n°: 2020-XXA

PROYECTO DE GRADO

Presentado ante la ilustre UNIVERSIDAD DE LOS ANDES como requisito parcial para obtener el Título de INGENIERO DE SISTEMAS

SISTEMA ADAPTATIVO DE PARÁMETROS PARA JUEGOS SERIOS EMERGENTES, UTILIZANDO ALGORITMO CULTURAL

Por

Br. Jesús Octavianosky Gutiérrez Sánchez
Tutor: Dr. José Aguilar
Cotutor: Dr. Junior Altamiranda
Facilitador: Ing. MSc. Francisco Díaz

Marzo 2020

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Sistema adaptativo de parámetros para juegos serios emergentes, utilizando algoritmo cultural

Br. Jesús Octavianosky Gutiérrez Sánchez

Proyecto de Grado — Sistemas Computacionales, xx páginas

Resumen: En el presente trabajo, se plantea un sistema adaptativo de parámetros para juegos serios emergentes, que permita la emergencia de propiedades en un juego, lo que cambia sus características, con el fin de adaptarse al contexto en el que se esté desarrollando. En trabajos anteriores se ha propuesto un motor para juegos serios emergentes basado en el algoritmo de optimización de colonia de hormiga, el cual permite la emergencia inicial de un juego serio. En éste trabajo se especifica uno de los componentes del sistema adaptativo del motor de videojuego, el cual permite su adaptación dinámicamente mientras se realiza el juego, adecuando sus propiedades al contexto-dominio educativo. Particularmente, el sistema de emergencia de propiedades desarrollado en este trabajo, está basado en el uso de algoritmos culturales, el cual establece un proceso de aprendizaje para ir modificando sus parámetros. Este sistema se prueba en el contexto de un salón de clases inteligente.

Palabras clave: Motor de Juegos Serios Emergentes, Sistema de Adaptación de Videojuegos, Sistema Adaptativo de Parámetros, Algoritmo Cultural, Salón de Clases Inteligentes.

A Dios Todopoderoso.

A mis padres: Ciria y Octaviano.

Índice

Índice de Figuras

Índice de Tablas

Agradecimientos

A Dios Todopoderoso, porque con él está el poder y la sabiduría, gracias por haber incrementado mi fe.

A mis padres: Ciria y Octaviano, que siempre los he tenido por guía y norte en mi vida, personas ejemplares y modelos de virtudes. Gracias por estar tan cerca de mí, cuidándome, animándome, apoyándome en todo momento, esperando con entereza la culminación de mis metas.

A la Universidad de los Andes, mi alma mater.

A todos los profesores de la facultad de Ingeniería, que interactuaron conmigo, tanto en el ciclo básico como en el profesional, gracias por formar parte en el desarrollo de mis conocimientos.

Al tutor Dr. José Aguilar, el cotutor Dr. Junior Altamiranda y el facilitador Ing. MSc. Francisco Díaz, por darme la oportunidad de integrarme en su equipo de trabajo, en conjunto con el reconocimiento, la confianza, el apoyo y el aporte de sus conocimientos y experiencias, para contribuir con esta investigación, al desarrollo de un motor de juegos serios emergentes.

Capítulo 1

Introducción

Según [1] desde el punto de vista de un videojuego, un juego serio emergente (JSE) debe basarse en un motor de juegos serios emergentes (MSJE) que permita manejar las diferentes formas de emergencia en un juego emergente (JE), pero a la vez, por sus características de juego serio (JS), esa forma de emergencia es guiada según un contexto muy específico (en nuestro caso, educativo) en un salón de clases inteligente (SaCI) [2]. 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 auto-gestionados, entornos en los que interactúan, entre otras cosas.

Un motor de juego (MJ) es un conjunto de programas que permiten la creación, la representación y la ejecución de un juego. Pueden ser vistos como librerías o framework, donde los desarrolladores se enfocan tanto en las mecánicas, como en las lógicas y demás características específicas del juego el cual planean concebir. Por lo tanto, se considera como el núcleo general que une todas las partes de un juego.

Hoy en día, existen una gran variedad de motores para que el usuario pueda crear sus propios juegos. Ahora bien, no se han desarrollado motores de juegos para juegos emergentes, y mucho menos para JSE.

1.1 Planteamiento del problema

En el área de las aulas inteligentes, en el contexto educativo, cuando se presenta el JSE al estudiante, ocurre la problemática de que el nivel de dificultad no sea el adecuado para el mismo (nivel de primaria o educación inicial más bajo, nivel de bachillerato o universitario más alto), ya que los jugadores tienen diferentes habilidades para jugar.

Esto hace que los jugadores se frustren si el juego es demasiado difícil, o pierdan el interés si ocurre lo contrario. Por lo tanto, es necesario capturar el nivel educativo de los estudiantes que se encuentran en el SaCI.

Una vez que se capturen estos datos, se procede a adaptar las propiedades del JSE mediante parámetros óptimos, usando en nuestro caso los algoritmos culturales, con el fin de obtener un JSE que presente el nivel que se espera.

1.2 Justificación

En éste trabajo, se propone el desarrollo de un sistema adaptativo de parámetros (SAP) para JSE que permite la emergencia de propiedades en JS, específicamente, cambiando las características del videojuego parametrizando sus propiedades, con el fin de adaptarse al contexto educativo en el que se esté desarrollando.

En trabajos anteriores se ha propuesto un motor para JSE basado en el algoritmo de optimización de colonia de hormiga (ACO) [1], que permite la emergencia inicial de un JS. En esta investigación se especifica el sistema adaptativo de videojuego (SAV), el cual permite su adaptación dinámica (durante la realización del JSE), y está compuesto por las subcapas de estrategias, secuencias y propiedades, que gestionan cada uno de los tipos de emergencia posible en un JSE, con la intención de adaptarlo dinámicamente al contexto-dominio educativo.

Por esta razón, aquí se analiza específicamente el comportamiento de la subcapa de emergencia de propiedades, modificando sus parámetros por medio de algoritmos culturales (AC).

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar un Sistema que permita la determinación de los parámetros óptimos de un JSE, para adecuarse a las características de los jugadores.

1.3.2 Objetivos específicos

- Caracterizar el uso de los JS y juegos emergentes.
- Caracterizar el proceso de optimización de parámetros usando AC.
- Especificar la arquitectura del motor de juegos para el proceso de optimización de parámetros en un JSE usando AC.

- Implementar el algoritmo cultural en JSE.
- Desarrollar un prototipo y realizar pruebas sobre un salón de clases.

1.4 Antecedentes

En [3], los JS se utilizan como herramienta en un proceso de aprendizaje, a fin de simular conceptos abstractos para que parezcan más realistas. Se enfrentan al problema de que los jugadores tienen diferentes habilidades para jugar. Esto hace que los jugadores se sientan frustrados si el juego es demasiado difícil, o se aburren si es demasiado fácil. En el ámbito del uso de técnicas inteligentes en videojuegos, en [4] presentan un análisis de las técnicas de inteligencia artificial más utilizadas en los videojuegos, y particularmente, en juegos del tipo de simuladores. Entre las técnicas indicadas en este trabajo se encuentran los sistemas de clasificación de aprendizaje (LSC, por sus siglas en inglés) las redes neuronales y los algoritmos genéticos, entre otros.

En [5] aplican un marco de optimización evolutivo basado en AC, como mecanismo de colaboración para distribuir el conocimiento de un juego en una población. El rendimiento de este mecanismo se compara con otros enfoques. Los resultados preliminares sugieren que este enfoque es mejor. [6] propone una nueva arquitectura para múltiples ACs, donde se incorpora un nuevo marco de selección multinivel (ML-MPCA). Se introduce un proceso de selección de dos niveles, a saber, la selección dentro del grupo y la selección entre grupos. Las personas interactúan con los otros miembros del grupo en un juego evolutivo que determina su estado físico. En [7] presenta un algoritmo evolutivo inspirado en la teoría de juegos para la 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 [8] se introduce un nuevo método de distribución de conocimiento para AC, y su rendimiento se compara con el enfoque clásico. [9] describe un enfoque para utilizar la computación humana en videojuegos convencionales. Para hacer esto, examinan las mecánicas de un juego y hacen que la computación humana (meta-heurísticos) coincida con ellos.

Por otro lado, en [24] se ha propuesto un juego emergente denominado Metrópolis, que permite generar comportamientos emergentes débiles (aparición de patrones) en el juego. Dicho juego fue extendido en [30], para permitir la emergencia fuerte en el juego, en este caso, para posibilitar la emergencia de los valores de los parámetros del juego, en función de las dinámicas que se van dando en el mismo. Finalmente, en [33] fue extendido como un juego serio emergente, para poder ser usado para capacitar a la ciudadanía en procesos de e-participación para la gestión urbana de una ciudad inteligente.

Vinculados a este proyecto se encuentran los siguientes trabajos: en [2] y [25] se especifica la arquitectura de un motor de juegos serios emergentes, detallándose todos sus componentes. Además, se describe como dicho motor se integra en un SaCI, usando la teoría de agentes. A su vez, en [1] y [29] se detalla uno de los componentes adaptativos del motor de juegos serios emergentes, el sistema de adaptación de secuencias, el cual permite la aparición de nuevas tramas, en función de la dinámica del entorno donde el JSE es usado. Para la gestión del proceso adaptativo se usan Algoritmo de Optimización de Colonia de Hormigas.

1.5 Alcance

Se va a llegar hasta un prototipo, sobre el cual se van a realizar experimentos, elaborar casos de estudio y ejecutar pruebas, en un contexto que emule un SaCI, para analizar el comportamiento del sistema adaptativo de parámetros (SAP).

1.6 Metodología

Como metodología de desarrollo de software, se usará el Proceso Unificado Racional (por sus siglas en inglés: RUP), junto con UML. Los mismos serán usados para el análisis, diseño, implementación y documentación.

1.6.1 Fases de la metodología

El proceso unificado se repite a lo largo de una serie de ciclos, que constituyen la vida de un sistema (ver Figura 1.1).

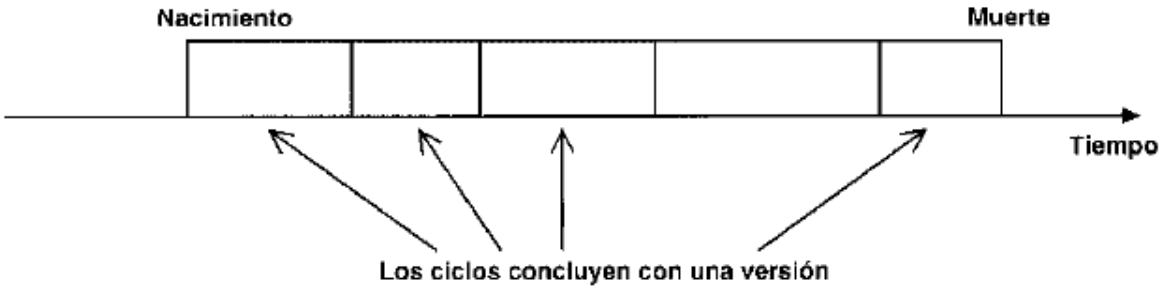

Figura 1.1. La vida de un proceso consta de ciclos, desde su nacimiento hasta su muerte [10]

Cada ciclo se desarrolla a lo largo del tiempo, y concluye con una *versión* del producto para los clientes. Un ciclo consta de cuatro fases: inicio, elaboración, construcción y transición; Cada fase se

subdivide, a su vez, en iteraciones o *mini-proyectos* (parte más pequeña del trabajo), que resultan en un incremento, y hacen referencia a pasos en el flujo de trabajo, que reflejan el crecimiento del producto (ver Figura 1.2).

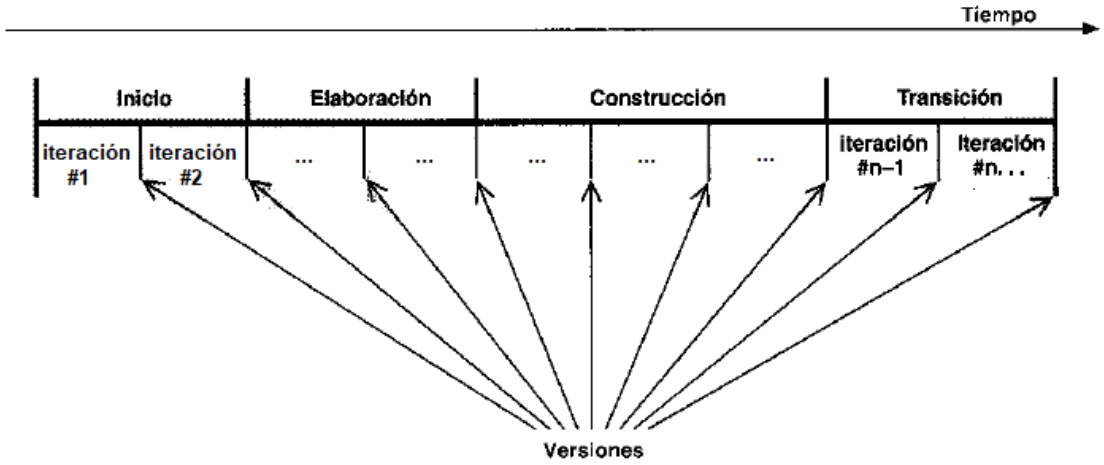

Figura 1.2. Un ciclo con sus fases e iteraciones [10]

La figura 1.3 muestra en la columna izquierda los flujos de trabajo (Requisitos, Análisis, Diseño, Implementación, y Prueba). Las curvas son una aproximación de hasta donde se llevan a cabo los flujos de trabajo en cada fase, recordando que cada fase se divide en iteraciones o mini-proyectos; Un mini-proyecto (ejemplo: Iter #1) pasa por los cinco flujos de trabajo para una parte de una fase (ejemplo: Iter. #1, en la primera mitad, de la fase de Inicio).

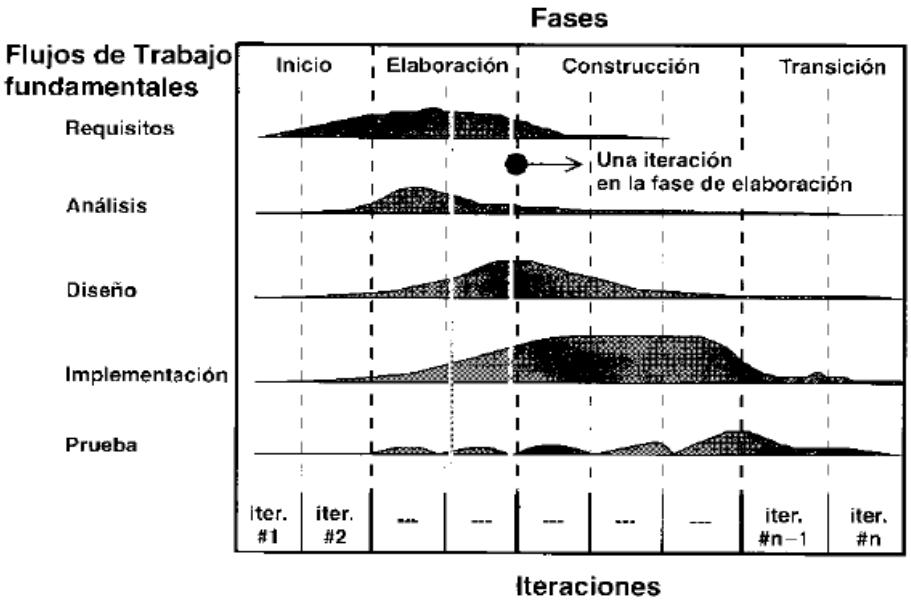

Figura 1.3. Los cinco flujos de trabajo (Requisitos, Análisis, Diseño, Implementación, Prueba) tienen lugar sobre las cuatro fases: Inicio, Elaboración, Construcción y Transición [10]

A continuación, se explican las cuatro fases.

- Inicio: consiste en definir el alcance del sistema, donde se desarrolla una descripción del mismo, a partir de una idea preliminar.
- Elaboración: consiste en definir la arquitectura. Se especifican en detalle la mayoría de los casos de uso del sistema y se diseña dicha arquitectura.
- Construcción: es donde se realiza el desarrollo, la creación del sistema. En ésta fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo, un producto preparado para ser entregado a la comunidad de los usuarios.
- Transición: cubre el período durante el cual el producto se convierte en versión beta, y es entonces que un número reducido de usuarios con experiencia prueba el producto e informa defectos y deficiencias, que posteriormente los desarrolladores corrigen e incorporan en una nueva versión mejorada [10].

