Escape del laberinto mutante

Matías Figueroa, Diego Rebollo, Daniel Támaro

Facultad de Ingeniería, Universidad de Concepción

Inteligencia Artificial

Prof. Mabel Vidal

30 de septiembre de 2025

Explicación del método de búsqueda escogido:

Como equipo, decidimos utilizar el algoritmo A* para el enfoque de búsqueda clásica. El algoritmo tiene un funcionamiento similar al algoritmo de *Dijkstra* y al algoritmo de *Best-First-Search*.

A grandes rasgos, lo que hace es comenzar desde el nodo origen (en este caso una celda en particular del laberinto, y ve todas las acciones que tiene disponibles, tras ello determina el "costo" asociado a cada una de estas acciones. En el caso del algoritmo A*, este valor corresponde a la suma entre el costo de llegar desde el origen hasta esa casilla y el costo estimado de llegar desde ese nodo hasta la salida. En nuestro caso, para la estimación de este último costo utilizamos la heurística de la distancia de Manhattan.

Cada una de estas acciones se agregan a una colección y en cada paso se elige la acción con costo más bajo, se retira de la colección de acciones y se expande nuevamente a partir ese nodo. Este proceso acaba cuando el agente llega a la casilla de salida o ya no quedan más acciones disponibles.

Para adaptarse a la naturaleza dinámica del laberinto, el algoritmo fue modificado de modo que, si una pared bloquea el camino calculado hacia la salida, este se descarta y se calcula una nueva ruta.

Explicación del algoritmo genético:

El algoritmo genético a grandes rasgos consiste en la generación de individuos los cuales pueden ser posibles soluciones para el problema en cuestión, a continuación, se explican las distintas partes de este algoritmo:

Individuos: Los individuos consisten en cadenas de 0's y 1's las cuales representan una secuencia de movimientos (Arriba: 00, Abajo: 01, Izquierda: 10 y Derecha: 11)

Fitness function: Para la fitness function la cual representa el puntaje que mide que tan bueno es un individuo se tienen los siguientes criterios:

- Para los individuos los cuales, si llegaban a la salida correcta del laberinto, se les asigna el valor de 1.0, el cual corresponde al valor máximo.
- Para los individuos que no llegan a ninguna salida se les asignaba un puntaje de acuerdo con que tan cerca quedaban de la salida.

- Para los individuos que llegan una salida falsa se les asignaba un puntaje de acuerdo con que tan cerca quedaban de la salida con una penalización del –50% por haber llegado a una salida falsa.

Para la medida con la que se verifica que tan cerca queda el individuo de la salida real se utilizó la distancia de Manhattan.

Single point crossover: Para realizar el crossover entre los individuos se eligen al azar dos individuos de la generación y se intercambiaban entre si un segmento de ellos mismos, luego mediante la fitness function se elige a aquel con el mejor puntaje.

Mutación: Luego del crossover, se realiza la mutación la cual simplemente consiste en cambiar el valor de un bit elegido aleatoriamente al individuo.

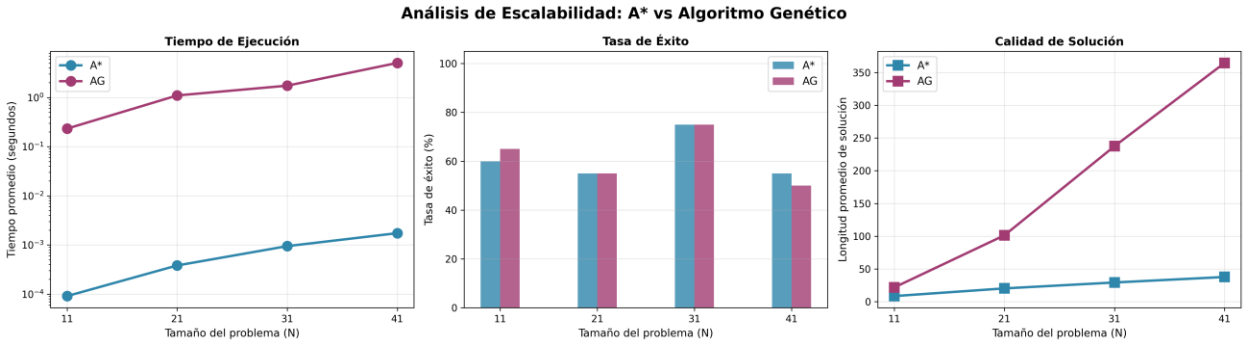
Elitismo: Para no desperdiciar o dañar individuos con buen puntaje se decidió por preservar a aquellos cuyo puntaje esté por encima de 9.0 dándoles un espacio asegurado en la generación siguiente.

Así, se va generando generación tras generación de estos individuos con el objetivo de mantener a los aquellos que llegan más lejos y desechar a los que van por caminos indeseados y así obtener al individuo correspondiente a la solución del laberinto.

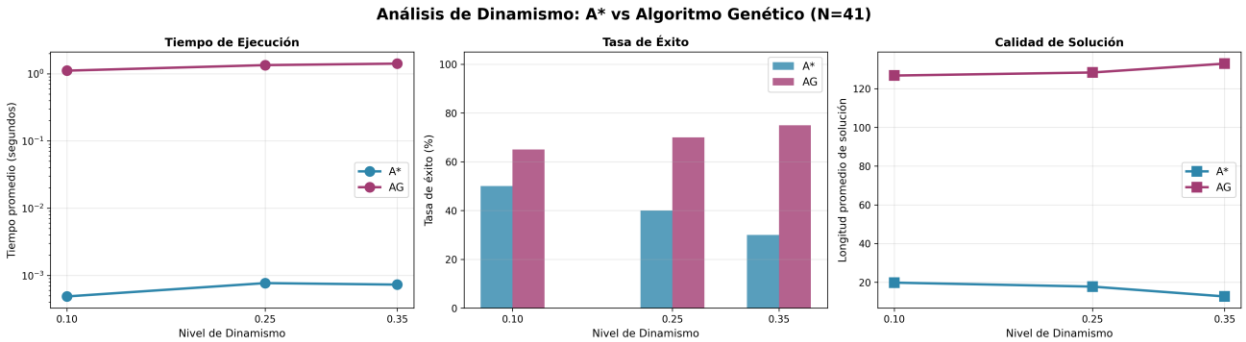
Tablas y gráficos comparativos:

A continuación, los resultados obtenidos en la experimentación:

Experimentación cambiando en el tamaño del laberinto manteniendo constante la probabilidad de que se mueva una pared (probabilidad del 5%).



Experimentación dados los cambios en el dinamismo del laberinto manteniendo constante el tamaño de este.



Reflexiones finales:

A través de este trabajo se ha podido determinar que el algoritmo A* ha tenido un desempeño muy superior en comparación a su contraparte genética en un ambiente con bajo dinamismo. En gran parte esto puede explicarse por la naturaleza estocástica del algoritmo genético, pudiendo llegar a tener diferencias de algunos ordenes de magnitud en tiempo de ejecución para un mismo “tamaño” de entrada.

Esta brecha en eficiencia se mantiene en escenarios altamente dinámicos. No obstante, en dichos entornos, el algoritmo genético demuestra una ventaja crucial, su tasa de éxito para encontrar una solución es significativamente mayor que la de A. Esto se debe a que el algoritmo genético, al trabajar con una población completa de posibles soluciones en lugar de una sola ruta como A, cuenta con alternativas diversificadas cuando el entorno cambia. Mientras A* debe recomenzar desde cero ante un bloqueo en su camino.