problemas de transporte multi-robot, análisis y clasificación de datos, entre otros. Una de las técnicas basadas en inteligencia colectiva más utilizada es el ACO [53]. Los algoritmos ACO se inspiran directamente en el comportamiento de las colonias reales de hormigas, para solucionar problemas de optimización combinatoria. Se

basan en una colonia de hormigas artificiales, esto es, agentes computacionales simples, que trabajan de manera cooperativa, y se comunican mediante rastros de feromona artificiales. Son algoritmos constructivos: en cada iteración del algoritmo, cada hormiga construye una solución al problema, recorriendo un grafo que contiene soluciones parciales.

### 2.2.3. JUEGOS EMERGENTES

En la literatura, hay varias definiciones de lo que es un juego emergente. Por ejemplo, en [5] indican que un juego emergente aparece cuando un conjunto relativamente simple de reglas conduce a complejas estrategias de juego.

En general, un juego emergente responde a las acciones del jugador, y de manera similar, es capaz de interactuar con los jugadores, lo que añade una nueva dimensión al juego [5]. Así, un juego emergente requiere un alto nivel de interactividad, para explotar la creatividad humana para encontrar soluciones. La emergencia en los juegos es posible gracias a la definición de simples reglas globales sobre el comportamiento y las propiedades del juego, así como por las interacciones entre el mundo del juego y los jugadores. La emergencia del juego permite que el juego sea más interactivo y reactivo, creando un amplio abanico de posibilidades para las acciones y estrategias, entre otras cosas. Según [5, 6], existen seis tipos de emergencia en un juego emergente:



1. Estrategia: 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 estrategias no han sido diseñadas, creadas, ni predefinidas por el diseñador del juego. Por ejemplo: estrategias de golpes y combos de ataque en videojuegos de pelea como “Street Fighter”.
2. 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 las escenas del juego. Por ejemplo: cambio de escenarios y ciudades en los “Los Sims”.
3. Propiedad: cambia las características y capacidades en los objetos, lo que puede conllevar a nuevos artefactos, personajes, etc. Eso puede implicar el cambio de normas, leyes y reglas en el videojuego. Por ejemplo: el cambio de vehículos o de los poderes de los protagonistas virtuales, como en “Megaman”.
4. Final: determina cuando debe terminar el videojuego, cambiando aspectos en el mismo. Por ejemplo, podría hacer emerger vidas infinitas, o finalizar el juego cuando se alcance un objetivo, entre otras cosas. Un ejemplo son las vidas infinitas en el juego “Mario Bros”.
5. Modelo de Negocio: 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 “Era de Imperios” la compra de madera, hierro y oro, etc.

6. Utilidad: hace emerger como se va a utilizar el juego emergente, 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.

## 2.3. TÉCNICAS DE COMPUTACIÓN INTELIGENTES

Según [59], las técnicas de computación inteligentes hacen referencia a técnicas de la inteligencia artificial usadas para la resolución de problemas de ese ámbito: razonamiento, aprendizaje, de búsqueda, entre otros. Existen muchas técnicas en esta área, algunas bioinspiradas. En la presente tesis se utilizan las descritas a continuación.

### 2.3.1. ALGORITMO DE COLONIA DE HORMIGA

ACO es un tipo de meta heurística basada en una población, inspirada en la conducta de las colonias de hormigas reales cuando buscan comida. Ha sido utilizada para resolver problemas de optimización combinatoria y está compuesto por [53, 54, 55]:



1. Espacio de solución: es un grafo o espacio que recorrerán las hormigas para obtener soluciones.
2. Hormigas: caminan en el grafo.
3. Solución: los nodos del grafo son marcados por una feromona, igual que los arcos que los interconectan. Cuando converge el algoritmo, los nodos son seleccionados según si su feromona pasa un umbral, igual que los arcos que salen de ellos, para ser parte de la solución final.
4. Feromonas: define lo deseable de los nodos y de los arcos que los interconectan, para pertenecer a la solución final.
5. Función Feromona: actualiza cada tipo de feromona, en función de la calidad de la solución propuesta.
6. Función Heurística: define la decisión heurística que toma una hormiga al estar en un nodo, con respecto a que otro nodo debe continuar visitando después de él.

El macroalgoritmo clásico de ACO [53, 54, 55] se muestra a continuación:

*Inicio*
*Inicializar parámetros, feromonas y grafo*
**Repita Mientras** *(no se cumpla terminación)*
*ConstruirSolucionesPorHormigas*
*ActualizarFeromona*
*ConstruirSoluciónFinal*
*Fin*

En general, las hormigas van recorriendo el grafo, y en la medida que los recorren van construyendo una solución posible al problema estudiado. Durante el recorrido, ellos van tomando la decisión de qué nodo visitar según una función heurística que considera la cantidad de feromona depositada en el entorno y el objetivo que se busca alcanzar en el problema.

Finalmente, cada hormiga deposita feromona en el recorrido realizado, según la calidad de la solución alcanzada. Matemáticamente, el macroalgoritmo ACO usa las siguientes ecuaciones. La hormiga $k$ se mueve del estado $x$ al estado $y$ según la siguiente probabilidad:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(n_{xy}^{\beta})}{\sum_{u \in J_x^k}(\tau_{xu}^{\alpha})(n_{xu}^{\beta})} \; si \; y \in J_x^k \qquad (2.1)$$



Donde: $\tau_{xy}$ es la cantidad de feromona que se ha depositado en el arco entre los nodos $x$ a $y$, $0 \leq \alpha$ es un parámetro para controlar la influencia de $\tau_{xy}$, $n_{xy}$ es la conveniencia de la transición del nodo $x$ $a$ $y$, $\beta \geq 1$ es un parámetro para controlar la influencia de $n_{xy}$, y $J_x^k$ es el conjunto de sitios que puede visitar la hormiga $k$ desde la posición $x$.

Cuando todas las hormigas han completado una solución, los rastros son actualizados por:

$$\tau_{xy} = (1-\rho)\tau_{xy} + \sum_k \Delta\tau_{xy}^{k} \qquad (2.2)$$

Donde: $\tau_{xy}$ es la cantidad de feromonas depositadas en el arco entre los nodos $x$ $e$ $y$, $\rho$ es el coeficiente de evaporación de feromonas, y $\Delta\tau_{xy}^{k}$ es la cantidad de feromonas depositadas por la hormiga $k$, lo cual depende de la calidad de la solución hallada por la hormiga.

### 2.3.2. ALGORITMO CULTURAL

Según [56], los algoritmos culturales fueron desarrollados por Robert G. Reynolds, como un complemento a la metáfora que usan los algoritmos de computación evolutiva (CE), que han sido exitosos en la resolución de diversos problemas de búsqueda y optimización, aún en situaciones con poco o ningún conocimiento del dominio. Sin embargo, ellos pueden mejorar su ejecución cuando se utiliza un conocimiento específico del problema, para guiarlos en su resolución.

Están basados en que la evolución cultural puede ser visto como un proceso de herencia en dos niveles: el nivel micro-evolutivo, que consiste en el material genético heredado por los padres a sus descendientes, y 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 2.5, se puede apreciar cada uno de los componentes de los algoritmos culturales:

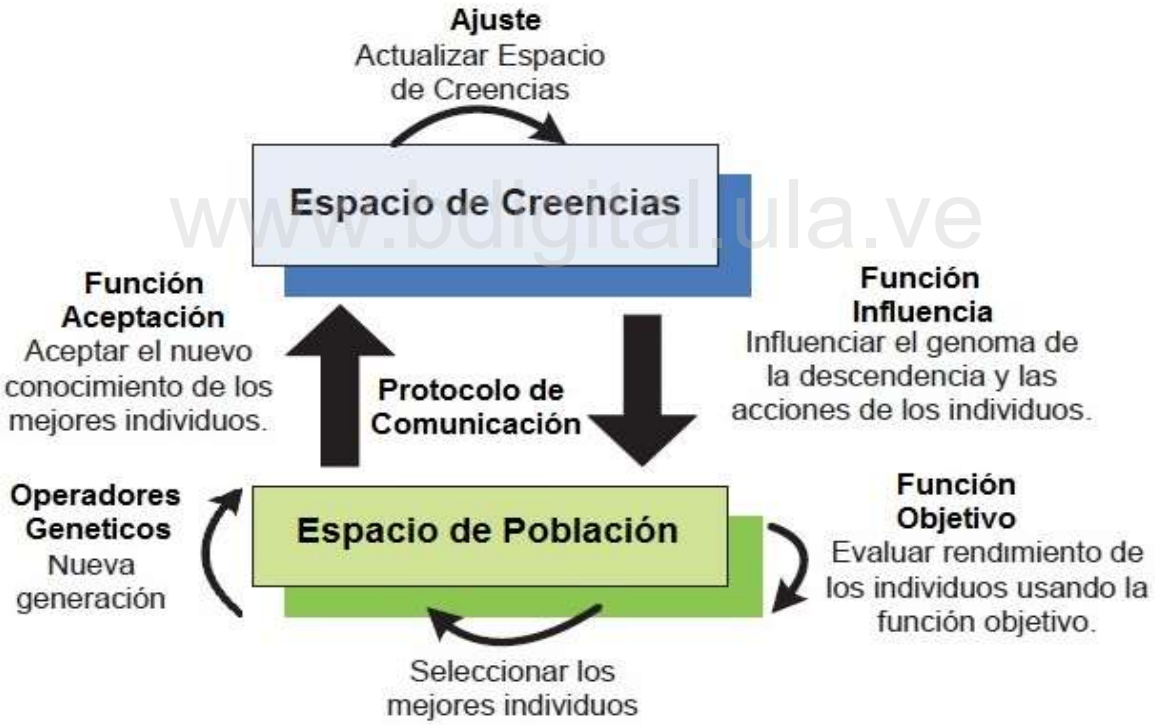



**Figura 2.5.** Estructura de los algoritmos culturales [56]

En específico, los componentes de un algoritmo cultural son:

1. Espacio de la Población: está formada por individuos. Cada individuo tiene un conjunto de características independientes de los otros, con las que es posible determinar su aptitud. A través del tiempo, tales individuos podrán ser reemplazados por algunos de sus

descendientes, obtenidos a partir de un conjunto de operadores aplicados a la población. Para ese espacio se definen:

a. **Función Objetivo:** permite evaluar el performance o el desempeño de cada individuo.
b. **Operadores Genéticos:** permiten la reproducción, modificación o variación de los individuos en el espacio de la población. Los más comunes son los operadores de cruce y mutación. A continuación, se presentan sus correspondientes descripciones [36]:
    i. *Operador de Cruce:* permiten la unión entre partes de individuos, produciendo su descendencia.
    ii. *Operador de Mutación*: modifican una parte de un individuo de manera aleatoria.

2. Espacio de Creencias: es donde se almacenarán 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:** contienen 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 de ellos, de los objetivos del dominio, entre otros conocimientos. Por ejemplo, conocimiento sobre el contexto donde se aplica el JSE.
    d. **Histórico:** son patrones temporales, comportamientos históricos a resaltar, etc.
    e. **Topográfico:** son patrones espaciales del comportamiento, características del espacio de soluciones, caminos de interés, etc.

3. Protocolo de Comunicación: las funciones de aceptación e influencia son los protocolos que permiten la interacción entre el espacio de creencias y el espacio de la población. Estas funciones actualizan ambos conocimientos, de la siguiente manera:
    a. **Función de Aceptación:** incorpora las experiencias individuales de la muestra 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 de Influencia:** determina como el conocimiento del sistema influye en los individuos de la población. Ejerce cierta presión, para que los hijos 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.

4. Macroalgoritmo Cultural: es parecido a los algoritmos evolutivos, en los que se va pasando por varias generaciones, en las cuales se va modificando la población con los operadores genéticos, pero también, en este caso, se va actualizando el espacio de creencias con nuevo conocimiento. A continuación, se presenta dicho macroalgoritmo.

*Inicio*
    $t = 0$
    *Iniciar* $JSE^t$
    *Iniciar* $B^t$
    ***Repetir***
        *Evaluar* $JSE^t$
        *Ajustar* ($B^t$, *Acepta* ($JSE^t$))
        *Variar* ($JSE^t$, *Influencia* ($B^t$))
        $t = t + 1$
    ***Hasta*** *(haber alcanzado la condición de finalizar)*
*Fin*

En el macro-algoritmo anterior, t es el número de generaciones, $JSE^t$ es el espacio de la población (donde están los individuos), $B^t$ es el espacio de creencia (conformado por un vector de elementos, en el que cada elemento está compuesto por la información de cada conocimiento considerado (situacional, normativo, dominio, histórico o topográfico)). En cada iteración (generación) se evalúan los individuos de la población $JSE^t$ por medio de la ecuación 2.1. Después, se ajusta (actualiza) el espacio de creencias $B^t$ con el conocimiento de los individuos de la actual población ($JSE^t$). A continuación, se varía (actualiza) la población $JSE^t$ usando los operadores genéticos (bajo la influencia del espacio de creencias $B^t$). Para ello, se seleccionan individuos de $JSE^t$ para generar nuevos individuos usando los operadores genéticos (algunos basados en la información en $B^t$), y se escogen los mejores entre los nuevos individuos y los actuales en $JSE^t$ para conformar la población de la siguiente generación ($JSE^{t+1}$). Lo anterior se hace hasta que el sistema converja.



### 2.3.3. SISTEMA CLASIFICADOR DIFUSO

Según [38, 57, 58], un sistema clasificador es un tipo de máquina de aprendizaje que se basa en reglas. Un SCD es un sistema clasificador 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 2.3:

$$S_i(t+1) = S_i(t) + Acti_i(\mathrm{t}) - PS_i + R_i(t) \qquad (2.3)$$

Dónde: i es el identificador de la regla; $Acti_i$(t) es el grado de activación de la regla; *PSi(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 sistema clasificador.

La función de ajuste para la función de pertenencia de un conjunto difuso $F$, cuando el operador condición es "O", viene dada por [58]:

$$S_F(t+1) = S_F(t) + Acti_i * \mu_{F[x_k]}, \qquad (2.4)$$

Y cuando el operador condición es "Y", viene dada por [58]:

$$S_F(t+1) = S_F(t) + Acti_i * \frac{1}{\mu_{F[x_k]}}, \qquad (2.5)$$

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 2.4 y 2.5 permite asignar más crédito al conjunto difuso que influyo más en el nivel de disparo de una regla [57, 58].

El peso de la regla se usa para poder tomar decisiones: a mayor cantidad de peso, más probabilidad de ser acertada y ser utilizada nuevamente; a menor cantidad de peso, el SCD tiende a borrar o no utilizar esta regla.



Las reglas se van adaptando a las decisiones exitosas tomadas (utilizando las de mayor peso continuamente, según los niveles de dificultad de las escenas), en función de los objetivos que debe cumplir el avatar o personaje principal del JSE.

Ejemplos del formato de reglas asociadas al proceso de actualización de estrategias del JSE serían:

Si <Evento=Aparece> entonces <Acción=Ocurre>
Si <Evento=No Aparece> entonces <Acción=No Ocurre>

A continuación, se dan ejemplos de algunas instanciaciones de las reglas, que podrían ser usadas para definir las estrategias en un JSE dado:

Si Evento 1=Aparece es Verdadero Y/O Evento 2=Aparece es Falso entonces
Acción=Ocurre es Alta Y/O Acción=Ocurre es baja

Esta regla indica que, si ocurre un evento 1 y/o otro evento 2 no ocurre, entonces se realizan las acciones y/o no al mismo tiempo.

El macroalgoritmo de un SCD para ajustar las reglas es:

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

En el macro-algoritmo anterior, el SCD se inicializa con las variables difusas y sus respectivos conjuntos difusos del sistema modelado, para después pasar a definirse las reglas que modelan el sistema usando esas variables. Después, en cada iteración se fusifican las entradas al sistema (eventos que ocurren en el contexto). Esos eventos constituyen los antecedentes de las reglas. A continuación, se invoca la función Activar, que es el razonador difuso del SCD, el cual permite determinar que reglas se disparan con esos eventos para generar las acciones determinadas en el consecuente de las reglas. Finalmente, la función Actualizar selecciona aquellas reglas que fueron efectivas (las más usadas/activadas) para generar nuevas reglas, y sustituir así las peores reglas (que no son usadas/disparadas o pocos usadas) por estas nuevas. Particularmente, la función Actualizar no se llama todo el tiempo, sino cada cierto número de generaciones/iteraciones, para evaluar el comportamiento de las reglas durante un intervalo de tiempo. Esto se hace hasta que dejen de llegar nuevos eventos al sistema.

# CAPÍTULO III: DISEÑO DEL SISTEMA

En el presente capitulo, se presenta el diseño del MJSE. Para ello, comenzamos por definir lo que entenderemos por JSE. Posteriormente, se especifica la arquitectura del motor de JSE. Seguidamente, se describen cada uno de sus sistemas adaptativos propuestos en este trabajo: el de secuencia que ajusta las tramas en un JSE, el de parámetros que adapta las propiedades del JSE, y el de estrategia que las adecua dinámicamente en el JSE. Finalmente, también especificamos el agente de JSE en un SaCI.

## 3.1. DEFINICIÓN DE JUEGOS SERIOS EMERGENTES

En general, los Juegos Serios son diseñados y desarrollados desde un objetivo distinto al de la pura diversión [4]. Eso ha hecho que haya sido usado en otros campos: salud, militar, ecología, social, entre otros. Por otro lado, el comportamiento de los juegos emergentes se basa en un despliega espontáneo, autónomo, y sin leyes explícitas, del juego, adecuándose a los jugadores [49]. Así, un juego emergente es un juego con vida propia, el cual toma en cuenta lo que haga el jugador y al entorno, para adaptarse.

Desde el punto de vista de un videojuego, un JSE es la mezcla de un juego serio y un juego emergente. Por ello, debe basarse en un MJSE que permita manejar las diferentes formas de emergencia en un juego emergente, pero a la vez, por sus características de Juegos Serios, esa forma de emergencia es guiada según un contexto muy específico (en nuestro caso, educativo). Entonces, el JSE posiciona al estudiante en un entorno de realimentación de información y motivación al logro, considerando: la definición explícita de un objetivo distinto a la pura diversión, y la posibilidad de superar desafíos adecuados a su capacidad humana y de aprender de sus propios errores. Así, el comportamiento del JSE es autónomo, adaptándose a las necesidades de los estudiantes, evaluando sus procesos de aprendizaje, entornos en los que interactúan, entre otras cosas.



## 3.2. ARQUITECTURA DE UN MOTOR DE JUEGOS SERIOS EMERGENTES

La arquitectura del MJSE está basada en el trabajo [55], la cual fue desarrollará para soportar un JSE en un SaCI (revisar [9] para más detalles). La arquitectura del MJSE consiste en capas jerárquicas (ver figura 3.1).

La arquitectura está conformada por tres capas, una que corresponde a los componentes de un clásico motor de videojuego (núcleo del motor de videojuego), y otras dos que se encargan de la gestión de la emergencia. Una de ellas permite la emergencia inicial del JSE adecuado al contexto donde será usado (subsistema de emergencia del videojuego, SEV), y la otra va adaptando el JSE mientras se usa a través de diferentes tipos de emergencia en el juego (subsistema adaptativo del videojuego, SAV): de secuencia, de estrategias, etc.

La descripción detallada de esas capas se encuentra en [1, 9] a continuación, se dará una descripción de ellas.


**Figura 3.1.** Arquitectura de adaptación del videojuego

### 3.2.1. NÚCLEO DE MOTOR DE VIDEOJUEGO

Es el elemento central del MJSE, en él se encuentra los seis submotores de base para cualquier videojuego [49], que se explicó en 2.1.3.2.



### 3.2.2. SUBSISTEMAS DE GESTION DE LA EMERGENCIA

Están compuesta por el SEV y de SAV, que son las capas de MJSE que permiten hacer emerger un JSE. Ambos subsistemas usan los siguientes submotores:

- *Submotor de inteligencia artificial:* 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 inteligencia artificial para permitir la emergencia en un JSE, explicado en 2.1.3.2.
  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 (ver figura 3.2). Para ello, recolecta la información del contexto, realiza la gestión de escenas y eventos, ensambla subtramas/secuencias 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.

Pasemos ahora a describir cada capa.

1. Subsistema de Emergencia del Videojuegos: en el caso del STE, permite hacer emerger la primera versión del JSE que será usado, según los objetivos que se debe cumplir con el JSE. El STE, cuando es invocado por el SEV, está compuesto por los siguientes componentes (ver figura 3.2 y [1, 55], para más detalles):

*a.* **Gestor de Materia:** determina la temática que se está tratando en el contexto para, a partir de allí, establecer el objetivo que debe cubrir el JSE. Así, crea una emergencia de utilidad definiendo para qué va a ser utilizado el videojuego.
*b.* **Gestor de Videojuegos:** 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 *Gestor de Materia*. Si no consigue al menos un videojuego parecido a lo buscado, llama al módulo siguiente.
*c.* **Módulo de Generación de JSE (MGJSE):** es el responsable del ensamblaje de un nuevo JSE usando las subtramas provistas por el *Gestor de Videojuegos*. En un trabajo previo, se ha definido el MGJSE basado en ACO [55], El MGJSE tiene imbricadas las funciones de los siguientes tres componentes del SEV, para generar inicialmente un JSE.
*d.* **Storyboard:** se encarga de generar los guiones narrativos o subtramas del JSE.
*e.* **Gestor de Escenas:** genera el ambiente o entorno, requerido por las tramas del JSE.
*f.* ***Sistemas de Eventos:*** se encarga de generar eventos especializados requeridos por el *Storyboard*, para generar los comportamientos deseados en el JSE.

*2.* Subsistema Adaptativo del Videojuego: en el caso del STE, permite ir adaptando al JSE durante el desarrollo del mismo. La capa SAV permite que en un JSE se generen comportamientos emergentes durante el juego, actuando sobre sus características de base. En particular, en [55] se proponen 6 niveles de emergencia en un JSE, que se dividen en dos módulos, a saber:



*a.* **Módulo de Emergencia Fuerte:** este módulo está compuesto de tres subcapas, que permiten las siguientes emergencias fuertes en el videojuego:
  i. *Estrategias:* se generan nuevas tácticas y logística en el juego.
  ii. *Secuencia:* se crean nuevos escenarios o tramas en los juegos, cambiando el ambiente y el contexto del juego.
  iii. *Propiedad:* cambia las características en los objetos en el JSE.
*b.* **Módulo de Emergencia Débil:** este módulo está compuesto por tres subcapas, que permiten las siguientes emergencias débiles:
  *i.* *Final:* patrones que determinan la culminación de los juegos, o continuar el juego de forma infinita.
  *ii.* *Modelo de Negocio:* surgimiento de mercados y servicios alrededor de los JSEs.
  *iii.* *Utilidad:* uso del JSE

En la presente tesis solo nos dedicaremos a implementar los módulos para la emergencia fuerte.

La figura 3.2 describe de manera general al STE, que como se comentó antes, es invocado tanto por el SEV como por el SAV, para permitir la emergencia en un JSE. En el caso del SEV, realiza la emergencia de la primera versión del JSE adaptado al contexto donde será usado, para lo cual usa los componentes de base para la construcción de un JSE (gestor de escenas, etc.).

En el caso del SAV, adecua el JSE durante su uso utilizando diferentes mecanismos de emergencia: estrategias, secuencias, etc. Todo ello ocurre sin modificar el núcleo de motor de videojuegos.

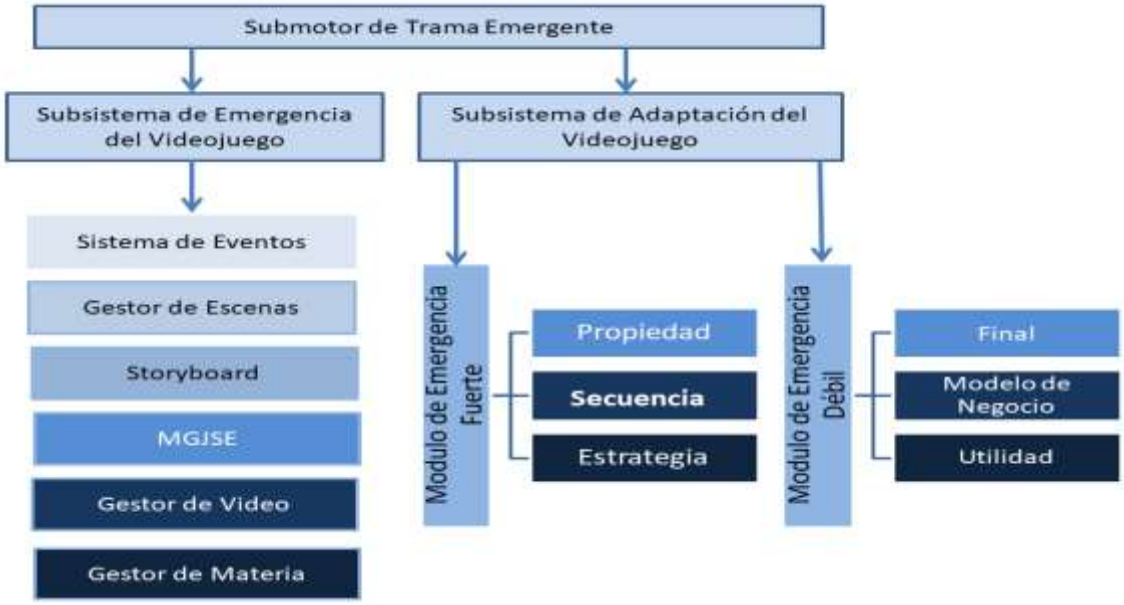





**Figura 3.2.** Submotor de trama emergente

## 3.3. SISTEMAS ADAPTATIVOS DEL MJSE

A continuación, describimos los tres sistemas adaptativos propuestos en esta tesis, que posibilitan la emergencia en un MJSE.

### 3.3.1. SISTEMAS DE ADAPTACION DE SECUENCIAS

El MJSE supervisa permanentemente el JSE en ejecución, con el fin de adaptarlo al contexto, invocando para ello al Sistema de Adaptación de Secuencias/Trama (SAT). El SAT permite la adaptación de las secuencias en un videojuego, de forma tal que en un JSE existente se generen comportamientos emergentes (cambio en las tramas o escenarios). Para realizar esas tareas, se apoyará en el submotor de inteligencia artificial y STE. La aparición de secuencia se caracteriza por crear nuevas tramas, cambiar el orden en las actual, o eliminar/añadir tramas en los juegos. Este tipo de emergencia utiliza ACO y un sistema de búsqueda en repositorios de videojuegos (ver figura 3.3). Para el desarrollo del SAT, el uso del ACO es adecuado porque permite considerar diferentes tramas o Juegos Serios para la actualización de un JSE. En particular, la posible configuración inicial de un JSE se define tras el análisis de diferentes combinaciones de Juegos Serios y/o tramas. Por otro lado, para adaptar un JSE en tiempo real a los temas impartidos en un SaCI, la combinación de las tramas que lo componen con nuevos Juegos Serios y/o tramas que se encuentran en Repositorios de Objetos de Aprendizaje (ROAs) permiten realizar dicha adaptación.

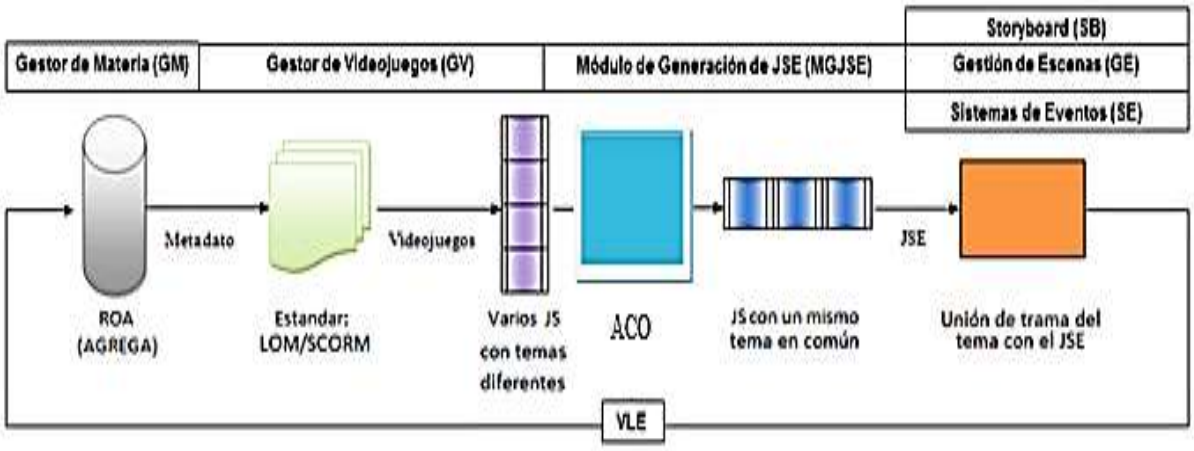



**Figura 3.3.** Componentes del STE

Específicamente, para la emergencia de secuencia es necesario determinar la brecha entre lo que está sucediendo en el contexto y lo que se definió en el diseño del JSE. Para ello, este componente define una métrica, denominada "medidor de narrativa y jugabilidad", que se encarga de medir dicha brecha, con el fin de determinar la disonancia ludonarrativa a cubrir. En general, el STE del MJSE, al invocar el SAT, se compone de los siguientes componentes (ver figura 3.4):



1. Gestor de Materias: determina toda la información sobre el tema deseado, según el contexto donde se utilizará el juego.
2. Gestor de Videojuegos: seleccionada de los ROAs los Juegos Serios/tramas más compatibles con la temática deseada previamente determinada. Para ello, este componente analiza los metadatos de los juegos serios en los ROAs, que se describen en uno de los estándares de metadatos de objetos de aprendizaje, como LOM (metadatos de objetos de aprendizaje) o SCORM [17].
3. El MGJSE: invoca al algoritmo ACO para actualizar (hacer emerger) una nueva versión del JSE compatible con el tema deseado.
4. Solución final: propuesta por el MGJSE**,** es una versión actualizada del JSE que ha modificado sus tramas/escenas.

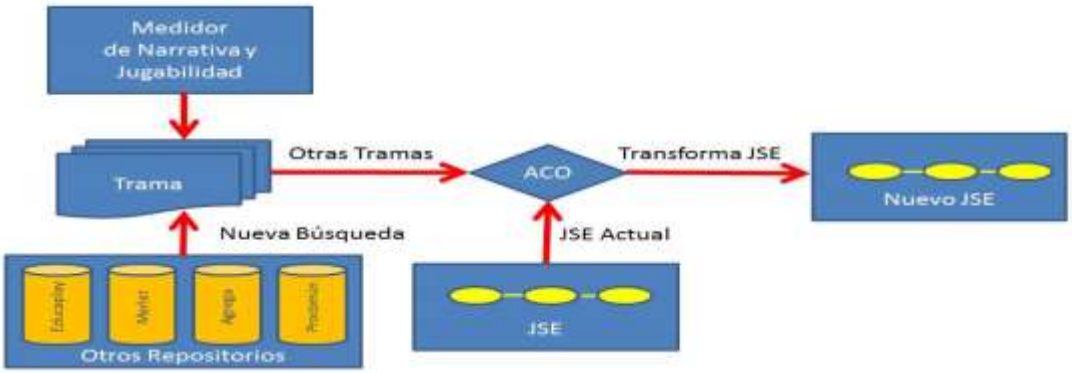



**Figura 3.4.** Proceso de emergencia de tramas en un JSE usando SAT

El MJSE supervisa la dinámica del JSE bajo ejecución, para irlo adecuando al contexto, haciendo emerger nuevas secuencias/tramas (llama para ello a ACO), pero también, eventualmente, usa otros sistemas adaptativos para hacer emerger en el JSE propiedades, estrategias, entre otras cosas [3].