Capítulo 2

Marco Teórico

En éste capítulo se introducen conocimientos básicos referentes a los juegos serios, sistemas emergentes, juegos serios emergentes y algoritmos culturales, que comprenden: sus conceptos, características, y demás argumentos que los describen.

2.1 Juegos Serios

El primer uso del término "Juego Serio" con un significado cercano a su uso actual, se presenta en [11], donde se define como juegos de mesa y juegos de cartas, pero proporciona una definición general que puede aplicarse con facilidad a los JS de la era actual: es un juego en un contexto con reglas entre adversarios, que intentan conseguir objetivos.

Según [12], el JS es una prueba mental, que se lleva a cabo frente a una computadora siguiendo un determinado protocolo, con la diversión como modo de formación, tomando en cuenta que sus objetivos están en el ámbito de la educación, sanidad, política pública y comunicación estratégica, entre otros.

En [13], los JSs lo definen como un grupo de videojuegos y simuladores, cuyo objetivo principal es la formación, antes que el entretenimiento. Esta área de desarrollo y creación de videojuegos ha surgido como una manera inteligente de combinar los beneficios de los videojuegos, su poder de penetración en la población, y las necesidades de educación y formación efectiva.

En el contexto de la informática, un JS se entiende como un programa de computadora o un simulador, que posee protocolos específicos, basados en hechos reales, cuyo principal objetivo consiste en educar y/o entrenar al usuario, con el fin de que adquiera las habilidades requeridas y que las aplique en un ámbito determinado, por ejemplo, la educación, la medicina, entre otros. Proveen un contexto de entretenimiento y auto-fortalecimiento, de manera que el usuario se motive, al mismo tiempo que aprende jugándolo.

2.1.1 Características

De acuerdo con [14], los JSs presentan algunas características importantes, como:

- Están destinados para la educación, el entrenamiento en habilidades determinadas, la comprensión de procesos complejos, sean sociales, políticos, económicos o religiosos. También sirven para publicitar productos y servicios.
- Están vinculados con algún aspecto de la realidad. Esto favorece la identificación del jugador con el área de la realidad que se está representando en el ambiente virtual. Por ejemplo, si se asume el rol en el juego de un dirigente político que debe tomar decisiones difíciles en las que se pone en peligro la vida de algunas personas.
- Constituyen un ambiente tridimensional virtual en el que se le permite una práctica "segura" a los aprendices en algunas áreas. En los casos de entrenamiento, por ejemplo, en el campo militar, se entrena a los soldados a manipular las armas.
- Hay intereses manifiestos en sus contenidos (políticos, económicos, psicológicos, religiosos, etc.).

También se pueden agregar las siguientes características:

- Se utilizan para aprender cómo los sistemas reaccionan en condiciones de continuo cambio.
- Quienes participan se enfrentan a ciertos datos preseleccionados, en circunstancias controladas.
- Pueden emplearse concertadamente modelos físicos, representaciones matemáticas y operadores humanos.
- Las personas participantes deben asumir papeles que implican diferentes grados de cooperación o rivalidad, y resolver conflictos entre jugadores o equipos tomando decisiones que reflejan su comprensión de los elementos esenciales del modelo.
- Se prevén sanciones, castigos o recompensas según las decisiones, las evaluaciones, etc., que se hagan durante el juego.
- Las decisiones modifican la situación. Se experimentan nuevas situaciones, tal que se establece una relación entre decisiones y cambios.
- El JS tiene un tiempo propio, más rápido que el real, en líneas generales. Todo sucede aceleradamente.

Adicionalmente, pueden hacer que el aprendizaje sea divertido. Aunque el término JS parezca bipolar, es decir, que parezca haber un contraste entre las cosas alegres como la diversión, el relax, y las cosas serias como la realidad, la responsabilidad, el hecho de que es un juego, no deja de ser seria la temática tratada.

2.1.2 Metodologías de desarrollo

En [15] proponen un proceso de diseño de JSs, que está constituido por 5 fases, cuya intención es captar todos los elementos del juego a partir de los objetivos y los requerimientos pedagógicos previamente planteados. Dichas fases abarcan desde la parte del diseño y desarrollo hasta las pruebas técnicas, en conjunto con la adquisición del conocimiento, con la fase final que comprende las continuas mejoras.

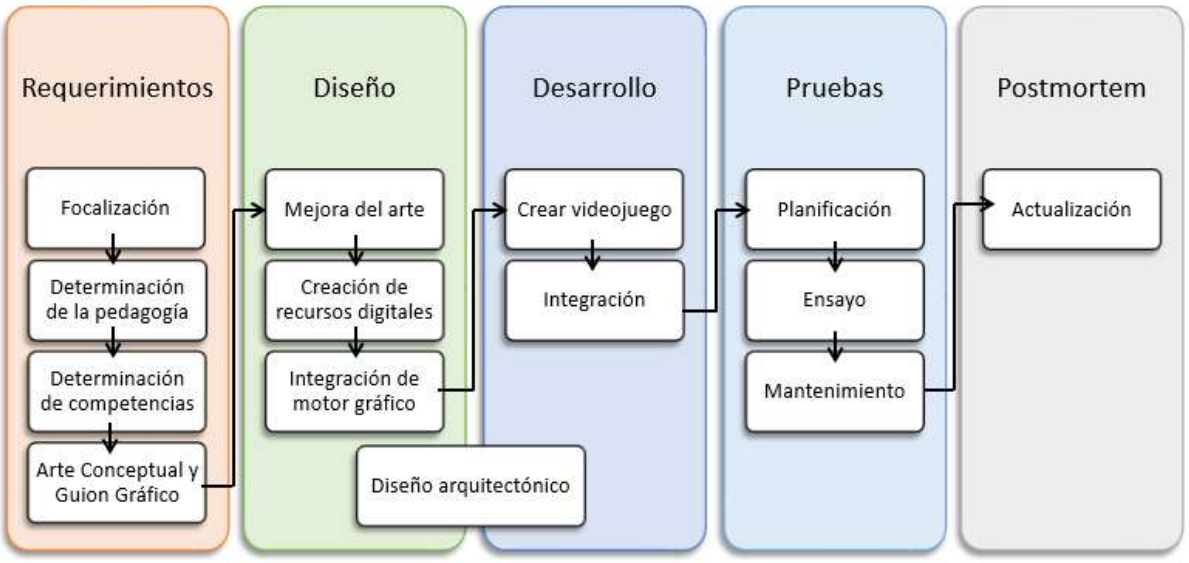

Figura 2.1. Propuesta de diseño de JSs [15]

Esta metodología de diseño está fundada en los paradigmas tradicionales de la ingeniería del software, y complementada por esfuerzos previos de desarrollo a gran escala de recursos digitales de aprendizaje [16]. A continuación, se presenta la descripción de cada fase.

- *Fase de requerimientos*: el objetivo es plantear las metas que cubren el juego, establecer los mecanismos pedagógicos, junto con el conocimiento a transmitir a los estudiantes, para determinar las competencias y el área de conocimiento, además de los conceptos artísticos e historia del juego.

- *Fase del diseño*: se crean todos los recursos digitales necesarios por el motor del juego, para la creación del videojuego, como son: ilustraciones 2D, modelos tridimensionales, mapas, objetos, materiales, superficies, sonido (diálogo, efectos especiales), música, etc.
- *Fase del desarrollo*: se crea el juego usando: capas, eventos, inteligencia artificial, estos se integran con menús, opciones etc.
- *Fase de prueba*: consiste en probar el videojuego en los aspectos: técnicos, adquisición del conocimiento, usabilidad, con el propósito de obtener datos estadísticos y así mejorar el juego.
- *Fase postmortem*: se analizan todos los procesos e información guardada durante el uso del videojuego para ser usado en desarrollos futuros.

En la literatura existen otras metodologías de diseño y desarrollo de juegos serios, como la propuesta en [16], que está alineada con la idea de desarrollar tareas educativas integradas en paralelo a un juego principal. Algunas de esas metodologías son descritas en [17, 18].

2.1.3 Motores de juegos

En [19], se explica que un motor de juego, también conocido como el núcleo de un videojuego, es un término que se refiere a un conjunto de librerías y demás herramientas tecnológicas, que son necesarias para crear, desarrollar e implementar un videojuego. Ese conjunto determina los distintos comportamientos que se presentan, por ejemplo, animaciones de todas las entidades registradas, así como también, el ambiente (sonido, efectos especiales), entre otras cosas.

Además, ese trabajo plantea una visión general de la arquitectura del motor, es decir, no importa de qué genero se trate el videojuego, dicha arquitectura tiene un propósito general. Se basa en una arquitectura definida por capas, mostrada en la figura 2.2. Esas capas son:

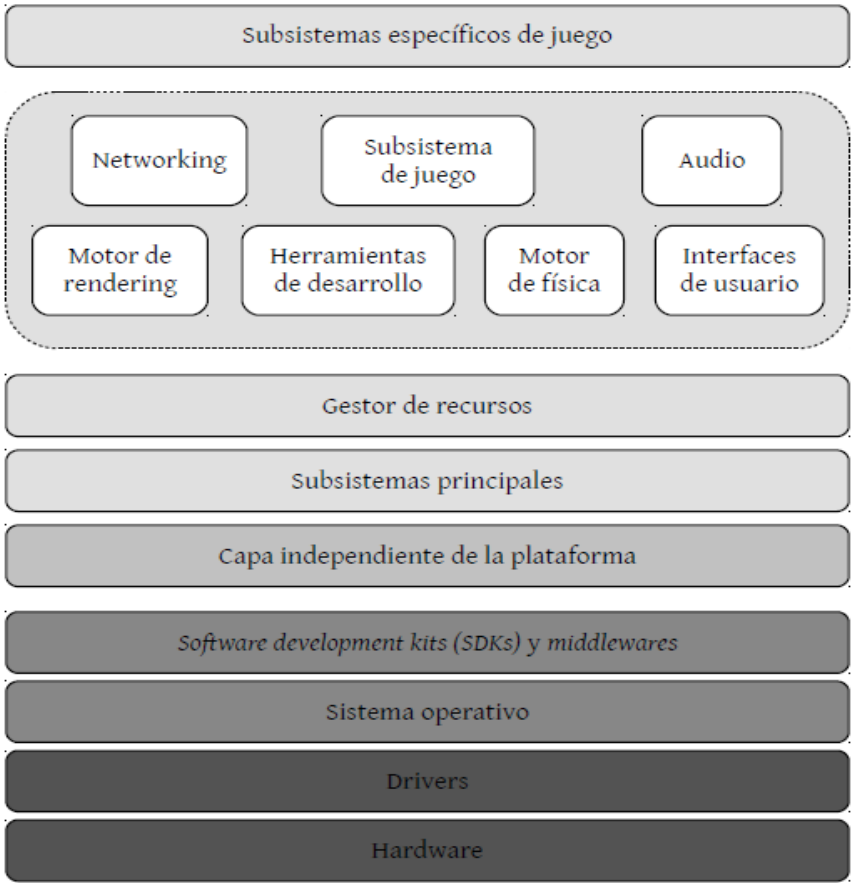

Figura 2.2. Arquitectura general de un motor de juegos [19]

- *Hardware*: capa referente a la plataforma en la que se ejecutará el motor de juego.
- *Drivers*: capa referente a aquellos programas de bajo nivel, que permiten la correcta gestión de determinados dispositivos, como, por ejemplo, tarjetas de expansión gráfica o de sonido.
- *Sistema operativo*: capa referente a la comunicación entre los procesos que se ejecutan y los recursos de hardware de la plataforma.
- *Software development kits (SDK) y middlewares*: capa referente al conjunto de librerías, o software, que abarcan el manejo de estructuras de datos, los gráficos 2D, 3D, sonido, la parte física, entre otros.
- *Capa independiente de la plataforma*: capa referente a la compatibilidad de ejecutar el motor de juego en diferentes ambientes, bien sea para computadoras personales, como para consolas de sobremesa (Nintendo Wii, PlayStation, Xbox), o teléfonos inteligentes.
- *Subsistemas principales*: capa referente a aquellas utilidades o bibliotecas que dan soporte al motor de juego, algunos de sus subsistemas más relevantes son:

- ✓ Biblioteca matemática: encargada del tratamiento de operaciones referentes al plano bidimensional como al tridimensional.
- ✓ Estructuras de datos y algoritmos: encargada de ofrecer una implementación personalizada de datos abstractos y algoritmos.
- ✓ Gestor de memoria: garantiza la asignación y liberación de la misma eficientemente.

- *Gestor de recursos*: capa referente a una interfaz unificada para el acceso de los componentes de software que existen en el motor, como, por ejemplo, animaciones, objetos 3D.
- *Subsistemas especializados*: compuesta por el motor de rendering, herramientas de desarrollo, motor de física, interfaces de usuario, networking, subsistema de juego, audio.
  - ✓ *Motor de rendering*: encargado del componente gráfico, consiste en una arquitectura por capas, como se presenta la figura 2.3.

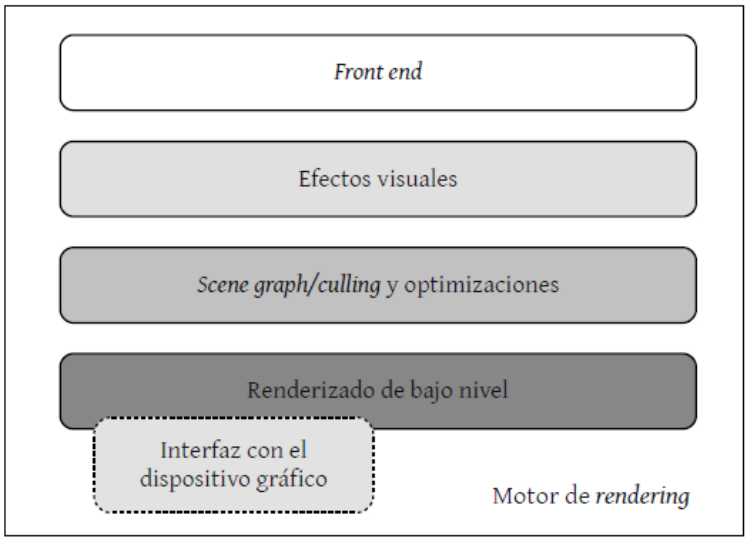

Figura 2.3. Visión conceptual de la arquitectura de un motor de rendering [19]

- ❖ Renderizado de bajo nivel: su objetivo es renderizar las primitivas (objetos geométricos creados a partir de parámetros configurables).
- ❖ Scene graph/culling y optimizaciones: responsable de seleccionar partes de la escena que se renderizarán, lo cual permite incrementar el rendimiento del motor de rendering.
- ❖ Efectos visuales: encargado de proporcionar aquellos efectos como el humo, el agua, el fuego, rayos, etc., que forman parte de un sistema de partículas.

- ❖ Front end: considera los distintos menús del juego, la superposición de contenidos 2D en una escena 3D, que permiten, entre otras cosas, la configuración del juego (dispositivos de entrada, audio, video), y la configuración del estado del personaje (inventarios de armas, herramientas, vestimenta, etc.).

- ✓ *Herramientas de desarrollo*: herramientas de propósito general para el desarrollo del motor de juego, así como también para la depuración.
- ✓ *Motor de física*: define los mecanismos para la detección, determinación y respuesta a colisiones entre las distintas entidades registradas en el juego.
- ✓ *Interfaces de usuario*: consiste en el tratamiento de los eventos de entrada (por parte del usuario) y salida (por parte del juego).
- ✓ *Networking*: consiste en la conectividad, tanto con el servidor del juego, el cual aloja por ejemplo los datos del jugador (sus estadísticas), como entre varios jugadores, conocido como el modo multijugador.
- ✓ *Subsistema de juego*: consiste en una arquitectura que reúne todas las propiedades de la realidad virtual, y administra los distintos personajes o entidades registradas. También se encarga de los protocolos registrados que gobiernan dicha realidad (ver figura 2.4).

