UNIVERSIDAD DE GRANADA

MÁSTER UNIVERSITARIO

EN CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

Trabajo de Fin de Máster

Desarrollo y Optimización de un Agente Inteligente para Scripts of Tribute mediante Algoritmos Evolutivos

Autor

Francisco David Castejón Soto

Director

Dr. Pablo García Sánchez

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, 1 Julio de 2025

Prefacio

Para la realización de este documento, se ha utilizado la plantilla LaTeX [7] específica para la ETSIIT.

Agradecimientos

A mi familia por el apoyo moral y económico que me han brindado de forma incondicional durante estos años fuera de casa. Y a mi pareja, por mostrarme más caminos de los que nunca hubiera pensado que existieran para mí.

Índice general

Índice de figuras

Índice de tablas

Parte I

Marco teórico

Capítulo 1

Introducción

La inteligencia artificial (IA) en videojuegos existe en una intersección entre la ciencia computacional y el arte del entretenimiento, dando vida a mundos virtuales y creando oponentes dignos de nuestras mejores estrategias. En este contexto, la IA se refiere únicamente a un conjunto de algoritmos y técnicas diseñadas para cumplir una función muy específica dentro del videojuego. Ejemplos de esto son la IA que "conduce" los coches en un juego de carreras, los Pokémon contra los que el jugador se enfrenta en una batalla o el simple movimiento de una línea de píxeles (a modo de pala contrincante) en el videojuego Pong. Esta acepción específica contrasta con la idea más extendida de IA, que posee un conjunto de definiciones más amplias y generalistas como "la máquina que aprende" o "el estudio y construcción de sistemas que hacen «lo correcto», en función de su objetivo" [12].

Independientemente de la definición que se utilice, lo cierto es que un factor clave en el desarrollo de un videojuego es la creación de una IA que resulte entretenida para el jugador. De la misma forma que en un libro o una película es necesaria una trama que plantee un desafío, como la lucha contra el Imperio Galáctico en Star Wars, o un compañero de aventuras que ayude al protagonista, como Sam en El Señor de los Anillos, en un videojuego es necesaria una IA que controle esos elementos que interactúan con el jugador, ya sean enemigos, aliados o incluso el propio entorno. Desde los propios inicios de la industria del videojuego, donde se construían máquinas específicas para ejecutar un juego concreto, como es el caso de Nim en 1948 [11], hasta los videojuegos modernos, donde se utilizan técnicas de IA más complejas, como el aprendizaje por refuerzo (más en la sección 3.2), la IA ha sido un componente esencial para crear experiencias de juego atractivas y desafiantes.

La intención de este trabajo es la de crear una IA contra la que jugar en un videojuego de estrategia. En concreto, la IA que controla a un agente autónomo (o simplemente "bot") en el videojuego de cartas *Tales of Tribute*. Es importante la distinción entre "juego" y "videojuego" en este caso, ya que las cartas no son

físicas, sino que únicamente existen en el universo digital del videojuego. Para guiar las decisiones del bot, un algoritmo evolutivo se encarga de ajustar los pesos que se utilizan para calcular la puntuación de cada jugada posible, eligiendo vorazmente la jugada que maximiza dicha puntuación. Así se ha conseguido crear un contrincante que juega de forma competitiva tanto contra el jugador humano como contra otros bots manejados por IAs totalmente diferentes.

Capítulo 2

Objetivos

2.1. Objetivo principal

El objetivo principal de este Trabajo de Fin de Máster se puede dividir en dos partes, la primera es ahondar en la investigación y entendimiento de un area tan extensa como las metaheurísticas, y en concreto los algoritmos evolutivos. La optimización de soluciones a problemas complejos mediante técnicas que tratan incluir conocimiento específico del dominio conjuntamente con la generación iterativa de otras soluciones parciales es un campo de estudio con un gran número de aplicaciones prácticas. Aunque el teorema de "No Free Lunch" nos advierte de que no existe una única técnica que sea la mejor para todos los problemas y desde todas las perspectivas [16], la experiencia empírica ha demostrado que los algoritmos evolutivos son una herramienta que se puede aplicar a la gran mayoría de problemas de optimización [13].

Es precisamente esa versatilidad la que ha propiciado su uso durante el desarrollo del segundo objetivo principal de este proyecto: la creación de un bot para un videojuegos de cartas. Este tipo de videojuegos cuentan con un inmenso número de variables, pues no solo se deben tener en cuenta las cartas existentes en las manos de cada jugador y en la pila de cartas, sino también las mecánicas[1] intrínsecas del juego, como la vida, los recursos de compra, la posibilidad de recuperar cartas usadas, o el uso de otro tipo de habilidades especiales. Todo esto hace que la creación de un bot que juegue de forma competitiva sea un reto interesante, y que el uso de técnicas de optimización evolutiva sea una herramienta adecuada para conseguirlo.