En esta emergencia se sigue el siguiente proceso (ver figura 3.4): inicialmente se calcula la disonancia ludonarrativa. 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 de tramas del actual JSE, y con las nuevas tramas conseguidas intenta transformarlo, o proponer un nuevo JSE, que siga la temática deseada.

SAT sigue el siguiente pseudocódigo:

a. ***Gestor de Materias:*** define los metadatos del tema deseado
b. **Gestor de Videojuegos:** busca ROAs similares al tema deseado
c. **MGJSE:** construye una nueva versión de JSE utilizando los ROAs seleccionados y el algoritmo ACO.

#### 3.3.1.1. MACROALGORITMO ACO PARA EL SAT



En nuestro caso, ACO usa agentes hormigas, y cada una construirá un JSE para un mismo objetivo (temática deseada). En particular, un videojuego seleccionado de un repositorio de juegos lo denominaremos subtrama, mientras que una trama de un JSE es la unión/fusión de una o más subtramas. ACO está compuesto por:

1. Espacio de solución: es el espacio que recorrerán las hormigas para obtener soluciones. Es un grafo compuesto por nodos que representan las subtramas seleccionadas desde los diferentes repositorios, y los arcos que establecen las relaciones de dependencia entre ellas (la secuencia lógica entre las subtramas). Además, el grafo incluye un subgrafo que representa el actual JSE en ejecución.
2. Hormigas: caminan en el grafo de subtramas.
3. Solución: las subtramas (nodos) están marcadas por una feromona, igual que los arcos que las interconectan. Cuando converge ACO, las subtramas se seleccionan en función de si sus niveles de feromonas superan un umbral, lo que establece la secuencia lógica de ejecución de las subtramas.
4. Feromonas: hay dos tipos de feromonas, una para las subtramas (nodos), que define que tan deseable es ($\tau_{(r)}$), y otra para los arcos, que definen los nodos contiguos deseables desde un nodo r ($\tau_{(r,s)}$). La primera feromona $\tau_{(r)}$ se utiliza para seleccionar las subtramas en la solución final, y la segunda $\tau_{(r,s)}$ para determinar la secuencia entre ellas. Los nodos

con un $\tau_{(r)}$ superior a un umbral dado pertenecen a la solución final (JSE), y los valores de $\tau_{(r,s)}$ se utilizan para determinar el camino (secuencia de subtramas en el JSE).

5. Función Feromona: actualiza cada tipo de feromona en función de la calidad del JSE propuesto.
6. Función Heurística: define la decisión que toma una hormiga para pasar al siguiente nodo no visitado.

ACO establece un proceso de aprendizaje colectivo entre las hormigas, con el fin de permitir el surgimiento de la nueva configuración del JSE. Así, la solución no es más que una secuencia de subtramas, dispuestas según una secuencia lógica entre ellas.

El macroalgoritmo que se usa es el clásico de ACO [55]. En específico, en nuestro caso consiste de una fase de inicialización de parámetros, un proceso iterativo hasta que el sistema converja, y la construcción de la solución final (ver sección 2.3.1). Dicho macroalgoritmo se detalla a continuación:

*a.* **Creación del grafo teórico (grafo inicial):** el grafo teórico es definido como *G= (N, E),* donde N es un conjunto de nodos que representan las subtramas (juegos serios seleccionados por el *Gestor de Videojuego* de ROAs), y E un conjunto de arcos que conectan todos los nodos de N (ver figura 3.5). 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 $(i, j) \in E$, tal que es 1 si existe una relación de dependencia secuencial entre dos nodos $(JS_i, JS_j, \in N)$, y 0 en caso contrario. Eso implica que $d_{ij} \neq 0$ cuando entre dos nodos hay una relación de dependencia entre ellos (ver figura 3.5).



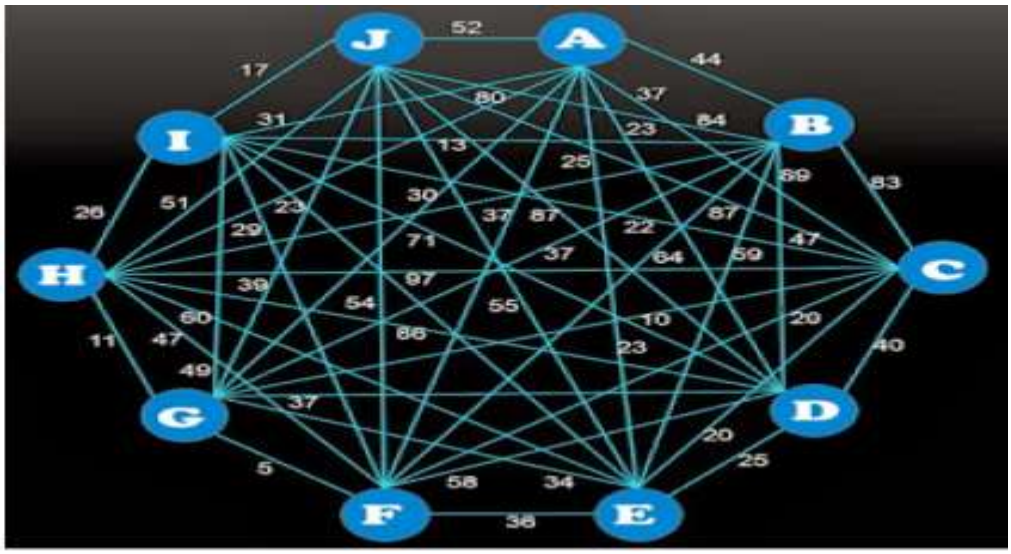



**Figura 3.5.** Grafo teórico

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 sus metadatos, y calculado por el Gestor de Videojuegos), y el nivel de feromona actualizado por las hormigas que transitan a través de ellos, que indica la deseabilidad de dicho nodo (ver figura 3.6).

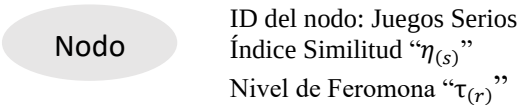



**Figura 3.6.** Nodo del grafo

*b.* **Construcción de la solución por parte de las hormigas (ConstruirSolucionesporHormigas):** 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 determinar 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 a 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*, según los niveles de feromonas del arco *r* y s ($\tau_{(r,s)}$), ponderada por el parámetro α que determina su influencia (α ≥ 0), y el índice de similitud de cada nodo s ($\eta_{(s)}$) en relación con el tema deseado, el cual es ponderado por otro parámetro β que determina su influencia (β ≥ 0). Esta función heurística se calcula para todos los nodos aún no visitados por la hormiga $k$ ( $J_r^k$), utilizando la siguiente ecuación:



$$\mathrm{P}_{(r,s)}^{k} = \frac{\tau_{(r,s)}^{\alpha} \cdot \eta_{(s)}^{\beta}}{\Sigma_{\mathrm{u}\,\epsilon\,\mathrm{J}_r^k} \tau_{(r,u)}^{\alpha} \cdot \eta_{(u)}^{\beta}} \text{ If } s \,\epsilon\, J_r^k \qquad (3.1)$$

Una hormiga puede culminar en cualquier momento la construcción de una solución (JSE), o continuar deambulando por los nodos, visitando cada uno de ellos, en base al siguiente algoritmo:

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

*c.* **Actualización de los Feromonas:** en este caso hay dos feromonas, uno para los arcos y otro para los nodos. Ambos se actualizan al final de cada iteración (camino/solución de cada hormiga), tal como se define en [55]. En ese sentido, cada hormiga actualiza la feromona de cada borde y cada nodo que visitó. Para ello se determina un índice de la

calidad del JSE propuesto por cada hormiga, el cual será utilizado durante el proceso de actualización. Este índice de calidad se calcula teniendo en cuenta el nivel de similitud de los JSE propuestos por cada hormiga en relación con el tema deseado (definido por *Gestor de Videojuegos*). El nivel de similitud del JSE propuesto por una hormiga k es el promedio de los índices de similitud "$\eta_{(s)}$ " de las *P* subtramas que definen el JSE propuesto por la hormiga *k* ($JSE^k$):

$$\Delta\tau_{(r)}^{k} = \frac{\Sigma_{k=1}^{P}\eta_{(k)}}{P} \; para\ la\ P \text{ subtrama} \in \text{JSE}^k$$

De esta forma, la *feromona* de cada nodo y arco se actualiza mediante las siguientes ecuaciones genéricas:

$$\tau_{(r,s)} = \tau_{(r,s)}(1-\rho) + \sum_{k\ and\ (r,s)\ \epsilon\ \text{JSE}^k} \Delta\,\tau_{(r,s)}^{k}$$

$$\tau_{(r)} = \tau_{(r)}(1-\rho) + \sum_{k\ and\ r\ \epsilon\ \text{JSE}^k} \Delta\,\tau_{(r)}^{k} \qquad (3.2)$$



Donde, ρ ∈ (0,1) es el coeficiente de evaporación de feromonas, y *k= 1, M,* y *M* es el número de hormigas en la colonia. En general, este proceso se lleva a cabo repetidamente, hasta que la colonia converge en un grupo de soluciones (JSE).

*d.* **Construcción de la solución final (ConstruirSolucionFinal):** 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 visita cada nodo del grafo, seleccionando los nodos con mayor valor de feromonas, y los arcos que los conectan se seleccionarán según su valor de feromonas (serán los que tengan mayor valor también), para garantizar que todos los nodos seleccionados (subtramas) formen un camino (esta será la secuencia lógica del nuevo JSE propuesto).

### 3.3.2. SISTEMA ADAPTATIVO DE PARÁMETROS

En esta sección se expone el diseño del Sistema Adaptativo de Parámetros (SAP), utilizando algoritmo cultural. A continuación, se explican cada uno de los elementos que conforman tanto el espacio de población como el espacio de creencias del algoritmo cultural, así como también, las funciones de aceptación e influencia. En el apéndice se detallan los aspectos de implementación.

### 3.3.2.1. DISEÑO DEL ESPACIO DE POBLACIÓN

Cada individuo representa las múltiples ejecuciones de un JSE dado por grupos de jugadores/estudiantes, los cuales pueden o no ser los mismos cada vez. En específico, un individuo está constituido por los valores de los diferentes parámetros del JSE que se usaron en cada uno de esos juegos, así como también, el valor de desempeño del mismo.

En la tabla 3.1: $P_j$ son los valores de los j-ésimo parámetros usados en ese juego en la i-ésima vez que se jugó, y $FO_i$ representa el desempeño del individuo (JSE en esa jugada).

**Tabla 3.1.** Estructura de un individuo en el espacio de población

| $P_1$ | $P_2$ | … | $P_j$ | $FO_i$ |
|---|---|---|---|---|

Por lo tanto, el espacio de población se define como el conjunto de estos individuos (múltiples ejecuciones del JSE), donde n representa la n-ésima vez que se jugó, como se muestra en la tabla 3.2.



**Tabla 3.2.** Estructura del espacio de población

| $Individuo_1$ | $Individuo_2$ | … | $Individuo_n$ |
|---|---|---|---|

La Función Objetivo (FO) permite evaluar el desempeño del JSE, según el objetivo para el cual se diseñó el mismo, de la siguiente manera:

$$FO_i = a * PA_i - b * LE_i \qquad (3.3)$$

Donde, $a$ y $b$ son constantes definidas por el usuario, que permiten normalizar las unidades en la función, $PA_i$ representa si el jugador alcanzó o no el objetivo definido para el JSE (Así, es 0 si no se alcanza, 1 si lo alcanza, y entre 0 y 1 los casos intermedios: $0 \leq PA_i \leq 1$), $LE_i$ es el lapso de tiempo que duró el JSE (o tiempo de ejecución. Si es 0 significa que el jugador duró poco tiempo, si es 1 que el jugador duró mucho, y entre 0 y 1, los casos intermedios: $0 \leq LE_i \leq 1$). La *FO* permite modelar quien jugó más rápido (o lo termina en menos movimientos), y, además, si el jugador logró alcanzar los objetivos para los que se hizo

el JSE. Si se obtiene un resultado negativo (< 0) significa que el jugador duró demasiado tiempo, y/o no logró alcanzar los objetivos requeridos. Por lo tanto, el mejor individuo será aquel que maximice la ecuación 3.3.

### 3.3.2.2. DISEÑO DEL ESPACIO DE CREENCIAS

Según la teoría de algoritmo cultural se tienen cinco conocimientos, pero en este trabajo se usan cuatro, de los cuales solo dos serán usados para los experimentos. Los cuatro son: Conocimiento Situacional, Conocimiento Normativo, Conocimientos de Dominio e Histórico, y los dos usados para el desarrollo del prototipo son: Conocimiento Situacional y Conocimiento Normativo. A continuación, se detallan dichos conocimientos.

1. Conocimiento Situacional: contiene ejemplos de éxitos de los JSEs. 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, que es calculada usando la *FO* de los individuos donde se usaron esos valores (ver tabla 3.3).



**Tabla 3.3.** Representación del conocimiento situacional

| $P_i$ | $V_j$ | $IO_j$ | $C_j$ |
|---|---|---|---|
| | | | |

2. Conocimiento Normativo: define los rangos de valores idóneos de cada uno de los parámetros del juego serio. En la tabla 3.4, LI y LS son los límites inferiores y superiores de cada parámetro $P_i$.

**Tabla 3.4**. Representación del conocimiento normativo

| $P^1$ | | $P^2$ | | … | $P^u$ | |
|---|---|---|---|---|---|---|
| $LI^1$ | $LS^1$ | $LI^2$ | $LS^2$ | | $LI^u$ | $LS^u$ |

3. Conocimiento de Dominio: en nuestro caso, indica los contextos donde se puede aplicar el juego serio, con los valores de parámetro adecuados para cada uno de ellos. En la tabla 3.5, $D_i$ representa el dominio i para el conjunto de parámetros $P_j$ de un JSE, 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, determinada usando la *FO* de los individuos en ese dominio.

**Tabla 3.5.** Representación del conocimiento de dominio

| $D_i$ | $FC_i$ | $P^1$ | $P^2$ | … | $P^u$ |
|---|---|---|---|---|---|
| | | valor ideal | valor ideal | | valor ideal |

4. 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 3.6), determinada usando la *FO* de los individuos que tienen esos valores de parámetros.

**Tabla 3.6**. Representación del conocimiento histórico

| $Evento_k$ | $P^1$ | $P^2$ | … | $P^u$ | $FE_k$ |
|---|---|---|---|---|---|



### 3.3.2.3. DISEÑO DEL PROTOCOLO DE COMUNICACIÓN

En esta parte, se establecen las reglas para la interacción entre el espacio de población y el espacio de creencias. Para ello, se usan las funciones de aceptación e influencia.

1. Función de Aceptación: según Reynolds [36], con un 20% de los individuos con mejor desempeño de la población, es suficiente para nutrir con sus experiencias el espacio de creencias. Este trabajo sigue ese criterio al usar esta función, que consiste en actualizar los diferentes tipos de conocimiento: en el conocimiento situacional, los valores de $C_j$ e $IO_j$, " j = 1, 2, … u; y en el conocimiento normativo, los valores de $LI_j$ y $LS_j$, " j = 1, 2, … u.

En el caso del *conocimiento 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 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.

En el conocimiento situacional, se actualiza de la siguiente manera: $IO_j$ es actualizado mediante la ecuación 3.4.

$$IO_j = NO_j + IO_j \qquad (3.4)$$

Donde, $NO_j$ es el nuevo número de ocurrencia del valor $V_j$ en la actual generación. A su vez, $C_j$ también se actualiza con la ecuación 3.5:

$$C_j = C_j * \overline{m} + \overline{C_j} * m \qquad (3.5)$$

Donde, $\overline{m}$ es el complemento del momento, es decir, $(1 - m)$, $\overline{C_j}$ es el promedio del valor $C_j$ de todos los individuos dentro del 20%, provenientes de la población actual con el valor de $V_j$. Finalmente, el momento $m$ viene dado por la ecuación 3.6:

$$m = \frac{\mu}{t} \qquad (3.6)$$



Tal que μ es una constante de momento entre 0 y 1, y t es el número de generaciones $(t = 1, 2, 3, \dots)$.

En el *conocimiento normativo*, los valores de $LI_j$ y $LS_j$ se actualizan si en la actual población, alguno de sus individuos tiene un valor superior a $LS_j$ o inferior a $LI_j$. 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 i-ésimo dominio $D_i$, se actualiza la matriz de la tabla 3.5 con una nueva fila. Si los valores ideales para un dominio i ya existente y 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 ecuación 3.4. 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:

$$FC_i = FC_i * \overline{m} + \overline{FC_\iota} * m \qquad (3.7)$$

Donde, $FC_i$ es la función de calidad actual, $\overline{FC_\iota}$ es el promedio del valor $FC_i$ de todos los individuos dentro del 20%, provenientes de la población actual en el dominio $D_i$.

Finalmente, en el *conocimiento histórico*, si los valores ideales de los parámetros ($P_i$) para un evento k deben cambiar, simplemente se sustituyen en la tabla 3.6. En este caso, los valores ideales para un evento k son los que maximizan la ecuación 3.4. Ahora bien, si solo se debe actualizar el valor de $FE_i$ de los actuales valores ideales, se debe usar la siguiente ecuación:

$$FE_i = FE_i * \overline{m} + \overline{FE_\iota} * m \qquad (3.8)$$

Donde, $\overline{FE_\iota}$ es el promedio del valor $FE_i$ de todos los individuos dentro de los 20%, provenientes de la población actual con el evento k.



2. Función de Influencia: se usa el conocimiento almacenado en el espacio de creencias para realizar operaciones de mutación guiada en el espacio de población. El *conocimiento situacional* permite una mutación directa a los valores $V_j$ de un Pi dado, dichos valores se determinan por su mejor calidad $C_j$. Si se usa el *conocimiento normati*vo, permite reajustar el valor de $V_j$ en $P_i$, generando un valor aleatorio que esté dentro de los límites (rango) definidos por dicho conocimiento. El *conocimiento de dominio* usa los valores ideales de $P_j$ para el dominio en el que se quiere diseñar el JSE, y *el conocimiento histórico* usa los valores ideales de $P_j$, según el evento k que se desee que aparezca en el JSE.

En la figura 3.7 se muestra un ejemplo de mutación dirigida por el conocimiento situacional.

Tabla de conocimiento situacional para el parámetro 3

| Vj | IOj | Cj |
|---|---|---|
| 10 | 5 | 170,6 |
| 7 | 3 | 176,3 |
| 2 | 2 | 154.3 |

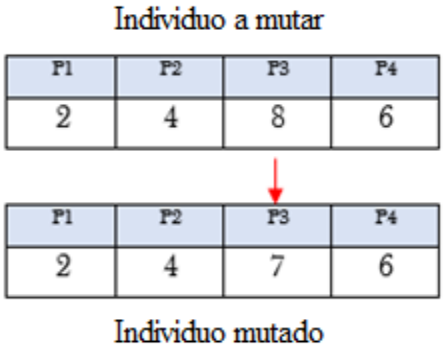
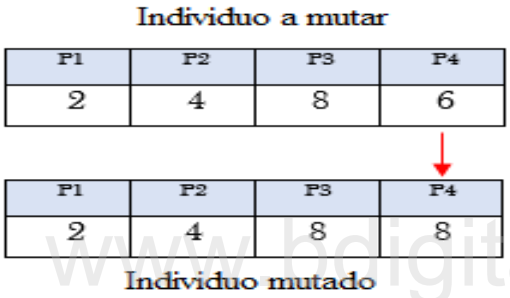



**Figura 3.7.** Ejemplo de mutación dirigida por conocimiento situacional

A su vez, en la figura 3.8 se muestra un ejemplo de mutación guiada por el conocimiento normativo.




Parámetro i elegido: 4

$LI^4$: 5

$LS^4$: 10

Valor generado: 8



**Figura 3.8.** Ejemplo de mutación dirigida por conocimiento normativo

### 3.3.3. SISTEMA ADAPTATIVO DE ESTRATEGIA

Este sistema adaptativo permite la emergencia de estrategias en un JSE, generando nuevas tácticas y logísticas en el juego, sin dejar de seguir las normas, leyes y reglas del mismo. Para realizar esas tareas, se apoya en el submotor de inteligencia artificial y STE. El punto relevante, en este caso, es definir como se modelarán las estrategias. En nuestro caso, ellas serán definidas por reglas, en las cuales en el antecedente de la regla se establece qué 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 un SCD. En la figura 3.9 se presenta el modelo de SCD a usar.

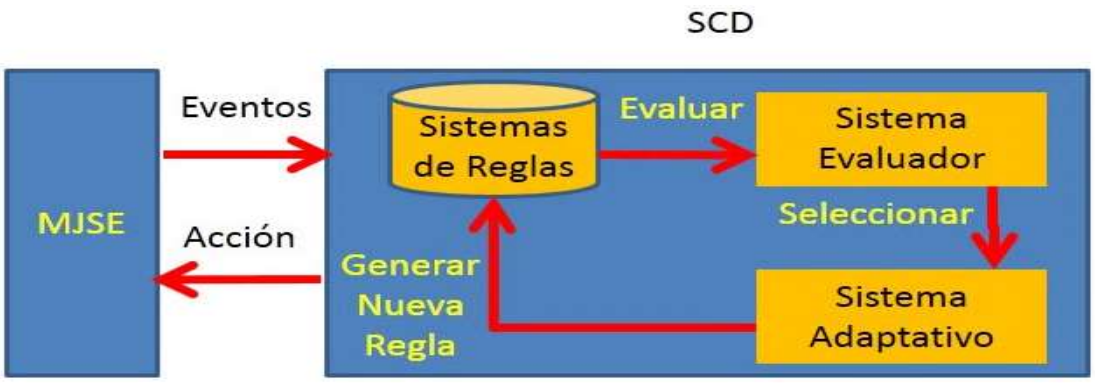



**Figura 3.9**. Modelo de SCD

1. MJSE*:* generan eventos cada uno de sus submotores
2. Sistemas de Reglas: este módulo contiene las reglas del sistema y recibe los eventos que van ocurriendo en el juego, para luego fusificarlos.
3. Sistema Evaluador*:* según los eventos recibidos, se activan las respectivas reglas. En menos palabras, es el razonador difuso del SCD.
4. Sistema Adaptativo: el SCD va adaptando las reglas, en función de sus comportamientos durante el juego. Aquellas más efectivas (más usadas, permiten alcanzar el objetivo del juego, etc.), perduran más en el tiempo, generando nuevas reglas.
5. A continuación, se definen todos los elementos que permiten caracterizar el SCD para la emergencia de estrategias.



### 3.3.3.1. VARIABLES DIFUSAS Y FUNCIONES DE PERTENENCIAS

A continuación, se describen las variables difusas que permiten caracterizar el contexto de un JSE.

1. *Las Variables Difusas y Conjuntos Difusos:* se definen para cada uno de los eventos de cada uno de los submotores, como de las acciones en el JSE. Ellas son*:*
   a. ***Contexto JSE (CJ):*** representan variables que definen el tema del JSE [60] ($CJ_x$ 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).
   b. **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 son: Verdadero (V), Más o menos (MM) y Falso (F).
   c. **Evento Acústico (EA):** representa tipos de eventos auditivos, como: cantar, gritar, el sonido de un rayo, el ruido al partirse un vaso, etc. ($EA_x$ donde [x=0,1,2,3…p]). Los conjuntos difusos son: V, MM y F.

*d.* **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. Los conjuntos difusos considerado para estas variables son: N, E, M y A.

*e.* **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, un videoclip, etc. ($EV_x$, donde [x=0,1,2,3…q] representa los tipos de eventos videos). Los conjuntos difusos son: V, MM y F.

*f.* **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.

*g.* **Acción de Destreza (AD):** son teclas especiales del videojuego que varían según el tipo de habilidad a permitir: saltar, agachar, abrir, cerrar, patear, agarrar, soltar, etc. o cualquier otra actividad en el JSE ($AD_x$ para [x=0,1,2,3…t]). Los conjuntos difusos son: N, E, M y A.

*h.* **Acción Avanzada (AA):** define una acción que es disparada por un algoritmo de inteligencia artificial después que obtiene una solución ($AA_x$ [x=0,1,2,3…v]). Por ejemplo, después de utilizar los árboles de comportamiento (BT) en Halo 2, o min-max en ajedrez. Los conjuntos difusos son: N, E, M y A.

2. *Las Funciones de Pertenencia:* se definen para 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* son caracterizadas por las funciones de pertenencia mostradas en la figura 3.10.



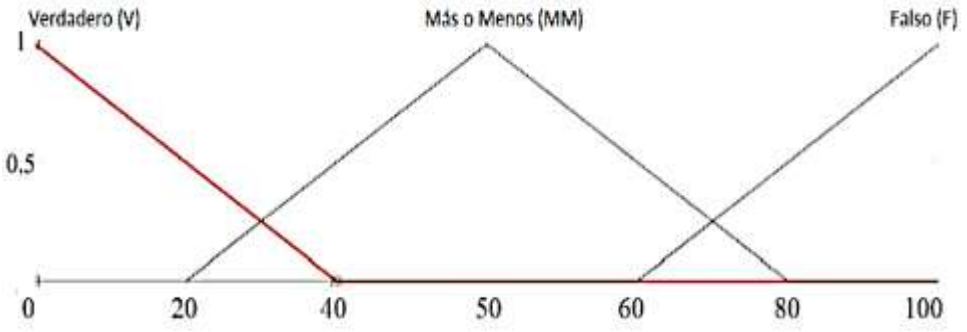



**Figura 3.10.** Función de pertenencia de EF, EA, EV

Cada una de esas variables difusas tiene un universo de discurso del [0%, 100%], y están compuestas por los conjuntos difusos V, MM y F (como se muestra en la figura 3.10). Cada conjunto difuso tiene una función de pertenencia triangular, variando entre 0-40%, 20-80% y 60-100%, respectivamente.

Las variables difusas CJ, AD, EC, AM, AD y AA se caracterizan por las funciones de pertenencia mostradas en la figura 3.11, y están compuestas por los conjuntos difusos N, E, M y A. Cada conjunto difuso tiene una función de pertenencia triangular, variando entre 0-20%, 10-50%, 40-80% y 70-100%, respectivamente.

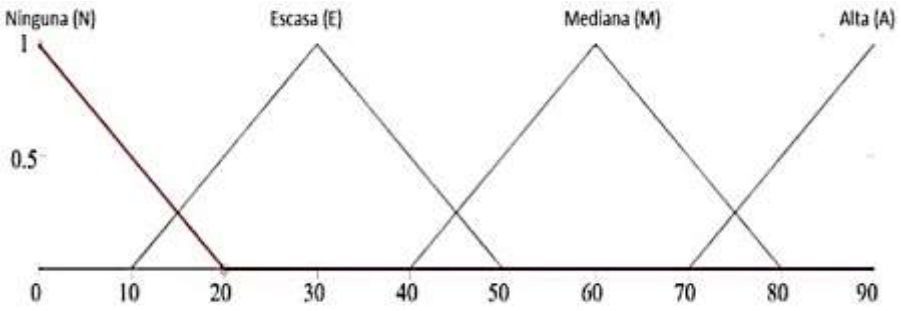



**Figura 3.11.** Funciones de pertenencia de CJ, AD, EC, AM, AD y AA

### 3.3.3.2. DEFINICION DE LAS REGLAS DE CONTROL GENÉRICAS

A continuación, se presenta ejemplos de reglas genéricas asociadas al proceso de actualización de estrategias, las cuales se dividen según el tipo de estrategia considerada:



1. JSE de agilidad: son juegos de saltos y poderes como, por ejemplo: Mario Bros, Donkey Kong, etc. Un ejemplo de sus posibles reglas sería:
   Si <EF=hueco> entonces
   <AM=saltar_adelante>
   Si <EF=enemigo> entonces
   <AD=disparar_enemigo>

2. JSE de conjetura: son juegos de cálculo como, por ejemplo: suma, resta, multiplicación, memoria, etc.
   Si <CJ=sumar>entonces <AA=calcular>
   Si <EO=valido>entonces <AM=mover>
   Si <EO=vacío>entonces <AM=mover>

3. 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 también otros grupos de reglas de estrategias, vinculadas a JSE de lucha, rompecabezas, entre otros.

A continuación, se dan ejemplos de algunas instanciaciones de las reglas genéricas, que podrían ser usadas para definir las estrategias 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 ocurre un evento físico de un hueco y otro evento físico de un enemigo, entonces se realizan las acciones de 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 de gritar, entonces se invoca a una técnica de inteligencia artificial que permita establecer la estrategia de protegerse.

## 3.4. USO DE UN JSE EN UN SALÓN DE CLASES INTELIGENTES

A continuación, se explica la caracterización de un SaCI y cómo trabaja el agente JSE en ese ambiente de aprendizaje:



### 3.4.1. CARACTERIZACION DE UN SALÓN DE CLASES INTELIGENTES

En [9,12] han definido un “Aula Inteligente”, llamado SaCI, basado en sistemas multiagentes, donde se definen dos tipos de agentes: uno para caracterizar el software presente en un SaCI, y el otro para el hardware. En el SaCI, todos sus componentes interactúan de manera autónoma entre sí, con el objetivo de mejorar los procesos de enseñanza que ocurren en él.

En particular, el SaCI plantea un modelo centrado en los estudiantes, para ayudar en los procesos de aprendizaje a través de dispositivos y programas que colaboran y apoyan su auto-enseñanza. Para ello, las aulas inteligentes poseen diferentes tipos de elementos, tanto de hardware (tales como cámaras, pizarras, celulares, tabletas inteligentes, entre otros), como de software Sistemas Tutoriales Inteligentes, Sistemas de Aprendizaje Colaborativo Apoyado por Computador, VLE, entre otros).



**Figura 3.12.** Un SaCI. Fuente: [62]

En la figura 3.12 observamos un ejemplo de un SaCI, donde la profesora busca un videojuego para impartir una clase de idiomas y pregunta como se dice niña en inglés “Girl”. Luego, aparece en el videojuego interactivo la cara de una niña. Esto permite ser más motivadora, visual y entretenida la clase, activando otros sentidos para los espectadores como el de la vista y el de audición, logrando una inmersión en el estudiante, tal que ellos practiquen mediante el videojuego el idioma.

También, en el ejemplo del SaCI de la figura 3.12, se observan niños interactuando con escritorios táctiles inteligente, mientras la profesora explica la clase en una pizarra inteligente con un videojuego. Si ese videojuego fuese un JSE, se requeriría de un MJSE que lo adapte a la explicación de la clase en tiempo real.