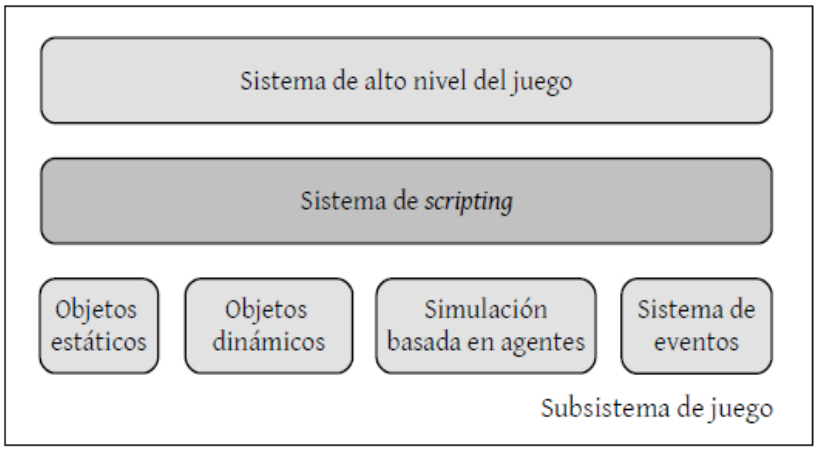

Figura 2.4. Visión conceptual de la arquitectura general del subsistema de juego [19].

- ❖ Objetos estáticos: gestiona aquellos elementos que no poseen movimiento, por ejemplo, edificios u otras estructuras.
- ❖ Objetos dinámicos: gestiona aquellos elementos que presentan movimiento, por ejemplo, rocas, sillas, cajas, etc. En juegos antiguos, los árboles no presentaban algún

tipo de movimiento, sin embargo, en la actualidad, se puede apreciar la dinámica tanto en el tronco y sus ramas, como en sus hojas.

- ❖ Simulación basada en agentes: ofrece la funcionalidad de la inteligencia artificial (IA), normalmente aplicada a los personajes NPC (por sus siglas en inglés: Non-player character) o no jugables. Por ejemplo, animales, u otras entidades que formen parte del ambiente virtual, que sean de ayuda o que rivalicen con el jugador, por lo que poseen un conjunto de protocolos determinados para tales propósitos.
- ❖ Sistema de eventos: ofrece soporte a la comunicación entre objetos, no importa la naturaleza o tipo.
- ❖ Sistema de scripting: consiste en el modelado de la lógica del juego, que comprende los distintos comportamientos de los objetos, entidades registradas en el mismo, por ejemplo, los NPC.

- ✓ *Audio*: consiste en todo el apartado sonoro del videojuego, los archivos con los diferentes formatos de audio que se utilizan, y la forma en cómo son invocados en el mismo.

- *Subsistemas específicos de juego*: en función del juego a desarrollar, se establecen distintos aspectos específicos referentes al mismo, como por el ejemplo, sistema de cámaras virtuales.

2.2 Juegos Serios Emergentes

2.2.1 Sistemas Emergentes

Antes de abordar el tema, es necesario una breve introducción de lo que es la emergencia. En [20], se señala que en la literatura hay muchas definiciones de la emergencia, que tienen visiones distintas, es decir, que dependen del ámbito en el que se usa.

De manera general, nos interesa en este trabajo definir a la emergencia como la propiedad de un sistema, dada por la aparición de ciertas propiedades y/o estructuras que están en un nivel superior de organización y complejidad, las cuales son impredecibles e irreductibles [20, 21]: "una característica de un sistema complejo se dice que es "emergente" sólo en el caso de que, a pesar de que surge de las propiedades y relaciones que caracterizan a sus componentes, no es ni predecible a partir de, ni reducible a esas características de bajo nivel".

Según [22], como definición relevante al presente trabajo, señala que es un suceso, accidente que sobreviene. Por lo tanto, la emergencia se puede entender, de manera general, como una reacción o fenómeno no planificado que surge ante uno o varios estímulos que se perciben en un ámbito

determinado, posiblemente, dados por la manifestación de diversos agentes, o de manera espontánea, es decir, que se produce sin alguna clase de intervención o estímulo exterior.

En [20], se habla de la definición de un sistema emergente como un sistema cuyo comportamiento final es discontinuo, del comportamiento observado en los niveles inferiores. Además, citan ejemplos como las hormigas en busca de alimento, o las aves que vuelan (comportamiento de enjambre).

En general, un sistema emergente se puede entender como un sistema de tipo complejo, es decir, que está compuesto por varios elementos que están entrelazados, de manera que esos lazos crean información extra no visible anteriormente por el observador, en el que su salida es presentada de forma espontánea o discontinua, lo que conlleva a un comportamiento emergente.

En [20], habla de la existencia de varios tipos de emergencia, entre los cuales están:

• *Con relación a la emergencia débil*: características sistémicas en el nivel "superior", que no se pueden predecir a pesar del conocimiento que se tenga de las características y leyes que rigen los componentes de ese sistema. Por lo tanto, no hay otra opción más que la observación de ése comportamiento emergente.

• *Con relación a la emergencia fuerte*: fenómenos emergentes pueden obtener nuevas capacidades causales que hace posible que los sistemas puedan ejercer una influencia de arriba hacia abajo. También, un sistema fuertemente emergente es aquel en el que los niveles superiores de complejidad poseen características genuinas que están ausentes de las partes constitutivas, no obstante, dichas características se pueden especular, por ejemplo, un determinado sistema de nivel superior, en el que, visto como un sistema emergente fuerte, presenta propiedades específicas que puedan variar por medio de una retroalimentación, de manera que pueda optimizarse, con el fin de mejorar su rendimiento.

2.2.2 Conceptualización de un juego serio emergente

En [23], habla de que un juego emergente se define como un juego que se expande de forma espontánea, es decir, sin leyes que lo gobiernen, adaptándose al usuario final (a los jugadores).

En [24], hablan del juego emergente "Metrópolis", cuya finalidad se plantea de la siguiente manera: Las ciudades se auto-gestionan por decisiones que son tomadas en conjunto por sus habitantes, que vienen siendo los jugadores, no hay un jugador especial (que tenga cierta característica que lo califique de importante). En el juego emergen ciertos patrones urbanísticos, debido a las decisiones antes mencionadas.

Haciendo referencia a los conceptos previos de juegos serios, en conjunto con los juegos emergentes, un juego serio emergente, se puede entender como un juego cuyos protocolos específicos están basados en hechos reales, además que presenta mecanismos de aprendizaje, con el fin de que el jugador pueda aprender al mismo tiempo que juega. Por otro lado, puede expandirse de forma espontánea, es decir, pueden surgir comportamientos emergentes, dando como resultado una variación en la historia o el guion, adaptándose a las necesidades del usuario.

2.2.3 Arquitectura del motor de juegos serios emergentes (MJSE)

La arquitectura del MJSE está basada en [1, 2, 25], la cual fue desarrollada para soportar un JSE en un SaCI [2, 26, 27, 28], y consta de las siguientes capas jerárquicas (ver figura 2.5):

Figura 2.5: Arquitectura del Motor de Juegos Serios Emergentes [25]

- *Núcleo del Motor de Videojuego (NMV)*: es el elemento central del MJSE, en él se encuentran los seis submotores de base para cualquier videojuego, los cuales son: submotor de gráficos (SG), submotor físico (SF), submotor de sonido (SS), submotor de interacción (SI), submotor de video (SV) y submotor de renderización (SR). Más detalles de esos submotores en [3, 29].
- *Subsistemas de Emergencia del Videojuego (SEV) y de Adaptación del Videojuego (SAV)*: que permiten hacer emerger un JSE. En específico, ambos subsistemas usan los siguientes submotores:
  - ✓ Submotor de IA (SIA): se encarga de introducir comportamientos inteligentes en los diferentes componentes del JSE. En general, en este componente se despliegan las diferentes

técnicas usadas de la IA para permitir la emergencia en un JSE. Un ejemplo de ello es la técnica ACO que usa el submotor de Trama Emergente (STE), cuando es invocado por el SEV para definir la primera versión del JSE que se comienza a ejecutar [1, 29].

- ✓ 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 [1, 29]. Para ello, recolecta la información del contexto y realiza la gestión de escenas y eventos, la configuración de subtramas/guiones del juego, entre otras cosas. El STE, en el caso del SEV, permite hacer emerger la primera versión del JSE que se ejecutará, según los objetivos que se deben cumplir con el JSE. A continuación, se muestra en la figura 2.6, el esquema del STE.

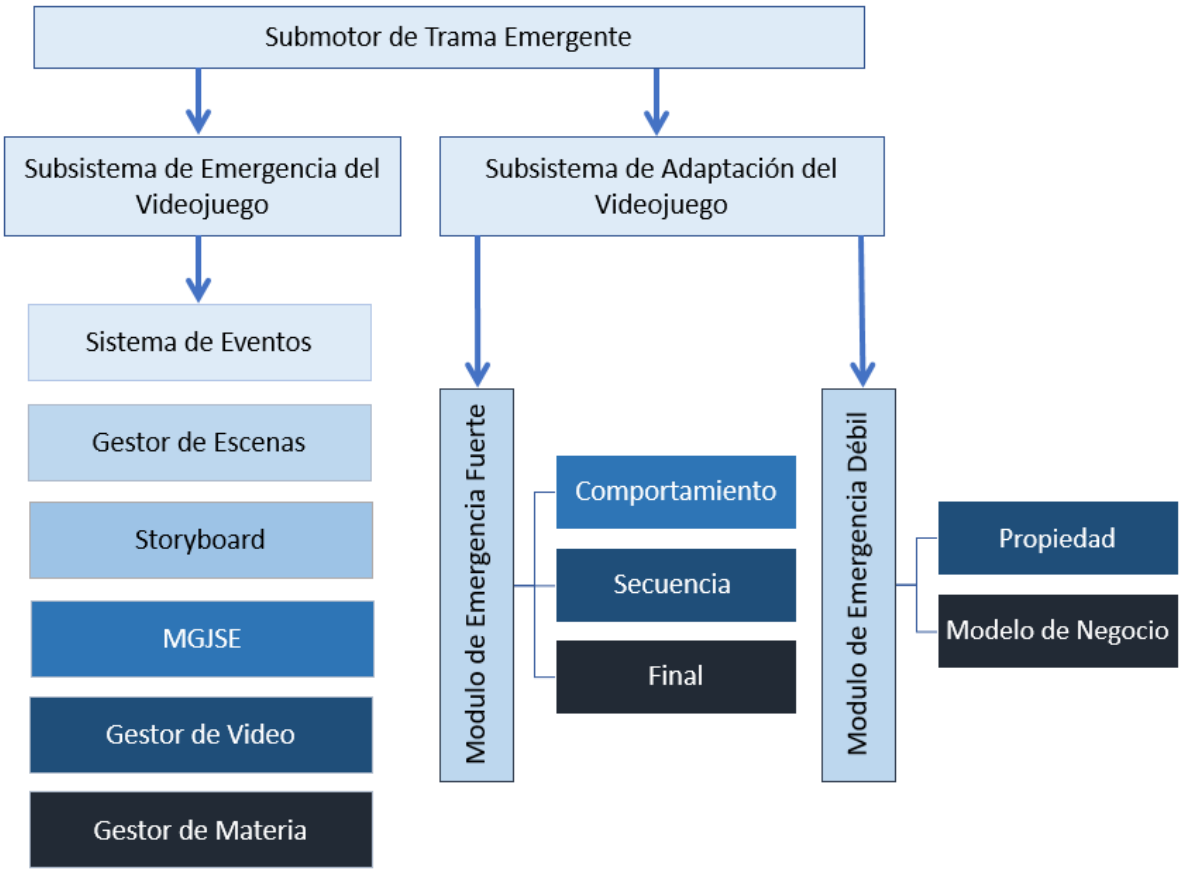

Figura 2.6: Submotor de Trama Emergente [25]

El STE está compuesto por dos subsistemas, el SEV, y el SAV. A continuación, se describen las capas de SEV.

Figura 2.7: Subsistema de Emergencia del Videojuego [25]

- *Gestor de Materia (GM)*: determina la temática que se está tratando en el contexto para, a partir de allí, establecer el objetivo pedagógico que debe cubrir el JSE. Así, crea una emergencia de utilidad, definiendo para qué va a ser utilizado el videojuego.
- *Gestor de Videojuegos (GV)*: busca en repositorios de videojuegos (por ejemplo, edugame, advergame, etc.), subtramas o videojuegos para esa temática. Dichas subtramas/videojuegos son definidos como Recursos de Aprendizaje (RA). Para la búsqueda, compara los metadatos de los RAs en los repositorios, con el definido por el GM. Si no consigue al menos un videojuego muy parecido a lo buscado, llama al módulo siguiente.
- *Módulo de Generación de JSE (MGJSE)*: es el responsable del ensamblaje de un nuevo JSE. En un trabajo previo, se ha definido el MGJSE basado en ACO [1, 29]. El MGJSE tiene imbricadas las funciones de los otros tres componentes del SEV (ver [29] para más detalles).
- *Storyboard (SB)*: se encarga de generar los guiones narrativos o subtramas del JSE.

- *Gestor de Escenas (GE)*: se encarga de generar el ambiente, mundo o entorno, requerido por las tramas del JSE.
- *Sistemas de Eventos (SE)*: se encarga de generar eventos especializados requeridos por el SB, para generar los comportamientos deseados en el JSE.

A continuación, se muestra la figura 2.8, que representa la capa del SAV.

Figura 2.8. Subsistema de Adaptación del Videojuego [25]

Esta capa permite que en un JSE se generen comportamientos emergentes durante el juego, actuando sobre sus características de base. En particular, esta capa permite la adaptación de las características de sus elementos, la emergencia de nuevas estrategias, secuencias de tramas, ambientes y eventos, en el videojuego. En [30] se proponen 6 niveles de emergencia en un JE, que se dividen en dos módulos a saber:

- *Módulo de Emergencia Fuerte*: este módulo está compuesto de tres subcapas que utiliza emergencia fuerte pudiendo manipular el videojuego, de la siguiente forma:
  - ✓ Comportamiento: se generan nuevas tácticas y estrategias en el juego, siguiendo reglas con variantes.
  - ✓ Secuencia: se crean nuevos escenarios, tramas o temáticas en los juegos, cambiando el ambiente y el contexto del juego.
  - ✓ Final: surgimiento de patrones que reflejan resultados finales de los juegos, que han sido adaptados por los jugadores.

- *Módulo de Emergencia Débil*: este módulo está compuesto de dos subcapas que utiliza emergencia débil, pudiendo manipular el videojuego de la siguiente forma:
  - ✓ Propiedad: cambia las características en los objetos.
  - ✓ Modelo de Negocio: por el surgimiento de tiendas, que forman parte de un sistema de negocio, en los juegos.

