Tarea 1: Escape del Laberinto Mutante

Gabriel Castillo, Diego Alday, Marcos Martínez

Departamento de Ingeniería Civil Informática y Ciencias de la Computación,

Facultad de Ingeniería

Universidad de Concepción

Inteligencia Artificial

Profesora: Mabel Vidal

29 de septiembre de 2025

Búsqueda Clásica: A*

En el equipo, ya se tenía conocimientos de varios algoritmos de búsqueda no informada, gran parte de éstos se habían visto en la asignatura de Estructuras de Datos. En cambio, algoritmos de búsqueda informada no se habían implementado aún y son “relativamente nuevos” para los integrantes, ya que es el primer acercamiento a éstos.

Es por ello, que se escogió implementar el algoritmo de búsqueda A*, ya que es una solución que implementa costos de camino y heurísticas, elementos que fueron novedosos para el equipo a la hora de decidir qué algoritmo implementar.

La implementación de A* es óptima ya que se creó con la heurística ‘Manhattan’, es decir, solo movimientos ortogonales (arriba, abajo, izquierda, derecha), con ello aseguramos que siempre entregue el camino más corto a la meta. Luego, teniendo el valor de la heurística (h) se suma al valor del costo de la casilla (g) y se puede obtener la función de costo total (f).

$$f(n) = g(n) + h(n)$$

Así, el algoritmo explora las celdas con el menor valor f(n).

También cuenta con un método *actuar* el cual se encarga de “activar” la búsqueda con la lógica de A* y lo guarda en una variable *camino*. Este método ayuda a recalcular cada vez que el laberinto cambia, ya que en el modulo principal está la variable booleana *cambio* que se encarga de avisar a este método si se debe actualizar la búsqueda o no.

Búsqueda Genética

El algoritmo genético (o búsqueda genética) es una búsqueda heurística inspirada en el proceso de evolución natural y la genética. Se utiliza para encontrar soluciones aproximadas a problemas complejos.

1. Se comienza con un conjunto de posibles soluciones (los cromosomas) generadas aleatoriamente.
2. Se evalúa la aptitud de cada solución para resolver el problema.
3. Se eligen las soluciones más aptas para la reproducción.
4. Las soluciones elegidas se combinan (cruce) y se genera una ‘mutación’ para crear una nueva generación de soluciones.
5. La nueva población reemplaza la antigua.

Todos estos pasos se repiten durante un número de generaciones hasta que se encuentra una solución o se cumple un criterio de parada.

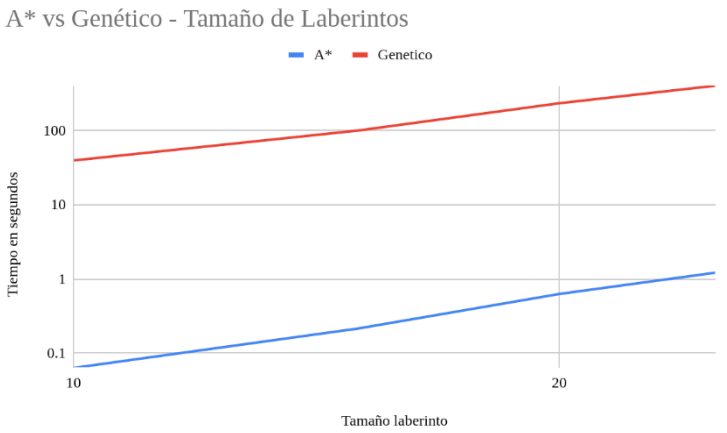
Para la implementación, se definió una lista de genes como: U, D, L, R, representando los movimientos posibles de arriba, abajo, izquierda, derecha en el laberinto. Se inicia con una población de 500 (500 cromosomas por cada generación) y un máximo de 500 generaciones. Además, se implementaron varios métodos que imitan las secuencias de pasos mencionada anteriormente, pero el más importante es la función de aptitud (conocida internacionalmente