### 3.4.2. AGENTE DE JSE EN UN SACI

El AJSE tiene como objetivo el hacer emerger JSEs con objetivos diferentes al de la pura diversión. Para ello, en función de la temática que se da en el SaCI, hace los ajustes en el JSE. EL AJSE interactúa con los agentes del SaCI: Gestor del ROA (Repositorio de Objeto de Aprendizaje), SA (Sistema Académico), Sistema Recomendador de Recursos Educacionales y el VLE (ver figura 3.13) [41]. Además, en general, explota diferentes insumos del contexto (dispositivos) y de los agentes del SaCI para adecuar los JSEs.

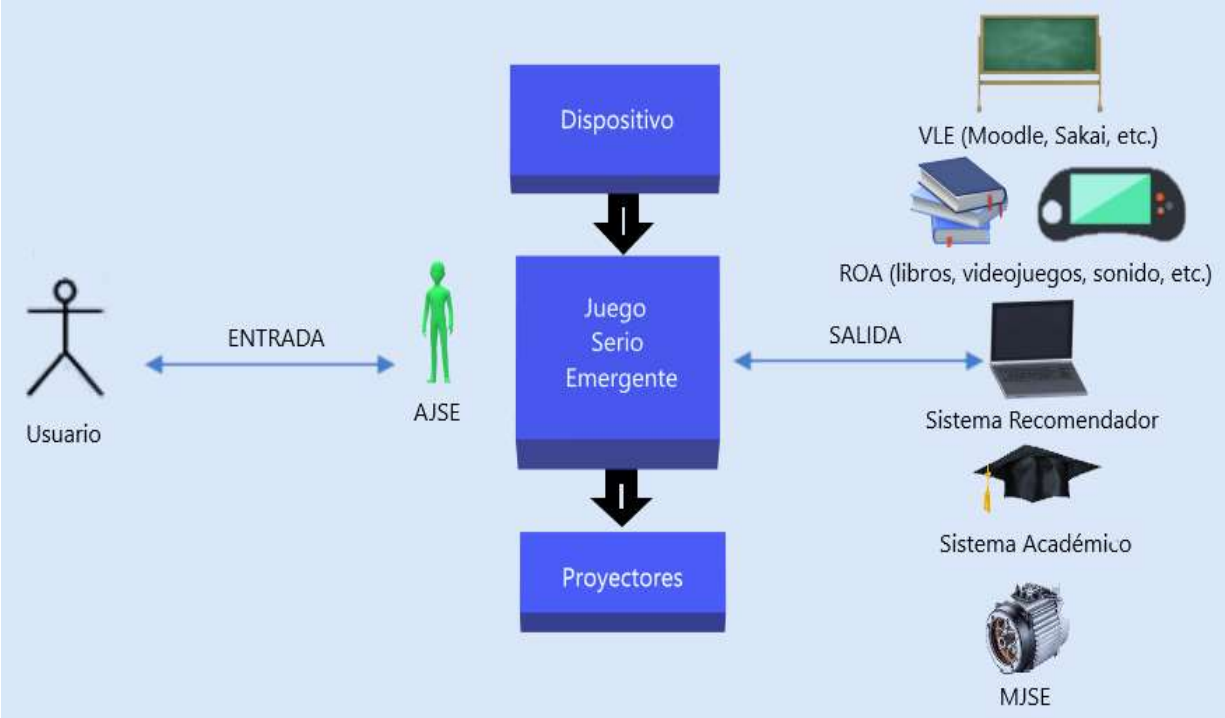



**Figura 3.13.** Modelo conceptual de un AJSE en un SaCI. Fuente [41]



El AJSE debe incorporarse al SaCI utilizando un middleware reflexivo autonómico para entornos de aprendizaje inteligente, llamado ambiente inteligente basado en la nube, propuesto en [9, 40]. Los modelos que describen a este agente, siguiendo la metodología MASINA [29, 30], son:

1. Modelo de Agentes del AJSE: este agente permite gestionar los JSE, para adaptarlos a las necesidades del tema tratado en el SaCI, tal que el usuario tenga una experiencia de aprendizaje adaptada a sus necesidades a través del SaCI. Sus objetivos específicos son:
   a. Presentar el JSE adaptado a los perfiles de usuario presentes en el SaCI. De este objetivo específico se derivan los siguientes sub-objetivos:
      i. Detectar el tipo de JSE que se necesita en un tema determinado, basado en el perfil de los usuarios;
      ii. Desplegar el JSE en los dispositivos físicos del SaCI.
   b. Presentar el JSE adaptado al tema educativo impartido en un momento dado en un SaCI:
      i. Adecuar el JSE al contexto actual del SaCI;
      ii. Ofrecer a los usuarios una experiencia en un tema, tal que el JSE potencie el proceso de aprendizaje.

En la tabla 3.7 presentamos un ejemplo de la descripción detallada de uno de los objetivos específicos.

**Tabla 3.7.** Objetivo del AJSE

| **Objetivo:** desplegar el JSE en los dispositivos presentes en el SaCI |
|---|
| **Tipo:** gestionar el JSE. |
| **Parámetros de entrada:** solicitud al AJSE, por medio de los agentes del SaCI.<br>**Parámetros de salida:** despliegue de un JSE en un dispositivo disponible en un SaCI.<br>**Condición de activación:** recepción de solicitud de JSE.<br>**Condición de finalización:** reproducción de JSE.<br>**Condición de éxito:** se reproducen los JSEs adaptados al tema de clases del SaCI para el usuario.<br>**Condición de fracaso:** no se reproducen los JSEs si: los dispositivos no son los correctos, la información del usuario no es la adecuada o existen errores de comunicación entre los agentes.<br>**Lenguaje de representación:** lenguaje máquina.<br>**Descripción:** reproduce los JSEs adaptados al tema de un perfil del usuario de un SaCI, en diferentes tipos de dispositivos de JSE. |

2. Modelo de tareas del AJSE: las tareas que este agente realiza, dependen de los servicios que debe brindar según sus objetivos. En general, tiene un grupo de tareas que realiza secuencialmente:
    a. Recolectar datos del VLE sobre el tema.
    b. Identificar requerimiento en el SaCI.
    c. Determinar el JSE.
    d. Adecuar el JSE.
    e. Incorporar el JSE en el SaCI.
    f. Ejecutar el JSE.



Además de esas tareas, el AJSE requiere realizar otras tareas. En ese sentido, algunas de esas tareas específicas que realiza el AJSE en el SaCI se muestran en la tabla 3.8.

**Tabla 3.8.** Tareas específicas del AJSE en el SaCI

| **Tareas** |
|---|
| *1.* Realizar consulta al SA sobre datos de perfiles de los usuarios. |
| *2.* Realizar consulta al VLE sobre el tema en que se está dando. |
| *3.* Revisar los Juegos Serios que están almacenados en ROA. |
| *4.* Usar las recomendaciones de ROA. |
| *5.* Activar a MGJSE, según el tema del SaCI. |
| *6.* Activar STE, según el tema del SaCI. |
| *7.* Recolectar subtramas de JSE y guardar en ROA. |
| *8.* Utilizar dispositivos para proyectar el JSE en el SaCI. |

*3.* Modelo de Inteligencia del AJSE: este agente debe tener un mecanismo de aprendizaje para responder rápidamente a situaciones que se le han presentado anteriormente, pero también, para adaptarse a los contextos y dominios que se les vayan presentando. A continuación, se presenta el modelo de inteligencia de este agente (tablas 3.9 y 3.10).

**Tabla 3.9.** Mecanismo de aprendizaje

| **Nombre:** JSE adaptados al tema. |
|---|
| **Tipo:** supervisado.<br>**Técnica de representación:** basada en reglas de SCD, en ACO y en un algoritmo cultural, para ensamblar un conjunto de subtramas adecuados a un tema dado en el SaCI.<br>**Fuente de aprendizaje:** JSE ensamblados en base al tema del SaCI y las características de usuarios para las que son adecuadas.<br>**Mecanismo de actualización:** retroalimentación |

**Tabla 3.10.** Mecanismo de razonamiento

| |
|---|
| **Fuente de información:** reglas que representan cómo se debe utilizar el JSE, según el tema, la actividad y los usuarios.<br>**Tipo de inferencia:** motor de reglas.<br>**Lenguaje de representación de conocimiento:** SCD.<br>**Relación utilidad tarea-objetivo:** resultados académicos satisfactorios de acuerdo el JSE usadas en el proceso de aprendizaje que ofrece el SaCI.<br>**Estrategias de búsqueda de soluciones:** deductivo. |



También, el AJSE invoca a otros mecanismos de la computación inteligente para adaptar a los JSE, tales como los algoritmos culturales, ACO y SCD.

*4.* Conversaciones del AJSE en un SaCI: algunas de las conversaciones predeterminadas donde interviene el AJSE en un SaCI son:

- Solicitud de recomendación de JSE al ROA.
- Diseño de un JSE.
- Ejecución de un JSE

Las tablas 3.11 y 3.12 describen en detalle una conversación: ejecución de JSE con el tema ya adaptado.

**Tabla 3.11.** Conversación: Ejecución de JSE con el tema ya adaptado

| **Objetivos:** proyección de JSE. |
|---|
| **Agentes participantes:** AJSE, Dispositivo, SA, Tutor, Estudiantes y VLE.<br>**Iniciador:** usuario autorizado del sistema.<br>**Actos de habla:** consultar perfil usuario en VLE, consultar dispositivos de proyección en un SaCI, desplegar JSE, adecuar JSE según interacciones usando el MGJSE.<br>**Precondición:** existir una solicitud de JSE, existir la comunicación entre los agentes involucrados, existir dispositivo de proyección de JSE.<br>**Condición de terminación:** proyección o detección de error de JSE.<br>**Descripción:** mediante esta conversación, el agente solicitante inicia la conversación que permite al AJSE interactuar con los actores involucrados en el servicio, para lograr saber que JSE va a ser utilizado, en el dispositivo de salida y en el tema indicado por el agente solicitante del servicio. |

**Tabla 3.12.** Esquema de coordinación de la conversación

| **Objetivo:** coordinar la comunicación entre los agentes implicados en la conversación. |
|---|
| **Tipo**: por defecto.<br>**Coordinación por defecto:** negociación entre AJSE y entre el agente dispositivo, VLE y SA.<br>*Mecanismo de comunicación*<br>**Tipo:** directa<br>**Técnica:** envío de mensajes<br>**Meta-lenguaje:** *Agent Communication Language*<br>**Ontologías:** ontología del dominio de JSE, ontología general de un SaCI |



La figura 3.14 muestra el diagrama de interacción de esa conversación. En este caso, el AJSE modifica las tramas del JSE con el SAV (revisar [41]). En la figura 3.14 se representa la conversación en el SaCI para este caso:

a. El agente VLE requiere los servicios del AJSE para explicar la sesión del curso que está gestionando en un SaCI.
b. El agente AJSE hace emerger las tramas en un JSE que permita explicar el curso, usando como insumo la información del curso y de los estudiantes.
c. Los estudiantes interactúan con el JSE, utilizando los agentes Dispositivos que lo despliegan, tales como una pizarra, una tableta, un celular inteligente, etc.
d. El AJSE va adecuando el JSE, en función de la interacción con el usuario. En particular, el AJSE va adaptando en línea al JSE haciendo emerger estrategias, secuencias y propiedades, entre otras cosas, acordes con las dinámicas del curso.

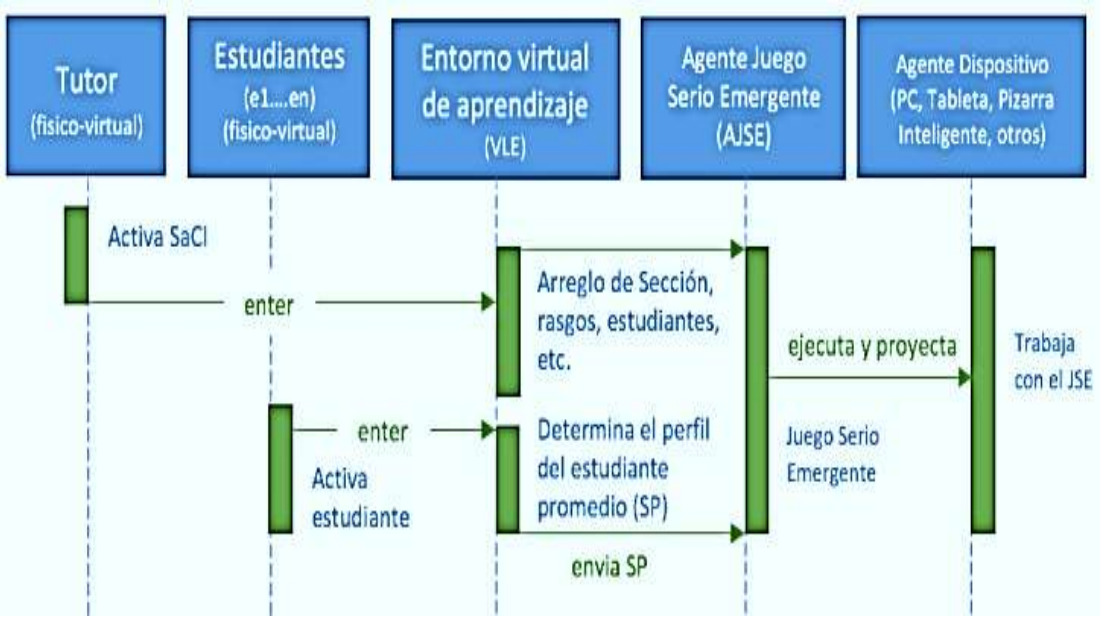



**Figura 3.14.** Diagrama de interacciones de la conversación de la tabla 3.11

# CAPÍTULO IV: EXPERIMENTOS Y ANÁLISIS DE RESULTADOS

En el presente capitulo se presentan las pruebas de entonación, experimentos y el análisis de los resultados de nuestro trabajo. Para ello, se presentan varios casos de estudio, con sus protocolos experimentales. Eso se hace para el SAT con ACO, el SAP con algoritmo cultural y el SAE con SCD.

## 4.1. CONTEXTO EXPERIMENTAL

Se considera un SaCI como el propuesto en [9, 63], en el que en el aula inteligente todos sus componentes son modelados usando el paradigma de sistemas multiagentes, el cual, caracteriza a sus dispositivos de hardware (pizarra inteligente, laptop, tableta, smartphone, etc.) y de software (VLE, Sistema Académico, etc.) como agentes [12]. El SaCI utiliza un middleware reflexivo autonómico para entornos de aprendizaje inteligente en la nube, llamado ambiente inteligente basado en la nube [9, 40]. Este middleware reflexivo autonómico está basado en sistemas multiagentes, y posibilita el despliegue de las diferentes comunidades de agentes del SaCI. Además, el ambiente inteligente basado en la nube es el monitor del SaCI y administra los servicios educativos en la nube para mejorar la experiencia de aprendizaje de los estudiantes. El gran valor de esta propuesta es que permite ver un SaCI como un sistema emergente auto-organizado, donde cada componente (agente) actúa individualmente, pero a través de las relaciones entre ellos se produce un efecto colectivo en el proceso de aprendizaje [63].



En particular, uno de esos agentes es el Agente de Juegos Serios Emergentes (AJSE), el cual gestiona los JSE de forma autónoma. AJSE se basa en el MJSE para la gestión autónoma de los JSE utilizados en el SaCI. De esta forma, utiliza todos los componentes del MJSE, tanto para generar como para actualizar en tiempo real los JSE utilizados en el SaCI. El AJSE, una vez recolectada la información del entorno del SaCI (perfil de los estudiantes, objetivos del actual proceso de aprendizaje, etc.), adapta el JSE a SaCI, llamando al Gestor de Materia. Para ello, el AJSE interactúa con los agentes del SaCI como se ve en la figura 3.13) [41]. En general, el AJSE explota diferentes insumos del contexto (dispositivos) e información de otros agentes del SaCI, para generar o adecuar los JSEs a las necesidades del proceso de aprendizaje ocurriendo en ese momento en el SaCI.

## 4.2. CASO DE ESTUDIO DEL SAT

En el Capítulo III apartado 3.3.1 se explica detalladamente tanto el diseño como la implementación del SAT, utilizando ACO. En esta sección se mostrará el funcionamiento del SAT en el contexto del SaCI.

### 4.2.1. DESCRIPCION FUNCIONAL DEL SAT

***Caso creación de JSE:***

A continuación, vamos a mostrar un ejemplo de cómo funcionaría el componente SAT del 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 4.1.

**Tabla 4.1.** 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 |

Además, los datos de los estudiantes y su contexto educativo son mostrados en la tabla 4.2. Toda esa información la captura el Gestor de Materia. Por otro lado, se usó el repositorio de aprendizaje http://agrega.educacion.es, el cual usa los metadatos de los objetos de aprendizaje contenidos en él en formato SCORM [17].

**Tabla 4.2.** 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.*** **Nivel:** Secundaria | **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 | |



Con la información provista por el Gestor de Materias, el Gestor de Videojuegos usa los metadatos SCORM de cada objeto de aprendizaje en AGREGA para hacer la comparación con el tema deseado. La tabla 4.3 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 juegos serios 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*

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

**Tabla 4.3.** 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) | médium | 0,5 |
| **duration** | 30 minutos (VLE) | PT0H10M0S | 0,4 |
| **interactivityLevel** | very high | very high | 1 |
| **semanticDensity** | very high (SA) | médium | 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** |



En nuestro caso, supongamos que ningún juego supera el umbral de similitud de 0,85, pero los 6 juegos de la figura 4.1 cumplen el valor de similitud de estar entre 0,85 y 0,4. Esos 6 juegos serán los usados por ACO para hacer emerger un JSE. Con ellos se construye el grafo que ACO usará (serán sus nodos, ver figura 4.2).

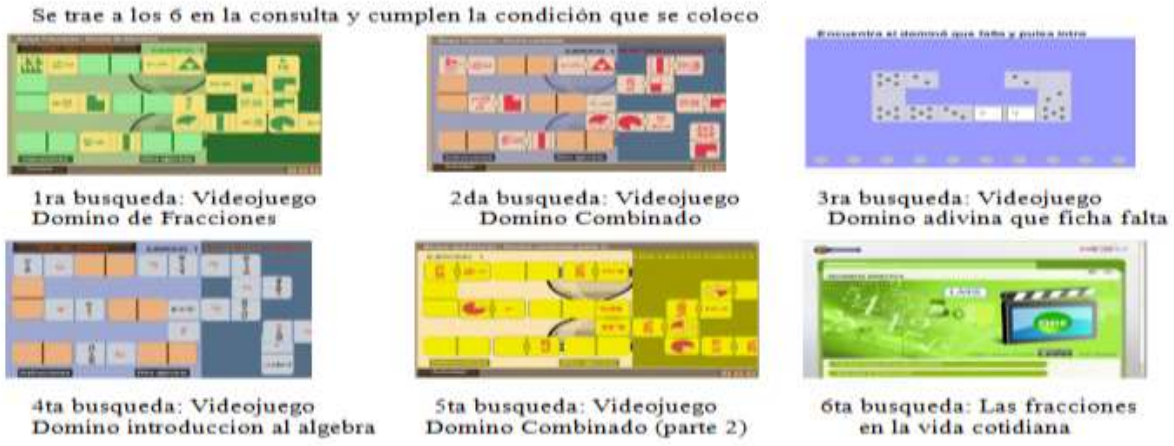



**Figura 4.1.** Videojuegos que usará ACO

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 figura 4.2), y toma la decisión del nodo a ser visitado utilizando el modelo de Monte Carlo basado en la ecuación 3.1.

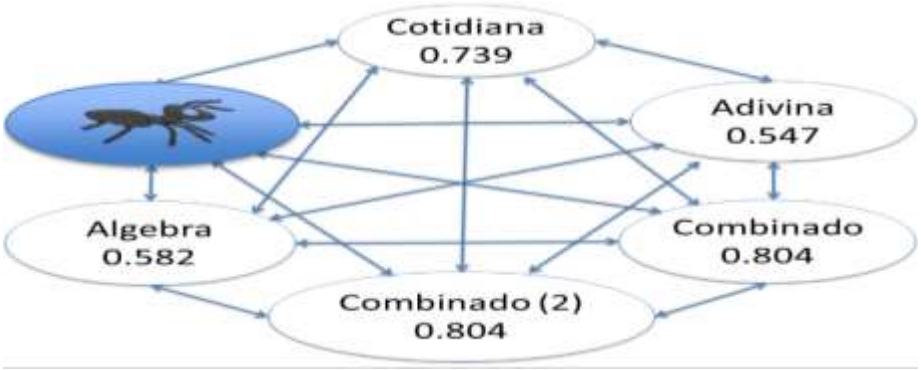



**Figura 4.2.** Elije aleatoriamente como nodo inicial "Fracciones"

Supongamos que, en ese caso, la hormiga decide trasladarse al nodo "Combinado" usando la ecuación 3.2, tal como se observa en la figura 4.3.

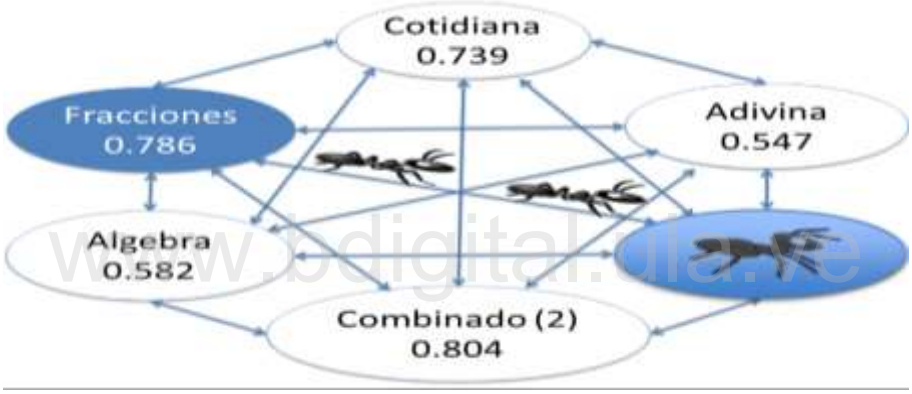





**Figura 4.3.** Elije moverse al nodo "Combinado"

Continuando con el proceso, de nuevo debe decidir donde moverse, y la hormiga decide ahora moverse al nodo "Combinado (2)" (ver figura 4.4).

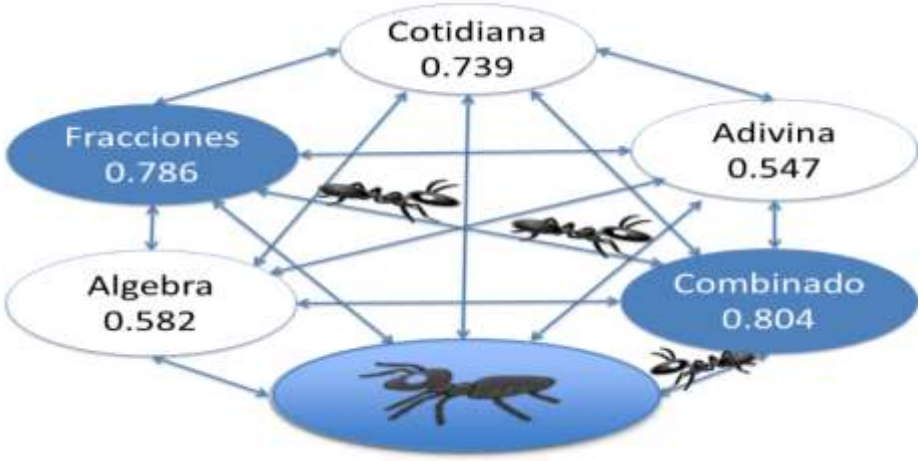



**Figura 4.4.** Elije aleatoriamente nodo "Combinado (2)"

La hormiga puede continuar con la construcción de la solución, o detenerse en cualquier momento. En este caso, suponemos que culmina cuando visita al nodo “Adivina” (ver figura 4.5).

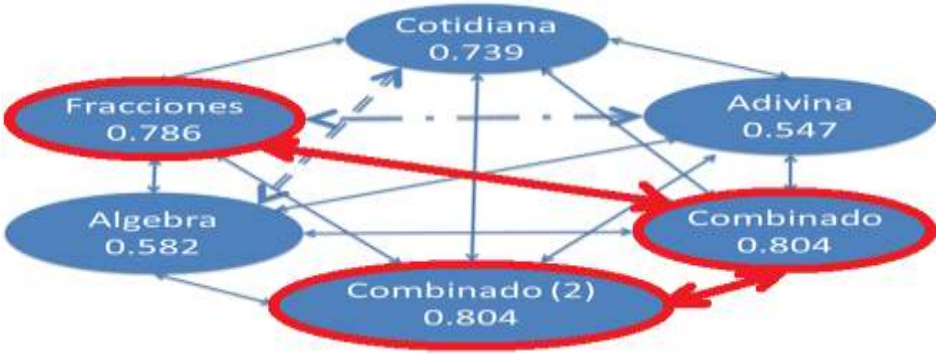



**Figura 4.5.** Solución final propuesta de JSE

En ese momento se inicia la actualización de las feromonas, lo cual es realizado usando la información de calidad de las propuestas de JSE de todas las hormigas de esa iteración (ver sección 3.3.1.1 en 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 (todas las hormigas siguen la misma ruta), se pasa a construir la solución final. En ese caso, se escogen los nodos cuya feromona pasa cierto umbral, y se conectan entre ellos según los valores más altos de los arcos que los unen. Si suponemos la figura 4.5 como la final, una vez que converge ACO, suponemos que los nodos que superan el umbral son solo 3, con los arcos que los unen (porque tienen los valores más altos de feromona), Todos están indicados en esa figura en color rojo.



Basado en ello, el JSE creado está compuesto por 3 subtramas, organizadas en la siguiente secuencia lógica:

- JS1: Fracciones – Combinado – Combinado (2)

Ese sería el JSE que emerge (ver figura 4.6). 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.


**Figura 4.6.** 3 videojuegos de fracciones utilizando domino. Fuente [18]

***Caso actualización de JSE:***

En este caso se supone que inicialmente el SEV propone como JSE el videojuego domino “Combinado” (ver figura 4.7):

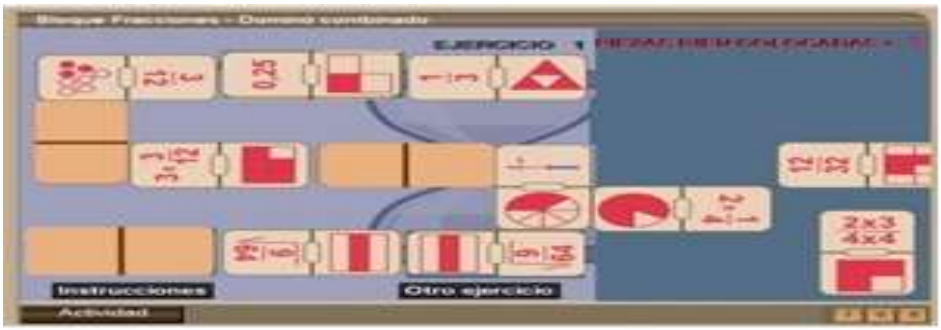



**Figura 4.7**. El videojuego de domino “Combinado”

Además, se supone que el JSE de domino “combinado” no cumple con las expectativas del profesor que está explicando la clase en el SaCI. Si este juego generado no se adecua convenientemente al tema de clase deseado, se debe actualizar el JSE buscando otros repositorios de aprendizaje con nuevas tramas o JSE, que permitan explicar las fracciones en una clase de matemáticas. En particular, SAT realiza los siguientes pasos:



1. Cálculo de medidor de narrativa y jugabilidad: determina la brecha entre lo que está aconteciendo en el SaCI (clase de fracciones) y lo definido en el diseñó actual del JSE “Combinado”.
2. Seleccionar nuevos repositorios: en este caso de estudio, se escoge http://www.educaplay.com, que es un repositorio de recursos de aprendizaje de diferentes tipos como juegos serios, videojuegos, etc.
3. Búsqueda de nuevas tramas: se seleccionan varios juegos serios del repositorio de Eduplay, según la temática deseada. Por ejemplo, los juegos serios seleccionadas del repositorio de Eduplay en este caso de estudio son Fracciones algebraicas y Cruce de fracciones, entre otro, como se observa en la figura 4.8.


**Figura 4.8.** Cinco JSE seleccionados de Eduplay

Para realizar la selección, se comparan sus metadatos con el medidor de narrativa y jugabilidad y se calcula una puntuación (la tabla 4.4 da un ejemplo para una de las tramas), según lo cual se seleccionan las tramas/JSE (serán los nodos del grafo que recorrerán las hormigas).

**Tabla 4.4.** 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** |

4. 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 4.8, para conformar el grafo que se usa para construir un nuevo JSE (figura 4.9). 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 de adecuación del JSE inicial.



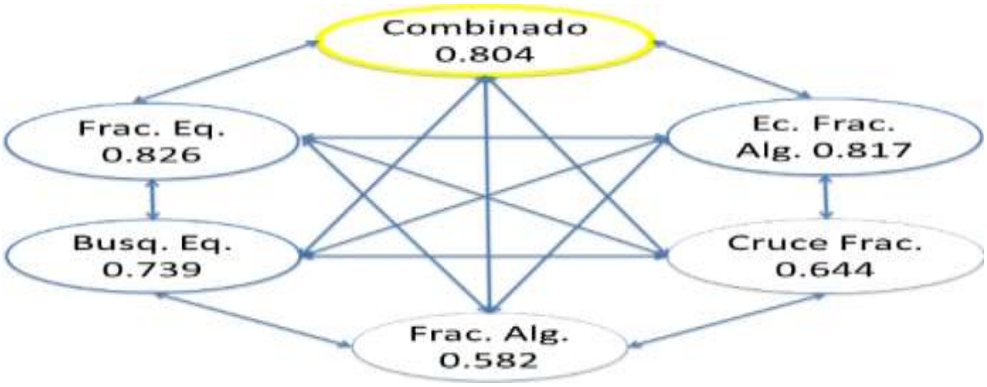



**Figura 4.9.** Grafo del JSE inicial con los 5 JSE de Eduplay

A partir del grafo inicial, empieza a actuar el algoritmo ACO (ver figura 4.10). El proceso que se sigue es el mismo explicado en [55] y anteriormente.

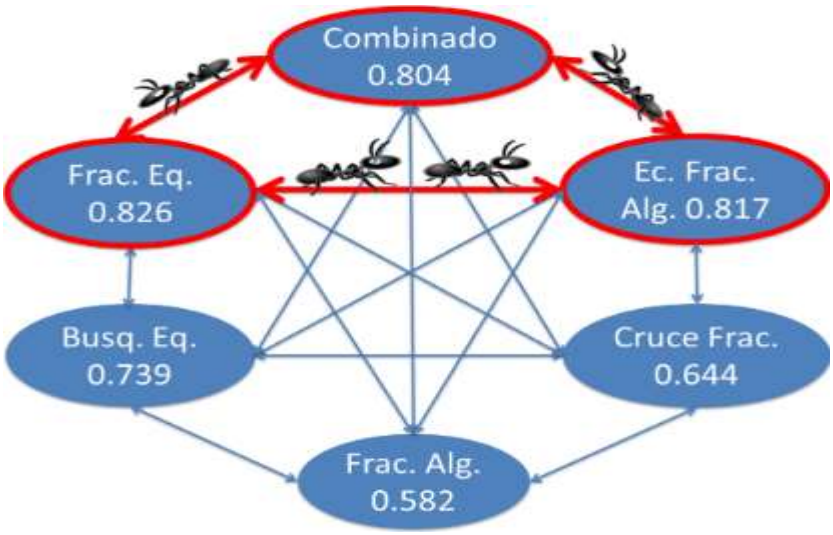



**Figura 4.10.** Nuevo JSE

La figura 4.10 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. Así, se observa que sigue utilizando este domino por su alta puntuación.



