Trabajo Final – Inteligencia Artificial I

2022

# Resumen

# Introducción

La búsqueda de caminos óptimos o *pathfinding* es un problema frecuente en múltiples dominios. Por ejemplo, en el sector de los videojuegos, principalmente aquellos videojuegos donde el entorno tiene forma de tablero, se suelen utilizar algoritmos de búsqueda de caminos para determinar el camino más corto para mover objetos de una posición inicial a una cierta posición final. Uno de los algoritmos de búsqueda de caminos es el algoritmo A-estrella (**A\***).

El algoritmo A\* utiliza una función de evaluación que toma en cuenta el valor heurístico del nodo a evaluar desde el actual, *n*, y el coste real del camino recorrido para llegar a dicho nodo *n*, desde el nodo inicial. El algoritmo es una combinación entre búsquedas del tipo primero en anchura con primero en profundidad.

En este trabajo, se desarrollará en Python un laberinto 2D basado en un tablero dividido en casillas. El usuario podrá seleccionar la posición inicial (punto de partida), final (objetivo) y podrá colocar barreras que servirán de barreras al algoritmo de búsqueda. El algoritmo A\* será utilizado para encontrar el camino óptimo (si existe) que permita trasladarse desde la posición inicial hasta la posición final con el menor costo. A su vez, se planea desarrollar también un algoritmo de generación aleatoria de laberintos.

# Estado del arte

Numerosos estudios tratan sobre la utilización del algoritmo A\* para la búsqueda de caminos óptimos en el contexto de un laberinto 2D. El articulo realizado por Ade Candra et al. analiza la utilización directa del algoritmo en este contexto, y muestra su lógica de funcionamiento en diagrama de bloques y su implementación en pseudocódigo, lo cual facilita su desarrollo en un entorno real de programación (Python). Xiang Liu et al. propone modificaciones a la función heurística y compara la calidad de las soluciones encontradas para laberintos “perfectos”, es decir, sin bucles ni regiones inaccesibles.

En cuanto a la generación automática de laberintos, el artículo de Peter Gabrovšek analiza comparativamente seis métodos de generación de laberintos. El autor propone inspeccionar el número de caminos sin salida, así como también, el número de casillas visitadas por diferentes agentes cómo método de caracterización de la dificultad del laberinto obtenido.

# Especificación del agente

# Diseño del agente

## Algoritmo de búsqueda A\*

## Generación automática de laberintos

Los algoritmos de generación de laberintos inspeccionados comienzan con una grilla rectangular que representa al propio laberinto. Generalmente, todas las casillas son consideradas como paredes y los algoritmos convierten paredes específicas en casillas transitables para formar el laberinto [1]. Otros algoritmos trabajan a la inversa, es decir, consideran inicialmente todas las casillas como transitables y luego van añadiendo paredes progresivamente. Algunos algoritmos incluso son capaces de trabajar de las dos formas. En general, los laberintos pueden tener bucles, pero en este caso nos concentraremos en aquellos laberintos simples.

En nuestro caso, consideraremos al laberinto como una *grilla* compuesta por *casillas* Las casillas están conectadas entre sí por medio de *vías*. El laberinto puede ser representado matemáticamente por medio de un *grafo*. Esta representación es intuitiva y permite escalar los algoritmos de generación a laberintos de mayor tamaño [2].

Se define **laberinto ideal** a aquel que respeta las siguientes restricciones:

1. El laberinto no posee bucles.
2. El laberinto no posee zonas inaccesibles.
3. El costo de moverse entre casillas adyacentes es siempre igual a 1.

### Randomized Prim’s Algorithm

Este algoritmo puede trabajar añadiendo paredes o casillas transitables a la estructura inicial.

# Desarrollo de la solución

# Ejemplo de aplicación

# Resultados

# Conclusión

# Referencias bibliográficas

[1]Gabrovšek, P. (2019). Analysis of maze generating algorithms. IPSI Transactions on Internet Research, 15(1), 23-30.