2.3 Algoritmos Culturales

Según [31], los algoritmos culturales (AC) 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 vista 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 2.9, se puede apreciar cada uno de los componentes de los AC:

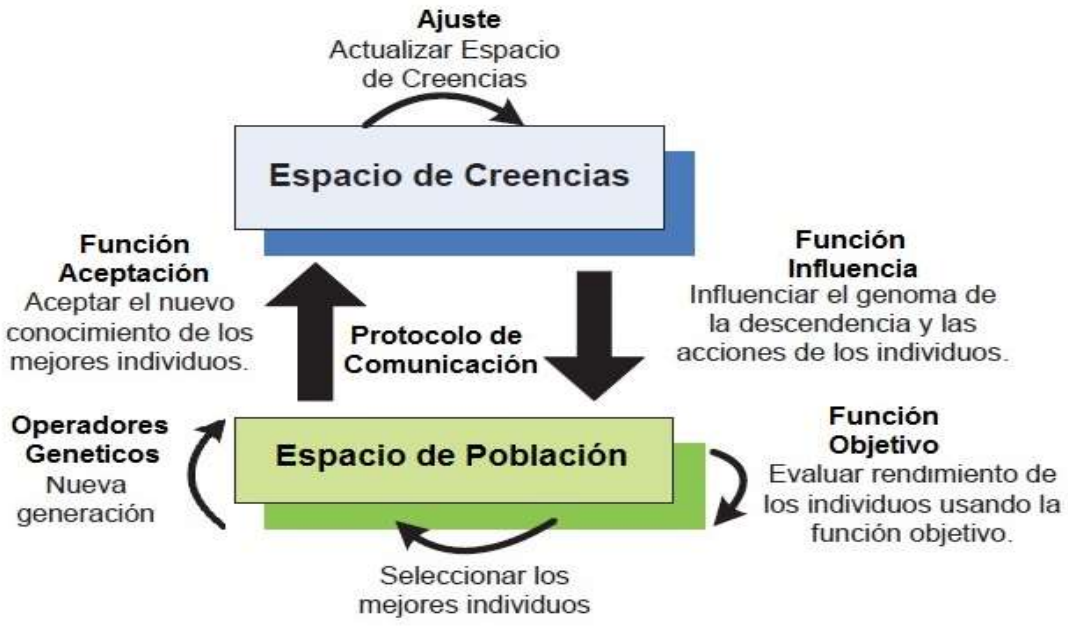

Figura 2.9. Estructura de los algoritmos culturales [31]

En específico, los componentes de un AC son:

- *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:
  - ✓ Función Objetivo: permite evaluar el performance o el desempeño de cada individuo.
  - ✓ Operadores Genéticos: permiten la reproducción, modificación o variación de nuevos individuos en el espacio de la población. Los más comunes son los operadores de cruce y mutación. A continuación, se presentan sus correspondientes descripciones [24]:
    - ❖ *Operador de Cruce*: permiten la unión entre partes de individuos, produciendo su descendencia.
    - ❖ *Operador de Mutación*: modifican una parte de un individuo de manera aleatoria.
- *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:
  - ✓ 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.
  - ✓ Normativo: son comportamientos ideales, rangos de valores idóneos de cada una de las variables, o las gamas de comportamientos aceptables.
  - ✓ 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.
  - ✓ Histórico: son patrones temporales, comportamientos históricos a resaltar, etc.
  - ✓ Topográfico: son patrones espaciales del comportamiento, características del espacio de soluciones, caminos de interés, etc.
- *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:

- ✓ 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.
- ✓ 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.

El macroalgoritmo cultural es parecido al de los algoritmos evolutivos, en el que se va pasando por varias generaciones, en las cuales se va modificando la población con los operadores genéticos y el conocimiento en el espacio de creencias, pero también, se va actualizando el espacio de creencias con nuevo conocimiento. A continuación, se presenta dicho macroalgoritmo:

Inicio

$t = 0$

Iniciar JSE^t *// Iniciar la población*

Iniciar B^t *// Iniciar el espacio de creencias*

Repetir

Evaluar JSE^t *// Evaluar la población*

Ajuste (B^t, *Acepte* (JSE^t)) *// Ajustar el espacio de creencias con el conocimiento de los individuos de la actual población (los seleccionados)*

Variación (JSE^t, *Influencia* (B^t)) *// Variar la población usando los operadores //genéticos (bajo la influencia del espacio de creencias)*

$t = t + 1$

Seleccionar JSE^t *de* $JSE^t - 1$ *// Realizar la selección*

Hasta (haber alcanzado la condición de finalizar)

Fin

Capítulo 3

Diseño e Implementación del SAP

En este capítulo se expone, tanto el diseño como la implementación del SAP, utilizando AC. Se explicarán cada uno de los objetos que conforman tanto el espacio de población como el espacio de creencias, así como también, las funciones de aceptación e influencia. La Figura 3.1 presenta el flujo arquitectónico del SAP, que describe los diferentes componentes del mismo, que serán descritos en las secciones siguientes. Para la codificación del programa, se usó el lenguaje de programación C++, con su biblioteca estándar de plantillas (por sus siglas en inglés, STD), en conjunto con el framework Qt en su versión 5.7.0, para definir la interfaz gráfica.

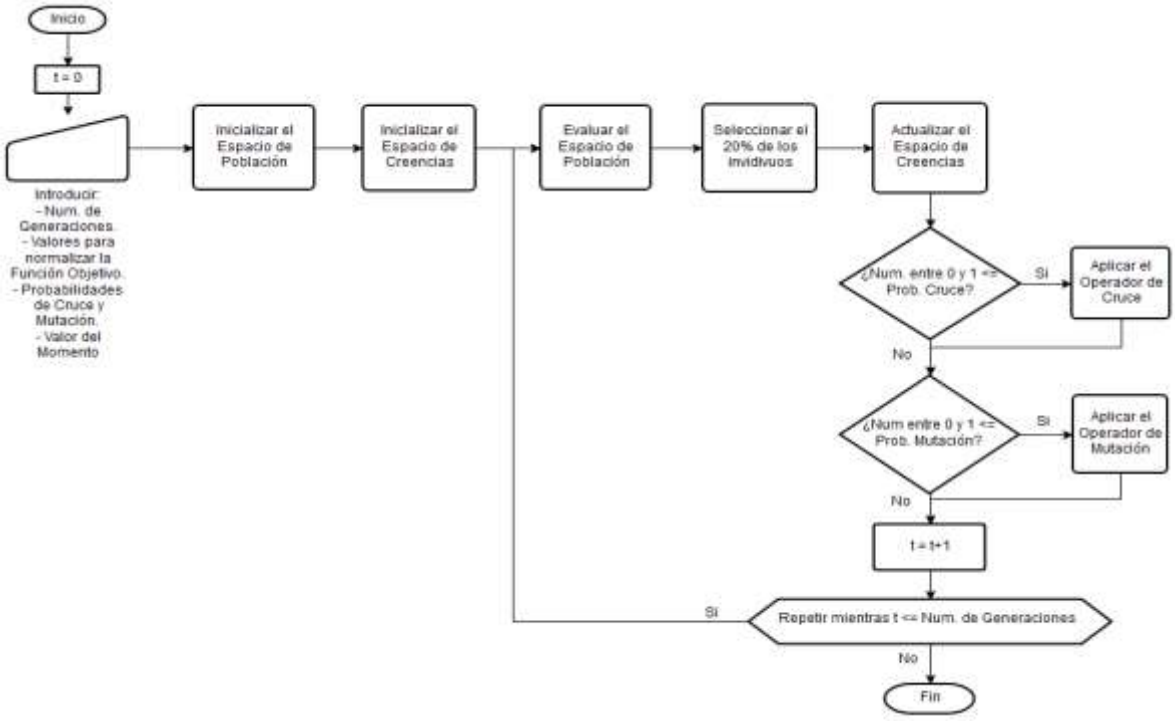

Figura 3.1. Diagrama de Flujo del SAP

3.1 Diseño del espacio de población

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

En la Tabla 3.1: Pj son los valores de los *j-ésimo* parámetros usados en ese juego en la *i-ésima* vez que se jugó, y FOi 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

En la figura 3.2, se muestra el diagrama de clases del tipo de dato abstracto (TDA) llamado Individuo.

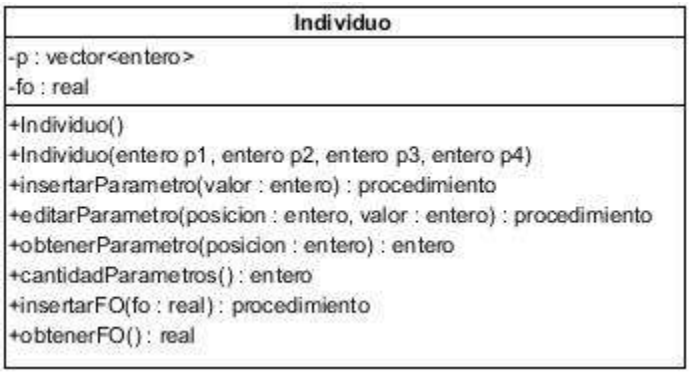

Figura 3.2. Diagrama de la clase Individuo

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:

$$FO_i = a * PA_i - b * LE_i \qquad (1)$$

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 (1).

3.2 Diseño del espacio de creencias

Según la teoría de AC se tienen cinco conocimientos, pero en este trabajo se explicarán *cuatro*, de los cuales solo *dos* serán usados para los experimentos. Los *cuatro* a explicar 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.

- Conocimiento Situacional: contiene ejemplos de éxitos de los JSE. En particular, los valores Vj de los parámetros Pi en dichos juegos, con sus índices de ocurrencia IOj a través de los diferentes juegos. Además, la calidad promedio de ese valor de parámetro Cj, 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	Vj	IO_j	C_j

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 3.3).

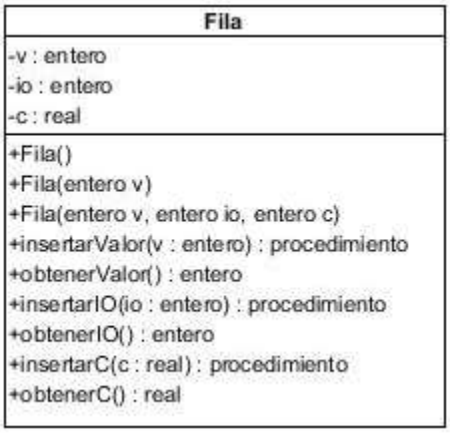

Figura 3.3. Diagrama de la clase Fila

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 3.4).

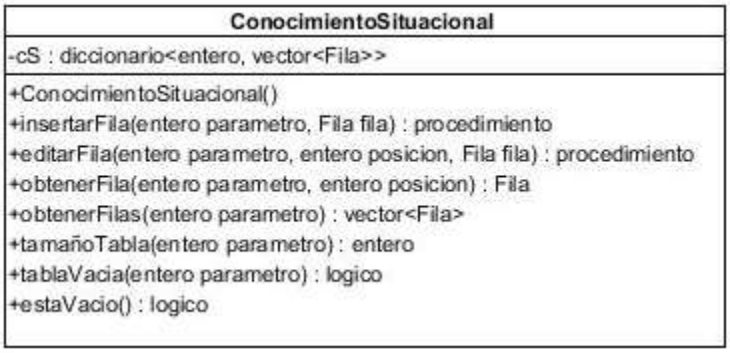

Figura 3.4. Diagrama de la clase ConocimientoSituacional

- Conocimiento Normativo: define los rangos de valores idóneos de cada uno de los parámetros del JS. En la tabla 3.4, LI y LS son los límites inferiores y superiores de cada parámetro Pi.

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

La implementación consiste en definir dos TDA: Limite y ConocimientoNormativo, ambos se muestran en las figuras 3.5 y 3.6, respectivamente. El TDA Limite describe la Tabla 3.4 para un parámetro dado, y el TDA ConocimientoNormativo la lista de parámetros con sus límites.

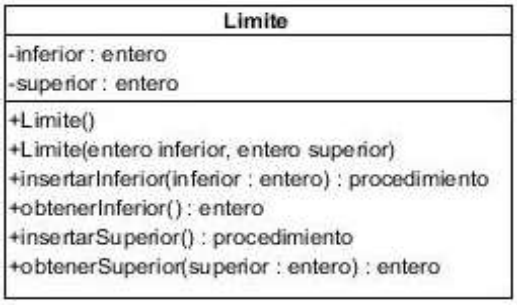

Figura 3.5. Diagrama de la clase Limite

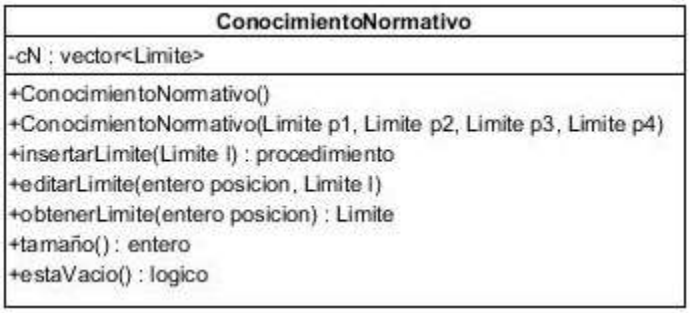

Figura 3.6. Diagrama de la clase ConocimientoNormativo

- Conocimiento de Dominio: en nuestro caso, indica los contextos donde se puede aplicar el JS, con los valores de parámetro adecuados para cada uno de ellos. En la tabla 3.5, Di representa el dominio i para el conjunto de parámetros Pj de un JSE, y la función de calidad (FCi) es calculada como el promedio de calidad de las veces que se ha usado el JSE en ese dominio, determinada usando el FO de los individuos en ese dominio.

Tabla 3.5. Representación del Conocimiento Dominio

Di	FC_i	P^1	P^2	...	P^u
		valor ideal	valor ideal		valor ideal

- 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 Pj para que el evento k ocurra, y la función (FEk) 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 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.

- Función de Aceptación: según Reynolds [6, 32], 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, $\forall$ j = 1, 2, … u; en el *conocimiento normativo*, los valores de LIj y LSj, $\forall$ j = 1, 2, … u. En el caso del *conocimiento dominio*, se puede actualizar porque hay un nuevo dominio Ds, o porque los valores ideales de los parámetros han cambiado, o porque FCs ha cambiado para los actuales valores ideales. Finalmente, en el *conocimiento histórico*, pueden cambiar los valores ideales de los parámetros, o el valor FEk para los actuales valores ideales.

En el *conocimiento situacional*, se actualiza de la siguiente manera: IOj es actualizado mediante la ecuación (2).

$$IO_j = NO_j + IO_j \qquad (2)$$

Donde, NO_j es el nuevo número de ocurrencia del valor Vj en la actual generación. A su vez, C_j también se actualiza con la ecuación (3):

$$C_j = C_j * \bar{m} + \bar{C_J} * m \qquad (3)$$

Donde, $\bar{m}$ es el complemento del momento, es decir, $(1 - m)$, $\bar{C_J}$ es el promedio del valor C_j de todos los individuos dentro del 20%, provenientes de la población actual con el valor de Vj. Finalmente, el momento m viene dado por la ecuación (4):

$$m = \frac{\mu}{t} \qquad (4)$$

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 c*onocimiento normativo*, los valores de LIj y LSj se actualizan si en la actual población, alguno de sus individuos tiene un valor superior a LSj o inferior a LIj. 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 Di, se actualiza la matriz de la tabla IV 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 (2). 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:

$$FC_i = FC_i * \bar{m} + \overline{FC_\iota} * m \qquad (5)$$

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 Di.

Finalmente, en el *conocimiento histórico,* si los valores ideales de los parámetros (Pi) para un evento k deben cambiar, simplemente se sustituyen en la tabla V. En este caso, los valores ideales para un evento k son los que maximizan la ecuación (2). Ahora bien, si solo se debe actualizar el valor de FE_i de los actuales valores ideales, se debe usar la siguiente ecuación:

$$FE_i = FE_i * \bar{m} + \overline{FE_\iota} * m \qquad (6)$$

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.

- Función de Influencia: se usa el conocimiento almacenado en el espacio de creencias para realizar operaciones de mutación guiada en el espacio de población. En general, se usan los siguientes operadores genéticos:
  - ✓ *Operador de cruce*: se realiza un cruce del material genético del padre y de la madre (dos individuos) por punto sencillo, con una probabilidad definida por el usuario. Consiste en elegir un valor aleatorio entre 1 y n parámetros que representa el punto, y después se realiza el *cruce*. Para ello, se selecciona el conjunto de parámetros desde 1 hasta el punto elegido de la madre; entonces, dicho conjunto se usa para definir los primeros genes del hijo (el nuevo individuo); finalmente, se selecciona el conjunto de parámetros restantes del padre para completar el material genético del hijo. Por ejemplo, ver la Fig. 3.6:

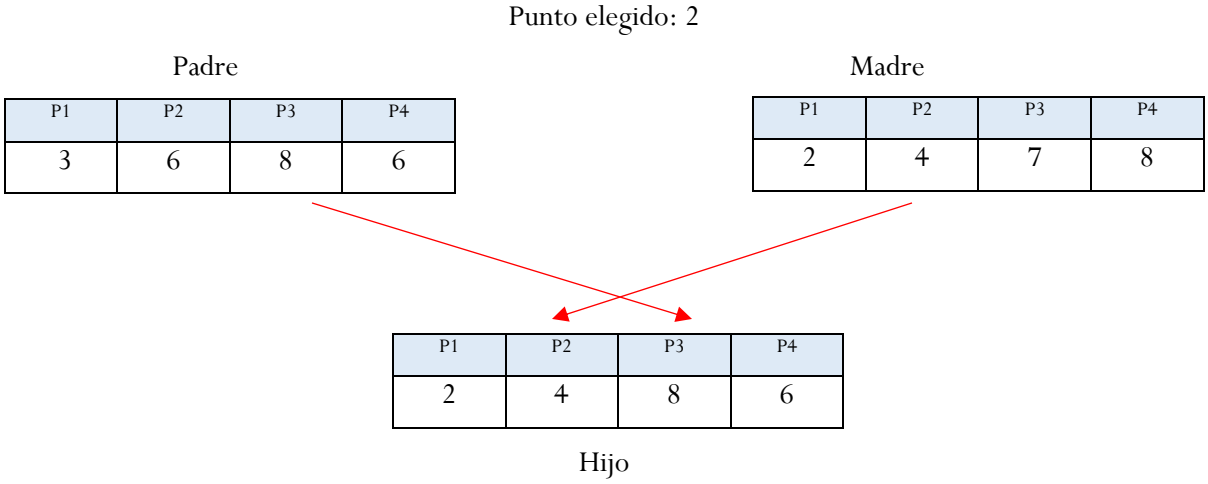

Figura 3.7. Ejemplo del operador de cruce

En caso de haber un solo individuo aceptado tras el proceso de selección, no se aplicará dicho operador, así como también, si el conjunto de parámetros es menor que 2.