### 4.2.2. EXPERIMENTOS

***Contexto Experimental:***

En este apartado se presentan varios experimentos con el objetivo de evaluar el funcionamiento del SAT basado en el algoritmo ACO [43, 55, 64]. Como ROA se utiliza el repositorio AGREGA (http://agrega.educacion.es). Para el desarrollo de los diferentes escenarios, se seleccionan y descargan manualmente los metadatos de varios juegos serios sobre diferentes temas (matemáticas, física, lenguaje, historia y geografía) para ser utilizados como datos de entrada. Durante el proceso de prueba, se aplicaron tres experimentos [43, 55, 64], con el fin de analizar los parámetros apropiados para ser utilizados en el SAT. Para estos experimentos, se consideran tres cursos (ver tabla 4.5).

**Tabla 4.5.** Cursos

| Curso | Tema | Tema | Tema |
|---|---|---|---|
| **1** | Español | Literatura | Comunicación verbal |
| **2** | Matemáticas | Álgebra | Raíz cuadrada. |
| **3** | Matemáticas | Fracciones | Fracciones equivalentes |

Para las pruebas se estudian tres parámetros del SAT:

1. Método de similitud: es la métrica utilizada para determinar la similitud entre dos palabras, que se utiliza para comparar los metadatos de los *juegos serios* de los *ROA* con el tema deseado. En particular, se estudian tres métodos de similitud (explicados con más detalle en [65]): Levenshtein, Jaro-Winkler y Jaccard. "Levenshtein" es un método de comparación de cadenas que se basa en cuántas transformaciones se deben realizar en dos cadenas para igualarlas (número mínimo de operaciones requeridas), este número determina la distancia entre ellas. El método "Jaro-Winkler" es un método basado en la distancia para la comparación de cadenas, que mide una distancia de edición entre dos secuencias utilizando una escala de prefijo que otorga calificaciones más favorables a las cadenas que coinciden desde el principio. Finalmente, el método "Jaccard" estudia el número de "Tokens" que componen cada cadena para realizar la comparación. "Jaccard y Jaro-Winkler" devuelven un número en el rango [0,1], que determina la cercanía entre las cadenas comparadas. En el caso de “Levenshtein”, este número debe normalizarse.

2. Umbral de similitud: de los 15 atributos utilizados para describir un juego serio en los metadatos (ver Tabla 4.4), solo cuatro se consideran de mayor importancia (título, idioma, descripción y palabras clave), ya que estos atributos definen el tema en un juego serio. Con estos atributos se define un número entre 0 y 1 para describir la similitud entre el tema deseado y el juego serio utilizando uno de los métodos de similitud anteriores para analizar cada atributo. Teniendo este valor como referencia, es posible definir un umbral de similitud entre un juego serio y el tema deseado. Por ejemplo, si la media del número obtenido de los cuatro atributos al comparar el tema buscado con el juego serio analizado es superior a 0,6 (es el índice de similitud del juego serio) y el umbral de similitud es 0,6, entonces podemos concluir que existe una similitud entre ellos. Así se obtiene la similitud de cada *juego serio* con el tema deseado, por lo que se debe analizar la sensibilidad de SAT respecto al valor umbral de similitud.

3. Número de Tramas Utilizadas en la Solución Final: en general, cuando se invoca ACO, se define el número de tramas a utilizar para la construcción del JSE (define la complejidad del grafo a construir por ACO). En ese sentido, debe estudiarse el número máximo de tramas utilizadas por el SAT.

   De esta manera, se evalúan los resultados finales proporcionados por el SAT según los siguientes rangos:
   a. ***Óptimo*:** el juego generado cumple totalmente con el tema deseado (el índice de similitud del JSE final es mayor que el umbral de similitud).
   b. **Aceptable***:* el juego generado tiene alguna relación con el tema deseado (el índice de similitud del JSE final está cerca del umbral de similitud).
   c. **Erróneo:** el juego generado no tiene relación con la temática deseada del curso (el índice de similitud JSE es menor).

***Experimentación:***

Durante el proceso de prueba, se aplicaron tres experimentos, con el fin de analizar los parámetros apropiados para ser utilizados en el SAT.

1. Método de similitud: el objetivo del primer experimento es estudiar el funcionamiento de varios métodos de comparación de cadenas. Definimos la hipótesis de que los métodos de comparación de las cuerdas tienen una gran influencia para determinar la similitud entre un juego serio y el tema deseado. Para el primer ensayo con cada método, para el resto de parámetros ACO y SAT se consideran sus mejores valores. Tras 30 corridas, los resultados medios obtenidos se muestran en la tabla 4.6.

**Tabla 4.6.** Resultados del primer experimento

| Método | Curso | Óptimo | Aceptable | Erróneo |
|---|---|---|---|---|
| **Jaro Winkler** | 1 | 100% | 0% | 0% |
| | 2 | 83% | 10% | 7% |
| | 3 | 87% | 7% | 6% |
| **Jaccard** | 1 | 97% | 0% | 3% |
| | 2 | 73% | 10% | 17% |
| | 3 | 80% | 7% | 13% |
| **Levenshtein** | 1 | 80% | 17% | 3% |
| | 2 | 73% | 17% | 10% |
| | 3 | 83% | 10% | 7% |

De acuerdo con los resultados de la tabla 4.6, el método de "Jaro-Winkler" presenta los mejores resultados para todos los casos, en comparación con los demás métodos de similitud. De acuerdo con estos resultados, nuestra hipótesis queda confirmada. En general, SAT requiere un método de comparación preciso porque define los *juegos serios* que se utilizarán para la adaptación de un JSE.



2. Umbral de similitud: el objetivo del segundo experimento es evaluar la sensibilidad del umbral de similitud. En este experimento establecemos la hipótesis de que el valor umbral de similitud tiene un impacto importante en el proceso de adaptación de un JSE porque define los tramas/*juegos serios* que se utilizarán para su adaptación. Se definen varios valores de umbral para esta prueba: 0,4, 0,65, 0,8 y 0,95. Después de 30 corridas para cada caso, los resultados obtenidos se muestran en la tabla 4.7.

**Tabla 4.7.** Resultados del segundo experimento

| Límite | Curso | Óptimo | Aceptable | Erróneo |
|---|---|---|---|---|
| **0,4** | 1 | 0% | 0% | 100% |
| | 2 | 0% | 0% | 100% |
| | 3 | 0% | 0% | 100% |
| **0,65** | 1 | 77% | 6% | 17% |
| | 2 | 20% | 33% | 47% |
| | 3 | 50% | 23% | 27% |
| **0,8** | 1 | 100% | 0% | 0% |
| | 2 | 40% | 30% | 30% |
| | 3 | 87% | 10% | 3% |
| **0,95** | 1 | 93% | 0% | 7% |
| | 2 | 27% | 20% | 53% |
| | 3 | 77% | 13% | 10% |

De acuerdo con la tabla 4.7, se puede observar cómo aumentando los umbrales de similitud desde el inicio, también aumentan los resultados óptimos y disminuyen los resultados erróneos, hasta alcanzar un número pico (0,95) donde ocurre lo contrario. Estos resultados se deben a que cuando el umbral aumenta, el algoritmo se vuelve más estricto con respecto a lo que se considera similar entre dos cadenas de caracteres. Sin embargo, cuando el umbral llega a 0,95, entonces tiene un impacto negativo en los resultados óptimos ya que SAT no puede explorar las tramas que no son tan similares (0,8) para construir el JSE. En general, para generar JSEs cercanos al tema deseado, es necesario utilizar un umbral de similitud más cercano a 0,8. Por lo tanto, la segunda hipótesis puede ser confirmada.

3. Número de subtramas de la solución final: el objetivo del tercer experimento es analizar el impacto en la calidad del resultado del número máximo de subtramas utilizadas por el SAT. Es importante recordar que cuando SAT termina, entonces realiza una fusión entre las tramas/*juegos serios elegidas* e integra sus atributos. La hipótesis con este experimento es que el número máximo de tramas/*juegos serios* utilizados para adaptar un JSE tiene un gran impacto en la calidad del JSE final (su adaptación a la temática deseada). Para esta prueba se utilizan diferentes números máximos de tramas (2, 3, 5 y 10). Después de 30 corridas para cada número máximo de tramas, los resultados promedio obtenidos se muestran en la tabla 4.8.

**Tabla 4.8.** Resultados del tercer experimento



| Subtrama | Curso | Óptimo | Aceptable | Erróneo |
|---|---|---|---|---|
| **2** | 1 | 63% | 30% | 7% |
| | 2 | 58% | 31% | 11% |
| | 3 | 60% | 30% | 10% |
| **3** | 1 | 67% | 33% | 0% |
| | 2 | 65% | 27% | 8% |
| | 3 | 65% | 25% | 10% |
| **5** | 1 | 87% | 10% | 3% |
| | 2 | 85% | 12% | 3% |
| | 3 | 83% | 9% | 8% |
| **10** | 1 | 43% | 0% | 57% |
| | 2 | 41% | 24% | 35% |
| | 3 | 37% | 3% | 60% |

De acuerdo con la tabla 4.8, cuando se utiliza un número muy elevado de subtramas, se generan JSEs que no son de buena calidad (no se adaptan a la temática deseada). Esto se debe a que SAT intenta adaptar un JSE con el máximo número de subtramas, agregando tramas/*juegos serios no muy buenas* (el algoritmo llena estos espacios vacíos con tramas que no cumplen con el tema deseado). Además, un número máximo pequeño de subtramas no es muy bueno porque no hay suficientes tramas para construir un JSE cercano a los temas deseados (el número de subtramas que tiene el SAT para cumplir con el tema deseado es pequeño). En nuestros experimentos, el valor ideal es cinco porque con este valor se obtiene el mayor número de JSE óptimos y el menor número de JSE incorrectos, con respecto a los otros valores del número de subtramas (por ejemplo, con ese valor se obtuvo para el curso 1 87% de JSEs óptimos, 10% aceptables y solo 3% incorrectos). Otro aspecto importante a destacar es que,

con un valor alto de un número máximo de subtramas, el tiempo de ejecución del SAT aumenta ya que el espacio de posibles soluciones que se debe explorar para adaptar el JSE es mayor (mayores combinaciones posibles de subtrama). Así, se confirma la tercera hipótesis.

Los tres experimentos nos permiten determinar los parámetros adecuados del SAT para ser utilizados en un SaCI. La combinación "óptima" de los parámetros depende del objetivo en el SaCI. Si el objetivo es ser estricto con el tema a buscar, entonces se requiere un buen número de tramas y umbrales altos, con un método estricto de comparación de cadenas. Por el contrario, si el objetivo es ser más permisible sobre los JSEs y su relación con el tema deseado, pero mejorando los tiempos de búsqueda, entonces se recomienda utilizar métodos de comparación menos estrictos, con umbrales de similitud bajos y número pequeño de subtramas.

En la figura 4.11 se muestra un ejemplo de un JSE óptimo proporcionado por el SAT. En el primer párrafo de la figura 4.11, aparece una lista de *juegos serios* para la asignatura de matemáticas (que está entre paréntesis), seguido del nombre del juego serio y su índice de similitud. En el segundo párrafo se observa la solución final generada por el algoritmo ACO (JSE generado), compuesta por tres *juegos serios* (que cumplen con el tema actual de las fracciones): "Las_fracciones_en_la_vida_cotidiana.xml", "Fracciones.xml" y "Domino_Combinado_(parte_2).xml". Finalmente, se observa el índice de similitud del JSE generado, el cual es igual a 0,859.

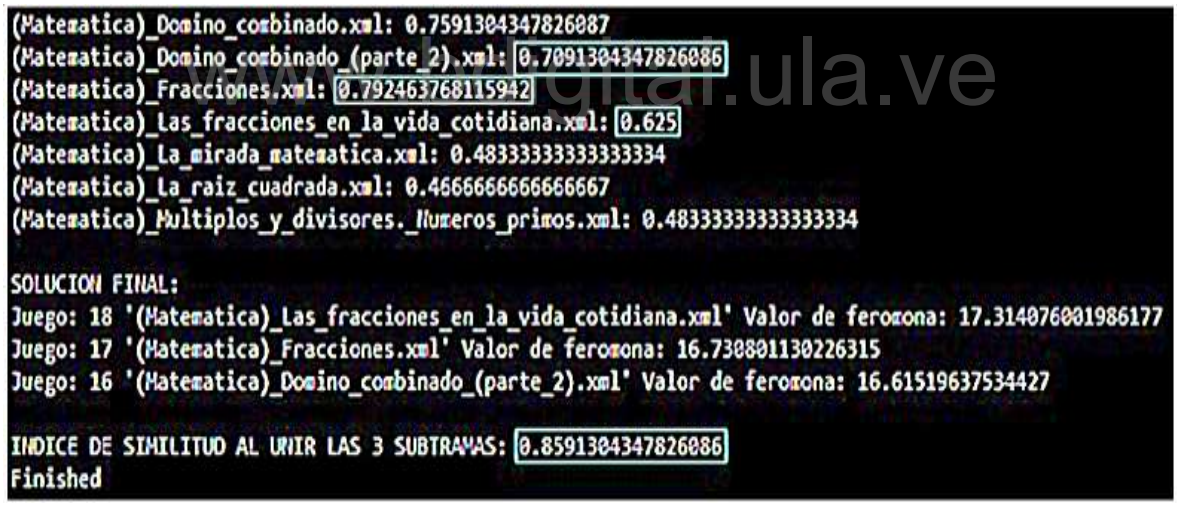



**Figura 4.11.** Ejemplo de resultado óptimo [43]

Por lo tanto, el JSE propuesto se compone de tres subtramas, que combinadas tienen un índice de similitud de 0,85.

**Análisis de Resultados:**

Los experimentos permiten comprobar la pertinencia de nuestra contribución: SAT permite adaptar un JSE al contexto educativo donde se está utilizando. En particular, SAT tiene un componente para determinar el tema deseado en un momento dado. Con esta información, SAT busca JSEs en repositorios de ROAs que se acerquen al tema deseado. Estos ROA se utilizan posteriormente para modificar el JSE

actual. Esta modificación consiste en utilizar las tramas iniciales del JSE, y hacer ajustes al mismo incorporando los nuevos ROA como tramas y, eventualmente, eliminar algunas de las subtramas que actualmente lo componen para alcanzar el tema deseado. Este proceso se realiza con un ACO. Así, al final, hay un JSE modificado que sigue el tema deseado.

Los experimentos analizan la sensibilidad de los parámetros del SAT para realizar la tarea anterior. Es posible verificar en los resultados obtenidos en las tablas 4.6, 4.7 y 4.8 que el JSE modificado sigue el tema deseado (es la solución óptima) si se eligen los valores adecuados para los parámetros del SAT. En concreto, el método de similitud debe ser "Jaro-Winkler", el umbral de similitud cercano a 0,80 y el número máximo de subtramas utilizadas debe ser cinco. El JSE modificado contendrá un conjunto de tramas (nuevas y antiguas) que juntos son adecuados para el tema deseado en el contexto actual.

## 4.3. CASO DE ESTUDIO DEL SAP

En el Capítulo III, apartado 3.3.2, se explica detalladamente tanto el diseño como la implementación del SAP, utilizando algoritmo cultural. En esta sección se mostrará el funcionamiento del SAP en el contexto del SaCI.

### 4.3.1. DESCRIPCION FUNCIONAL DEL SAP



Este caso de estudio se considera la adaptación de varios JSEs usando a SAP según [66].

***Cálculo mental de la raíz cuadrada de 6 números:***

En este caso se supone un curso de matemáticas en un SaCI con 20 estudiantes, y se requiere un JSE para explicar y resolver ejercicios sobre las raíces cuadradas, para lo cual se supone que inicialmente el subsistema de emergencia del videojuego propone como JSE inicial el videojuego del cálculo mental de la raíz cuadrada de 6 números, obtenido del repositorio AGREGA (http://agrega.educacion.es) (ver figura 4.12):

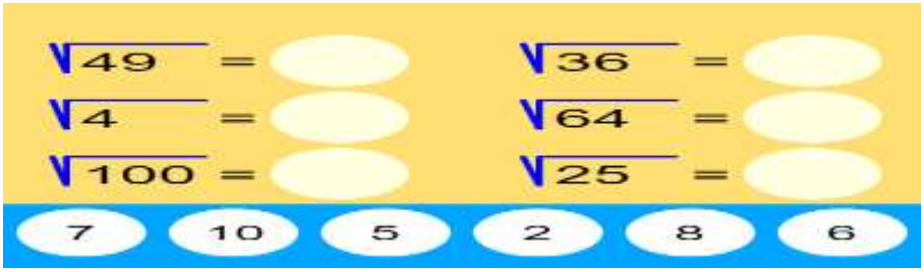



**Figura 4.12**. El JSE: "Si el cálculo mental de la raíz cuadrada de 6 números" [66]

El JSE de cálculo mental de la raíz cuadrada de 6 números consiste en una escena donde aparecen seis raíces cuadradas (zona amarilla), cuyos radicandos son números cuadrados perfectos. Los números en la zona azul son las soluciones. Luego, el jugador tendrá que colocarlos en los espacios. Una vez que

se hayan colocado todas las fichas (números) en los lugares, el JSE mostrará la palabra "CORRECTO" o "INCORRECTO" en la zona azul.

El SaCI utiliza el SAP para determinar los valores ideales de los parámetros del JSE. El JSE "cálculo mental de la raíz cuadrada" tiene 3 parámetros:

- $P_1$ = Valor numérico máximo del radicando.
- $P_2$ = Raíces cuadradas a calcular (los espacios para colocar las fichas).
- $P_3$ = Las fichas (números) que tiene el jugador (ya que el jugador puede tener más fichas que raíces cuadradas a calcular).

Los criterios de calidad son el tiempo y la puntuación, los cuales son definidos por el profesor de la clase. En este caso, se definen como:

- Intervalo de tiempo (LE)*:* un límite de tiempo de 600 segundos (10 minutos).
- Puntuación acumulativa (PA): si el valor es superior a 500 puntos, el jugador cumplió con la mayoría de los objetivos usando el JSE. El valor máximo es de 1000 puntos.

La tabla 4.9 muestra la población inicial del algoritmo cultural (ver sección 3.3.2.1), que reflejan varias veces que los estudiantes jugaron el JSE.



**Tabla 4.9.** Población inicial

| # | $P_1$ | $P_2$ | $P_3$ | *FO* |
|---|---|---|---|---|
| **1** | 9 | 1 | 4 | 631,4 |
| **2** | 64 | 1 | 4 | 500,4 |
| **3** | 81 | 4 | 3 | 255,6 |

1. El primer estudiante: lo jugó con los parámetros $P_1$ = 9, $P_2$ = 1 y $P_3$ = 4, con *FO* = 631,4.
   El valor de *FO* se calculó de la siguiente manera: *FO* = 1.000 * 0,944 - 600 * 0,521 = 631,4
   Donde, *PA* normalizado es = 944 puntos, y *LE* normalizado es = 312,6 segundos.
   Según los criterios de calidad, se observa que el estudiante alcanzó los objetivos JSE (944 puntos) en aproximadamente 5 minutos y 22 segundos (313 segundos).

2. El segundo estudiante: lo jugó con los parámetros $P_1$ = 64, $P_2$ = 1 y $P_3$ = 4, con *FO* = 500,4.
   El valor de *FO* se calculó de la siguiente manera: *FO* = 1.000 * 0,774 - 600 * 0,456 = 500,4
   Donde, *PA* normalizado es = 774 puntos, y *LE* normalizado es = 273,6 segundos.
   Según los criterios de calidad, se observa que el estudiante alcanzó los objetivos JSE (774 puntos) en aproximadamente 4 minutos y 50 segundos (274 segundos).

3. El tercer estudiante: lo jugó con los parámetros $P_1$ = 81, $P_2$ = 4 y $P_3$ = 3, con *FO* = 255,6.

En este caso, *PA* normalizado es de 414 puntos y *LE normalizada* es de 158,4 segundos. Según los criterios de calidad, se observa que el alumno no alcanzó los objetivos JSE (414 < 500 puntos) en un tiempo aproximado de 3 minutos (158 segundos).

Después de 200 generaciones del algoritmo cultural (ver sección 3.3.2.3), la evolución del mejor individuo se observa en la figura 4.13.

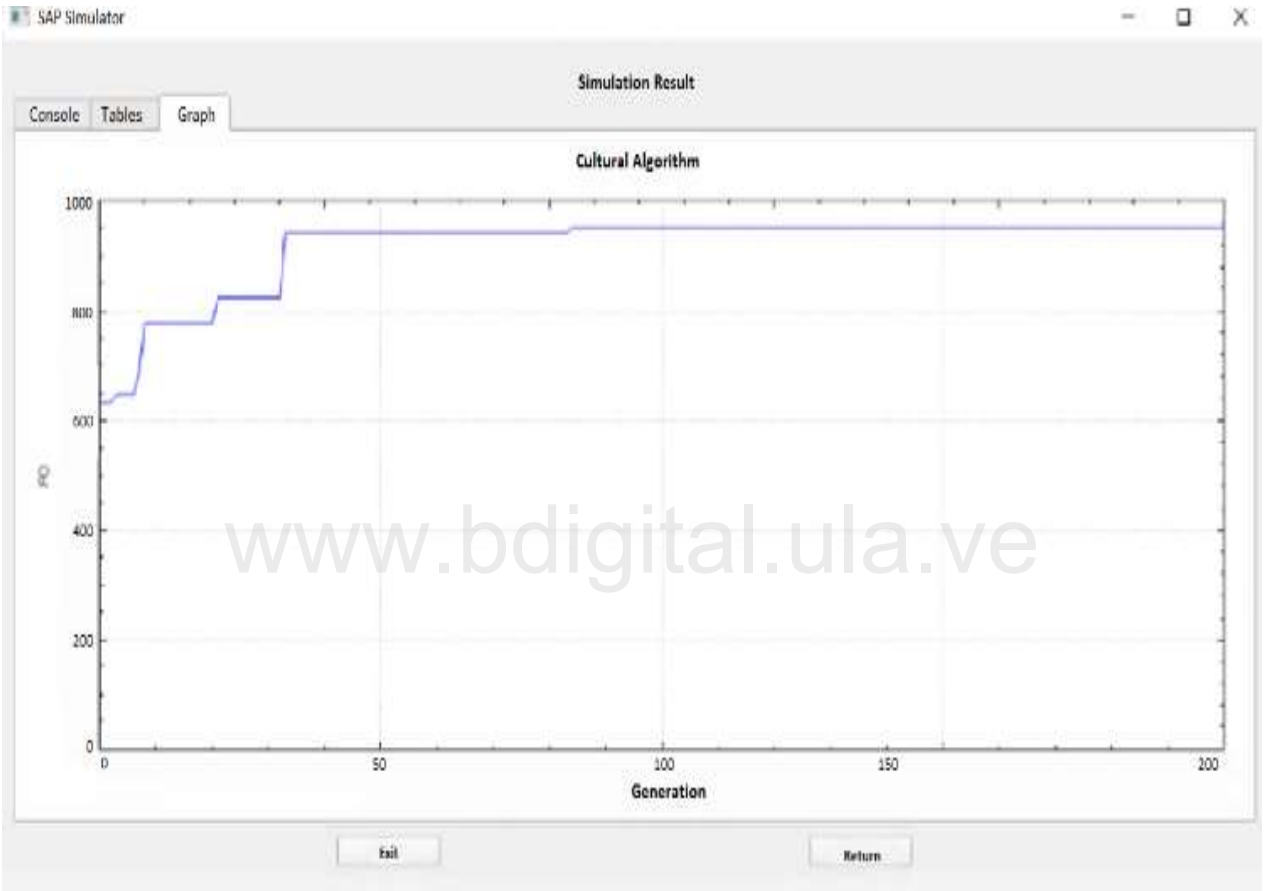





**Figura 4.13.** El mejor individuo durante las generaciones [66]

En la figura 4.13, el mejor individuo se presenta en la generación 84, con *FO* = 950,4. El *PA* normalizado es de 978 puntos y *LE normalizado* es de 27,6 segundos. En particular, para este caso de estudio, el SAP sugiere los parámetros $P_1 = 49$, $P_2 = 4$ y $P_3 = 4$, para que el jugador alcance los objetivos del "cálculo mental de raíz cuadrada" de JSE (978 puntos) en el menor tiempo posible (~ 28 segundos).

***Aprende las notas de la escala C:***

Ahora, supongamos un curso de música y el tema a estudiar son las "Notas Musicales" [66]. Luego, el *SEV* propone como JSE "*Aprender las notas de la escala C*", obtenido del enlace: https://aprendomusica.com/const2/27aprendonotas8/aprendonotas8.html (ver figura 4.14).

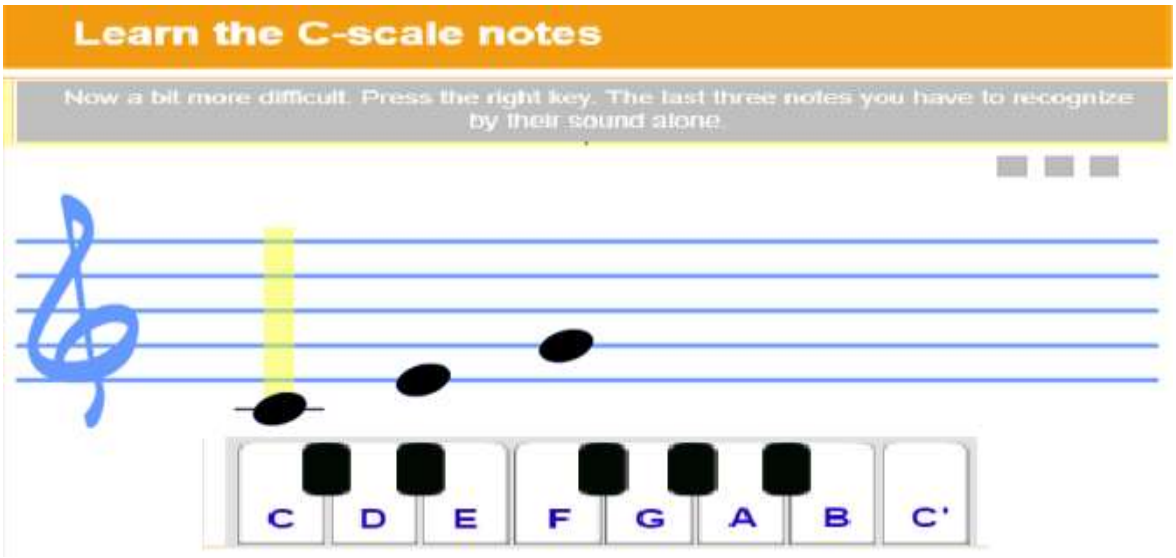



**Figura 4.14.** JSE: "Aprende las notas de la escala C" [66]

El videojuego " Aprende las notas de la escala C " consiste en reconocer a través del sentido del oído tres notas musicales, las cuales se mostrarán cuando el jugador presione la tecla correcta del piano, en el momento correcto, determinadas como una franja amarilla vertical. En un principio, el pentagrama contiene: la clave de sol (G) ver figura 4.14, cuya función es determinar la altura que corresponde a cada nota musical según la posición (espacio o línea) que ocupan en ella, y tres notas musicales (DO, RE-MI, FA) que sirven de guía para resolver los JSE.



**Tabla 4.10.** Notas musicales en español e inglés

| Español | Inglés |
|---|---|
| **DO** | C |
| **RE** | D |
| **MI** | E |
| **FA** | F |
| **SOL** | G |
| **LA** | A |
| **SI** | B |

El JSE consta de tres niveles, representados por los cuadrados grises, que cambian de color a verde si se alcanzan los objetivos de ese nivel, y a rojo o naranja en caso contrario. Este JSE tiene cuatro parámetros:

- $P_1$ = Claves que tiene el jugador.
- $P_2$ = Notas para adivinar en el pentagrama.
- $P_3$ = Penalización por mala jugada.
- $P_4$ = Niveles.

Los criterios de calidad son los mismos definidos anteriormente. En particular, el tiempo y la puntuación son definidos por el profesor de la clase. En este caso, se definen como:

- *Intervalo* de tiempo: el límite de tiempo es de 60 segundos (1 minuto).
- *Puntuación acumulativa:* si el valor es superior a 500 puntos, el jugador alcanzó la mayoría de los objetivos JSE y el valor máximo es de 1.000 puntos.

La tabla 4.11 muestra la población inicial del algoritmo cultural (ver sección 3.3.2.1), con varias veces que inicialmente los estudiantes jugaron el JSE.

**Tabla 4.11.** Las primeras veces que se juega el JSE

| # | $P_1$ | $P_2$ | $P_3$ | $P_4$ | *FO* |
|---|---|---|---|---|---|
| **1** | 4 | 2 | 7 | 2 | 781,8 |
| **2** | 7 | 3 | 6 | 2 | 287 |
| **3** | 7 | 2 | 9 | 2 | 241,8 |

En la tabla 4.11:

1. El primer estudiante: lo jugó con los parámetros $P_1 = 4$, $P_2 = 2$, $P_3 = 7$, $P_4 = 2$, con *FO* = 781,8, que se calculó de la siguiente manera:



   *FO* = 1.000 * 0,81 - 60 * 0,47 = 781,8

Donde, el *PA* normalizado es de 810 puntos, y *LE normalizado* es de 28,2 segundos. Según los criterios de calidad, se observa que el estudiante alcanzó los objetivos del JSE (810 puntos) en aproximadamente 28 segundos.

2. El segundo estudiante: lo jugó con los parámetros $P_1 = 7$, $P_2 = 3$, $P_3 = 6$, $P_4 = 2$, con *FO* = 287. Para este estudiante, el *PA normalizada* es de 290 puntos y *LE normalizada* es de 3 segundos. Según los criterios de calidad, el estudiante no alcanzó los objetivos del JSE (290 <500 puntos).

3. El tercer estudiante: lo jugó con los parámetros $P_1 = 7$, $P_2 = 2$, $P_3 = 9$, $P_4 = 2$, con *FO* = 241,8. Para este estudiante, el *PA normalizada* es de 270 puntos y *LE normalizada* es de 28,2 segundos. Según los criterios de calidad, el alumno no alcanzó los objetivos del JSE (270 <500 puntos), jugando aproximadamente 28 segundos.

Después de 200 generaciones, la evolución del mejor individuo se observa en la figura 4.15. Dicho mejor individuo aparece en la generación 141.