[1] Gran parte de lo que define un juego es sus mecánicas, sus reglas, lo que los jugadores deben llevar a cabo para ganar. La labor del creador de videojuegos es conseguir un conjunto de reglas que estén equilibradas y permitan disfrutar a los jugadores [15].

2.2. Objetivos específicos

- OG1: Analizar en profundidad las mecánicas del juego "Tales of Tribute", el entorno de desarrollo "Scripts of Tribute" y adquirir las competencias necesarias en el lenguaje de programación C#.
- OG2: Diseñar e implementar un agente inteligente en C# para "Scripts of Tribute", cuya toma de decisiones se base en una evaluación heurística del estado del juego mediante una función de fitness ponderada, incorporando conocimiento experto del dominio.
- OG3: Desarrollar un marco de optimización en Python para ajustar los pesos de la función de fitness del agente, implementando y comparando dos estrategias principales: algoritmos coevolutivos y entrenamiento supervisado contra agentes de referencia de "Scripts of Tribute".
- Desarrollar un conjunto de herramientas para la visualización y análisis de los datos generados durante el entrenamiento.
- OG5: Evaluar cuantitativamente el rendimiento del agente entrenado mediante las diferentes estrategias, utilizando métricas relevantes como la tasa de victorias contra distintos oponentes.
- OG6: Analizar comparativamente la efectividad y eficiencia de las estrategias de optimización implementadas (coevolución vs. entrenamiento contra referentes), discutiendo sus ventajas y desventajas en el contexto específico de "Scripts of Tribute".
- OG7: Investigar y contextualizar el enfoque desarrollado frente a otras técnicas predominantes en competiciones similares de IA en juegos, como Monte Carlo Tree Search (MCTS), analizando las razones de su éxito en ediciones anteriores.

Capítulo 3

Antecedentes y estado del arte

3.1. Inteligencia artificial en videojuegos comerciales

En esta sección se realiza un breve repaso a las técnicas de inteligencia artificial más utilizadas a lo largo de la historia de los videojuegos, así como algunos ejemplos del estado del arte en este ámbito. Los usos de estas ténicas no solo abarcan su función más habitual de antagonista o compañero, sino que también abarca otros aspectos como la generación de contenido procedural o la deformación realista de las mallas [2] de los personajes animados.

3.1.1. Funciones hash

Una de las técnicas más sencillas y antiguas para controlar partes específicas de un videojuego son las funciones hash. Este tipo de funciones simplemente reciben un conjunto de parámetros como entrada y devuelven un valor o acción a realizar. Dada su simplicidad, generan comportamientos predecibles, pero requieren de muy pocos recursos para su procesamiento y son fáciles de implementar. Un ejemplo clásico de IA basada en funciones hash es el en juego *Space Invaders* [14] donde los invasores estaban controlados con funciones hash.

3.1.2. Máquinas de estados finitos

Las máquinas de estados finitos son un modelo de computación conceptual que describe un sistema que solo puede encontrarse en un estado a la vez, y que puede cambiar a uno de sus otros estados como respuesta a ciertos eventos. Este mecanismo se ha utilizado y sigue utilizándose para un gran número de aplicaciones dentro de los videojuegos. Por ejemplo, en el juego *Pacman*, la IA de los

[2] Un "mesh", o malla en español, es una estructura tridimensional formada por una colección de vértices, aristas y caras que definen la forma de un objeto 3D dentro de un videojuego [2].

fantasmas se rije por el estado en el que se encuentran, como çazando.º "siendo cazados" [10]. Otro uso está en la gestión de las animaciones de los personajes, donde cada estado corresponde a una posición o movimiento específico de la malla, que al mezclarse mediante interpolaciones generan animaciones. Al igual que las funciones hash, requieren de pocos recursos para su procesamiento, pero su naturaleza determinista y la posibilidad de entrar en bucles sin salida si no están bien diseñadas, limitan su uso en los escenarios más exigentes.