- ✓ *Operador de mutación*: los individuos (con excepción de los padres) tienen la probabilidad de mutar un parámetro Pi, con un valor de probabilidad definida por el usuario, tal que el valor de mutación es generado de forma aleatoria en el rango del dominio de esa variable. Así, en este caso, el *conocimiento normativo* tendrá un comportamiento parecido a la mutación clásica.
- ✓ *Operador de mutación guiada.* El *conocimiento situacional* permite una mutación directa a los valores Vj de un Pi dado, dichos valores se determinan por su mejor calidad Cj. Si se usa el *conocimiento normativo*, permite reajustar el valor de Vj en Pi, generando un valor aleatorio que esté dentro de los límites (rango) definidos por dicho conocimiento. El *conocimiento de dominio* usa los valores ideales de Pj para el dominio en el que se quiere diseñar el JSE, y *el conocimiento histórico* usa los valores ideales de Pj, según el evento k que se desee que aparezca en el JSE.

En la figura 3.8, se muestra un ejemplo de mutación dirigida por el conocimiento situacional. A su vez, en la figura 3.9 se muestra un ejemplo de mutación guiada por el conocimiento normativo.

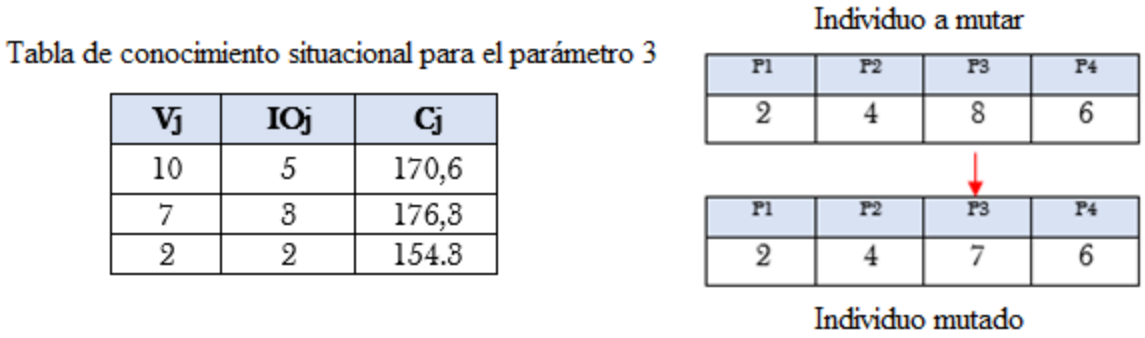

Figura 3.8. Ejemplo de mutación dirigida por conocimiento situacional

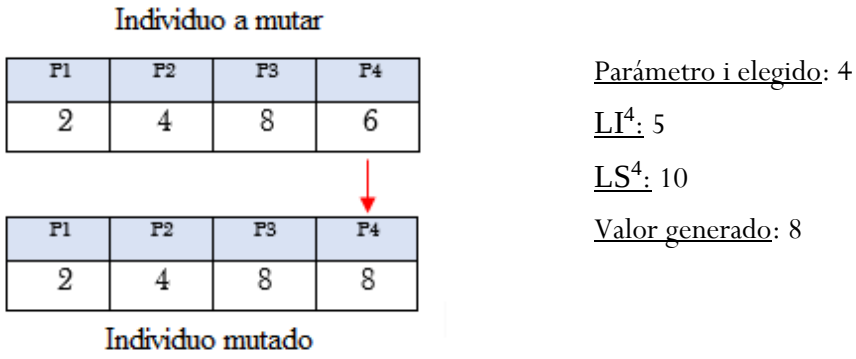

Figura 3.9. Ejemplo de mutación clásica

3.4 Macro-Algoritmo

Recordando el macro-algoritmo del AC visto en el capítulo anterior, junto con la información previamente observada, se expone a continuación el nuevo macro-algoritmo del mismo, adaptado al presente trabajo.

3.4.1 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 (cs y cn) 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.

3.4.2 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.

3.4.3 Ciclo Repetitivo

Por cada iteración, ocurre la siguiente secuencia:

1. Evaluación de la población: cada individuo es evaluado por medio de la ec. (1).
2. 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.
3. 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%.
4. 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).
5. 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.
6. Aplicación del operador de mutación: requiere de la nueva población y de la probabilidad de mutación, para generar nuevos individuos.

3.4.4 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
    // Definición de los objetos
    t <- 0, numGeneraciones, a, b, probCruce, probMutacion, constanteMomento
    vector<Individuo> EspaciodePoblacion
    ConocimientoNormativo cn
    ConocimientoSituacional cs
    // Inicialización
    Introducir(numGeneraciones, a, b, probCruce, probMutacion, constanteMomento)
    Inicializar(EspaciodePoblacion)
    Inicializar(cn)
    Hacer
        // Evaluar cada individuo usando la FO
        EspaciodePoblacion <- Evaluar(EspaciodePoblacion, a, b)
        // Ordenar la población de mayor a menor valor de desempeño.
        EspaciodePoblacion <- Ordenar(EspaciodePoblacion, MayoraMenor)
        // Seleccionar el 20% de los individuos con su mejor desempeño
        Aceptados <- Seleccionar(EspaciodePoblacion)
        // Actualizar el conocimiento normativo
        Actualizar(cn, Aceptados)
        // Actualizar el conocimiento situacional
        Actualizar(cs, Aceptados, t, constanteMomento)
        // Aplicar el operador de cruce
        EspaciodePoblacion <- Cruce(EspaciodePoblacion, Aceptados, probCruce)
        // Aplicar el operador de mutación
        EspaciodePoblacion <- Mutacion(EspaciodePoblacion, probMutacion)
        t = t + 1
    Mientras que (t ≤ numGeneraciones)
Fin
```

Capítulo 4

Análisis del comportamiento del SAP

En éste capítulo se definen todas las pruebas que se le hicieron al sistema, con el objetivo de evaluar su calidad. Además, se describen dos casos de estudio en un SaCI.

4.1 Experimentos a realizar

Los experimentos a realizar son:

- Pruebas del prototipo: tiene como objetivo determinar los valores adecuados de los diferentes parámetros del SAP (en particular, del algoritmo cultural), los cuales serán los valores por defecto para el correcto funcionamiento del mismo. En particular, los valores a optimizar son la probabilidad de usar los operadores, el número de generaciones, el tamaño de la población inicial y el valor de μ.
- Pruebas de la calidad del JSE generado por el sistema: tiene como objetivo evaluar la calidad del JSE final generado por el SAP, como resultado de la optimización de sus parámetros.

4.1.1 Pruebas del prototipo:

JSE 1: Videojuego de domino "Combinado"

En primera instancia, se usa el videojuego de domino "Combinado", obtenido del repositorio Agrega (http://agrega.educacion.es) (ver figura 4.1).

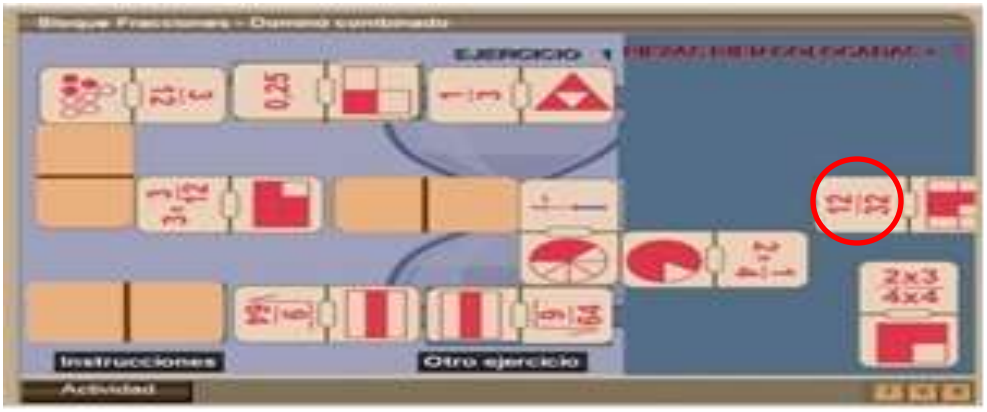

Figura 4.1. El videojuego de domino "Combinado"

El videojuego de domino "Combinado" consiste en que cada uno de los dominós tiene dos zonas opuestas entre sí, ya sea izquierda (arriba) o derecha (abajo). Estas zonas van a tener: una figura, fracción, porcentaje, etc., que va a dar un resultado de fracciones matemáticas. Esas zonas deben ser iguales entre dos dominós adyacentes en el tablero, y al jugar un domino la figura debe ser igual al espacio vacío adyacente a alguna otra pieza de domino que este en el tablero. Por ejemplo, en la figura 4.1, el círculo rojo señala la zona izquierda del domino que tiene como valor 12/32, este debe ser colocado en una zona vacía que tenga el mismo resultado en uno de los dominós en el tablero, como por ejemplo $\sqrt{}$ (9/64), que es 0.375. Este último se encuentra en el espacio vacío abajo a la izquierda.

En general, este JSE tiene 3 parámetros que se pueden optimizar:

- P1 = Fichas (dominós) del jugador
- P2 = Fichas en el tablero
- P3 = Son las operaciones de fracciones posibles, dependerá del perfil del jugador. Por ejemplo:
  - ✓ Suma y resta de fracciones con denominador común.
  - ✓ Suma y resta de fracciones con distinto denominador.
  - ✓ Multiplicación de fracciones.
  - ✓ División de fracciones.
  - ✓ Raíz cuadrada de fracciones.

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

P^1		P^2		P^3	
LI^1: 1	LS^1: 5	LI^2: 1	LS^2: 5	LI^3: 1	LS^3: 5

Tabla 4.1. Conocimiento normativo del videojuego de domino "Combinado"

A continuación, se muestran los valores iniciales de los distintos parámetros del SAP, introducidos por el usuario.

Individuos	Generaciones	Prob. Cruce	Prob. Mutación	μ	a	b	PA	LE
10	20	0.1	0.1	0.1	100	100	[0, 1]	[0, 1]

Tabla 4.2. Valores iniciales del SAP para el estudio del videojuego de domino "Combinado".

Con estos valores se procede a realizar la primera simulación, para determinar los valores idóneos del SAP. Un ejemplo de los resultados obtenidos se muestra en la figura 4.2. Ella muestra solamente el mejor individuo en cada generación, según su valor de FO. En la gráfica se puede ver que después de 11 generaciones no mejoran los resultados del mejor individuo.

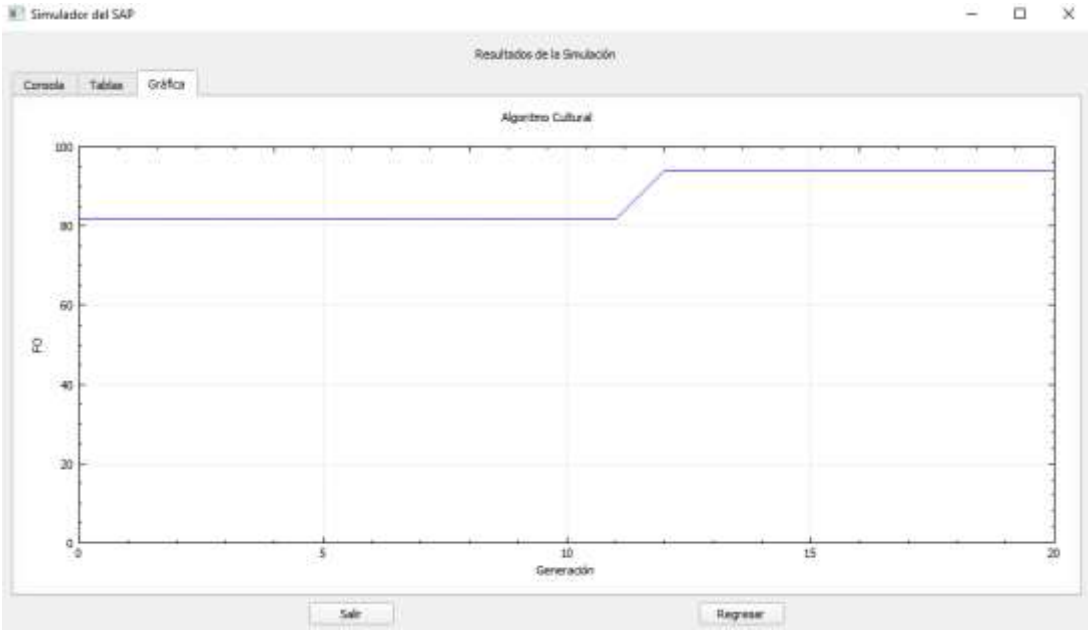

Figura 4.2. Gráfica del mejor individuo en cada generación para el videojuego "Combinado"

Las tablas del conocimiento situacional, normativo y de los mejores individuos, se muestra en la figura 4.3.

Figura 4.3. Tablas del conocimiento situacional, normativo y de los mejores individuos para el videojuego "Combinado"

Se puede ver como conocimiento situacional a los valores de los parámetros que provienen de los individuos aceptados, cada uno con sus índices de ocurrencia y calidad promedio, cuya columna se ordenó de mayor a menor para apreciar el mejor valor para realizar la mutación dirigida. Como conocimiento normativo tenemos los límites inferior y superior para parámetro, siguiendo el formato: $[[LI^1, LS^1] [LI^2, LS^2] [LI^3, LS^3]]$ por cada generación. La figura también muestra la tabla de los mejores individuos en cada generación. El individuo se presenta con el formato: [P1, P2, P3], en conjunto con el valor de su FO, cuya columna se ordenó de mayor a menor para apreciar el cambio final de la curva presentada en la figura.

La consola se muestra en la figura 4.4, y registra los datos de entrada, en conjunto con todas las operaciones del SAP. Ella muestra el comportamiento del SAP a través de las generaciones.

Figura 4.4. Resultados del SAP en consola, para el videojuego de domino "Combinado"

En la figura 4.5 se muestra la "Nueva Población (después del cruce)", pero en este caso es la última generación para esta simulación, por lo que se convierte en la población final que posteriormente se explica con detalle.

```
  Conocimiento situacional:
Parametro       Filas/Parametro
1               [[5, 30, 83.022] [4, 12, 65.2708]]
2               [[4, 25, 93.8922] [3, 12, 65.2708] [2, 5, 82.9677]]
3               [[5, 16, 81.6804] [2, 26, 83.022]]

  Aplicando el operador de cruce:
4. El punto de cruce: [2]
  EL Padre: 5 4 2 | 93.8922;      La Madre: 5 2 2 | 82.9677
                El Hijo es: 5 2 2 | 0

  Nueva Población (después del cruce):
1: 5 4 2 | 93.8922
2: 5 2 2 | 82.9677
3: 5 4 5 | 81.6804
4: 5 2 2 | 0
5: 1 2 2 | 27.8294
6: 4 4 5 | 22.8884
7: 4 4 2 | 7.94815
8: 5 4 2 | 7.5813
9: 5 4 2 | -29.2948
10: 5 4 2 | -30.8359

  Aplicando el operador de mutación:
No hay variación de la población.