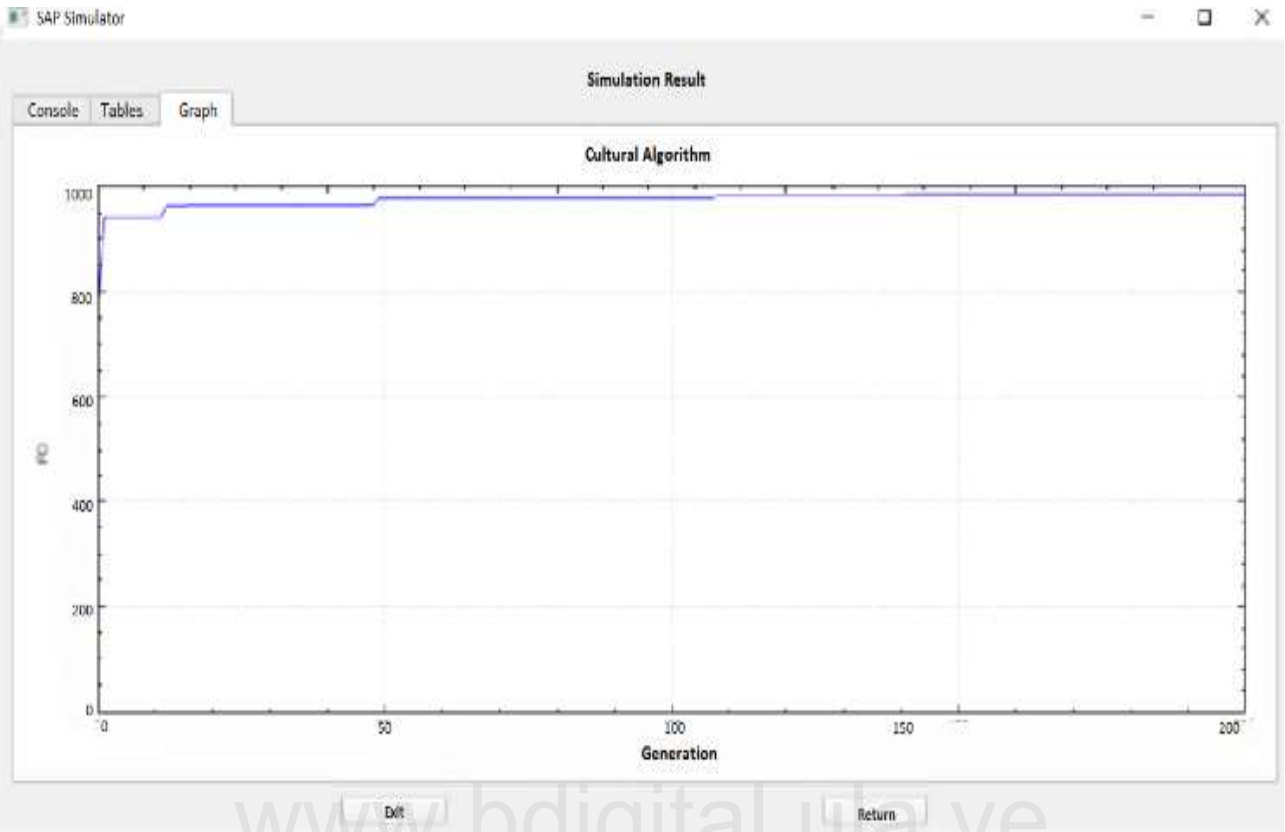





**Figura 4.15.** El mejor individuo durante las generaciones [66]

El mejor individuo aparece a partir de la generación 141, con *FO* = 983,4. En este caso, el *PA normalizada* es de 990 puntos, y *LE normalizada* es de 6,6 segundos. Por tanto, el SAP sugiere los parámetros $P_1 = 4$, $P_2 = 1$, $P_3 = 9$ y $P_4 = 1$, para que el jugador alcance los objetivos del JSE (990 puntos) en el menor tiempo posible (~ 7 segundos).

### 4.3.2. EXPERIMENTOS

En este apartado se define el experimento según [66], realizado para evaluar su calidad.

**Pruebas del prototipo:**

Su objetivo es determinar los valores adecuados de los diferentes parámetros del SAP (en particular, el *algoritmo cultural*), que serán los valores por defecto para su correcto funcionamiento. En concreto, los valores a optimizar son: la probabilidad de utilizar los operadores, el número de generaciones, el tamaño de la población inicial, y el valor de $\mu$ *(que es una constante de momento entre 0 y 1 como se definió en la ecuación 3.6).* Este experimento se realiza con un JSE denominado "Polígonos con el Tangram", obtenido del repositorio AGREGA (http://agrega.educacion.es ) (ver figura 4.17).

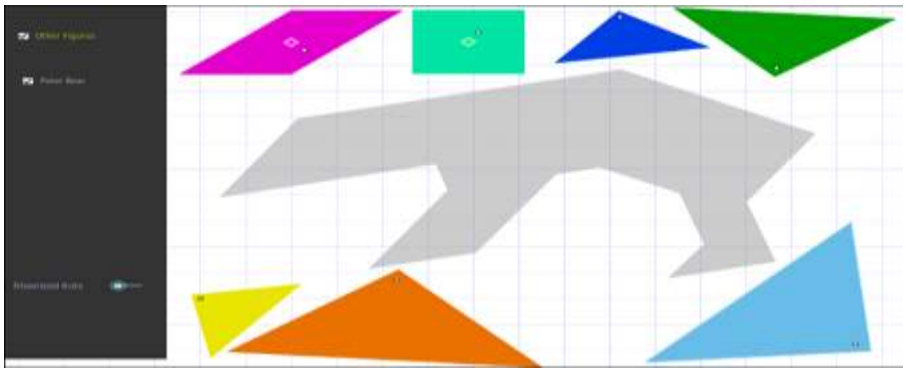


**Figura 4.16.** El JSE: "Polígonos con el Tangram" [66]

Este JSE consiste en resolver el Tangram, que es un juego de puzles de origen chino compuesto por siete piezas, obtenidas al dividir un cuadrilátero: cinco triángulos (dos grandes, uno mediano y dos pequeños) y dos cuadriláteros (un cuadrado y un romboide), con el objetivo de formar una figura compuesta por 13 posibles polígonos convexos (triángulo, cuadrado, rectángulo, romboide, trapezoide isósceles, rectángulo trapezoide, pentágono trapezoide, pentágono, 4 hexágonos). Esta figura tiene alguna de las siguientes formas: hombre sentado, oso polar (como se muestra en la figura 4.15), cachorro, velero, cisne, flecha negra y flecha blanca.



Los parámetros de ese JSE a considerar son:

$P_1$ = Figuras para construir
$P_2$ = Polígonos convexos que tiene el jugador

**Proceso de calibración:**

Los límites iniciales para cada parámetro, requeridos para inicializar el conocimiento normativo, se muestran en la tabla 4.12.

**Tabla 4.12.** Conocimiento normativo

| $P_1$ | | $P_2$ | |
|---|---|---|---|
| $LI_1 : 1$ | $LS_1 : 7$ | $LI_2 : 1$ | $LS_2 : 13$ |

Los valores iniciales para iniciar la simulación se encuentran en la tabla 4.13.

**Tabla 4.13.** Valores iniciales de los SAP

| Individuos | Generación | Probabilidad de Cruce | Probabilidad de Mutación | $\mu$ | $A$ | $B$ | $PA$ | $LE$ |
|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 0,1 | 0,1 | 0,1 | 100 | 100 | 0,1 | 0,1 |

En esta prueba solo se utilizan los conocimientos situacionales y normativos. En la figura 4.17 se muestran las tablas del conocimiento situacional y normativo, y los mejores individuos de cada generación. En el caso del conocimiento situacional se muestran los valores de los parámetros $P_1$ y $P_2$ que provienen de los mejores individuos, cada uno con su ocurrencia y calidad media. Como conocimiento normativo, se muestran los límites inferior y superior de los parámetros, siguiendo el formato: [$LI_1$, $LS_1$] y [$LI_2$, $LS_2$] para cada parámetro y generación. La figura también muestra la tabla de los mejores individuos de cada generación. Los individuos se presentan en el formato [$P_1$, $P_2$], junto con el valor de su *FO.*


**Figura 4.17.** Conocimiento situacional, normativo y el mejor individuo de cada generación [66]

En la figura 4.17 se observa cómo el conocimiento situacional de ambos parámetros influye en los mejores individuos. El mejor individuo aparece a partir de la generación 13, manteniéndose así hasta la convergencia del algoritmo.

Ahora, se analiza la sensibilidad de los parámetros del SAP. La primera prueba es variando el número de individuos en la población final (ver tabla 4.14).

**Tabla 4.14.** Variaciones del número de individuos

| Número de individuos | FO del mejor individuo |
|---|---|
| **30** | 77,3442 |
| **50** | 90,6724 |
| **70** | 86,7574 |
| **80** | 93,8339 |
| **100** | 90,7341 |

Los resultados indican un mejor valor de *FO* para 80 individuos de la población. Por lo tanto, el número de individuos se establece en 80.

La segunda prueba es variando la probabilidad del operador de cruce (ver tabla 4.15).

**Tabla 4.15.** Variaciones de la probabilidad del operador de cruce

| Probabilidad | *FO* del mejor individuo |
|---|---|
| **0,3** | 95,2848 |
| **0,5** | 96,6373 |
| **0,7** | 92,4736 |
| **0,9** | 96,0042 |
| **1,0** | 96,207 |

Los resultados indican que con una probabilidad de cruce del 50% se obtiene un mejor valor de *FO*. Por lo tanto, la probabilidad del operador de cruce se establece en 0,5.

La tercera prueba es variando la probabilidad del operador de mutación (ver tabla 4.16).

**Tabla 4.16**. Variaciones de la probabilidad del operador de mutación

| Probabilidad | *FO* del mejor individuo |
|---|---|
| **0,3** | 96,6871 |
| **0,5** | 96,207 |
| **0,7** | 96,2725 |
| **0,9** | 94,6158 |
| **1,0** | 97,7206 |



Los resultados indican que con un 100% de probabilidad de mutación se obtiene un mejor valor de *FO*. Por lo tanto, la probabilidad de mutación se establece en 1,0.

La cuarta prueba es variando el número de generaciones (ver tabla 4.17).

**Tabla 4.17.** Variación del número de generaciones

| Número de generaciones | FO del mejor individuo |
|---|---|
| **30** | 97,7206 |
| **50** | 97,7206 |
| **100** | 99,1939 |
| **200** | 99,8165 |

Los resultados indican que con 200 generaciones se obtiene el mejor valor de *FO*. Por lo tanto, el número de generaciones se establece en 200.

La última prueba es variando $\mu$ (ver tabla 4.18).

**Tabla 4.18.** Variación de μ

| *μ* | *FO* del mejor individuo |
|---|---|
| **0,3** | 99,1843 |
| **0,5** | 99,1843 |
| **0,7** | 99,3492 |
| **0,8** | 99,3492 |
| **0,9** | 99,3492 |
| **1,0** | 99,1843 |

Los resultados indican un aumento en el valor de *FO*, con μ entre 0,3 y 0,7. Luego, con μ entre 0,7 y 0,9, comienza a disminuir. Por lo tanto, el valor μ se establece en 0,7.

### 4.3.3. ANÁLISIS DE RESULTADOS PARA EL SAP

Ahora, analizamos la calidad de los JSEs según [66] generados por el SAP, utilizando los parámetros definidos en el sub apartado anterior. El JSE se juega 26 veces en estas pruebas. También, para estas pruebas los criterios de calidad son:



1. Timespan: es el valor de *LE,* en segundos. Se supone que se establece un límite de tiempo para cumplir con los objetivos de los JSEs. Inicialmente se definen 300 segundos (5 minutos) por lo que, si se supera este valor, hasta un máximo de 600 segundos (10 minutos). Entonces, significa que el jugador tardó demasiado en cumplir los objetivos. Si se alcanza el valor máximo, entonces no se ha alcanzado la meta.

2. Alcance de los objetivos: es el valor de *PA*, puntuación acumulada. La hipótesis es que 1.000 puntos es el valor cuando se cumplen todos los objetivos del JSE, mientras que con un valor en torno a los 500 puntos se puede considerar alcanzados la mayoría de los objetivos alcanzados.

   Los resultados de los criterios de calidad para la JSE "Polígonos con el Tangram" se muestran en la figura 4.18.

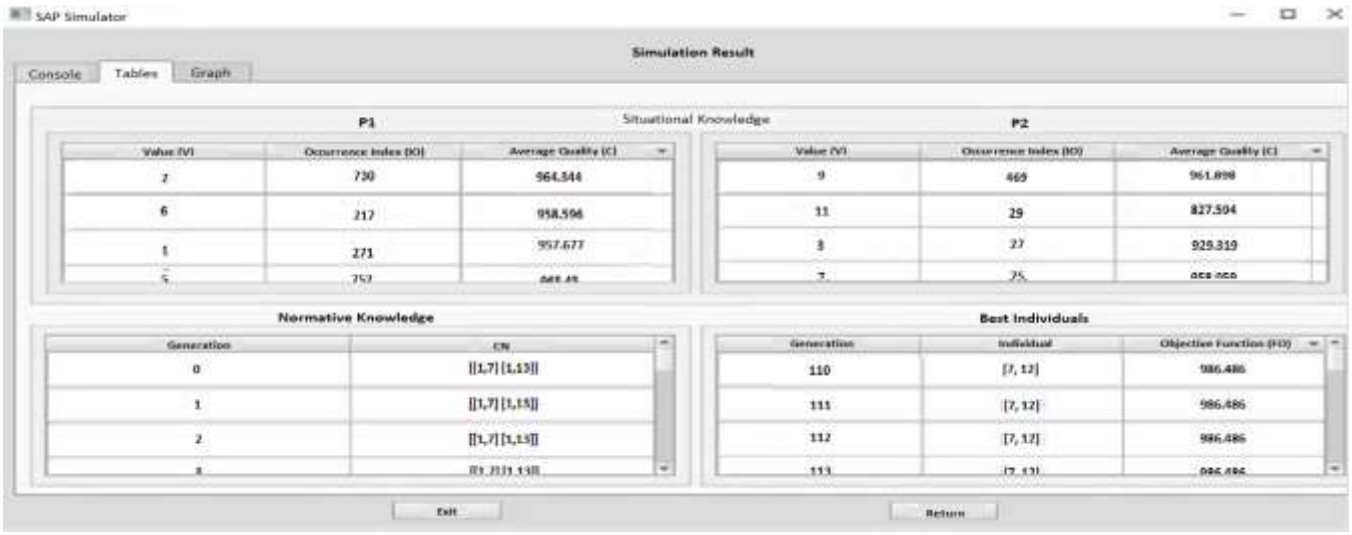



**Figura 4.18.** Resultados de la prueba de calidad [66]

La figura 4.18 muestra al mejor individuo, con un *FO* de 986.486, en la generación 110, con los siguientes parámetros:

- $P_1$ = 7 (figuras de jugadores)
- $P_2$ = 12 (convexo polígonos)



El valor de *FO* se calculó de la siguiente manera:
*FO* = 1.000 * 0,99674 - 600 * 0,017090833 = 986,4855

Donde, el *PA normalizado es* = 996.74 ≈ 997 puntos, y *LE normalizado es* = 10.2545499 ≈ 10 segundos.
Como se puede observar en la *FO*, con estos parámetros en el JSE, los alumnos pueden cumplir con los dos criterios de calidad: duración del juego y objetivos alcanzados.

Después de haber realizado simulaciones con otros JSEs utilizados en un SaCI, optimizados por nuestro SAP, se concluye lo siguiente:

- Si hay un gran número de individuos en la población, entonces es necesario definir un número de generaciones lo suficientemente grande como para garantizar que los resultados sean satisfactorios.
- En cuanto a las probabilidades de cruce y mutación: considerando 0,5 y 1, respectivamente, se obtendrán los mejores resultados. Sin embargo, si son bajos, entonces es necesario definir un número muy grande de generaciones para obtener los mejores resultados.
- El valor μ se definió como 0,7. Se demostró a través de los experimentos, que a mayor tamaño (> 0.7) hay una disminución en el valor de *FO* del mejor individuo.

## 4.4. CASO DE ESTUDIO DEL SAE

En el Capítulo III, apartado 3.3.3, se explica detalladamente tanto el diseño como la implementación del SAE, utilizando un SCD. En esta sección se mostrará el funcionamiento del SAE en el contexto del SaCI.

### 4.4.1. DESCRIPCION FUNCIONAL DEL SAE

**Contexto en un SaCI:**

Inicialmente, el MJSE [55] selecciona un JSE para la carrera de Programación de Videojuegos, específicamente para la materia Software Testing, con el objetivo de dar una introducción al curso de videojuegos clásicos. El SAT nos arroja por defecto el popular y famoso videojuego "Súper Mario Bros" (figura 4.19) como JSE, con alta puntuación, ya que además de ser entretenido, tiene las capacidades de generar ejercicios para desarrollar movimientos en el joystick de forma sencilla y clara como: saltar, correr y disparar.


**Figura 4.19.** El famoso videojuego "Súper Mario Bros"

El jugador asume los roles de Mario o de Luigi presionando «select». El objetivo es recorrer el Reino Champiñón para derrotar a las fuerzas del Rey Koopa y salvar a la Princesa Peach. Si reciben un contacto enemigo, se pierde una vida, por ello, los hermanos Mario/Luigi tienen un primer ataque que consiste simplemente en saltar sobre el enemigo —presionando la tecla «A» en el control—, siendo los champiñones conocidos como Goombas (Hongos), los primeros en aparecer. Igualmente, es posible saltar sobre los Koopa Troopas (Tortugas), y saltando una segunda vez sobre ellos, es posible lanzar su caparazón. Al patear este caparazón, se puede derrotar también a los enemigos que se encuentran delante, con el inconveniente de que, si hay un obstáculo, el caparazón regresa y puede herir a Mario o Luigi. Si alguno de ellos recoge un champiñón aumenta de tamaño, y pueden ser heridos hasta dos veces antes de perder una vida (conociéndose esta transformación como Super Mario/Luigi). Por otro lado, cogiendo una flor, se obtiene la habilidad de lanzar bolas de fuego con un máximo de dos por vez —presionando una vez la tecla «B» en el control—. Algunos enemigos no pueden ser derrotados saltando sobre ellos; estos solo pueden ser eliminados con un caparazón o con las bolas de fuego, o bien al ser tocados por Mario/Luigi estrella [67].

**Emergencia Fuerte de Estrategias:**

Esta sección describe el proceso de emergencia de estrategia gestionado por el AJSE. A partir de los datos del contexto del SaCI proporcionados por los otros agentes, y con el JSE generado inicialmente por el SEV, el SAE usa al SCD para ir adecuando las reglas que definen las estrategias del JSE, las cuales van emergiendo a través del tiempo.

1. Eventos y Acciones: mediante el SCD se establecen los diferentes tipos de eventos (parámetros) y acciones que pueden suceder en el juego, con sus respectivos valores. Para nuestro fin, en el sistema divide los parámetros de la siguiente forma:
   a. **Enemigo:** es el primer parámetro que detecta el sistema, los cuales son: los hongos, las tortugas, las plantas, los reyes, etc. (ver figura 4.20). Se miden de la siguiente forma, según su nivel de agresividad:
      I. *Tipo 1:* (1,0 – 3,3) es estático y/o dispara, por ejemplo: una planta piraña.
      II. *Tipo 2: (*3,4 – 6,6) es dinámico y no dispara; por ejemplo: hongos o tortugas.
      III. *Tipo 3:* (6,7 – 9,9) es dinámico y dispara; por ejemplo: el Rey Koopa disparando.
      IV. *Ninguno:* (0,0 – 0,9) ausencia de enemigos en la escena (no es necesario disparar).

Planta (Piraña) | Koopa Troopa (Tortugas) | Goomba (Hongo) | Bowser (Rey Koopa)

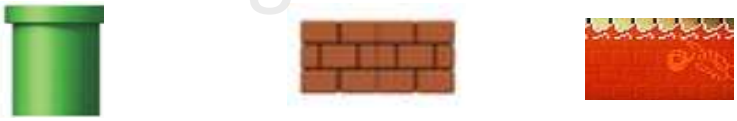
**Figura 4.20.** Enemigos

   b. **Hueco:** es el segundo tipo de parámetro, pueden haber de diferentes tamaños, como se explica a continuación (ver figura 4.21):
      I. *Corto:* (1,0 – 3,3) de tamaño muy pequeño, donde cabe un Mario/Luigi, y muchas veces no es necesario correr para agarrar impulso para saltar, o incluso se pasa corriendo rápido (sin saltar).
      II. *Medio:* (3,4 – 6,6) de tamaño medio pequeño, donde caben dos o tres Marios/Luigis, y muchas veces no es necesario correr para agarrar impulso y saltar.
      III. *Largo:* (6,7 – 9,9) de tamaño muy grande, donde caben cuatro o más Marios/Luigis. Es bastante amplio, se necesita correr muy rápido para saltar y pasarlo.
      IV. *Ninguno:* (0,0 – 0,9) ausencia de hueco en la escena (no es necesario saltar).

**Figura 4.21.** Huecos

c. **Obstáculos:** son el tercer tipo de parámetro, y se divide de la siguiente manera (ver figura 4.22):
   I. *Tubo:* (1,0 – 3,3) es un tubo verde que se puede saltar o se puede introducir dentro de él.
   II. *Muro:* (3,4 – 6,6) son bloques de piedra que construyen los escenarios, se puede estar por encima de ellos y saltar.
   III. *Lava:* (6,7 – 9,9) es fuego hirviendo que aparece en los escenarios. Si Mario o Luigi lo tocan se mueren.
   IV. *Ninguno: (0,0 – 0,9)* no existe ningún obstáculo en la escena (no es necesario saltar).





Tubo Muro Lava

**Figura 4.22**. Obstáculos

d. **Armas:** es el cuarto parámetro a utilizar en el sistema, y se divide de la siguiente forma (ver figura 4.23):
   I. *Bala:* (1,0 – 3,3) es lanzada por diferentes enemigos, como: cañones, tubos, o a veces salen de la nada.
   II. *Nube:* (3,4 – 6,6) se encuentra estática en el cielo del videojuego. Si se le da la espalda comienzan a perseguir a Mario/Luigi hasta que lo logran tocar (restando su energía).
   III. *Bomba:* (6,7 – 9,9) está caminando en sitios estratégicos de la tierra del videojuego, y si Mario/Luigi está muy cerca se activa y explota.
   IV. *Ninguno:* (0,0 – 0,9) no existe armas en la escena, no es necesario esquivar.

Bala Nubes Bomba

**Figura 4.23.** Armas

En este videojuego, Mario o Luigi solo pueden hacer tres acciones, como muestra la tabla 4.19, que son saltar con la tecla «A» (será acción A), disparar con la tecla «B soltando» (acción B), y correr con la tecla «B sin soltar presionando hacia la derecha o izquierda para agarrar más velocidad» (acción C).

**Tabla 4.19.** Eventos y acciones en el JSE

| Estado | Variables | Valores posibles |
|---|---|---|
| **Entradas** | | |
| **Evento** | Enemigo | Tipo: 1, 2 y 3 |
| | Hueco | Corto, Medio y Largo |
| | Obstáculo | Tubo, Muro y Lava |
| | Armas | Bala, Nube y Bomba |
| **Salida** | | |
| **Acciones** | A | Saltar |
| | B | Disparar |
| | C | Correr |

2. Estrategias: se establecen como reglas para el contexto del JSE. Ellas establecen cuando se debe saltar, disparar o correr. En particular, se deben establecer reglas genéricas alrededor de las escenas. La tabla 4.20 muestra algunos ejemplos. Por ejemplo, la regla 1 establece que cuando se encuentra un enemigo 2 (Tortuga u Hongo), entonces se debe realizar una acción de disparar. Las reglas 2, 3 y 4 indican que, si hay un evento hueco u obstáculo tubo, entonces debe realizarse la acción Saltar. Finalmente, la regla 5 indica que cuando ocurre el evento arma Bala, entonces debe realizarse la acción Correr.



**Tabla 4.20.** Ejemplo de Reglas Genéricas

| N° | Regla |
|---|---|
| **R1** | Si <Enemigo=2> entonces <B=Disparar> |
| **R2** | Si <Hueco=Corto> entonces <A=Saltar> |
| **R3** | Si <Hueco=Medio> entonces <A=Saltar> |
| **R4** | Si <Obstaculo=Tubo> entonces <A=Saltar> |
| **R5** | Si <Arma =Bala>entonces <C=Correr> |

3. Ejecución del JSE: a continuación, se muestra un ejemplo del desarrollo del JSE:

En el siguiente ejemplo (ver figura 4.24), se observa que Mario capturo una flor de fuego y puede lanzar fuego. En ese caso, se activa la regla R1 por Enemigo=2 (Hongo), tal que la acción a realizar es B=Disparar.

**Figura 4.24.** Mario dispara

En otro ejemplo diferente, observamos una escena donde Mario se ha vuelto pequeño (ver figura 4.25). En esta escena se activa la regla R3 por Hueco = Medio, pero también por R4, ya que se observa Obstáculo = Tubo. Ahora, también se observa Arma = Bala, lo que activa R5. Eso daría como resultado que Mario deba A = Saltar y C = Correr al mismo tiempo.

Figura **4.25.** Mario actúa

**Funcionamiento del SAE:**

Esta sección describe la aplicación del SCD en el videojuego *Súper Mario Bros.* Para ello, se utilizó el código fuente de [61], y se definieron las variables lingüísticas requeridas por el JSE antes descrito, usando la interfaz gráfica de la figura 4.26. Inicialmente, se introduce la Probabilidad de Mutación = 0,05, la Población = 20, el Numero de Generaciones = 5, y el Tamaño de Torneo = 5.




**Figura 4.26.** Interfaz del SCD

Esa interfaz tiene las siguientes acciones:

1. Introducir archivo: se introduce la población de individuos, que está en el archivo Reglas.data, el cual contiene diferentes variantes de las reglas según los parámetros definidos antes: *Enemigo, Hueco, Obstáculo, Armas y Clase*. Estas reglas permiten saber en qué momento se debe (*Saltar y Disparar)*, (*Correr y Saltar*) o (*Correr o Disparar*), etc. (ver figura 4.27).

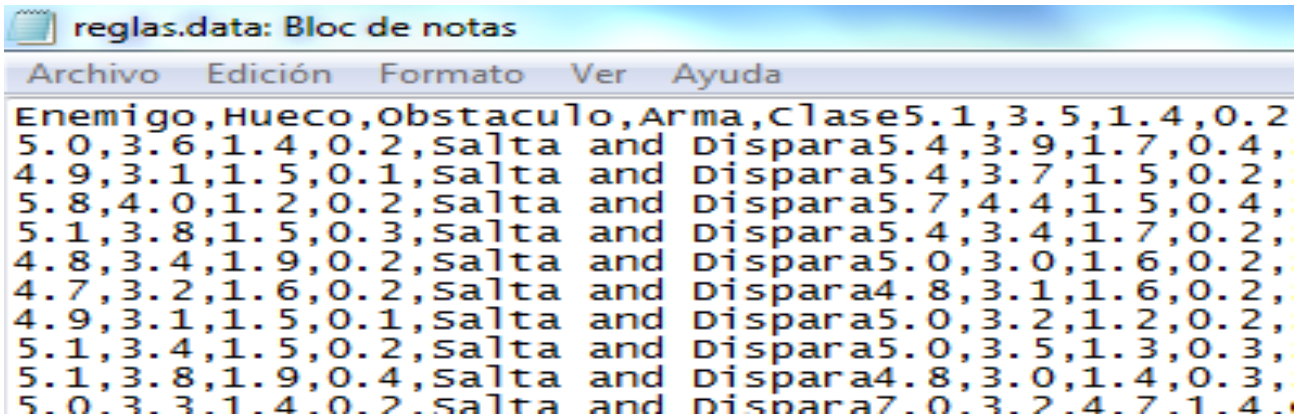



**Figura 4.27.** Reglas.data

Los valores de las variables de los parámetros van de 0.1 hasta 9.9, midiendo la intensidad con que ocurre aleatoriamente. Esto va a determinar cuándo un parámetro está activo o no.



2. Porcentaje Entrenar: luego se coloca un dataset (Datos de Entrenamiento), que son juegos previos simulados de 150 individuos. Se usa solo el 20% para entrenar (30 individuos). En la aplicación aparece lo siguiente:
   a. Los datos usados para el entrenamiento (ver figura 4.28) del dataset original *Reglas.data* de entrenamiento (ver figura 4.29), con el formato 0.XXXXXX para poder fuzzificar.