3.1.3. Árboles de comportamiento

Como evolución a las máquinas de estados finitos, surgieron los árboles de comportamiento, permitiendo un mayor grado de complejidad y flexibilidad en la toma de decisiones de los personajes. Los árboles de comportamiento son grafos dirijidos acíclicos, con nodos hoja que representan acciones y varios tipos de control de flujo que determinan el orden de ejecución de las hojas [3]. Una de sus características más destacadas es su modularidad, lo que permite reutilizar y combinar comportamientos fácilmente. Un fantastico ejemplo de su correcto uso se expuso en la charla de la "Game Developers Conference" (GDC) de 2005, sobre la IA de *Halo 2*, donde se explica que su capacidad de reutilización de árboles y el ser más sencillos de entender para los diseñadores del juego, hizo que los programadores optaran por su implementación en lugar de las máquinas de estados finitos [8].

3.1.4. Planificación de acciones orientada a objetivos

En contraste con los árboles de comportamiento, que generan una jerarquía de caminos, la planificación de acciones orientada a objetivos (GOAP, por sus siglas en inglés) es más parecida a la planificación de rutas, donde se busca el camino más óptimo para alcanzar un objetivo en función del estado del mundo. Cada uno de los pasos que llevan al objetivo tiene una serie de precondiciones que deben cumplirse, a los cuales se les puede asignar un coste. Es tal su similitud con la planificación de rutas, que este tipo de IA suelen utilizar internamente algoritmos de búsqueda como A*[3] para encontrar el camino más óptimo. Una de las mejores implementaciones de este tipo de IA se encuentra en el título de 2005 *F.E.A.R.*, donde los enemigos y aliados utilizan GOAP para "planificar" su comportamiento una manera mucho más proactiva que sus predecesores [9].

3.1.5. Aprendizaje por refuerzo y redes neuronales

El aprendizaje por refuerzo (RL, por sus siglas en inglés) es un subcampo del aprendizaje automático que trata de mejorar el comportamiento de un agente a

[3] A* también se utiliza en la búsqueda de caminos tradicional dentro de los videojuegos, es decir, para saber la ruta que debe tomar un objeto por el mapa para llegar a un destino, y no solo en GOAP.

través del feedback que recibe de su entorno. Utilizando un sistema de recompensas y penalizaciones, el agente aprende a tomar decisiones que maximicen su recompensa total a lo largo del tiempo. Aunque el RL empezó con técnicas tabulares, los algoritmos recientes incorporan redes neuronales en diferentes partes del proceso [6]. Quizás el ejemplo que más se suele ver en la literatura de videojuegos es el uso de RL para controlar vehículos autónomos. Una implementación de esta tecnología en un videojuego comercial está en el título *Star Wars: Outlaws*, donde las motos controladas por IA utilizan un sistema de RL para moverse por el mapa o perseguir al jugador [5]. Sus desarrolladores argumentan que les permitió tener un sistema de control que se adapta a las diferentes rutas del juego sin importar los cambios en el mapa que haya entre versiones.

Pero el uso de redes neuronales no se limita a la toma de decisiones. Su capacidad para realizar inferencias rápidas a partir de datos complejos las hace ideales para otras tareas, como la deformación de mallas. El objetivo en este caso es evitar el fenómeno conocido como el "valle inquietante" (*Uncanny Valley*), que se produce cuando una réplica de un ser humano es muy realista, pero no perfecta, generando una sensación de extrañeza en el espectador. Un personaje que sonríe, por ejemplo, pero cuyos músculos faciales alrededor de los ojos permanecen inmóviles, es un caso clásico de este efecto. Para intentar superar este problema, Epic Games optó por crear un sistema para Unreal Engine que utiliza modelos de redes neuronales que simulan el comportamiento de la musculatura y los tejidos blandos bajo la piel al extenderse y contraerse [4]. Un ejemplo de uso de esta reciente tecnología se puede encontrar en la demo técnica que CD Projekt Red mostró en el Unreal Fest de 2025 [1]. En ella, se mostraba como se modelaba internamente la musculatura de un caballo para que sus animaciones de movimiento fueran mucho más realistas.

3.2. Metaheurísticas en videojuegos

3.2.1. Generación de contenido procedural

3.2.2. El desafío de las metaheurísticas en videojuegos comerciales

3.3. Algoritmos evolutivos en juegos de cartas

3.3.1. Optimización de agentes de Hearthstone mediante un algoritmo evolutivo

3.3.2. Desarrollo de un agente para el concurso de IA Tales of Tribute

TODO: Analizar la estrategia de MCTS de los ganadores de la competición de IA de Tales of Tribute.