=== FIN DEL ALGORITMO CULTURAL ===
```

Figura 4.5. Población final de la primera simulación del SAP en consola, para el videojuego de domino "Combinado"

La población final es el resultado de variar el espacio de población con los operadores genéticos a través de las generaciones (en este caso 20 iteraciones/generaciones), tomando en cuenta sus probabilidades de ocurrencia, a partir de los valores introducidos por el usuario. Esa población final se detalla en la tabla 4.3.

#	P1	P2	P3	FO
1	5	4	2	93.8922
2	5	2	2	82.9677
3	5	4	5	81.6804
4	5	2	2	0
5	1	2	2	27.8294
6	4	4	5	22.8884
7	4	4	2	7.94815
8	5	4	2	7.5813
9	5	4	2	-29.2948
10	5	4	2	-30.8359

Tabla 4.3. Población final de la primera simulación del SAP, para el videojuego de domino "Combinado"

A continuación, se hace un análisis de sensibilidad de los parámetros del SAP, con el fin de conseguir los valores finales por defecto de cada uno. La primera prueba es variando el número de individuos en la población final (ver Tabla 4.4).

Número de individuos	FO mejor individuo
30	88.7822
50	96.6871
70	97.378
80	98.1961
90	94.7378

Tabla 4.4. Variación del número de individuos para el videojuego de domino "Combinado"

Los resultados nos indican que hay un cierto incremento en la FO del mejor individuo, si se considera una población con más de 50 individuos. Pero dicho incremento deja de ocurrir si el tamaño de la población sobrepasa los 90 individuos. Así, el número de individuos queda establecido en 80.

La segunda prueba es variando la probabilidad de cruce (ver Tabla 4.5).

Pb cruce	FO mejor individuo
0.3	94.7437
0.5	96.6871
0.7	94.6158
0.9	94.6158
1.0	94.465

Tabla 4.5. Variación de la probabilidad de cruce para el videojuego de domino "Combinado"

Los resultados nos indican que con una probabilidad de cruce del 50%, se obtiene un mejor valor de FO, aunque la variación es mínima. Así, la probabilidad de cruce se establece en 0.5, por ser con el valor que se consiguió el FO más alto.

La tercera prueba es variando la probabilidad de mutación (ver Tabla 4.6).

Pb. Mutación	FO mejor individuo
0.3	94.4578
0.5	94.6158
0.7	96.2725
0.9	94.465
1.0	97.378

Tabla 4.6. Variación de la probabilidad de mutación para el videojuego de domino "Combinado"

Los resultados nos indican que con una probabilidad de mutación del 100%, se obtiene un mejor valor de FO, pero igual que antes, con una variación muy leve entre los valores. Así, la probabilidad de mutación se establece en 1.0, por ser con el valor que se consiguió el FO más alto.

La cuarta prueba es variando el número de generaciones (ver tabla 4.7).

Número de generaciones	FO mejor individuo
30	97.378
50	97.378
100	97.378
200	98.5987

Tabla 4.7. Variación del número de generaciones para el videojuego de domino "Combinado"

Los resultados nos indican que con 200 generaciones se obtiene el mejor valor de FO, pero de nuevo, con muy poca variación entre los valores. Así, el número de generaciones se establece en 200.

La quinta prueba es variando μ.

μ	FO mejor individuo
0.3	99.0617
0.5	99.3492
0.7	99.5503
0.8	99.5503
0.9	98.7308
1.0	98.7308

Tabla 4.8. Variación de μ para el videojuego de domino "Combinado"

Los resultados nos indican un valor de la FO muy parecido para los diferentes valores de μ. Así, el valor μ queda establecido en 0.7, por ser con el valor que se consiguió el FO más alto.

JSE 2: "Polígonos con el Tangram",

El segundo experimento es con un JSE denominado "Polígonos con el Tangram", obtenido del repositorio Agrega (http://agrega.educacion.es) (ver figura 4.6).

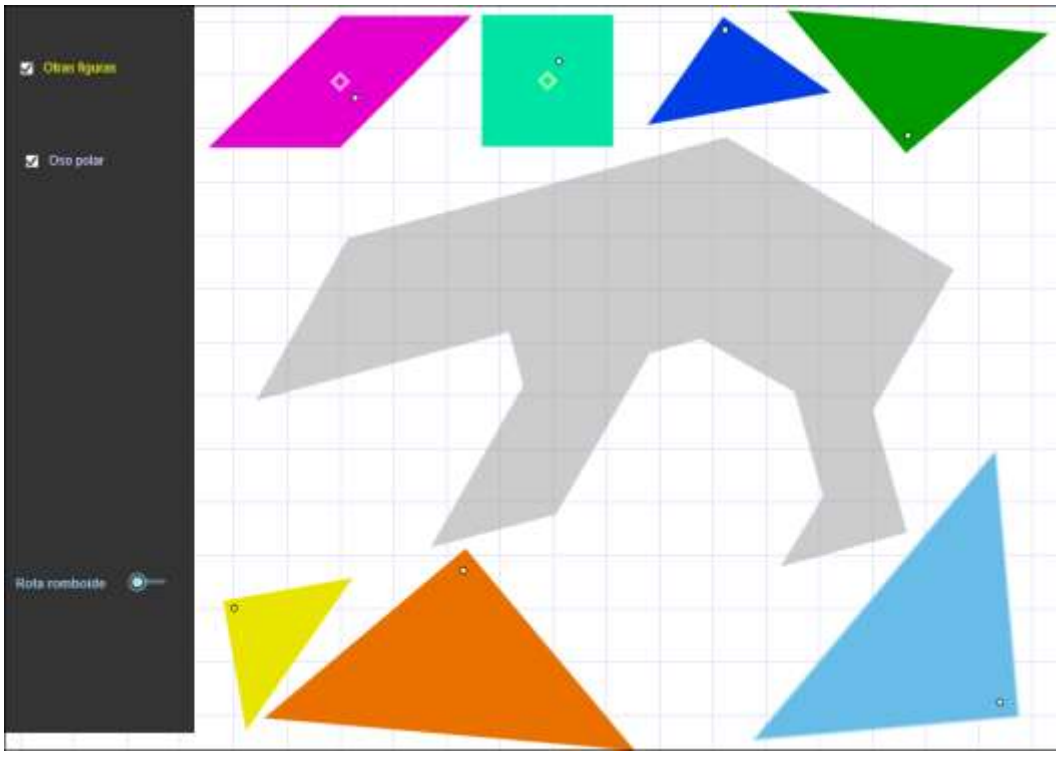

Figura 4.6. Videojuego "Polígonos con el Tangram".

El videojuego "Polígonos con el Tangram" consiste en resolver el *Tangram*, el cual es un juego de rompecabezas (o en inglés: puzzle) de origen chino compuesto por siete piezas, obtenidas por división de 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, trapecio isósceles, trapecio rectángulo 1, trapecio rectángulo 2, pentágono 1, pentágono 2, hexágono 1, hexágono 2, hexágono 3, hexágono 4). Dicha figura presenta cualquiera de las siguientes formas: hombre sentado, oso polar (como se muestra en la figura 4.6), perrito, velero, cisne, flecha negra y flecha blanca.

Los parámetros de ese JSE a considerar son:

- P1 = Figuras con las que cuenta el jugador

- P2 = Polígonos convexos a formar

Los límites iniciales para cada parámetro, requeridos para inicializar el conocimiento normativo, se muestran en la tabla 4.9.

P^1		P^2	
LI^1: 1	LS^1: 7	LI^2: 1	LS^2: 13

Tabla 4.9. Conocimiento normativo del videojuego "Polígonos con el Tangram"

Los valores iniciales para empezar con la simulación están en la tabla 4.10.

Individuos	Generaciones	Prob. Cruce	Prob. Mutación	μ	a	b	PA	LE
10	20	0.1	0.1	0.1	100	100	[0, 1]	[0, 1]

Tabla 4.10. Valores iniciales del SAP para el videojuego "Polígonos con el Tangram".

Después de realizar la primera simulación, se muestra la gráfica del mejor individuo en cada generación en la figura 4.7.

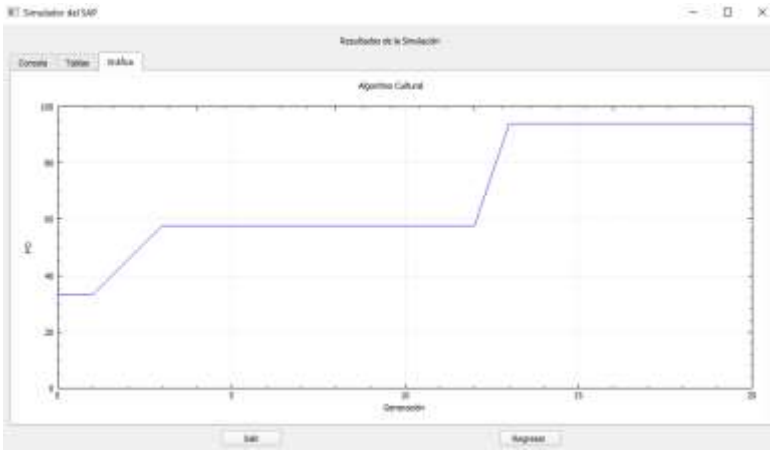

Figura 4.7. Gráfica del mejor individuo para el videojuego "Polígonos con el Tangram".

En la figura 4.8 se muestran las tablas del conocimiento situacional, normativo y de los mejores individuos, según la misma descripción hecha para la figura 4.3.

Figura 4.8. Tablas del conocimiento situacional, normativo y de los mejores individuos para el videojuego "Polígonos con el Tangram".

En la figura 4.8 se observa como el conocimiento situacional de ambos parámetros influyen en el mejor individuo, el cual aparece a partir de la generación 13, manteniéndose así hasta alcanzar la generación 20, como sucede en la figura 4.7.

Ahora pasamos a hacer el análisis de sensibilidad de los parámetros del SAP. La primera prueba es variando el número de individuos en la población final (ver Tabla 4.11).

Número de individuos	FO mejor individuo
30	77.3442
50	90.6724
70	86.7574
80	93.8339
100	90.7341

Tabla 4.11. Variación del número de individuos para el videojuego "Polígonos con el Tangram"

Los resultados nos indican un mejor valor FO para 80 individuos, por lo tanto, el número de individuos queda establecido en 80.

La segunda prueba es variando la probabilidad de cruce (ver Tabla 4.12).

Pb cruce	FO mejor individuo
0.3	95.2848
0.5	96.6373
0.7	92.4736
0.9	96.0042
1.0	96.207

Tabla 4.12. Variación de la probabilidad de cruce para el videojuego "Polígonos con el Tangram"

Los resultados nos indican que con una probabilidad de cruce del 50%, se obtiene un mejor valor de FO. Así, la probabilidad de cruce se establece en 0.5, por ser con el valor que se consiguió el FO más alto.

La tercera prueba es variando la probabilidad de mutación (ver Tabla 4.13).

Pb. Mutación	FO mejor individuo
0.3	96.6871
0.5	96.207
0.7	96.2725
0.9	94.6158
1.0	97.7206

Tabla 4.13. Variación de la probabilidad de mutación para el videojuego "Polígonos con el Tangram"

Los resultados nos indican que con una probabilidad de mutación del 100%, 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 4.14).

Número de generaciones	FO mejor individuo
30	97.7206
50	97.7206
100	99.1939
200	99.8165

Tabla 4.14. Variación del número de generaciones para el videojuego "Polígonos con el Tangram"

Los resultados nos indican que, a 200 generaciones, mejor valor de FO se obtiene, por lo tanto, el número de generaciones se establece en 200.

La quinta prueba es variando μ (ver Tabla 4.15).

μ	FO 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

Tabla 4.15. Variación de μ para el videojuego "Polígonos con el Tangram"

Los resultados nos indican un incremento del valor de la FO, con μ entre 0.3 y 0.7, luego tras estabilizarse con μ entre 0.7 y 0.9, empieza a disminuir. Así, el valor μ queda establecido en 0.7.

Ahora analizamos el comportamiento de los parámetros en los dos JSE, para determinar los valores por defecto, para el correcto funcionamiento del prototipo del SAP.

En 26 simulaciones por cada JSE, se encuentra una afinidad en el valor de los parámetros del SAP para obtener los mejores valores de FO durante los experimentos, usando los "valores óptimos" mostrados en la tabla 4.16.

Individuos	Generaciones	Prob. Cruce	Prob. Mutación	Valor μ	a	b	PA	LE
80	200	0.5	1.0	0.7	100	100	[0, 1]	[0, 1]

Tabla 4.16. Valores óptimos del SAP

4.1.2 Pruebas de la calidad del JSE generado

Para estas pruebas, las reglas de calidad que se definen son:

- Lapso de tiempo (segundos). Se parte de la hipótesis que se establece un límite de tiempo para cumplir con los objetivos del JSE elegido. Inicialmente, se definen 300 segundos (5 minutos), de manera que, si se sobrepasa ese valor, hasta un máximo de 600 segundos (10 minutos), significa que el jugador tardó demasiado en cumplir con los objetivos. Si se llega al máximo, se supone que simplemente no los alcanzo.
- Alcance de los objetivos (puntuación acumulada). Se parte de la hipótesis de que 1000 puntos es el valor cuando se cumplen todos los objetivos del JSE, mientras que con un valor alrededor de 500 puntos se puede considerar la mayoría de los objetivos alcanzados.

Usando los valores por defecto obtenidos en la sección anterior, con los valores de a = 1000 y b = 600 para normalizar las variables PA y LE que componen la fórmula de la FO, se procede a realizar la prueba de la calidad de los videojuegos presentados en las pruebas anteriores.

JSE 1: Videojuego de domino "Combinado"

A continuación, se presentan los resultados de la prueba de calidad para el videojuego de domino "Combinado" (ver figura 4.9).

Figura 4.9. Resultados de la prueba de calidad para el videojuego de domino "Combinado"

En la figura 4.9 se observa el mejor individuo, con una FO de 984.314, en la Generación 105, con los siguientes parámetros:

- P1 = 1 (fichas del jugador)
- P2 = 4 (fichas en el tablero)
- P3 = 4 (operaciones de fracciones)

El valor de la FO fue calculado de la siguiente manera:

$$FO = 1000 * 0{,}986282 - 600 * 0{,}00328135 = 984.314$$

PA normalizado es = 986.282 ≈ 986.3 puntos.

LE normalizado es = 1.96881 ≈ 2 segundos.

Como se puede observar en la FO, con esos parámetros en el JSE los estudiantes pueden cumplir con los dos criterios de calidad: duración del juego y objetivos alcanzados.

JSE 2: "Polígonos con el Tangram",

En esta sección se presentan los resultados de la prueba de calidad para el JSE "Polígonos con el Tangram" (ver figura 4.10).

Figura 4.10. Resultados de la prueba de calidad para el videojuego "Polígonos con el Tangram"

En la figura 4.10 se observa el mejor individuo, con una FO de 986.486, en la Generación 110, con los siguientes parámetros:

- P1 = 7 (figuras del jugador)
- P2 = 12 (polígonos convexos)

El valor de la FO fue calculado de la siguiente manera:

$$FO = 1000 * 0.99674 - 600 * 0.0170908333333333 = 986.4855$$

PA normalizado es = 996.74 ≈ 997 puntos.

LE normalizado es = 10.2545499999999998 ≈ 10 segundos.

De nuevo, como se puede observar en la FO, con esos parámetros en el JSE los estudiantes pueden cumplir con los dos criterios de calidad: duración del juego y objetivos alcanzados.

4.1.3 Análisis de los resultados:

Después de haber realizado 54 simulaciones con los dos JSEs, se concluye lo siguiente:

- Si existe una gran cantidad de individuos en la población, entonces es necesario definir un número de generaciones lo suficientemente grande para garantizar que los resultados sean satisfactorios (ver tablas 4.7 y 4.14).
- Con respecto a las probabilidades de cruce y mutación: al considerarse el 0,5 para el caso del cruce y 1 para la mutación, se obtendrán los mejores resultados (ver tablas 4.5, 4.6, 4.12 y 4.13). No obstante, si son bajas, entonces es necesario definir un número de generaciones muy grande para obtener los mejores resultados (ver figura 4.2).
- Se ha observado que los valores de los límites que componen el conocimiento normativo para cada videojuego, no cambian a través de las generaciones (ver figuras 4.3 y 4.8). Esto se debe a que los límites que se definieron al inicio para dichos parámetros corresponden a los límites del dominio de los mismos. Este conocimiento es importante optimizar cuando no se conocen dichos límites.