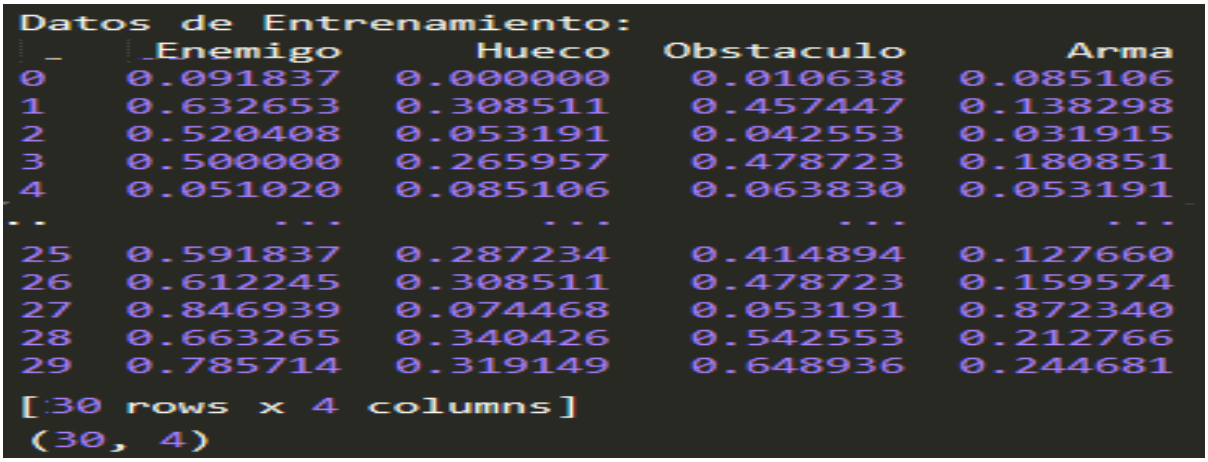



**Figura 4.28.** Datos usados de entrenamiento

```
Datos del Test:
      Enemigo      Hueco   Obstaculo      Arma
0    0.551020   0.414894    0.287234  0.042553
1    0.061224   0.382979    0.106383  0.978723
2    0.653061   0.340426    0.563830  0.244681
3    0.091837   0.042553    0.351064  0.106383
4    0.561224   0.255319    0.404255  0.117021
..        ...        ...         ...       ...
115  0.020408   0.053191    0.170213  0.765957
116  0.091837   0.010638    0.053191  0.861702
117  0.040816   0.095745    0.031915  0.351064
118  0.591837   0.425532    0.021277  0.021277
119  0.438776   0.319149    0.117021  0.010638

[120 rows x 4 columns]
(120, 4)
```

**Figura 4.29.** Datos del test

b. Genera los datos para entrenar (ver figura 4.30), generando un mensaje de entrenamiento exitoso al finalizar. La función objetivo considera los siguientes valores para cada acción realizada:

-1 = No valido
0 = Salta y Dispara
1 = Correr y Saltar
2 = Correr o Disparar



```
Datos para Entrenar y buscar el Objetivo:
0     0
1     1
2     0
3     2
4     1
     ..
70    1
71    0
72    0
73    1
74    0
Length: 75, dtype: int32
Test para el Objetivo:
0     2
1     1
2     2
3     0
4     1
     ..
70    1
71    1
72    0
73    2
74    2
Length: 75, dtype: int32
Entrenamiento Exitoso.
```

**Figura 4.30**. Datos de finalización

3. Generar y Clasificar Reglas con Algoritmos Genéticos: la última generación tiene las mejores reglas (ver figura 4.31):

```
Generacion numero : 4
Mejor precision ajustada : 0.05263157894736842
Regla 1: [1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0]   (-1, 0)
Regla 2: [1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0]   (1, 1.0)
Regla 3: [0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0]   (-1, 0)
Regla 4: [1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1]   (0, 0.48888888888888876)
Regla 5: [1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0]   (2, 0.4844658992449561)
Regla 6: [1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0]   (0, 0.6367521367521367)
Regla 7: [0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1]   (0, 0.43513513513513513)
Regla 8: [0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0]   (2, 0.43421052631578944)
Regla 9: [0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0]   (-1, 0)

Calculo de complejidad: 12
```

**Figura 4.31.** Última generación

4. Regla del Sistema**:** se redacta literalmente el resultado de la figura 4.31 para que sea entendido por el usuario (ver figura 4.32). Ahora bien, las nuevas escenas serán controladas por estas nueve reglas generadas.

```
Reglas del sistema
Regla 1: If Enemigo = Hongo and Hueco = Largo and Obstaculo = Lava and Arma = Nube Then Not Valid
Regla 2: If Enemigo = Planta and Hueco = Medio and Obstaculo = Muro and Obstaculo = Tubo and Arma = Bala Then Corre and Salta
Regla 3: If Hueco = Largo and Arma = Bal Then Not Valid
Regla 4: If Enemigo = Planta and Hueco = Medio or Hueco = Largo and Obstaculo = Lava and Arma = Bomb Then Salta and Dispara
Regla 5: If Enemigo = Hongo and Hueco = Corto and Obstaculo = Muro and Arma = Nube Then Corre or Dispara
Regla 6: If Enemigo = Hongo or Enemigo = Tortuga or Enemigo = Planta and Hueco = Medio and Obstaculo = Muro or Obstaculo = Lava and Arma =
    Nube Then Salta and Dispara
Regla 7: If Obstaculo = Muro or Obstaculo = Lava and Arma = Nube or Arma = Bomb Then Salta and Dispara
Regla 8: If Obstaculo = Tubo and Arma = Bal Then Corre or Dispara
Regla 9: If Hueco = Largo and Obstaculo = Lav Then Not Valid

Salida de cada regla
Not Valid, Corre and Salta, Not Valid, Salta and Dispara, Corre and Dispara, Salta and Dispara, Salta and Dispara, Corre and Dispara, Not
    Valid
```



**Figura 4.32.** Regla del sistema

Podemos observar que la regla 2, a pesar de tener el obstáculo muro de sobra, es la que más se parece a nuestra escena, y además, posee una forma de actuar correcta: correr y saltar.

En este caso, se usaron datos simulados de haber jugado antes el JSE. Si fuera un contexto real, una vez actualizada las reglas, se inicia un nuevo ciclo (iteración) del SCD cada cierto tiempo. Así, si se dejase el JSE durante un tiempo funcionando en el SaCI para un curso y grupo de usuarios, ira cada cierto tiempo invocándose al SAE para ir estabilizando el grupo de reglas de estrategias adecuadas para ese contexto en el tiempo. En menos palabras, este mecanismo hace emerger el grupo de reglas estratégicas del JSE para un contexto dado para un SaCI.

### 4.4.2. EXPERIMENTOS

En este apartado se estudia los resultados de la eficacia de nuestra propuesta. Para ello, se generó un entorno virtual donde se asumió el contexto del SaCI anterior, y en cada prueba se jugó uno de los siguientes juegos:

1. Donkey Kong: es un videojuego en el que el jugador elige un personaje de un simio, que viaja en carros y barriles, usa armas para disparar, y vuela un avión para transportarlo de un lugar a otro. Es posible caminar, saltar y correr sobre la plataforma, balanceándose y despejando el camino a través de los niveles del juego.
2. Super Tux: es un videojuego de plataforma de aventuras de desplazamiento lateral que presenta al pequeño pingüino Tux como el personaje principal del juego. El juego tiene veintiséis niveles en los que el jugador puede correr y saltar sobre plataformas, y matar a los enemigos saltando sobre sus cabezas.
3. Kirby's Adventure: es un videojuego de desplazamiento lateral y rompecabezas de plataforma que presenta las aventuras de Kirby. El juego permite hacer rodar y saltar a Kirby de una plataforma a otra. Es posible copiar las habilidades de los enemigos, y usarlas contra ellos.
4. Crash Bandicoot: es un videojuego de un solo jugador, de desplazamiento lateral, donde la historia sigue un complot para reducir el tamaño del planeta Tierra por parte del malvado Doctor Neo Cortex, utilizando un arma gigantesca llamada Planetary Minimizer. El jugador actúa como el protagonista principal de la serie llamada Crash Bandicoot, y su tarea es recolectar cristales a lo largo de los niveles para restaurar la Tierra a su tamaño original.



La función de aptitud promedio de la población inicial (primera invocación) y después de 10 invocaciones, se compara para determinar si el SCD mejoró el sistema de reglas (ver tabla 4.21). Recuerde que la función de aptitud está definida por el número de veces que se usa una regla, lo cual es un indicador de supervivencia (se evapora si no se invoca una regla, y se refuerza cuando se usa), pero también, está determinada por su utilización debido a la calidad de la estrategia expresada por la regla (ver sección 3.3.3).

**Tabla 4.21.** Evaluación del desempeño del SCD

| Videojuego | Función de aptitud promedio inicial | Función de aptitud promedio final |
|---|---|---|
| 1 | 1.2 | 4.2 |
| 2 | 0.63 | 3.6 |
| 3 | 0.9 | 4.1 |
| 4 | 0.6 | 3.8 |

En la tabla 4.21 se muestra que la calidad del sistema de reglas mejora considerablemente en los cuatro videojuegos, tal que la función de aptitud promedio final es superior sobre la inicial.

## 4.5. COMPARACIÓN CON OTRAS PROPUESTAS

A continuación, se hace la comparación del SAT, SAP Y SAE en base a criterios que permiten obtener una métrica con respecto a otros trabajos.

**Comparación del SAT con otros trabajos:**

En la tabla 4.22 se compara nuestra propuesta de SAT con otros trabajos, en base a los siguientes criterios:

a. ¿Es un juego serio? Con esta pregunta, queremos determinar si la propuesta es un juego serio.
b. ¿Tiene o confiere habilidades para adaptarse al juego? Con esta pregunta queremos determinar si se proponen capacidades adaptativas.
c. ¿Permite cualquier tipo de emergencia? Con esta pregunta queremos determinar si las propiedades del juego se desarrollan de forma espontánea, autónoma y sin leyes explícitas, adaptándose a los jugadores.
d. ¿Es parte de un Motor de Juego? Con esta pregunta queremos determinar si la propuesta es parte de un Motor de Juego, siendo uno de sus componentes.



**Tabla 4.22.** Comparación con otros trabajos recientes

| Criterios | Rasim y otros [68] | Martínez y otros [69] | D'Amato y otros [32] | Tregel y otros [33] | Subbaraj y otros [70] | Chen y otros [71] | Jamieson y otros [72] | Daylamani - Zad y otros [31] | Barajas y otros [73] | Trabajo actual |
|---|---|---|---|---|---|---|---|---|---|---|
| **a** | X | X | | | | | | | X | X |
| **b** | X | | X | X | | X | X | X | | X |
| **c** | | | | | | | X | X | | X |
| **d** | X | | | | | | | | | X |

En [68], Rasim y otros proponen el desarrollo de juegos serios utilizando un motor adaptable, que utiliza una técnica de aprendizaje automático para el proceso de adaptación de los juegos serios a diferentes jugadores. El motor adaptativo consta de reglas basadas en la descripción de las competencias del jugador, estilos de aprendizaje, estados cognitivos, entre otras cosas. Martínez y otros [69] proponen una arquitectura para el desarrollo de videojuegos educativos para incentivar a usuarios inexpertos a interactuar con algoritmos de inteligencia artificial. El juego permite al usuario experimentar a través de cambios en los parámetros de los algoritmos de ACO.

D'Amato y otros [32] propone un enfoque basado en ACO para el diseño cooperativo de una red de líneas aéreas, y proporciona un ejemplo para rutas de tráfico aéreo intercontinental. En [33], las rutas se optimizan utilizando un algoritmo ACO mejorado, para generar dinámicamente una ruta personalizada. Subbaraj y otros [71] proponen un modelo de juego no cooperativo para enrutamiento seguro y tolerante a fallas en redes móviles ad hoc (MANET) basado en ACO. En [71], ACO se utiliza para explorar el entorno y comunicar información sobre los recursos, con el fin de proporcionar capacidades de adaptación al juego.

En el trabajo [72] se describen dos juegos que demuestran la viabilidad del uso de algoritmos heurísticos para problemas de optimización combinatoria: el primero es un juego de estrategia que hace emerger recursos en tiempo real usando ACO. El segundo es un juego de carreras, donde las técnicas de inteligencia artificial adaptan características numéricas como la velocidad, la agilidad y la potencia de salto, para mejorar el rendimiento del corredor durante la competencia con otros jugadores.

Por otro lado, en [31] se propone un juego de estrategia, formado por agentes que basan su comportamiento en las formas de forrajeo y defensivas de las colonias de abejas, para adaptarse a un entorno humano. De esta forma, el juego consta de múltiples agentes cooperativos autónomos. En [73], Barajas y otros proponen un procedimiento cuyo objetivo es garantizar la correcta implementación de los aspectos pedagógicos (actividades y contenidos de aprendizaje, aprendizajes esperados, etc.) en la producción de juegos serios. Una vez aplicado el proceso de producción, midieron la calidad del prototipo aplicando el juego serio en un salón de clases.



Solo cuatro de los artículos anteriores analizados proponen o se centran en el desarrollo de juegos serios ([68, 69, 73] y nuestra propuesta), pero no están orientados a los JSEs. Por otro lado, varios trabajos proponen mecanismos para la adaptación de un juego ([31, 32, 33, 68, 71, 72] y nuestra propuesta), pero solo dos para juegos serios ([68] y nuestra propuesta). A su vez, algunos de los mecanismos de adaptación de los juegos se pueden catalogar como un tipo de emergencia que permiten en los juegos ([31, 72] y nuestra propuesta). Finalmente, solo dos trabajos están vinculados al diseño de motores juegos serios, pero el nuestro es el único para MJSE.

En resumen, algunos de los trabajos anteriores confieren capacidades adaptativas, pero no están orientados a formar parte de un MJSE. Por otro lado, algunos permiten la emergencia de conductas en los juegos, permitiendo principalmente emergencias de estrategias y propiedades, pero no permiten la emergencia de secuencias en los juegos guiadas por el contexto, como en nuestro caso. Además, no hay ninguno orientado a JSE, y pocos orientados a juegos serios. Finalmente, la mayoría no tiene capacidades de adaptación ni es parte de un MJSE.

Así, la principal diferencia entre nuestra propuesta y trabajos anteriores es que nuestra propuesta es parte de un MJSE. Además, es el único que permite el surgimiento de tramas/secuencias en un JSE, según las características del contexto donde será utilizado. Para ello utiliza un algoritmo ACO que le permite adaptar dinámicamente un JSE.

**Comparación del SAP con otros trabajos:**

La tabla 4.23 presenta una comparación de nuestro enfoque con otros trabajos, de acuerdo con los siguientes criterios:

a. ¿Permite la adaptación de los parámetros de un juego?
b. ¿Es un *Juego Serio*?
c. ¿Permite algún tipo de *emergencia*?
d. ¿Es parte de un motor de juego?
e. ¿Utiliza un *Algoritmo Cultural*?

**Tabla 4.23.** Comparación con otras obras

| Criterios | Aguilar y otros [7] | Figueira y otros [30] | Rasim y otros [68] | Waris y otros [36] | Singh y otros [74] | Yang [37] | Reynolds y Kinnaird - Heether [75] | Jamieson [72] | Khodabakhshian y Hemmati [76] | Abdolrazzagh-Nezhad y otros[77] | Presente trabajo |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **a** | X | X | | | X | X | | | | | X |
| **b** | | | X | | | | | | | | X |
| **c** | X | X | X | X | X | X | X | X | X | X | X |
| **d** | | X | X | | | | | | | | X |
| **e** | | | | X | | | X | | X | X | X |



Aguilar y otros [7] describen la adaptación del *juego emergente* "Metropolis" a través de la adaptación del radio, el cual define la influencia de un edificio con respecto a otro, y se modifica en tiempo real para facilitar la emergencia urbana, de modo que los edificios preferidos por los jugadores tengan un radio corto. Figueira y otros [30] presentan un marco para adaptar los parámetros del juego de acuerdo con las habilidades del jugador actual, utilizando diferentes modelos de adaptación basados en perfiles de jugador. Rasim y otros [68] proponen un motor adaptativo basado en reglas preestablecidas para *juegos serios* utilizando técnicas de inteligencia artificial, pero no existe ningún tipo de surgimiento en el juego. Waris y otros [36] definen un enfoque para la gestión de redes sociales basado en *algoritmos culturales,* para la distribución del conocimiento. Permite el surgimiento de comunidades de conocimiento utilizando mecanismos de distribución que favorecen la cooperación y competencia entre los participantes.

Singh y otros [74] adaptan un juego de dos niveles usando una arquitectura multinivel: en el primer nivel los jugadores interactúan; y los jugadores de segundo nivel se dividen según ciertas reglas. La adaptación se define en términos del número de participantes en un juego. Yang [37] propone la generación de nuevos jugadores artificiales utilizando operadores de aprendizaje e imitación. Un operador de aprendizaje ajusta comportamientos y estrategias

mediante el estudio de información histórica. Un operador de imitación reproduce las estrategias y acciones de otros jugadores para mejorar el rendimiento de un jugador.

Reynolds y Kinnaird - Heether definen un método de distribución del conocimiento en [75]. Jamieson y otros [72] describen juegos que tienen problemas de optimización combinatoria en su dinámica: el primer juego incorpora aspectos de recolección de recursos en sus estrategias. El segundo juego utiliza información de los personajes (son corredores), como velocidad, agilidad y poder de salto, para encontrar las mejores soluciones a diferentes problemas del juego (por ejemplo, la ruta a seguir). Khodabakhshian y Hemmati [76] ajustan los parámetros de la metaheurística utilizada en el juego (la *potencia* técnica *de estabilización del sistema),* usando un algoritmo cultural. Abdolrazzagh-Nezhad y otros [77] modelan múltiples objetivos usando la teoría de conjuntos aproximados y un algoritmo cultural.

Los trabajos [7, 30, 37, 68, 76] utilizan una técnica para adaptar los parámetros al nivel del jugador. Los trabajos basados en algoritmos culturales permiten la adaptación de los juegos al entorno [36, 74, 21, 75, 77] pero no son juegos emergentes. No existen trabajos que describan motores para JSEs [68], y los juegos emergentes desarrollados hasta ahora con la capacidad de emerger el valor de los parámetros son exclusivamente para el juego específico donde se utilizan [7]. Nuestra propuesta es parte de un motor JSE y permite una fuerte aparición de parámetros en cualquier JSE de acuerdo con el entorno dado (como el SaCI).



**<u>Comparación de SAE con otros trabajos</u>:**

La comparación que se propone realizar es basada en si permiten emergencias de estrategias en el JSE, si usan la lógica difusa, qué técnica difusa usan, y cómo las usan en el juego (ver tabla 4.24).

**Tabla 4.24.** Comparación con otros trabajos recientes

| Técnica Aplicada | Rasim y otros [68] | Saeed y otros [78] | Camci y otros [79] | Esfahlan y otros [38] | Pan y otros [39] | Vasudeva y otros[80] | Subbaraj y otros [70] | Presente Trabajo |
|---|---|---|---|---|---|---|---|---|
| Emergen estrategias | | | | X | X | | | X |
| Capacidad Adaptativa del Juego | X | | | X | X | | | X |
| Usan Lógica Difusa | | X | X | X | X | X | X | X |
| Para qué | Motor Adaptativo | Simulación Vehicular | Control Cuadracopteros | Estrategias de Rehabilitación Física | Toma de Decisión | Toma de Decisiones | Certificado de Autoridad y Confianza | MJSE |
| Técnica difusa | | Control Difuso | TSK | Mamdani's | FTS | Teoría de Juegos Difusa | Reglas | SCD |

El desarrollo de juegos serios en [68] involucra un motor adaptable, que mediante la aplicación de una máquina de aprendizaje puede adaptarse a diferentes jugadores. En [78] aplican la teoría de juegos y control basado en lógica difusa para la simulación vehicular, que

debe librar obstáculos, derrumbes o deslizamiento de tierra, roca o lodo, debido a diferentes razones: deforestación, terremotos, erupciones volcánicas, fuertes lluvias, entre otros. En [79] se utiliza el modelo difuso Takagi-Sugeno-Kang (TSK) para controlar cuadricopteros.

En [38] proponen un juego serio para la rehabilitación física basado en un Kinect de Xbox One, que permite el seguimiento de articulaciones en un pedal del timón. El sistema usa reglas difusas del tipo Mamdani's que definen las estrategias de rehabilitación, las cuales se van adecuando de acuerdo al jugador, en función del rendimiento de la persona en la terapia y su nivel de habilidad física. En [39] se describen videojuegos con incertidumbre estocástica, donde se utiliza un 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 [80] se diseñan reglas de inferencia difusas para las decisiones de transferencia y la selección de la estación base destino en redes HeTNets, teniendo en cuenta la eficiencia energética/espectral. El sistema de inferencia utiliza la teoría de juegos basado en lógica difusa para abordar este problema. En [70] se propone un modelo de juego no cooperativo para el ruteo seguro y tolerante a fallos en redes móviles ad hoc (MANET) basado en ACO. También utiliza un enfoque integrado de confianza y certificado de autoridad basado en lógica difusa, para el intercambio seguro de datos, aislando los nodos dañinos de la comunicación.



En comparación con los anteriores trabajos, en el presente trabajo se propone un sistema para la gestión de la emergencia de estrategia, usando un SCD, para JSE. Es el único trabajo que propone incorporar la emergencia de estrategias en MJSE, como mecanismo adaptativo de los JSEs. La emergencia permitida en otros trabajos, es vinculada a tomas de decisiones durante el desarrollo del juego, lo cual no impacta la propia dinámica del sistema (en nuestro caso, el JSE).

# CAPÍTULO V: CONCLUSIONES Y TRABAJOS FUTUROS

## 5.1. CONCLUSIONES

Los JSEs, a diferencia de otros juegos, tienen el potencial de adaptarse a un contexto narrativo emergente, como un proceso de enseñanza-aprendizaje en un aula inteligente. En particular, adaptan su comportamiento al contexto donde serán utilizados. Para ello, permiten la aparición de determinados comportamientos/propiedades en el juego. El uso de JSEs adaptados a un contexto del SaCI, permite que sea más dinámico y entretenido el proceso de aprendizaje en los estudiantes. Para lograr esto, se requiere un MJSE que permita diferentes tipos de emergencias.

En esta tesis presentamos el componente SAV para un MJSE, el cual permite la emergencia fuerte, de tal manera de adaptar un JSE al entorno. Tres componentes del SAV fueron probados en un SaCI, de tal manera de permitir al JSE adecuarse al proceso de enseñanza-aprendizaje que se esté dando en un momento dado en el SaCI. Así, las opciones de adaptación que posibilita el SAV sobre un JSE en un SaCI están dirigidas a tres tipos de emergencia fuerte: la *emergencia de secuencia*, que permite la aparición de nuevos escenarios durante el juego; la *emergencia de propiedades*, de tal manera de adecuar los parámetros del JSE al contexto estudiantil del SaCI en un momento dado; y finalmente, la *emergencia de estrategias*, en el sentido de posibilitar nuevas formas, reglas, etc., de comportamiento en el JSE. A continuación, profundizaremos los logros alcanzados en estos tres tipos de emergencia.



Para la *emergencia de secuencia* se realizó un SAS basado en un algoritmo ACO. Particularmente, se implementó un mecanismo que permite la aparición de secuencias/tramas en un JSE. El objetivo del SAS es posibilitar que emerja dinámicamente un JSE óptimo para un tema determinado utilizando una serie de juegos serios. Para ello, el algoritmo ACO permite la aparición de tramas en un JSE a partir del manejo ordenado y sistemático de un conjunto de nuevas subtramas vinculadas a un contexto y dominio deseado, para fusionarse con el JSE actual. El algoritmo ACO selecciona de forma autónoma las mejores subtramas que se utilizarán en el JSE actualizado

Se realizaron experimentos para probar la calidad del SAS para hacer surgir un juego serio para un tema deseado. Los experimentos analizaron la sensibilidad del SAS con respecto al método de similitud, el número de subtramas y el umbral de similitud utilizados. De acuerdo con los resultados de los experimentos, es posible hacer emerger un JSE que siga el tema deseado, teniendo un alto impacto en la calidad de la adaptación del JSE. Con respecto a las limitaciones del SAS, los ROA deben estar conectados al MJSE para poder buscar en ellos los potenciales juegos serios a usar en el proceso de diseño del JSE que desarrolla el algoritmo ACO. Además, otra limitación es su escalabilidad. Aquí se probó para la adaptación de un JSE tomando los Juegos Serios desde un ROA, pero se debe probar utilizando varios ROA. Ahora bien, no existe un trabajo anterior similar al nuestro, el cual tiene dos características

principales: una es ser un componente de un MJSE, y la otra que permite la emergencia de tramas/secuencias en un JSE en función del contexto (tema deseado).

En cuanto a la *emergencia de propiedades*, en este trabajo se desarrolló un SAP que determina los parámetros óptimos de un JSE, para adaptarse a las características de los jugadores. Las métricas utilizadas evalúan si la adecuación de los JSE permite al jugador alcanzar los objetivos del JSE en un tiempo corto. Específicamente, se propuso utilizar un *algoritmo cultural* para la determinación de los valores óptimos de los parámetros del JSE, utilizando diferentes tipos de conocimiento sobre los JSE: situacional, normativo, de dominio, e histórico. En los experimentos solo se consideraron los tipos de conocimiento situacional y normativo. La razón es que permiten caracterizar los comportamientos de los usuarios, con el fin de adaptar los JSEs para seguirlos. Los otros tipos de conocimientos definen principalmente información general sobre el contexto.

El protocolo experimental presenta un análisis de sensibilidad de los parámetros del *algoritmo cultural*, para determinar los valores por defecto de las variables del SAP permitiendo optimizar los parámetros del JSE. Particularmente, el JSE adapta sus parámetros al contexto pedagógico en un SaCI, utilizando el componente SAP del MJSE. En concreto, el SAP define los valores óptimos de los parámetros del JSE, tal que permite que los estudiantes en el SaCI puedan lograr los objetivos de aprendizaje en un tiempo corto.



En cuanto a la *emergencia de estrategias*, se desarrolló el componente SAE en el MJSE que permite el surgimiento de nuevas reglas, tácticas, etc., para adaptar la lógica de un JSE al contexto actual en un SaCI. Específicamente, en este trabajo, las estrategias se definieron en forma de reglas. Para el proceso adaptativo se utilizó un SCD, que gestiona las estrategias según sus usos durante el juego, con el fin de crear nuevas reglas y hacer desaparecer las que no sirven. El protocolo experimental desarrollado en esta tesis fue aplicado en el videojuego Súper Mario Bros, tal que el SCD actualizo las reglas del juego en cuanto a las combinaciones de movimientos y poderes permitidos en el avatar Mario o Luigi.

En conclusión, el SAV permite tres tipos de emergencias de forma coherente, seleccionando las mejores tramas con el SAS en función del tema que se explicando en el SaCI, estableciendo los parámetros del JSE por intermedio de SAP según el nivel los estudiantes que se encuentran en el aula y por último, seleccionando con el SAE las mejores reglas a usar en el JSE según los objetivos en el SaCI. De esta forma, estos mecanismos pueden trabajar en conjunto y de forma sincronizada dentro del SAV en el MJSE, con el fin de permitir que el JSE se adapte al proceso de enseñanza-aprendizaje que se está dando en un momento dado con el tema en el SaCI.

## 5.2. TRABAJOS FUTUROS

Como trabajos futuros, se requieren hacer pruebas del MJSE en diferentes procesos de enseñanza-aprendizaje, y evaluar su impacto en los mismos usando indicadores pedagógicos que permitan mostrar su efectividad en dichos procesos. Otro trabajo futuro es probar la integración de todos estos tipos de emergencia en un único ambiente. También, hay trabajos futuros propios a cada una de los tipos de emergencia, que se explican a continuación:

1. Emergencia de Secuencias: como trabajo futuro, el prototipo ACO se debe probar en diferentes contextos, en particular, en un SaCI real, para evaluar su integración con el entorno, y en particular, capturar la información del tema deseado de manera autónoma. Otro aspecto importante a realizar es evaluar la sensibilidad de los parámetros del macroalgoritmo ACO.

2. Emergencia de Propiedades: los próximos trabajos deben analizar cómo el SAP puede explotar el conocimiento del dominio, histórico o topográfico, para adaptar el JSE. Por otro lado, en nuestro trabajo se optimiza el JSE tal que los estudiantes puedan jugar adecuadamente el JSE. Ahora bien, se debe hacer una investigación en un contexto donde los estudiantes deban ir subiendo de niveles en el JSE, haciéndolo cada vez más complicado, en vez de optimizarlo. Eso implica definir ese conocimiento en el contexto de un SaCI.



3. Emergencia de Estrategias: como trabajos futuros, se debe ampliar el desarrollo de reglas generales para otros géneros de videojuegos, como son: juegos de rol, de deportes, vehículos, rompecabezas, entre otros.

# BIBLIOGRÁFIA

[1] J. Aguilar, J. Altamiranda, F. Díaz and D. Mosquera, “Motor de Juego Serios en ARMAGAeco-c,” Revista Científica UNET, Vol. 28, Nº 2, pp. 100-110, 2016.

[2] P. Petridis, I. Dunwell, D. Panzoli, S. Arnab, A. Protopsaltis, M. Hendrix and S. de Freitas, “Game engines selection framework for high-fidelity serious applications,” International Journal of Interactive Worlds, Vol. 2012, pp. 418638, 2012.

[3] J. Aguilar, J. Cardozo, C. González and B. Rengifo, “Una aproximación a los Juegos Emergentes, Metrópolis, Simulador de Ciudades Autogestionada,” Proceeding XXXXVII Conferencia Latinoamericana de Informática, pp. 38- 46, 2013.

[4] L. Lima, D. Torres and E. Ramírez “Un Juego Serio para la Preservación de la Fauna Silvestre en Peligro de Extinción en Venezuela,” Proceedings Tercera Conferencia Nacional de Computación, Informática y Sistemas (CoNCISa), pp. 50-60, 2015.

[5] J. Steven, “Sistemas Emergentes: O qué tienen en común hormigas, neuronas, ciudades y software,” Ediciones Turner/Fondo de Cultura Económica, Madrid, España, 2004.

[6] R. Brandão and A. Neves, “Design da Ludonarrativa: princípios da narratología aplicados ao game design para concepção de mecánicas,” Proceedings XIII SBGames, pp. 112-119, 2014.

[7] J. Aguilar, J. Altamiranda, F. Díaz, J. Cordero, D. Chávez and J. Gutiérrez de Mesa, “Metropolis: an Emerging Serious Game for the Smart City,” DYNA, Vol. 86, pp. 215-224, 2019.

[8] J. Aguilar, J. Altamiranda, F. Díaz, J. Cordero, D. Chávez and J. Gutiérrez de Mesa, “Metropolis: Emergence in a Serious Game to Enhance the Participation in Smart City Urban Planning,” Journal of the Knowledge Economy, Vol. 12, pp. 1594–1617, 2021.
DOI: https://doi.org/10.1007/s13132-020-00679-5

[9] P. Valdiviezo, J. Cordero, J. Aguilar and M. Sanchez, “Conceptual Design of a Smart Classroom Based on Multiagent Systems,” International Journal of Advanced Information Science and Technology (IJAIST), Vol.39, Nº 39, pp. 471-477, 2015.

[10] J. Aguilar, L. Chamba-Eras and J. Cordero, “Specification of a Smart Classroom Based on Agent Communities,” Advances in Intelligent Systems and Computing (AISC), Vol. 444, pp. 1003-1012, 2016.

[11] J. Hernández, E. Benítez and C. Mezura, “Ambientes Inteligentes en Contextos Educativos: Modelo y Arquitectura,” Research in Computing Science 77. pp. 55–65, 2014.

[12] M. Sánchez, J. Aguilar, P. Valdiviezo and J. Cordero, “A Smart Learning Environment based on Cloud Learning,” International Journal of Advanced Information Science and Technology (IJAIST), Vol .39, Nº 39, pp. 39–52, 2015.

[13] C. James, Y. Eun, Y. Seung, W. Bradford, P. Jonathan and L. Jennifer, “Serious Games Get Smart: Intelligent Game-Based Learning Environments,” Association for the Advancement of Artificial Intelligence, pp. 31–49, 2014.

[14] D. Clark, E. Tanner-Smith and S. Killingsworth, “Digital Games, Design, and Learning: A Systematic Review and Meta-Analysis,” Review of Educational Research, Vol. 86, Nº 1, pp. 79 – 122, 2016.

[15] J. Fox, L. Pittaway and I. Uzuegbunam “Simulations in Entrepreneurship Education: Serious Games and Learning Through Play,” First Published: Entrepreneurship Education and Pedagogy, Vol. 1, Nº 1, pp. 61–89, 2018.

[16] G. Ben-Sadoun, V. Manera, J. Alvarez, G. Sacco and P. Robert, “Recommendations for the Design of Serious Games in Neurodegenerative Diseases,” Frontiers in Aging Neuroscience, Vol. 10, Nº 13, pp. 1-7, 2018.

[17] Y. El Borji and M. Khaldi, “An IEEE LOM Application Profile to Describe Serious Games «SG-LOM»,” International Journal of Computer Applications, Vol. 86, Nº 13, pp. 1–8, 2014.

[18] A. Alí, A. Shoten, S. Göbel and A. Amrich, “Playful Interactions and Serious Games,” Journal of Ambient Intelligence and Smart Environments, IOS Press, Vol. 1, pp. 1-5, 2014.