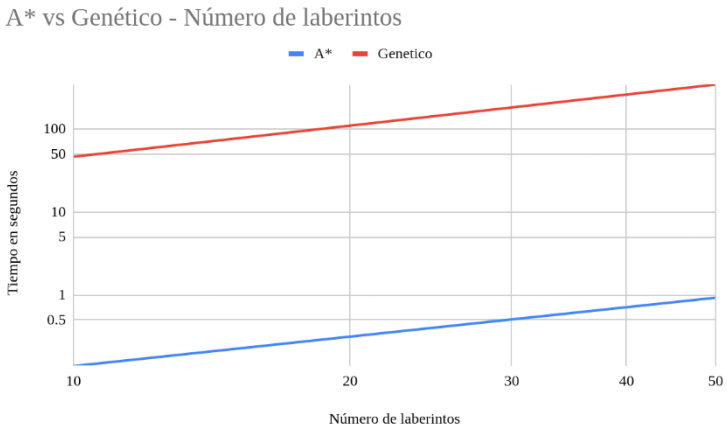
como *fitness function*). Ésta mide qué tan buena es la solución mediante un sistema de puntuación, se dan más de 2000 puntos por llegar a la meta y se castiga con -50 si se sale de los límites del laberinto para que el agente aprenda los caminos válidos.

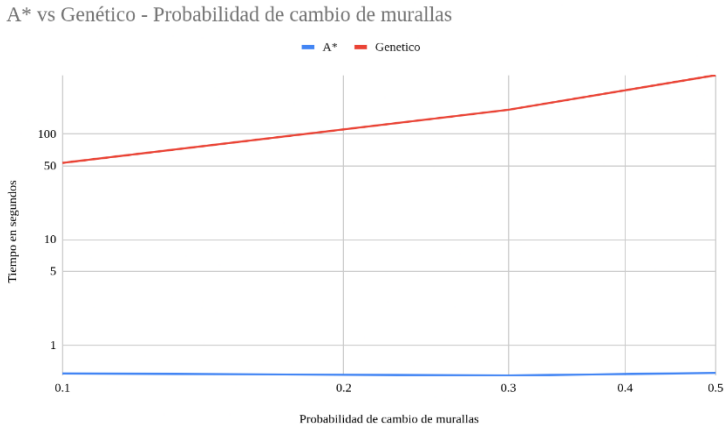
Resultados

A continuación, se presentan los gráficos de las tres pruebas que se implementaron; cantidad de laberintos variables, tamaño del laberinto variable y diferentes probabilidades de cambios en las murallas (un laberinto cada vez más dinámico). Todos están en escala logarítmica, ya que la disparidad en los tiempos entre A* y Algoritmo Genético era en demasía.

Toda la implementación de los algoritmos y pruebas se encuentran en el repositorio de Diego Alday: [BUG25/TAREA-1-IA: TAREA 1 INTELIGENCIA ARTIFICIAL 2025-2]







Reflexiones

En todos los casos de pruebas el A* sale victorioso. Este fenómeno se le atribuye a la estrategia que sigue el algoritmo, en donde solo vuelve a trazar el camino hasta alguna salida si alguna pared móvil le ha bloqueado el camino ya calculado. Por otro lado, el algoritmo genético ha presentado una mayor sensibilidad al aumento en el número de laberintos, la probabilidad de cambio de las paredes y el tamaño del laberinto ingresado. En este caso se le atribuye como causante a la gran cantidad de procesos iterativos que debe de realizar el algoritmo, lo que aumenta su costo de ejecución.

A modo de cierre, se concluye que el algoritmo más apropiado para laberintos con paredes móviles es el A* debido a su eficiencia y estabilidad. No obstante, el algoritmo genético podría llegar a ser más útil y flexible en situaciones en donde el entorno es incierto y se requiere optimización.