- El valor μ quedó definido en 0.7. Se mostró a través de los experimentos que al ser más grande (> 0.7) hay una disminución del valor FO del mejor individuo (ver tablas 4.8 y 4.15).

4.2 Caso de estudio

A continuación, se procede a demostrar el funcionamiento del SAP en el contexto de SACI.

4.2.1 Contexto de aplicación: SaCI

Se considera un SaCI como el propuesto en [26, 27], tal que todos sus componentes son modelados usando el paradigma de Sistemas Multiagentes (SMA); es decir, sus dispositivos de hardware (pizarra inteligente, laptop, tableta, smartphone, etc.) y de software (Entorno Virtual de Aprendizaje, Sistema Académico, etc.) son definidos como agentes. SaCI utiliza un middleware reflexivo autonómico para entornos de aprendizaje inteligente en la nube, llamado AmICL, propuesto en [26, 27, 28]. Este middleware reflexivo

autonómico está también basado en SMAs, y posibilita el despliegue de las diferentes comunidades de agentes de SaCI.

En particular, uno de esos agentes es el Agente de Juegos Serios Emergentes (AJSE), el cual gestiona los JSE de forma autónoma. El AJSE, una vez recolectada la información del entorno de SaCI (perfil de los estudiantes, objetivos del actual proceso de aprendizaje, etc.), adapta el JSE a SaCI. Para ello, el AJSE interactúa con los agentes de SaCI: Gestor del Repositorio de Objeto de Aprendizaje (ROA), Sistema Académico (SA), Sistema Recomendado (SR) de Recursos Educacionales y el Entorno Virtual de Aprendizaje (por sus siglas en inglés, VLE) (ver figura 4.11).

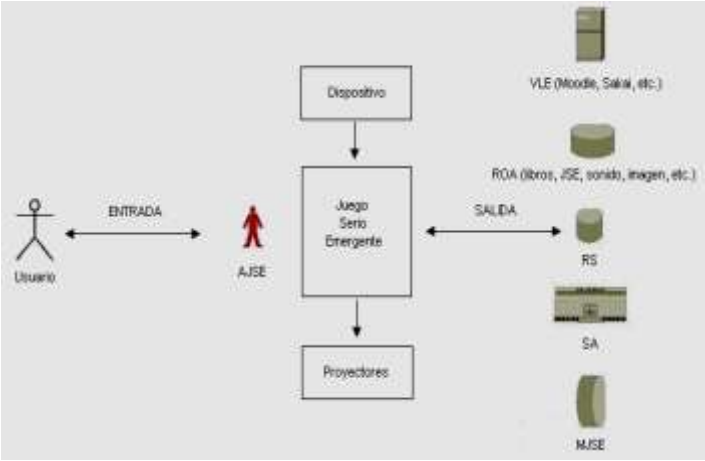

Figura 4.11. Modelo conceptual de un AJSE en un SaCI [33]

4.2.2 Contexto dinámico pedagógico para analizar el comportamiento del SAP

Una vez en el SaCI, se parte de la hipótesis que se está en una clase de matemáticas con 3 estudiantes, y se requiere definir un JSE con el objetivo de explicar y resolver ejercicios, dependiendo del tema escogido. El tema a estudiar es el de las raíces cuadradas, y se selecciona un videojuego del dominio en raíces compatible con el tema de aprendizaje.

Se supone que inicialmente el SEV 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):

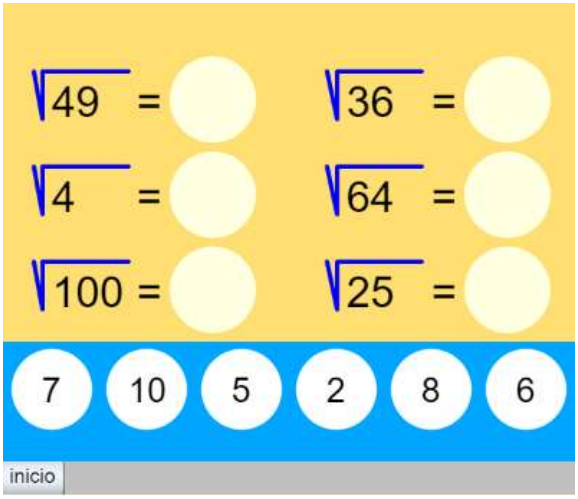

Figura 4.12. El videojuego del cálculo mental de la raíz cuadrada de 6 números

El videojuego del cálculo mental de la raíz cuadrada de 6 números, consiste en una escena donde aparecen en la parte de arriba (franja amarilla) seis raíces cuadradas cuyos radicandos son números cuadrados perfectos. Los números de abajo son sus soluciones, que vienen en forma de fichas circulares, las cuales el jugador tendrá que ubicarlas en los espacios correctos. Una vez se hayan ubicado todas las fichas en los lugares correctos, el videojuego mostrará la palabra "CORRECTO" en la franja azul.

Ahora bien, digamos que se quiere estudiar otro tema perteneciente a otra clase, como por ejemplo "Las notas musicales" en la clase de "Música". Entonces, el SEV puede proponer como JSE el videojuego "Aprendo las notas de la escala de DO, en el modo de escuchar", obtenido del enlace: https://aprendomusica.com/const2/27aprendonotas8/aprendonotas8.html (ver figura 4.13).

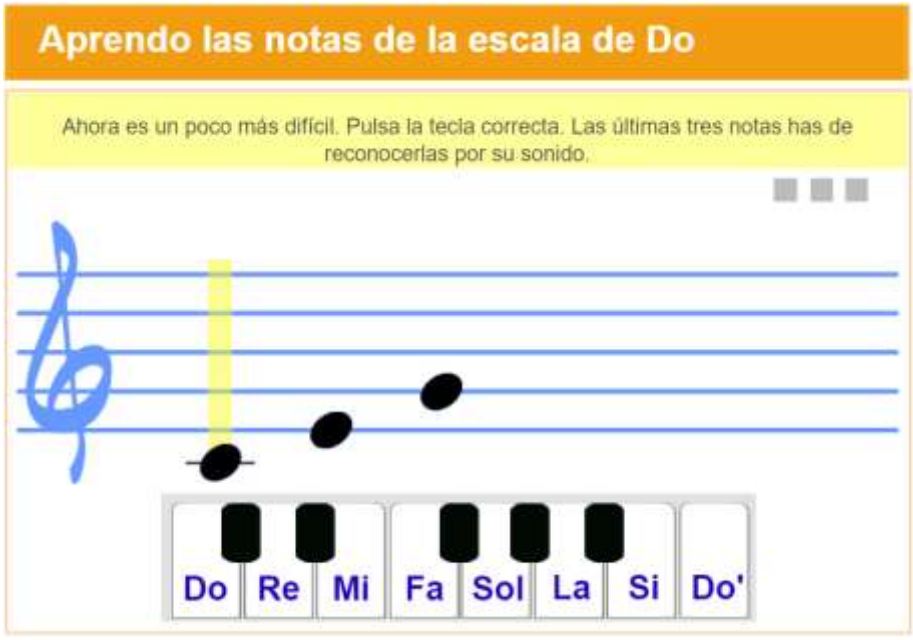

Figura 4.13. El videojuego "Aprendo las notas de la escala de DO, en el modo de escuchar"

El videojuego "Aprendo las notas de la escala de DO, en el modo de escuchar", consiste en reconocer por medio del sentido del oído, tres notas musicales, las cuales se mostrarán en el pentagrama, cuando el jugador pulse la tecla de piano correcta, en el tiempo correcto, determinado como una franja vertical amarilla. En un principio, dicho pentagrama contiene: la clave de sol, cuya función es determinar la altura que corresponde a cada nota musical, según la posición (el espacio o línea) que ocupan en el mismo, y tres notas musicales, que sirven de guía para resolver el videojuego.

Dicho videojuego consta de tres niveles, representados por los cuadrados de color gris, el cual, cambia de color a verde, si se alcanzan los objetivos de ese nivel, y rojo o naranja de lo contrario.

4.2.3 SAP en SaCI

A continuación, se explica cómo SaCI usa el SAP para la emergencia de propiedad. Particularmente, SAP determina los valores idóneos de los parámetros del JSE.

JSE1: videojuego del cálculo mental de la raíz cuadrada.

Ese videojuego tiene 3 parámetros:

- P1 = Máximo valor numérico del radicando.
- P2 = Raíces cuadradas a calcular (los espacios para colocar fichas).
- P3 = Las fichas (números) que tiene el jugador (ya que el jugador puede tener más fichas que raíces cuadras a calcular).

Las reglas de calidad son las mismas definidas en la sección 4.1.2. Ahora bien, el límite de tiempo y la puntuación son definidas por el profesor de la clase. En este caso, se definen como:

- Tiempo: un límite de tiempo de 600 segundos, o el equivalente en minutos, 10.
- Alcance de los objetivos: si el valor es superior a los 500 puntos, entonces el jugador cumplió con la mayoría de los objetivos del JSE, y el valor máximo es 1000 puntos.

La tabla 4.17 muestra las primeras veces que se jugó el JSE1.

#	P1	P2	P3	FO
1	9	1	4	631.4
2	64	1	4	500.4
3	81	4	3	255.6

Tabla 4.17. Población inicial del videojuego del cálculo mental de la raíz cuadrada

- El primer estudiante lo jugó con los parámetros P1 = 9, P2 = 1 y P3 = 4, con FO = 631.4.

  El valor de la FO fue calculado de la siguiente manera:

  $$FO = 1000 * 0.944 - 600 * 0.521 = 631.4$$

  PA normalizado es = 944 puntos.

  LE normalizado es = 312.6 segundos.

Según las reglas de calidad, se observa que el estudiante alcanzó los objetivos del JSE (944 puntos) en aproximadamente 5 minutos y 22 segundos (313 segundos).

- El segundo estudiante lo jugó con los parámetros P1 = 64, P2 = 1 y P3 = 4, con FO = 500.4.

  El valor de la FO fue calculado de la siguiente manera:

FO = 1000 * 0.774 - 600 * 0.456 = 500.4

PA normalizado es = 774 puntos.

LE normalizado es = 273.6 segundos.

Según las reglas de calidad, se observa que el estudiante alcanzó los objetivos del JSE (774 puntos) en aproximadamente 4 minutos y 50 segundos (274 segundos).

- El tercer estudiante lo jugó con los parámetros P1 = 81, P2 = 4 y P3 =3, con FO = 255.6.

  El valor de la FO fue calculado de la siguiente manera:

  FO = 1000 * 0.414 - 600 * 0.264 = 255.6

  PA normalizado es = 414 puntos.

  LE normalizado es = 158.4 segundos.

Según las reglas de calidad, se observa que el estudiante no alcanzó los objetivos del JSE (414 < 500 puntos) en un lapso de tiempo de aproximadamente 3 minutos (158 segundos).

Después de 200 generaciones, se observa la evolución del mejor individuo en la figura 4.14. Además, la tabla 4.18 muestra que a partir de la generación 84 se obtiene ese mejor individuo.

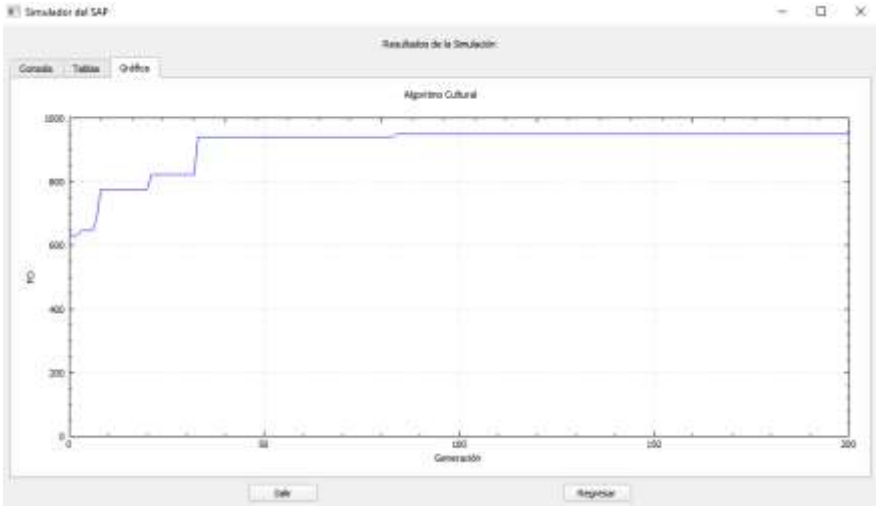

Figura 4.14. Gráfica del mejor individuo para el videojuego del cálculo mental de la raíz cuadrada

Mejores Individuos

Generación	Individuo	Función Objetivo (FO)
84	[49, 4, 4]	950.4
85	[49, 4, 4]	950.4
86	[49, 4, 4]	950.4
87	[49, 4, 4]	950.4
88	[49, 4, 4]	950.4

Tabla 4.18. Tabla donde se observa el mejor individuo para el videojuego del cálculo mental de la raíz cuadrada

En la Tabla 4.18 se puede ver el mejor individuo que ocurre en la generación 84, con FO = 950.4, el cual fue calculado de la siguiente manera:

FO = 1000 * 0.978 - 600 * 0.046 = 950.4

PA normalizado es = 978 puntos.

LE normalizado es = 27.6 segundos.

Particularmente, el SAP sugiere los parámetros P1 = 49, P2 = 4 y P3 = 4, para que el jugador alcance los objetivos del videojuego del cálculo mental de la raíz cuadrada (978 puntos) en el menor tiempo posible (~28 segundos).

JSE2: videojuego "Aprendo las notas de la escala de DO, en el modo de escuchar",

En este videojuego se consideran 4 parámetros:

- P1 = Teclas que tiene el jugador.
- P2 = Notas a adivinar en el pentagrama.
- P3 = Penalización por mala jugada.
- P4 = Niveles.

Las reglas de calidad son las mismas definidas en la sección 4.1.2. El límite de tiempo y la puntuación son definidas por el profesor de la clase, en este caso, como siguen:

- Tiempo: un límite de tiempo de 60 segundos (1 minuto).
- Alcance de los objetivos: si el valor es superior a los 500 puntos, entonces el jugador cumplió con la mayoría de los objetivos del JSE, y el valor máximo es 1000 puntos.

La tabla 4.19 muestra las primeras veces que se jugó el JSE 2.

#	P1	P2	P3	P4	FO
1	4	2	7	2	781.8
2	7	3	6	2	287
3	7	2	9	2	241.8

Tabla 4.19. Población inicial del videojuego "Aprendo las notas de la escala de DO, en el modo de escuchar"

- El primer estudiante lo jugó con los parámetros P1 = 4, P2 = 2, P3 = 7, P4 = 2 con FO = 781.8, el cual fue calculado de la siguiente manera:

$$FO = 1000 * 0.81 - 60 * 0.47 = 781.8$$

PA normalizado es = 810 puntos.

LE normalizado es = 28.2 segundos.

Según las reglas de calidad, se observa que el estudiante alcanzó los objetivos del JSE (810 puntos) en aproximadamente 28 segundos.

- El segundo estudiante lo jugó con los parámetros P1 = 7, P2 = 3, P3 = 6, P4 = 2 con FO = 287, el cual fue calculado de la siguiente manera:

$$FO = 1000 * 0.29 - 60 * 0.05 = 287$$

PA normalizado es = 290 puntos.

LE normalizado es = 3 segundos.

Según las reglas de calidad, se observa que el estudiante no alcanzó los objetivos del JSE (290 < 500 puntos), y duro 3 segundos.

- El tercer estudiante lo jugó con los parámetros P1 = 7, P2 = 2, P3 = 9, P4 = 2, con FO = 241.8, el cual fue calculado de la siguiente manera:

$$FO = 1000 * 0.27 - 60 * 0.47 = 241.8$$

PA normalizado es = 270 puntos.

LE normalizado es = 28.2 segundos.

Según las reglas de calidad, igualmente se observa que el estudiante no alcanzó los objetivos del JSE (270 < 500 puntos), jugando 28 segundos aproximadamente.

Después de 200 generaciones, se observa la evolución del mejor individuo en la figura 4.15, y en la tabla 4.20 que el mejor individuo aparece en la generación 141.

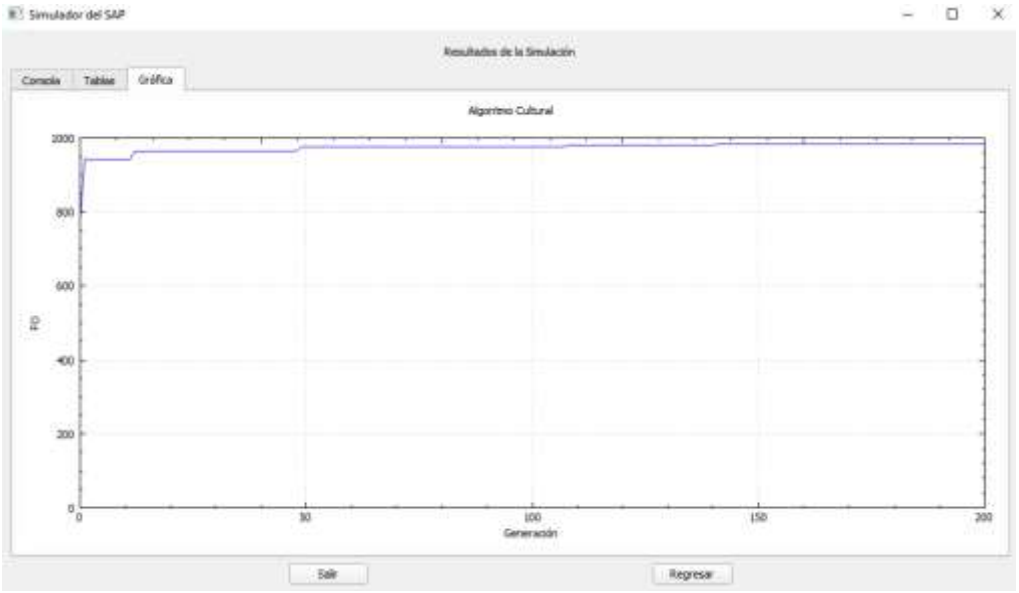

Figura 4.15. Gráfica del mejor individuo para el videojuego "Aprendo las notas de la escala de DO, en el modo de escuchar"

Mejores Individuos

Generación	Individuo	Función Objetivo (FO)
141	[4, 1, 9, 1]	983.4
142	[4, 1, 9, 1]	983.4
143	[4, 1, 9, 1]	983.4
144	[4, 1, 9, 1]	983.4
145	[4, 1, 9, 1]	983.4

Tabla 4.20. Tabla donde se observa el mejor individuo para el videojuego "Aprendo las notas de la escala de DO, en el modo de escuchar"

En la tabla 4.20 se puede ver que el mejor individuo aparece desde la generación 141, con FO = 983.4, el cual fue calculado de la siguiente manera:

$$FO = 1000 * 0.99 - 60 * 0.11 = 983.4$$

PA normalizado es = 990 puntos.

LE normalizado es = 6.6 segundos.

Por lo tanto, el SAP sugiere los parámetros P1 = 4, P2 = 1, P3 = 9 y P4 = 1, para que el jugador alcance los objetivos del videojuego (990 puntos), en el menor tiempo posible (~7 segundos).

Capítulo 5

Comparación con otros trabajos

La tabla 5.1 compara nuestra tesis con otros trabajos, en base a 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 Algoritmo Cultural?

TABLA 5.1. Comparación con otras obras.

Criterio	[34]	[30]	[3]	[5]	[6]	[7]	[8]	[9]	[35]	Presente trabajo
A		X			X	X				X
B			X							X
C	X	X	X	X	X	X	X	X	X	X
D			X							X
E	X			X			X		X	X

El artículo [34] presenta el uso de los AC para ajustar los parámetros de un estabilizador de un sistema de potencia (PSS). Esta técnica muestra ser robusta y computacionalmente eficiente, en comparación con otros algoritmos metaheurísticos. Las simulaciones se llevan a cabo en dos sistemas típicos de energía eléctrica.

En [30] se permite la adaptación de un JE a través de la adecuación del radio que define la influencia de un edificio con respecto a otros en el juego "Metrópolis". Este radio se modifica de acuerdo con el comportamiento de los jugadores, de modo que los edificios compatibles, utilizados principalmente por los jugadores, tienden a tener un corto rango de acción (radio), para facilitar la