[19] D. Fernández and J. Hamari, “Game-based Climate Change Engagement: Analyzing the Potential of Entertainment and Serious Games,” Proceedings of the ACM on Human-Computer Interaction, Vol. 5, Nº 226, pp. 1–2, 2021. DOI: https://doi.org/10.1145/3474653

[20] R. Chan, H. Zhang and X. Tao, "Serious Game Design for Stroke Rehabilitation," Intl Journal of Information Technology, Vol. 23, 2017.

[21] A.C. Alves, T.V. Ficagna and A.G. Alves, “Estudo de Caso no Desenvolvimento da Inteligência Artificial do Jogo Bonefighters,” Anais do IX Computer on the Beach, pp. 412-421, 2018.

[22] K. Shafi and H. Abbass, “A Survey of Learning Classifier Systems in Games,” IEEE Computational intelligence magazine, pp. 42–55, 2017.

[23] B. Gracia, M. González, G. Sanagustín and S. Romero, “Análisis Motores gráficos y su aplicación en la industria,” TecsMedia, Publicaciones electrónicas del Gobierno de Aragón (ITAINNOVA), pp. 1-16, 2015.

[24] H. Agustín and M. Garde, “Desarrollo de un Motor de Eventos para Videojuegos,” Red de Interconexión de los Recursos Informáticos (RedIRIS), pp. 1-36, 2016.

[25] K. H. Sharif and S. Yousif Ameen, "Game Engines Evaluation for Serious Game Development in Education," Proceedings of the 2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1-6, 2021, DOI: 10.23919/SoftCOM52868.2021.9559053.

[26] M. Filho, R. Brandão and A. Neves, “Narrativa Centrada no Jogador: Uma Análise da Relação entre a Narrativa Emergente e as Mecânicas nos Jogos Digitais,” SBC – Proceedings SBGames, pp. 175-184, 2014.

[27] M. Bigogno, V. Réda and M. La Carreta, “Dissonância Ludonarrativa X Suspensão da Descrença: quando o gameplay desmente a narrativa ou quando o jogador apenas a aceita,” Proceedings of SBGames, pp. 1068-1071, 2017.

[28] S. Chauvin, G. Levieux, J. Donnart and S. Natkin. "An Out-of-Character Approach to Emergent Game Narratives," Proceedings of the 9th International Conference on the Foundations of Digital Games, pp. 1-4, 2014.

[29] M.P. Recke and S. Perna, "An Emergent Narrative System to Design Conducive Educational Experiences," Universities and Entrepreneurship: Meeting the Educational and Social Challenges (Contemporary Issues in Entrepreneurship Research), Vol. 11, pp. 185-198, 2021. DOI: https://doi.org/10.1108/S2040-724620210000011012

[30] F. M. Figueira, L. Nascimento, J. da Silva Junior, T. Kohwalter, L. Murta and E. Clua, "BinG: A Framework for Dynamic Game Balancing using Provenance,". Proceedings of 2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), pp. 408-417, 2018. DOI: 10.1109/SBGAMES.2018.00016

[31] D. Daylamani-Zad, L. B. Graham and I. Paraskevopoulos, “Th. Chain of command in autonomous cooperative agents for battles in real-time strategy games,” Proceedings of Journal of Computers in Education, pp 1–32, 2018.

[32] E. D’Amato, E. Daniele and L. Mallozzi, “Designing Networks in Cooperation with ACO,” Proceedings of Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences. Computational Methods in Applied Sciences, Vol. 48, pp. 255-267, 2019.

[33] T. Tregel, P. Müller, S. Göbel and R. Steinmetz, “Where's Pikachu: Route Optimization in Location-Based Games,” Proceedings of 10th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games), 2018.

[34] N. A. Hasanah, L. Atikah, D. Herumurti and A. Yunanto, "A Comparative Study: Ant Colony Optimization Algorithm and Backtracking Algorithm for Sudoku Game," 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), pp. 548-553, 2020. DOI: 10.1109/iSemantic50169.2020.9234267

[35] M. Khalid, F. Al-Obeidat, A. Tubaishat, B. Shah, S. Razzaq, F. Maqbool and M. Ilyas, “An Assortment of Evolutionary Computation Techniques (AECT) in gaming,” Neural Comput & Applic, Vol. 13, Nº 3, pp. 1-14, 2020. DOI: https://doi.org/10.1007/s00521-020-05295-7

[36] F. Waris and R. Reynolds, “Optimizing AI Pipelines: A Game-Theoretic Cultural Algorithms Approach,” Proceedings EEE Congress on Evolutionary Computation (CEC) pp. 1-10, 2018.

[37] G. Yang, “Game Theory-Inspired Evolutionary Algorithm for Global Optimization,” Algorithms, Vol. 10, pp. 1-15, 2017.

[38] H. Esfahlan, S. Cirstea, A. Sanaei and G. Wilson, “An adaptive self-organizing fuzzy logic controller in serious game for motor impairment rehabilitation,” Proceeding IEEE Int. Symp. Ind. Electron, pp. 1311-1318, 2017.

[39] H. Pan, Y. Li, Y. Cao and D. Li, “Reachability in Fuzzy Game Graphs,” Proceedings IEEE Transactions on Fuzzy Systems. Vol. 25, Nº 4, pp. 984-972, 2017.

[40] L. Chamba and J. Aguilar, “Design of an Augmented Reality Component from the Theory of Agents for Smart Classrooms,” IEEE Latinoamerica Transactions, Vol.14, Nº 8, pp. 3826-3838, 2016.

[41] F. Díaz, J. Aguilar and J. Altamiranda, “Specification of a Managing Agent of Emergent Serious Games for a Smart Classroom,” IEEE Latin America Transactions, Vol. 8, Nº 1, pp. 51-58, 2020.

[42] F. Díaz, J. Aguilar and J. Altamiranda, “Sistema Adaptativo para la Generación de Comportamientos Emergentes en Juegos Serios Emergentes”, Computación y Sistemas, Vol. 24, No.3, pp. 1029–1051, México, 2020.

[43] F. Díaz, J. Aguilar, J. Altamiranda, N. P. Garcia and A. Pinto, “An Adaptive System for Emerging Serious Games using a Swarm Intelligence Algorithm,” IEEE Transactions on Games, vol. 14, no. 4, pp. 598-609, EEUU, Dec. 2022.

[44] F. Díaz, J. Aguilar, J. Altamiranda and E. Montoya, “An Emerging Serious Game Engine with a Parameter Adaptive System based on an evolutionary approach”. Proceeding XLVI Latin American Computer Conference (CLEI 2020), pp. 39-47, Loja, Ecuador, 2020.

[45] F. Díaz, J. Aguilar and J. Altamiranda, “Adaptive strategy system for serious emerging games using a diffuse classifier system,” Revista Ciencia e Ingeniería. Facultad de Ingeniería de la Universidad de los Andes, Vol. 42-3, N° 3, pp. 285-296. Mérida, Venezuela, 2021.

[46] M. Sánchez, “Buenas Prácticas en la Creación de Serious Games (Objetos de Aprendizaje Reutilizables),” Universidad de Málaga, Facultad de Ciencias de la Comunicación. Campus de Teatinos. pp. 1-9, España, 2015.

[47] F. J. Álvarez, A. Barajas and J. Muñoz, “Serious Game Design Process, Study Case: Sixth Grade Math,” Creative Education, Vol. 5, pp. 647-656, 2014.

[48] D. Vallejo and C. Martín, “Desarrollo de Videojuegos 1: Arquitectura del Motor de Videojuegos (2ª Edición),” Universidad de Castilla la Mancha, España, 2013.

[49] J. Aguilar, “Introducción a los Sistemas Emergentes,” Talleres Gráficos, Universidad de Los Andes, Mérida, Venezuela, 2014.

[50] T. Wolf and T. Holvoet, “Emergence and Self-Organisation: a statement of similarities and differences,” Lecture Notes in Artificial Intelligence, Vol. 3484, pp. 96-110, 2004.

[51] RAE, “La Real Academia de Ingeniería presenta un diccionario técnico que incorpora 50.000 entradas de 1.500 glosarios,” Europa Press, Madrid, España, 2014.

[52] J. García, “Teoría y ejercicios prácticos de Dinámica de Sistemas,” Barcelona, España, Edición, 2019.

[53] J. Aguilar, M. Mendonça and N. Perozo, “An Emergent Ontology for Ambient Intelligence based on an Ant Colony Optimization algorithm,” XL Latin American Computing Conference (CLEI), pp. 1 – 11, 2014.

[54] J. Aguilar, J. Altamiranda, F. Díaz, J. Gutiérrez and A. Pinto, "Sistema Adaptativo de Tramas para Juegos Serios Emergentes basado en el Algoritmo de Optimización de Colonia de Hormigas,” Proceeding XLV Latín American Computer Conference (CLEI 2019), Panamá, 2019.

[55] J. Aguilar, J. Altamiranda and F. Díaz, “Design of a Serious Emerging Games Engine Based on the optimization Algorithm of Ant Colony,” DYNA, Vol. 85, Nº 206, pp. 311-320, Medellin, Colombia, 2018.

[56] J.Terán, J. Aguilar and M. Cerrada “Cultural Learning for Multi-Agent System and Its Application to Fault Management,” Proceedings IEEE World Congress on Computational Intelligence (IEEE WCCI), pp. 2188 – 2195. 2014.

[57] J. Aguilar, Y. Menolascina and F. Rivas, “Compiler Design for Fuzzy Classifier Systems,” WSEAS Transactions on Systems, Vol. 4, Nº 4, pp. 262-267, 2005.

[58] J. Aguilar and M. Cerrada, “Un Sistema Clasificador Difuso para el Manejo de Fallas,” Revista Técnica de la Facultad de Ingeniería, Universidad del Zulia, Vol. 23, Nº 2, pp. 98-108, 2000.

[59] F. Abad, J. Zaldo, H. Villares, M. Walter and A. Yuncal, “Métodos y técnicas de inteligencia artificial: ¿cuáles son y para qué se usan?,” Revista APD Septiembre-Octubre, Hr Revolution Nº 359 Zurbano 90, 28003 Madrid, España, 2021.

[60] J. Aguilar, M. Mendonça, M. Jerez and M. Sánchez, “Emergencia ontológica basada en análisis de contexto, como servicio para ambientes inteligentes,” DYNA, Vol. 84, Nº 200, pp. 28-37, 2017.

[61] R. Mona, D. Quentin and Y. Joseph, “Implementation of a Fuzzy Rule-Based Classifier using Genetic Algorithms,” Proceedings Github, pp. 1-12. 2017. Available article online: https://github.com/Apoptoz/FL_Classifier/blob/master/Article/article.pdf

[62] ABC Sociedad “Aulas a lo Star Trek: la clase del futuro,” ABC, 2012. [online]. Available: https://www.abc.es/sociedad/abci-aulas-star-trek-201211230000_noticia.html [Último acceso: 2019].

[63] M. Sanchez, J. Aguilar, J. Cordero and P. Valdiviezo, “Basic features of a Reflective Middleware for Intelligent Learning Environment in the Cloud (IECL),” Proceeding Asia-Pacific. Conference on Computer Aided System Engineering, 2015.

[64] A. Trejo, J. Aguilar, F. Díaz, “Implementación del Algoritmo de Colonia de Hormigas para un Motor de Juegos Serios Emergentes,” Proyecto de Grado, Universidad de los Andes Mérida, Venezuela, 2019.

[65] J. Lu, W. Wang, Ch. Li and H. Wang, "String similarity measures and joins with synonyms," Proceedings ACM SIGMOD International Conference on Management of Data, pp. 373–384, 2013.

[66] J. Aguilar, F. Díaz, J. Gutiérrez, J. Altamiranda, N. Pérez and Á Pinto, “Parameter Adaptive System to Learning Processes for Emerging Serious Games using Cultural Algorithms”, En revisión.

[67] TM Nintendo, “Manual de Instrucciones,” Sharp Corporation. pp 1-19, 2006.

[68] T. Rasim, A. Langil, S. Munir and Y. Rosmansyah, “A survey on adaptive engine technology for serious games,” Proceedings of International Seminar on Mathematics, Science, and Computer Science Education, Vol. 1708, Nº 1, 2016. DOI: https://doi.org/10.1063/1.4941161

[69] J.L.E. Martínez, A.S. López and M. A. J. Maldonado, “On the Use of Ant Colony Optimization for Videogames,” Proceedings of Advances in Artificial Intelligence and Soft Computing. Vol. 9413, pp. 238-247, 2015.

[70] S. Subbaraj and P. J. Savarimuthu, “EigenTrust-based non-cooperative game model assisting ACO look-ahead secure routing against selfishness,” Journal on Wireless Communications and Networking, pp. 1-20, 2014. DOI: 10.1186/1687-1499-2014-78.

[71] X. Chen, Y. S. Ong, L. Feng, M. H. Lim, C. Chen and C. S. Ho, “Towards believable resource gathering behaviours in real-time strategy games with a memetic ant colony system,” Procedia Computer Science, Vol. 24, pp. 143–151, 2013. DOI: https://doi.org/10.1016/j.procs.2013.10.037

[72] P. Jamieson, J. Grace, J. Hall and A. Wibowo, “Metaheuristic Entry Points for Harnessing Human Computation in Mainstream Games,” Proceedings International Conference on Online Communities and Social Computing, Vol. 8029, pp. 156–163, 2013.

[73] A. Barajas, F. J. Álvarez, J. Muñoz, and A. C., A. C. Oviedo, “Process for modeling competencies for developing serious games,” Journal of Computers in Education, Vol. 18, Nº 3, pp. 146-160, 2016.

[74] D. Singh, P. Moradian and Z. Kobti, “A Multilevel Cooperative Multi-Population Cultural Algorithm,” 2018 Intelligent Systems and Applications (INISTA), 2018.

[75] R. Reynolds and L. Kinnaird - Heether, "Population Mechanics and Cultural Algorithms in the Development of a Cultural Engine," Proceedings 017 IEEE Symposium Series on Computational Intelligence (SSCI), 2017.

[76] A. Khodabakhshian and R. Hemmati, "Multi-machine power system stabilizer design by using cultural algorithms," International Journal of Electrical Power & Energy Systems, Vol. 44, Nº 1, pp. 571-580, 2013.

[77] M. Abdolrazzagh-Nezhad, H. Radgohar and S. Najme "Enhanced cultural algorithm to solve multi-objective attribute reduction based on rough set theory," Mathematics and Computers in Simulation, Vol. 170, pp. 332-350, 2020.

[78] Y. Saeed, L. S Aziz and K. Ahmed, "Obstacle Management in VANET using Game Theory and Fuzzy Logic Control," Proceedings International Journal of Scientific & Engineering Research, Vol 4. N° 1, pp. 9-13. 2013.

[79] E. Camci and E. Kayacan, "Game of Drones: UAV Pursuit-Evasion Game with Type-2 Fuzzy Logic Controllers Tuned by Reinforcement Learning," Proceeding IEEE International Conference on Fuzzy Systems, pp. 618-625, 2016.

[80] K. Vasudeva, S. Dikmese, S. Güvenç, A. Mehbodniya, W. Saad and F. Adachi, "Fuzzy-Based Game Theoretic Mobility Management for Energy Efficient Operation in HetNets," IEEE Access, Special Section on Future Networks: Architectures, Protocols and Applications, Vol. 5, pp. 7542-7552, 2017.

[81] J. Aguilar, F. Díaz and J. Altamiranda, "Strategy Adaptive System to Learning Processes for Emerging Serious Games using a Fuzzy Classifier System", En revisión.

# APENDICE

## APENDICE A: ARTÍCULOS PUBLICADOS EN EL MARCO DE LA TESIS

Los siguientes artículos han sido publicados mientras se desarrolló la tesis:

1. F. Díaz, J. Aguilar, J. Altamiranda, N. P. Garcia and A. Pinto, "An Adaptive System for Emerging Serious Games using a Swarm Intelligence Algorithm," IEEE Transactions on Games, vol. 14, no. 4, pp. 598-609, EEUU, Dec. 2022

   Disponible en línea:

   a. DOI: https://doi.org/10.1109/TG.2021.3118273

   b. https://www.dropbox.com/home/Doctorado/SAVCsecuencia/Ingles?preview=SASDef.pdf

2. F. Díaz, J. Aguilar and J. Altamiranda, "Adaptive strategy system for serious emerging games using a diffuse classifier system," Revista Ciencia e Ingeniería. Facultad de Ingeniería de la Universidad de los Andes, Vol. 42-3, N° 3, pp. 285-296. Mérida, Venezuela, 2021.



   Disponible en línea:

   a. http://190.168.5.19/index.php/cienciaeingenieria/article/view/17271

3. F. Díaz, J. Aguilar, J. Altamiranda, J. Cordero, D. Chavez and J. Gutierrez, "Metropolis: Emergence in a Serious Game to Enhance the Participation in Smart City Urban Planning," Journal of the Knowledge Economy, Springer Science+Business Media, LLC, part of Springer Nature, Suiza, Vol. 12, pp. 1594–1617, 2021.

   Disponible en línea:

   a. DOI: https://doi.org/10.1007/s13132-020-00679-5

   b. https://link.springer.com/epdf/10.1007/s13132-020-00679-5?sharing_token=ev3tmYaR4II7KhrL-eo7eve4RwlQNchNByi7wbcMAY6a3A2Ob2_Md832muOnfY_CGC9l5BURD3hCJFTF1fKgtowP7yRlgXswVcf8qAupFuleSf1KKKewMf3xXyffX2TR3b0mUs8-89_584WsmAau8W7GyRhShR-b9G9kAWCmOd8%3D

4. F. Díaz, J. Aguilar and J. Altamiranda, “Specification of a Managing Agent of Emergent Serious Games for a Smart Classroom,” IEEE Latin America Transactions, Vol. 18, Nº 1, pp. 51- 58, México, 2020.

   Disponible en línea:

   a. https://latamt.ieeer9.org/index.php/transactions/article/view/465/370

5. F. Díaz, J. Aguilar, J. Altamiranda and E. Montoya, “An Emerging Serious Game Engine with a Parameter Adaptive System based on an evolutionary approach”. Proceeding XLVI Latín American Computer Conference (CLEI 2020), pp. 39-47, Loja, Ecuador, 2020.

   Disponible en línea:

   a. http://bit.do/fReBr

6. F. Díaz, J. Aguilar and J. Altamiranda, “Sistema Adaptativo para la Generación de Comportamientos Emergentes en Juegos Serios Emergentes”, Computación y Sistemas, Vol. 24, No.3, pp. 1029–1051, México, 2020.

   Disponible en línea:



   a. http://bit.do/fJ5ti

7. F. Díaz, J. Aguilar, J. Altamiranda, J. Cordero, D. Chávez and J. Gutiérrez de Mesa “Metropolis: an Emerging Serious Game for the Smart City”. DYNA, Vol. 86 N° 211, pp. 215-224, Colombia, 2019.

   Disponible en línea:

   a. https://revistas.unal.edu.co/index.php/dyna/article/view/80864

   b. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0012-73532019000400215

8. F. Díaz, J. Aguilar, J. Altamiranda, J. Gutiérrez de Mesa and Á. Pinto, "Sistema Adaptativo de Tramas para Juegos Serios Emergentes basado en el Algoritmo de Optimización de Colonia de Hormigas”. Proceeding XLV Latín American Computer Conference (CLEI 2019), Panamá, 2019.

   Disponible en línea:

   a. https://ieeexplore.ieee.org/document/9073804

b. http://clei2019.utp.ac.pa/storage/app/uploads/public/5d8/cf7/576/5d8cf75762f35112494695.pdf

9. F. Díaz, J. Aguilar and J. Altamiranda "Design of a Serious Emerging Games Engine Based on the optimization Algorithm of Ant Colony". DYNA, Vol. 85, Nº 206, pp. 311-320, Medellin, Colombia, 2018.

   Disponible en línea:

   a. https://revistas.unal.edu.co/index.php/dyna/article/view/69881/67762

10. F. Díaz, J. Aguilar, J. Altamiranda and D. Mosquera "Motor de JS para ARMAGAeco-c". Revista Científica UNET. Vol. 28, Nº2, pp. 100-110 ISSN: 1316-869X11C, Táchira, Venezuela, 2016.

    Disponible en línea:

    a. https://studylib.es/doc/8920357/revista-cientifica-unet-vol-28-nro-2

## APENDICE B: ARTÍCULOS ACTUALMENTE EN REVISION EN EL MARCO DE LA TESIS

Los siguientes artículos están en proceso de revisión y no han sido publicados:

1. J. Aguilar, F. Díaz, J. Gutiérrez, J. Altamiranda, N. Pérez and Á Pinto, “Parameter Adaptive System to Learning Processes for Emerging Serious Games using Cultural Algorithms”.

2. J. Aguilar, F. Díaz and J. Altamiranda, “Strategy Adaptive System to Learning Processes for Emerging Serious Games using a Fuzzy Classifier System”.

# APENDICE C: ARQUITECTURA COMPUTACIONAL DEL SISTEMA

Nuestro sistema adaptativo de JSE para un MJSE está compuesto por tres subsistemas, cada uno responsable de adaptar un aspecto específico del JSE (tramas, parámetros o estrategias). A su vez, cada uno de ellos está basado en una técnica del área de inteligencia artificial diferente. Específicamente, el sistema adaptativo de secuencias/tramas se basa en ACO, el sistema adaptativo de parámetros se basa en algoritmos culturales, y finalmente, el sistema adaptativo de estrategias se basa en SCD. Por consiguiente, la especificación de cada subsistema (diseño e implementación) es diferente. En los siguientes apéndices se presenta la especificación detallada de cada uno, los cuales se integran como es indicado en la sección 3 (ver Figuras 3.1 y 3.2).

## C.1 ESPECIFICACIÓN DEL SISTEMA ADAPTATIVO DE SECUENCIA

En este apéndice se presentan todos los aspectos del diseño e implementación del SAT basado en el algoritmo ACO [43, 55]. En particular, se explicará cada uno de los componentes principales del Módulo de Recuperación de Trazas y del Módulo de Emergencia de Secuencias. En la Figura C.1 se muestra de manera general, el flujo y los diferentes componentes del STE.




**Figura C.1** Diagrama de flujo del SAT [64]

1. Diseño del Módulo de Recuperación de Trazas: según [77] está compuesto por dos componentes principales: el componente de parseo de metadatos y el componente de comparación. Ambos componentes cumplen con un objetivo en común, el procesamiento de los recursos de aprendizaje (RA), para definir qué tan parecido es el contenido de los RA del tipo Juegos Serios con respecto a la temática que se esté buscando.

   Cada RA está caracterizado por un metadato que sigue el estándar *Learning Object Metadata* (LOM) [17]. Dicho archivo (usualmente codificado en XML) tiene la finalidad de describir a través de una serie de atributos la temática de cada Juego Serio (RA). A continuación, se hace una descripción de los atributos tomados en cuenta durante el proceso de parseo (ver Figura C.2):

   a. **Title**: Es el nombre del RA.
   b. **Language**: Es el lenguaje para el que fue hecho el RA.
   c. **Description**: Contiene una descripción del contenido del RA. Por ejemplo: "Juego de dominó interactivo que relaciona sus piezas según las diferentes representaciones fraccionarias gráficas o numéricas".
   d. **Keyword**: Palabras claves que representan el tema principal del RA. Por ejemplo: fracciones, dominó, cálculo, probabilidad.
   e. **Coverage**: Describe la zona geográfica o región en la que es aplicable el RA.
   f. **Format**: Identifica el software necesario para utilizar el RA.
   g. **TypicalAgeRange**: Edad intelectual del destinatario típico del RA. Por ejemplo: "7-9", "0-5", "15".
   h. **Difficulty**: Este elemento describe lo difícil que resulta el uso del RA para los usuarios típicos. Por ejemplo: muy difícil, difícil, medio, fácil, muy fácil.
   i. **Duration**: Tiempo aproximado o típico para asimilar el RA.
   j. **InteractivityLevel**: El grado de interactividad que caracteriza a ese RA. Se mide como, muy alto, alto, medio, bajo, muy bajo.
   k. **SemanticDensity**: La densidad semántica de un RA puede ser estimada en función de su tamaño, ámbito, o en el caso de recursos auto-regulados, tales como audio y video, su duración. La densidad semántica de un RA es independiente de su dificultad. Se mide como: muy alto, alto, medio, bajo, muy bajo.
   l. **IntendedEndUserRole**: Usuario(s) principal(es) para el(los) que ha sido diseñado el RA.
   m. **Context**: El entorno principal para el que fue diseñado el RA.
   n. **CognitiveProcess**: Es el tipo específico del proceso cognitivo del RA.
   o. **Cost**: Indica si el RA requiere pago para su uso.

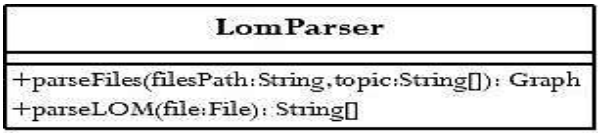



**Figura C.2** Diagrama de la clase LomParser

En nuestro caso, las tramas de Juegos Serios es lo que se buscará como RA.

Una vez realizado el proceso de parseo, en donde se utilizan los metadatos LOM de los RA para extraer los 15 atributos previamente descritos, es necesario comparar la información de cada atributo de la trama (RA) con respecto a la información del curso establecida (tópico que se desea impartir específicamente en el curso).

Eso determina el interés/parecido de dicha trama (RA) con respecto al curso a través de un valor numérico. Durante el proceso de comparación se hacen uso de los siguientes criterios según [64] (ver Figura C.3):

- Para el atributo “Title” se comparan las palabras que conforman al título del tópico deseado con las palabras del título del RA. Su valor determina la cercanía de dicha comparación, se devuelve 1 para un resultado correcto y 0 para un resultado incorrecto.
- Para el atributo "Language" se compara el lenguaje del tema deseado con el lenguaje del RA. Si son iguales, el valor obtenido es 1, si son diferentes 0.
- Para el atributo "Description" se comparan las palabras que corresponden a la descripción del tópico deseado con cada una de las palabras que conforman la descripción del RA. El valor obtenido corresponde al mayor entre todas las comparaciones, en un rango [0,1].
- Para el atributo "Keyword" se compara un conjunto de palabras clave separadas por coma, tanto del tópico deseado como del RA. Su valor determina la cercanía de dicha comparación, se devuelve 1 para un resultado correcto y 0 para un resultado incorrecto.
- Para el atributo "Coverage" se compara la cobertura del tema deseado con la cobertura del RA. Si son iguales el valor obtenido es 1, si son diferentes 0.
- Para el atributo "Format" se comparan un conjunto de formatos separados por coma, tanto del tópico deseado como del RA. Si alguno de los formatos coincide, el valor obtenido es 1, de lo contrario es 0.
- Para el atributo "TypicalAgeRange" se compara la edad especificada en el tópico deseado con la edad especificada en el RA. Si son iguales el valor devuelto es 1, si hay una diferencia de +/- 3 el valor devuelto es 0,5, de lo contrario el valor devuelto es 0.
- Para el atributo "Difficulty" se compara la dificultad especificada en el tópico deseado con la dificultad especificada en el RA. Usando una tabla como referencia, de acuerdo a la similitud entre las comparaciones, se devuelve un valor que puede ser 0, 0,25, 0,50, 0,75 o 1.
- Para el atributo "Duration" se compara la duración en minutos especificada en el tópico deseado con la duración en especificada en el recurso de aprendizaje. Si son

iguales el valor devuelto es 1, si hay una diferencia de +/- 30 minutos el valor devuelto es 0,5, de lo contrario el valor devuelto es 0.

- Para el atributo "InteractivityLevel" se compara el nivel de interactividad especificado en el tópico deseado con el nivel de interactividad especificado en el RA. Usando una tabla como referencia, de acuerdo a la similitud entre las comparaciones, se devuelve un valor que puede ser 0, 0,25, 0,50, 0,75 o 1.
- Para el atributo "SemanticDensity" se compara la densidad semántica especificada en el tópico deseado con la densidad semántica especificada en el RA. Usando una tabla como referencia, de acuerdo a la similitud entre las comparaciones, se devuelve un valor que puede ser 0, 0,25, 0,50, 0,75 o 1.
- Para el atributo "IntendedEndUserRole" se comparan un conjunto de palabras separadas por coma, tanto del tópico deseado como del RA. Si alguna de las palabras coincide, el valor obtenido es 1, de lo contrario es 0.
- Para el atributo "Context" se comparan un conjunto de palabras separadas por coma, tanto del tópico deseado como del RA. Si alguna de las palabras coincide, el valor obtenido es 1, de lo contrario es 0.
- Para el atributo "CognitiveProcess" se comparan un conjunto de palabras separadas por coma, tanto del tópico deseado como del RA. Si alguna de las palabras coincide, el valor obtenido es 1, de lo contrario es 0.
- Para el atributo "Cost" se compara el costo del tema deseado con el costo del RA. Si son iguales el valor obtenido es 1, si son diferentes 0.



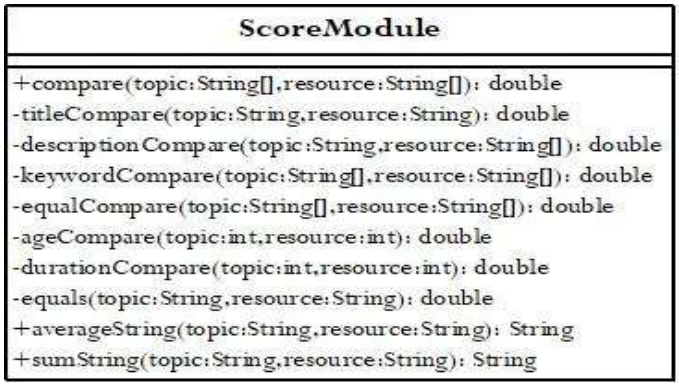



**Figura C.3** Diagrama de la clase ScoreModule [43, 55]

El promedio de los valores se convierte en el índice de similitud entre el RA y el tema deseado. Una vez que se realiza la comparación de todos los Juegos Serios con el tema deseado, el Módulo de Recuperación de Trazas ejecuta el algoritmo de la sección 4.2.1. Ese algoritmo determina si hay algún Juego Serio que cumple con el tema deseado, de lo contrario, si consigue algunos RA (tramas) más o menos similares a la temática buscada, intenta hacer emerger un Juego Serio para el tema deseado usando ACO (ver sección a continuación).

2. Diseño del Módulo de Emergencia de Secuencias: utiliza a ACO, que se encarga de crear agentes hormigas, con el fin de construir un JSE que cumpla con un objetivo específico (temática deseada). En particular, un videojuego seleccionado para formar parte del grafo de solución lo denominaremos subtrama, mientras que una trama de un JSE es la unión/fusión de una o más subtramas para generar una solución final [43, 55]. ACO está compuesto por:

   a. **Hormigas**: son los agentes que caminan en el grafo de subtramas (ver Figura C.4).