Capítulo 4

Plataforma de trabajo

4.1. El videojuego: Tales of Tribute

4.2. El entorno de simulación: Scripts of Tribute

4.2.1. La competición IEEE-CoG

4.2.2. Arquitectura del entorno

Capítulo 5

Planificación del proyecto

5.1. Recursos utilizados

5.1.1. Recursos software

5.1.2. Recursos hardware

5.2. Metodología de desarrollo

5.3. Distribución temporal y cronograma

Parte II

Metodología

Capítulo 6

Diseño del agente autónomo

6.1. Arquitectura de la toma de decisiones

6.2. Definición de los pesos

6.3. Lógicas específicas y heurísticas

Capítulo 7

Sistema de entrenamiento evolutivo

Parte III

Experimentación y conclusiones

Capítulo 8

Diseño experimental

TODO: Probar diferentes combinaciones en el modo hibrido. TODO: Hablar sobre los individuos generalistas (todos los pesos parecidos) y especialistas (0-1 en los pesos) en los mejores pesos. TODO: Ver la diferencia entre usar el hall of fame y no usarlo ya que no debería añadir info de una train run a otra. Mirar en la literatura que es exactamente hall of fame. TODO: Para comparar los modos de entrenamiento, intentar que todos los algoritmos se ejecuten durante la misma cantidad de tiempo (aunque tengan tamaños de poblaciones diferentes). TODO: Lanzar sin el hall of fame para evitar las diferencias entre runs.

8.1. Configuración de los experimentos

8.2. Métricas de evaluación

Capítulo 9

Resultados y discusión

9.1. A nivel de población

9.1.1. Evolución del fitness

9.1.2. Pesos medios

9.1.3. Pesos a lo largo de las generaciones

9.1.4. Variación de los pesos finales

9.2. A nivel de líderes

9.2.1. Evolución del fitness

9.2.2. Pesos medios

9.2.3. Pesos a lo largo de las generaciones

9.2.4. Variación de los pesos finales

Capítulo 10

Conclusiones

10.1. Objetivos alcanzados

10.2. Líneas de trabajo futuro

Bibliografía

[1] CD Projekt Red. The witcher 4 unreal engine 5 tech demo 4k | state of unreal | unreal fest orlando. URL: https://www.youtube.com/watch?v=aorRfK478RE.

[2] Universidad Europea. ¿qué es mesh en el desarrollo de videojuegos? | blog CC. Section: Blog. URL: https://creativecampus.universidadeuropea.com/blog/meshes/.

[3] Epic Games. Behavior tree in unreal engine - overview | unreal engine 5.6 documentation | epic developer community. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/behavior-tree-in-unreal-engine---overview.

[4] Epic Games. ML deformer framework in unreal engine | unreal engine 5.6 documentation | epic developer community. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/ml-deformer-framework-in-unreal-engine.

[5] Colin Gaudreau and Andreas Lasses. Game AI summit: No brakes! machine learning vehicles in 'star wars outlaws'. URL: https://gdcvault.com/play/1035556/Game-AI-Summit-No-Brakes.

[6] Majid Ghasemi, Amir Hossein, and Dariush Ebrahimi. Comprehensive survey of reinforcement learning: From algorithms to practical challenges. URL: https://arxiv.org/html/2411.18892v2#bib.bib84.

[7] Juan Julián Merelo Guervós. JJ/plantilla-TFG-ETSIIT. original-date: 2019-03-19T11:58:46Z. URL: https://github.com/JJ/plantilla-TFG-ETSIIT.

[8] Damian Isla. Managing complexity in the halo 2 AI system. URL: https://www.gdcvault.com/play/1020270/Managing-Complexity-in-the-Halo.

[9] Orkin Jeff. GDC vault - three states and a plan: The AI of f.e.a.r. URL: https://gdcvault.com/play/1013282/Three-States-and-a-Plan.

[10] Mike. Game programming concepts–finite state machines. URL: https://gamefromscratch.com/game-programming-concepts-finite-state-machines/.

[11] Raymond Redheffer. A machine for playing the game nim. 55(6):343–349. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00029890.1948.11999249. doi:10.1080/00029890.1948.11999249.

[12] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall. URL: https://aima.cs.berkeley.edu/global-index.html.

[13] Jose Torres-Jiménez and Juan Pavón. Applications of metaheuristics in real-life problems. 2(4):175–176. doi:10.1007/s13748-014-0051-8.

[14] Wikipedia. Artificial intelligence in video games. Page Version ID: 1292216335. URL: https://en.wikipedia.org/w/index.php?title=Artificial_intelligence_in_video_games&oldid=1292216335.

[15] Wikipedia. Diseño de juegos. Page Version ID: 167396849. URL: https://es.wikipedia.org/w/index.php?title=Dise%C3%B1o_de_juegos&oldid=167396849.

[16] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. 1(1):67–82. URL: https://ieeexplore.ieee.org/document/585893, doi:10.1109/4235.585893.