emergencia urbana. Por otro lado, [3] propone un motor que adapta el JS a los diferentes jugadores, utilizando técnicas de inteligencia artificial. El motor utiliza reglas preestablecidas, que se adaptan. En [5] se usan ACs para la gestión de redes sociales de distribución del conocimiento. Los mecanismos de distribución apoyan tanto la cooperación como la competencia entre los participantes de la red. Permite el surgimiento de comunidades de conocimiento.

[6] propone un proceso de adaptación de un juego de dos niveles: un primer nivel donde las personas interactúan con los otros jugadores; y el segundo nivel donde el grupo se divide, de acuerdo con ciertas reglas. La adaptación que propone este trabajo es en términos del número de participantes en los juegos. En [7], el GameEA utiliza una población de jugadores artificiales, y genera nuevos jugadores artificiales mediante un operador de imitación y un operador de aprendizaje. El operador de imitación adopta estrategias y acciones de otros jugadores, para mejorar su competitividad. Por otro lado, el operador de aprendizaje permite ajustar estrategias y comportamientos mediante el análisis de la información histórica.

En [8] se introduce un nuevo método de distribución del conocimiento, llamado mecanismo de población. Ese método de gestión de conocimiento es realizado usando un AC, que se puede utilizar para una amplia gama de problemas, ya que permite la adecuación automática del espacio de creencia. [9] describe dos juegos que usan algoritmos heurísticos para problemas de optimización combinatoria en su dinámica: el primer juego está diseñado para incorporar en los juegos de estrategia aspectos de recolección de recursos, utilizando el ACO para el problema del vendedor ambulante. El segundo juego es de carreras, donde se utiliza las características numéricas del corredor, como la velocidad, la agilidad y la potencia de salto, para buscar las mejores soluciones para diferentes problemas en el juego (por ejemplo, la ruta que deben tomar).

En [35] se modela una reducción de atributos de objetivos múltiples (MOAR) mediante el diseño de una función de costo que permite minimizar dicho número de atributos usando el coeficiente de dependencia máximo de la teoría de conjuntos aproximados (RST por sus siglas en ingles). Para ello, implementan un AC y cinco algoritmos, para comparar sus resultados en doce conjuntos de datos.

En general, vemos que muchos de los trabajos realizados con AC permiten procesos adaptativos en el entorno donde son usados [5, 6, 8], pero no han sido usados para que permitan la emergencia en motores de juegos, y menos en motores de JSE. Tampoco la emergencia ha sido incorporada en motores de juego [3], y los JEs hasta ahora desarrollados que permiten la emergencia de parámetros, son

exclusivamente para ese juego [30]. Nuestra propuesta es orientada a un MJSE, para permitir la emergencia fuerte de parámetros en cualquier JSE ejecutado en un entorno dado.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1 Conclusiones

A nuestro conocer, no existen trabajos que permitan la emergencia del comportamiento de los parámetros de un juego, y menos de un JSE, a partir del comportamiento de los jugadores (ver tabla 5.1). Los JSE, a diferencia de otros juegos, tienen el potencial de adecuarse a un contexto narrativo emergente, como el generado en un proceso de enseñanza-aprendizaje en un aula inteligente. Así, logran adaptar su comportamiento al contexto donde serán usados. Para ello, requieren la emergencia de ciertos comportamientos. Uno de ellos es la emergencia de sus propiedades, estudiada en esta tesis.

En particular, esta tesis desarrolló un sistema que permite la determinación de los parámetros óptimos de un JSE, para adecuarse a las características de los jugadores. Específicamente, se propuso utilizar los ACs para ese proceso, utilizando diferentes tipos de conocimiento sobre el JSE: situacional, normativo, dominio e histórico. El AC permite caracterizar el proceso de optimización de parámetros como un proceso evolutivo, que explota el conocimiento en el espacio de creencias (el mejor conocimiento obtenido a través de las generaciones para cada parámetro).

En este trabajo se utilizaron solo los conocimientos situacional y normativo, debido a la complejidad que se presentó al momento de implementar los conocimientos de dominio e histórico. Particularmente, con estos dos conocimientos se logra tener suficiente información del entorno para general un meta-conocimiento que permita mejorar los JSEs según el comportamiento de los jugadores (ver figuras 4.3 y 4.8).

Durante el desarrollo de los experimentos, se analizó la sensibilidad de los parámetros del AC, es decir, los valores de las variables del SAP, logrando determinar los valores por defecto del mismo, los cuales influyen en el proceso de optimización de parámetros del JSE (la búsqueda del mejor individuo de

la población final). En tal sentido, para los dos JSE en los que se analizó dicha sensibilidad se lograron obtener los mismos valores por defecto para diferentes corridas.

Con respecto al caso de estudio en un SaCI, se explica como el JSE adecua sus parámetros al contexto dinámico pedagógico, usando el componente SAP del MJSE. Dichos parámetros se logran alcanzar de manera eficiente usando el SAP basado en ACs, específicamente, son definidos en el mejor individuo de la población final (su cromosoma tiene los valores óptimos de los parámetros del JSE).

6.2 Trabajo Futuro

En primer lugar, se debe hacer un análisis experimental más amplio del comportamiento emergente que permita nuestro enfoque, y su impacto en el proceso de enseñanza, en particular, en los estudiantes. Estos experimentos se deben hacer en distintos niveles de educación, como son: preescolar, básica, secundaria, diversificada y universitaria, cambiando el grado de dificultad de JSEs en cada una de ellas.

Con respecto a analizar su efecto en el proceso de enseñanza, ya sea guiado por un SaCI o no, se deben usar varias métricas pedagógicas que permitan evaluar el impacto de los JSEs adaptados por el SAP. También, es preciso analizar la integración del algoritmo con otros tipos de emergencias posibles en un JSE (comportamiento, trama, entre otros).

Otro aspecto importante, es incorporar los otros tipos de conocimiento al espacio de creencias del SAP que no se consideraron en este trabajo, como son: los conocimientos de dominio e histórico, y evaluar el impacto de los mismos.

Bibliografía

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

[2] Aguilar J., Altamiranda J., Díaz F., (2020), "Specification of a Managing Agent of Emergent Serious Games for a Smart Classroom". IEEE Latin America Transactions, 18, 51-58.

[3] Rasim T., Langil A., Munir S., and Rosmansyah Y. (2016), "A survey on adaptive engine technology for serious games", Proceedings of International Seminar on Mathematics, Science, and Computer Science Education (MSCEIS 2015) AIP Conf. Proc. 1708, 1708 (1).

[4] Shafi K., and Abbass H. A., (2017), "A Survey of Learning Classifier Systems in Games", IEEE Computational intelligence magazine, pp. 42-55.

[5] Waris F., and Reynolds R., (2018), "Optimizing AI Pipelines: A Game-Theoretic Cultural Algorithms Approach", IEEE Congress on Evolutionary Computation (CEC), pp. 1-10.

[6] Singh D., Moradian P., and Kobti Z. (2018), "A Multilevel Cooperative Multi-Population Cultural Algorithm", School of Computer Science University of Windsor, Canada, pp. 1-5.

[7] Yang G. (2017), "Game Theory-Inspired Evolutionary Algorithm for Global Optimization", Algorithms, 10, 1-15.

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

[9] Jamieson P., Grace L., Hall J., and Wibowo A. (2013) 'Metaheuristic Entry Points for Harnessing Human Computation in Mainstream Games', International Conference on Online Communities and Social Computing (OCSC 2013). 8029, pp. 156-163.

[10] Jacobson I., Booch G., and Rumbaugh J. (2000) "El proceso unificado de desarrollo de software", Pearson Educación, S. A. Madrid, España.

[11] Abt C., (1970), "Serious Games", Viking Press, New York. EEUU.

[12] Zyda, M. (2005), 'From visual simulation to virtual reality to games', Computer, 38, 25-32,

[13] Marcano B. (2006) "Estimulación emocional de los videojuegos: efectos en el aprendizaje", Estudio de los comportamientos emocionales en la red. Revista Electrónica Teoría de la Educación. Educación y Cultura en la Sociedad de la Información, 7(2), http://campus.usal.es/~teoriaeducacion/rev_numero_07_02/n7_02_beatriz_marcano.pdf

[14] Chipia J. (2011), "Juegos Serios: Alternativa Innovadora", Revista CLED. 1, 1-18.

[15] Rodríguez F., Barajas S., and Muñoz J. (2014), "Serious Game Design Process, Study Case: Sixth Grade Math", Creative Education. 5, 647-656.

[16] Barbosa A., Pereira P., Dias J., Silva F., (2014), "A New Methodology of Design and Development of Serious Games", International Journal of Computer Games Technology, pp. 1-8.

[17] Carrión M., Santorum M., Aguilar J., Peréz M. (2019) "iPlus Methodology for Requirements Elicitation for Serious Games" 22nd Iberoamerican Conference on Software Engineering, pp. 434-447.

[18] Carrión M., Santorum M., Aguilar J., Peréz M. (2017) "A Participatory Methodology for the Design of Serious Games in the Educational Environment", II International Conference on Innovation and Trends in Engineering.

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

[20] Aguilar J. (2014), "Introducción a los sistemas emergentes", Talleres Gráficos, Universidad de Los Andes, Mérida, Venezuela.

[21] Jean M. (1997), "Emergence et SMA, en IA distribuée et systèmes multi-agents", Hermès, pp. 323-341.

[22] Real Academia Española [RAE] (2018), "Emergencia", 23.ª ed. [versión 23.2 en línea]. Recuperado de: <https://dle.rae.es/?id=EiX5X40>.

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

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

[25] Aguilar J., Altamiranda J., Díaz F., and Mosquera D. (2016), "Motor de Juego Serios en ARMAGAeco-c", Revista Científica UNET, 28 (2), 100-110.

[26] Aguilar J., Valdiviezo P., Cordero J., and Sánchez M. (2015), "Conceptual Design of a Smart Classroom Based on Multiagent Systems" Proceeding of the Int. Conf. Artificial Intelligence (ICAI'15), 471-477.

[27] Sánchez M., Aguilar J., Valdiviezo P. and Cordero J. (2015), "A Smart Learning Environment based on Cloud Learning," International Journal of Advanced Information Science and Technology (IJAIST), 39(39), 36-49.

[28] Sánchez M., Aguilar J., Cordero J. and Valdiviezo P. (2015), "Basic features of a Reflective Middleware for Intelligent Learning Environment in the Cloud (IECL)," Proceeding Asia-Pacific Conf. on Computer Aided System Engineering (APCASE), pp. 1-6.

[29] Aguilar J., Altamiranda J., Díaz F. (2018), "Design of a serious emerging games engine based on the optimization algorithm of ant colony", Revista DYNA, 85 (206), 311-320.

[30] Aguilar J., Altamiranda J. and Chávez D. (2016), "Extensiones a Metrópolis para una Emergencia Fuerte," Revista Venezolana de Computación, 3(2), 38-46.

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

[32] Azevedo D. (2012), "Algoritmos Culturais com Abordagem Memética e Multipopulacional Aplicados a Problemas de Optimização," Tese de Doutorado em Engenharia Eléctrica em UFPA/ITEC/PPGEE Campus Universitario Do Guamá, Belém, Pará, Brasil

[33] Aguilar J., Díaz F., Altamiranda J., Cordero J., Chávez D., Gutiérrez J. (2019), "Metrópolis: un juego serio emergente en una ciudad inteligente", Revista DYNA, 86 (211), 215-224

[34] Khodabakhshian A. and Hemmati R. (2013), "Multi-machine power system stabilizer design by using cultural algorithms", International Journal of Electrical Power & Energy Systems, 44(1), 571-580.

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

Anexos

Aquí empieza el texto.