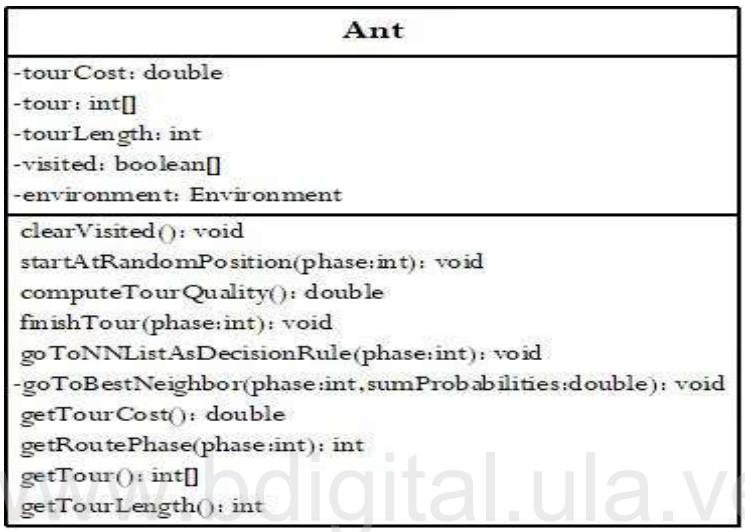





**Figura C.4** Diagrama de la clase Ant [43, 55]

   b. **Espacio de solución**: espacio que recorrerán las hormigas para obtener soluciones [43]. Es un grafo compuesto por nodos que definen las subtramas seleccionadas desde los diferentes repositorios, y arcos que establecen las relaciones de dependencia entre ellas, cuando existen. Los arcos establecen la secuencia lógica entre las subtramas. La figura C.5 describe la clase que define el ambiente (grafo).
   c. **Solución**: las subtramas (nodos) son marcadas por una feromona en el grafo, igual que los arcos que los interconectan. Cuando converge el algoritmo ACO, las subtramas 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.
   d. **Feromonas**: hay dos tipos, una para las subtramas (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.
   e. **Función Feromona**: actualiza cada tipo de feromona en función de la calidad del JSE propuesto.
   f. **Función Heurística**: define la decisión que toma una hormiga, al estar en una subtrama (nodo), con respecto a que otra subtrama (nodo) debe continuar visitando desde ella.

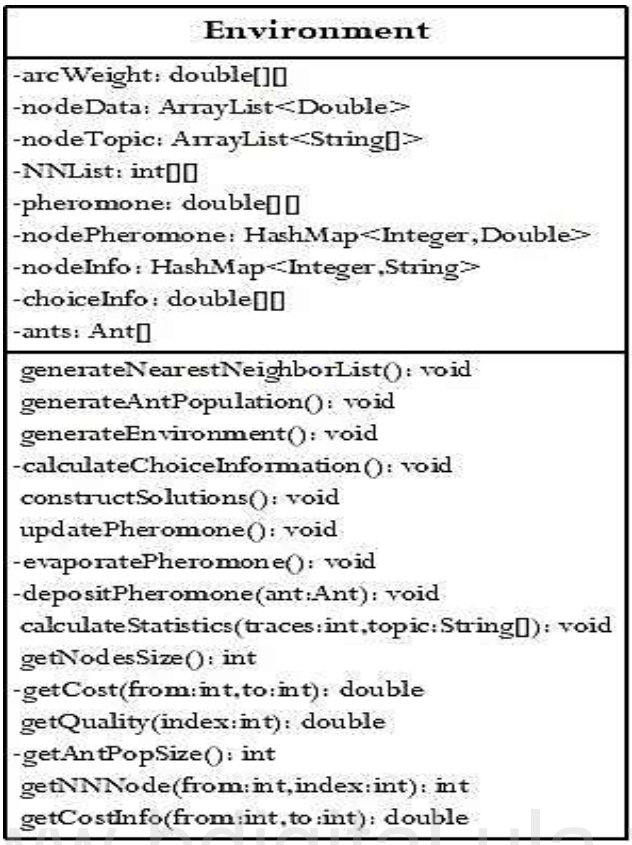





**Figura C.5** Diagrama de la clase Environment [43, 55]

ACO permite un proceso de aprendizaje colectivo entre las hormigas, para hacer emerger la nueva configuración del JSE. Así, la solución no es más que una secuencia de subtramas, ordenadas según una secuencia lógica entre ellas.

3. Macroalgoritmo ACO para el SAT: es el clásico de ACO [24]. En específico, en nuestro caso consiste de una fase de inicialización de parámetros, un proceso iterativo hasta que el sistema converja, y la construcción de la solución final. Dicho macroalgoritmo se detalla en la sección 3.3.1.1 y se representa así:

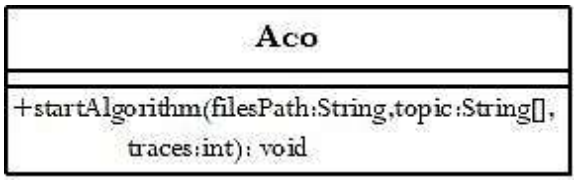



**Figura C.6** Diagrama de la clase ACO [43, 55]

## C.2 ESPECIFICACIÓN DEL SISTEMA ADAPTATIVO DE PARÁMETROS

A continuación, se explican cada uno de los objetos que conforman tanto el espacio de población como el espacio de creencias del algoritmo cultural [81], así como también, las funciones de aceptación e influencia.

La figura C.7 presenta el flujo arquitectónico del SAP, que describe los diferentes componentes del mismo, que serán descritos en las secciones 3.3.2 en donde, se inicializa t que es un contador, luego se introduce el número de generaciones que se van a evaluar, los valores para normalizar la función objetivo, valor del momento, probabilidades de cruce y mutación.

Posteriormente se inicializa el espacio de población y de creencia, se evalúa espacio de población generalmente con 20% de muestra de individuos y se pasa actualizar el espacio de creencias, se selecciona un numero entre 0 y 1 para la probabilidad de operador cruce o mutación luego pasa a otra generación finalizando en la cantidad que se colocó al inicio de la mutación.

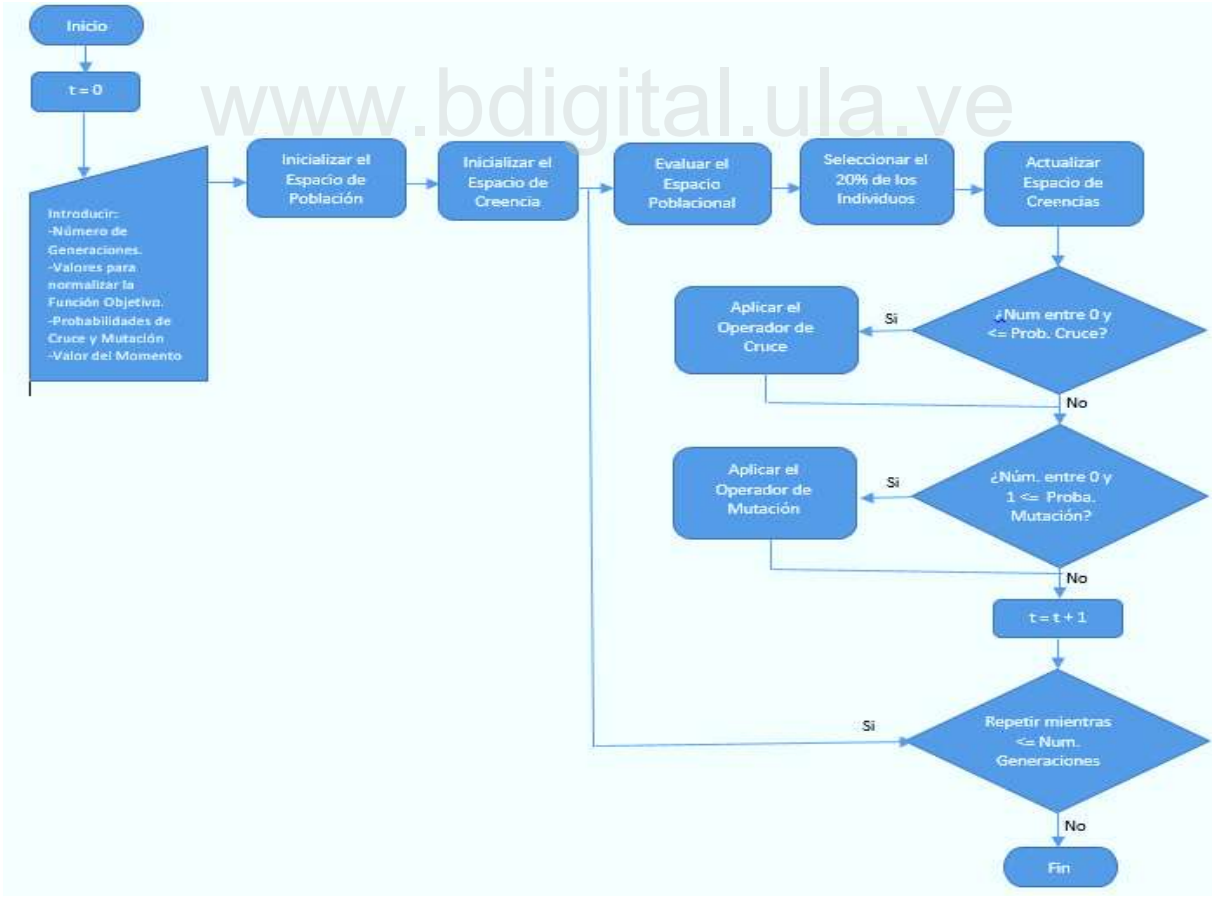



**Figura C.7.** Diagrama de Flujo del SAP [66]

1. Macroalgoritmo cultural para las propiedades: presentado en la sección 2.3.2 adaptado a la presente tesis es implementado de la siguiente forma:

   ***a.*** **Definición de los objetos:** el tiempo (*t*) es un entero positivo, el espacio de población es un vector en el que cada elemento está compuesto por un individuo, ambos conocimientos ($c_s$ y $c_n$) se instancian con sus clases (ConocimientoSituacional y ConocimientoNormativo). El número máximo de generaciones (numGeneraciones), *a, b,* la probabilidad de cruce (probCruce), la probabilidad de mutación (probMutación) y el momento, son constantes definidas por el usuario.

   *b.* **Inicialización:** La función "Introducir" consiste en capturar los valores que vienen del usuario, para inicializar las constantes previamente mencionadas; mientras que la función "Inicializar" define los valores iniciales que tendrán el espacio de población y el conocimiento normativo. En el espacio de población, se inicializa cada individuo con sus debidos parámetros; en el conocimiento normativo, los límites inferiores y superiores que cada parámetro.

   *c.* **Ciclo Repetitivo:** Por cada iteración, ocurre la siguiente secuencia:



      i. *Evaluación de la población:* cada individuo es evaluado por medio de la ecuación 3.1.

      ii. *Ordenamiento de la población:* de mayor a menor, para facilitar la selección, tanto del mejor individuo como del 20% de ellos, los cuales representan el conjunto de los aceptados para la reproducción en la nueva población.

      iii. *Selección del 20% de los individuos:* se define un vector llamado "Aceptados", el cual se construye a partir del recorrido del vector ordenado: se va seleccionando cada individuo, hasta llegar al límite que define el 20%.

      iv. *Actualización de los conocimientos:* permite definir los nuevos valores para los conocimientos: *normativo* (requiere el conjunto de los aceptados) y *situacional* (requiere tanto del conjunto de los aceptados, como del tiempo actual y del momento).

      v. *Aplicación del operador de cruce:* requiere del espacio de población, del conjunto de los aceptados, y de la probabilidad de cruce, para generar nuevos individuos.

      vi. *Aplicación del operador de mutación:* requiere de la nueva población y de la probabilidad de mutación, para generar nuevos individuos.

*d.* **Pseudocódigo:** a continuación, se describe en alto nivel, el nuevo macro-algoritmo. El mismo no cambia mucho con respecto al de un Algoritmo Cultural clásico, solo que en este caso está usando nuestras definiciones previas de funciones, operadores genéticos y conocimientos en el espacio de creencias (normativo y situacional).

*Inicio*
*Introducir (numGeneraciones, a, b, probCruce, probMutacion, constanteMomento)*
*Inicializar (EspaciodePoblacion) e Inicializar($c_n$)*
***Repita***
*EspaciodePoblacion <- Evaluar (EspaciodePoblacion, a, b)*
*EspaciodePoblacion <- Ordenar (EspaciodePoblacion, MayoraMenor)*
*Aceptados <- Seleccionar (EspaciodePoblacion)*
*Actualizar ($c_n$, Aceptados)*
*Actualizar ($c_s$, Aceptados, t, constanteMomento)*
*EspaciodePoblacion <- Cruce (EspaciodePoblacion, Aceptados, probCruce)*
*EspaciodePoblacion <- Mutacion (EspaciodePoblacion, probMutacion)*
*t = t + 1*
***Hasta** (t ≤ numGeneraciones)*
*Fin*

2. Diseño del Espacio de Población: en la figura C.8, se muestra el diagrama de clases del tipo de dato abstracto (TDA) llamado Individuo.




Figura C.8. Diagrama de la clase Individuo [66]

Para la implementación del mismo, se definió un TDA llamado Fila, que contiene los valores de V, IO, y C, los cuales representan sus columnas (ver figura C.9).


**Figura C.9.** Diagrama de la clase Fila [66]

Luego, se definió un TDA llamado ConocimientoSituacional, cuyo atributo es un diccionario, y cada elemento está compuesto por un par, el cual consiste en una clave y un valor. La clave es el id del parámetro (0, 1, …, n-1), donde n es el número de parámetros, mientras que el valor es un vector de filas, que representa la tabla para cada parámetro (ver figura C.10).




**Figura C.10.** Diagrama de la clase ConocimientoSituacional [66]

La implementación consiste en definir dos TDA: Límite y ConocimientoNormativo, ambos se muestran en las figuras C.11 y C.12, respectivamente. El TDA Limite describe la Tabla 3.4 para un parámetro dado, y el TDA ConocimientoNormativo la lista de parámetros con sus límites.

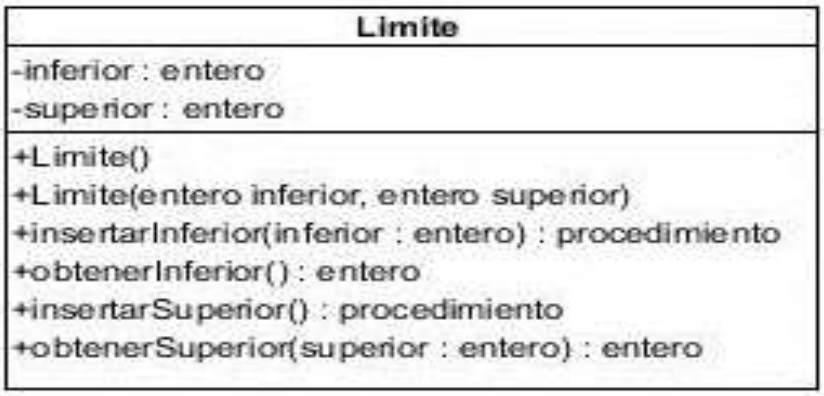



**Figura C.11.** Diagrama de la clase Limite [66]




**Figura C.12.** Diagrama de la clase ConocimientoNormativo [66]

## C.3 ESPECIFICACIÓN DEL SISTEMA ADAPTATIVO DE EXTRATEGIA

1. Macroalgoritmo SCD para la Estrategia: en la teoría de la sección 2.3.3, nos muestra un SCD que se completa con la presente información a continuación, se crea el nuevo macroalgoritmo para la presente tesis, en el cual se utiliza algoritmos genéticos como evaluadores y sistemas adaptativos, que se basa en [62]. Este código ha sido personalizado de acuerdo con nuestro diseño del SCD [81]. En particular, selecciona las mejores reglas que son las reglas más activas (función fitness). El resto del código se puede reutilizar cambiando las variables difusas y las funciones de pertenencia de acuerdo con nuestras reglas de control genéricas. Finalmente, el procedimiento de generación de nuevas reglas es el mismo propuesto en [62]. La figura C.13 muestra el diagrama de flujo del algoritmo genético, para luego utilizar el SCD. A continuación, describimos las siguientes fases:

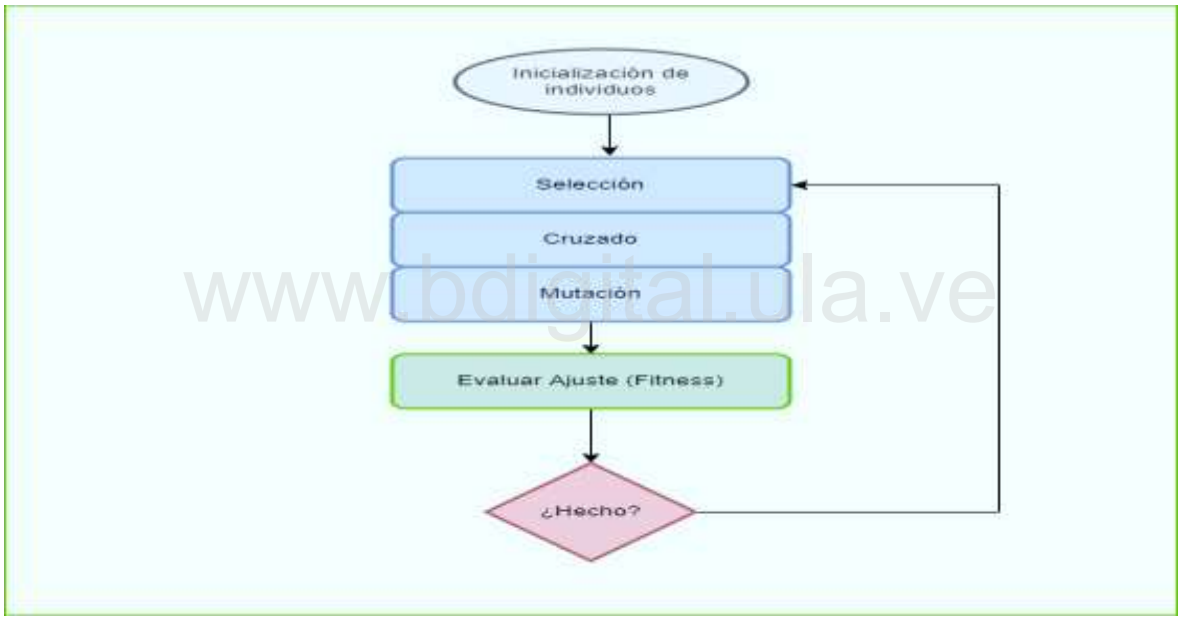





**Figura C.13.** Algoritmo Genético [81]

*a.* **Inicialización**: se inicializa cada individuo con sus debidos parámetros.
*b.* **Ciclo Repetitivo**: por cada iteración, ocurre la siguiente secuencia:
   i. *Selección:* de la cantidad de individuo a entrenar.
   ii. *Cruzado:* se cruzan los individuos del conjunto de entrenamiento.
   iii. *Mutación:* se mutan los individuos del conjunto de entrenamiento
   iv. *Evaluar Ajuste (fitness):* se evalúa después del entrenamiento, cual o cuales es/son el/los mejor(es) individuos según su aptitud
   v. *¿Hecho?:* fin del ciclo repetitivo una vez que haya seleccionado el mejor o los mejores individuos de una muestra dada por el usuario al inicio del programa.
*c.* **SCD:** se llama e inicializa al SCD definida en la figura 3.14:

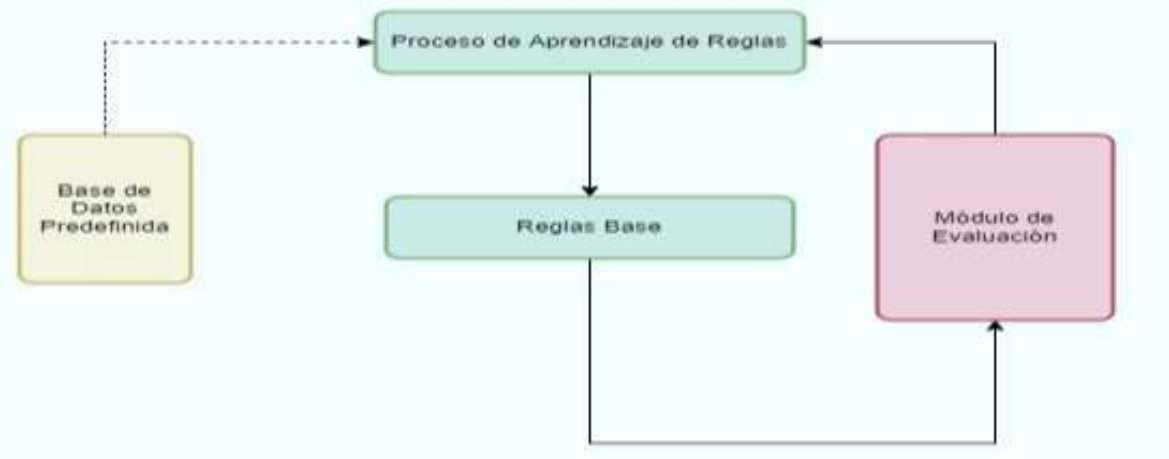



**Figura C.14.** SCD [81]

i. *Base de Datos Predefinida:* introduce los datos de las reglas echas por un especialista mediante un archivo.
ii. *Proceso de Aprendizaje de Reglas:* donde las reglas van detectando mediante varias interacciones cuales son las más efectivas.
iii. *Regla Base:* conjunto de reglas iniciales que permiten será aplicadas en el JSE.
iv. *Módulo de evaluación:* revisa cuales son las mejores reglas, internamente realiza lo que aparece en la siguiente figura C.15



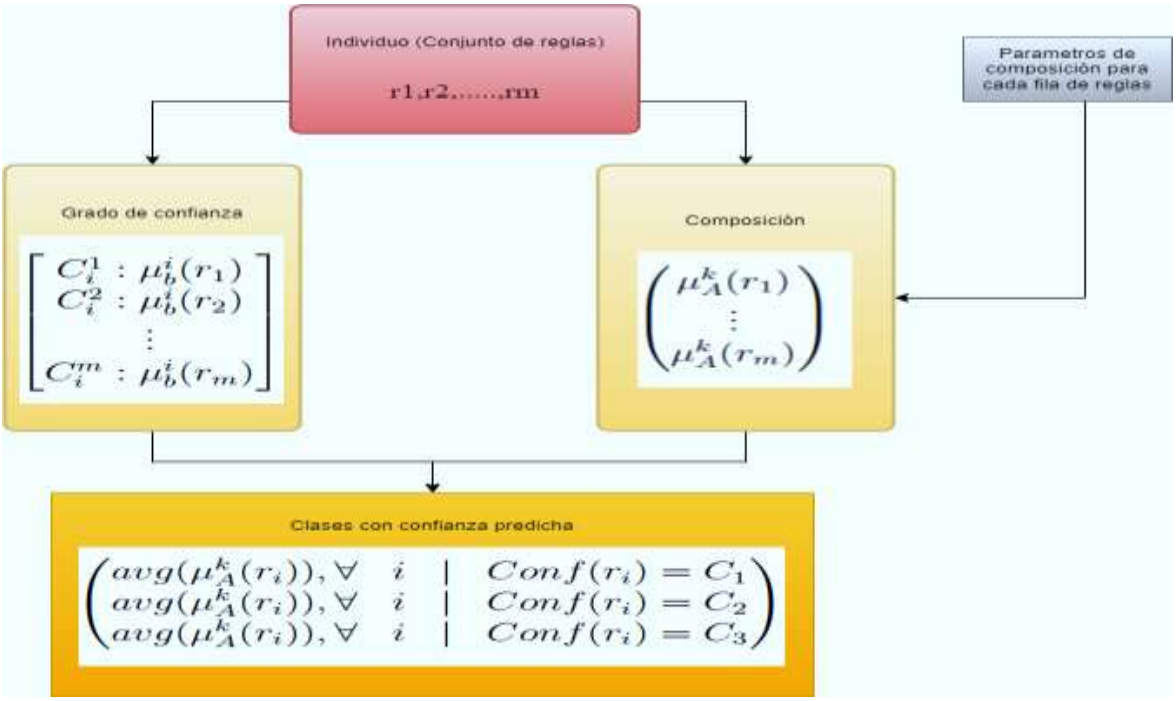



**Figura C.15.** *Módulo de evaluación [81}*

- ***Individuo:*** es el conjunto de reglas *que va desde* $r_1$, $r_2$ ….., $r_m$
- ***Grado de confianza:*** *según* [62], el vector de confianza de la mejor clase dada una regla, medimos el grado de verdad de esa regla usando la formula descrita anteriormente en nuestros datos de entrenamiento para cada clase. La función getConf da el vector:

$$\begin{bmatrix} C_1 : \mu_b^1(r) \\ C_2 : \mu_b^2(r) \\ C_3 : \mu_b^3(r) \end{bmatrix}$$

Con $C_i$ es la clase y $\mu_b^i$ el grado de verdad de esa regla *r* en la clase *i*.

Un Individuo se compone de un conjunto de reglas, por lo que la función getConfVect da el vector:

$$\vec{Conf} = \begin{bmatrix} C_i^1 : \mu_b^i(r_1) \\ C_i^2 : \mu_b^i(r_2) \\ \vdots \\ C_i^m : \mu_b^i(r_m) \end{bmatrix}$$

Con *m* que es el número de reglas. Para cada regla, tomamos el máximo $\mu_b$ de getConf. Este vector representa cada una de las reglas de un individuo, que clase se adapta mejor y con qué confianza.



- ***Composición:*** el siguiente paso es hacer una composición difusa en cada una de las reglas, dada una instancia de datos, aquí [x1; x2; x3; x4] [62].

La función getM*u*A toma dos parámetros: una instancia de datos y una regla. Recordando esta fórmula:

$$A = [x_{p(1)} = \bigcup_{j \in D(1)} A(p(1), j)] \bigcap \cdots \bigcap [x_{p(s)} = \bigcup_{j \in D(s)} A(p(s), j)]$$

Elegimos medir a $\mu_A$ así:

$$\mu_A = min \begin{pmatrix} max(A(p(1), j), \forall j \in D(1)) \\ \vdots \\ max(A(p(s), j), \forall j \in D(s)) \end{pmatrix}$$

Tomemos nuestra regla de ejemplo:

If Enemigo = 2 and Hueco = Corto or Hueco = Medio and Obstáculo= Tubo and Arma= Bala

Y una instancia de datos (normalizados): x1 = 0:97; x2 = 0:86; x3 = 0:97; x4 = 0:88.

De la misma manera que lo hicimos para nuestro vector de confianza, usamos la función getMuVect:

$$\overrightarrow{\mu_A^k} = \begin{pmatrix} \mu_A^k(r_1) \\ \vdots \\ \mu_A^k(r_m) \end{pmatrix}$$

Para el k-ésimo dato de los datos de prueba.

- ***Clases con confianza predicha:*** *según* [62], ahora queremos obtener la confianza predicha de clases para una instancia. $\overrightarrow{Conf}$ nos da la clase más adecuada para cada regla, y $\overrightarrow{\mu A}$ su confianza dada una instancia de datos.

Confianza de las clases previstas: $\begin{pmatrix} avg(\mu_A^k(r_i)), \forall \ i \ | \ Conf(r_i) = C_1 \\ avg(\mu_A^k(r_i)), \forall \ i \ | \ Conf(r_i) = C_2 \\ avg(\mu_A^k(r_i)), \forall \ i \ | \ Conf(r_i) = C_3 \end{pmatrix}$



Aquí, la clase predicha será simplemente el promedio máximo.

*d.* **Diagrama de Clases:** en la figura C.16, se muestra los diagramas de clases del tipo de dato abstracto (TDA) llamados SetTriangulo que gráfica con los mínimos y máximo la función de pertenencia de variables y conjuntos difusos, Individuo que se compone de un conjunto de reglas que da el vector y la aptitud, Población que es la cantidad de individuos con el tamaño de la muestra detectando a los mejores.

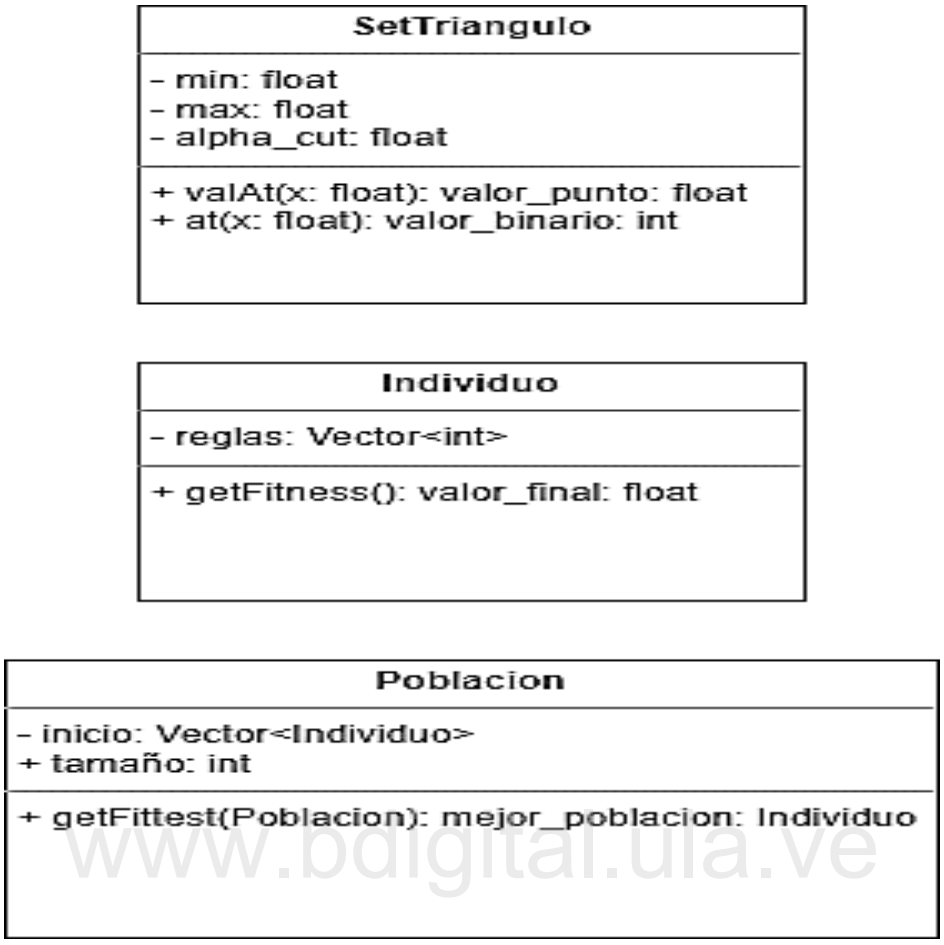





**Figura C.16.** Diagrama de Clases [81]

*e*. **Pseudocódigo:** a continuación, se describe en alto nivel el nuevo macro-algoritmo. El mismo no cambia mucho con respecto al de un SCD clásico, solo que en este caso está usando nuestras definiciones previas de funciones, operadores genéticos.

Inicio
    *Inicializar* (Individuos)
    ***Repita Mientras*** (existan individuos)
        *Selección, Cruzado, Mutación*
        *Evaluar Ajuste* (Fitness)
    *Inicializar* SCD
    ***Repita Mientras*** (existan eventos)
        *Fusificar* (evento)
        *Activar* (SCD, evento)
        *Actualizar* (SCD)
Fin