Taller 1 – Inteligencia Artificial

Daniel Vergara 202320392

Mariana Rodriguez 202421258

Martin de Angulo 202421628

a. Análisis

- Depth first search: El algoritmo dfs implementado tiene complejidad O(b^m) en tiempo, siendo b el factor de ramificación y m el máximo número de acciones, ya que en el peor caso explora todos los caminos hasta la profundidad máxima. Por otro lado, la complejidad en espacio es O(bm), porque solo almacena el camino actual y los nodos hermanos pendientes por revisar en la pila. En este caso este algoritmo es completo, ya que el espacio de estados no es infinito. Pero no es óptimo debido a que no minimiza el número de pasos ni el costo de su solución, por lo que puede y suele encontrar el objetico a través de caminos costosos.
- Breadth first search: El algoritmo bfs implementado tiene complejidad O(b^d) en tiempo y en espacio, siendo b el factor de ramificación y d la profundidad, ya que expande todos los nodos nivel por nivel hasta encontrar el objetivo y porque debe almacenar todos los nodos del nivel más profundo alcanzado. En este caso este algoritmo es completo, ya que de la misma manera que el dfs, está en un espacio finito. Pero no es óptimo, ya que no todas las acciones tienen el mismo costo, por lo que el algoritmo minimiza el número de pasos, mas no el costo de estos.
- Uniform cost search: El algoritmo ucs implementado tiene complejidad O(b^(C*/e)) tanto en tiempo como en espacio, siendo C* el costo optimo de la solución y e el costo mínimo por acción, ya que expande todos los caminos que tengan costo acumulado menor que el óptimo. En este caso este algoritmo es completo, ya que es un espacio finito y todos los costos son positivos, lo que permite que siempre encuentre un camino para llegar al objetivo, si existe. Además, es óptimo, ya que garantiza encontrar el camino con menor costo.
- A estrella con SimpleSurvivorProblem: El algoritmo A* aplicado a rescate de un solo sobreviviente tiene complejidad O(b^d), siendo b el factor de ramificación y d la profundidad, ya que en el peor caso la heuristica no aporta información útil, por lo que A* se comporta como usc, mientras que cuando la heurística si aporta información útil A* recorre muchos menos nodos. En este caso este algoritmo es completo debido a que el espacio es finito, los costos de las acciones son positivos y la heurística de Manhattan y la Euclidiana son admisibles. Además, es

óptimo porque siempre encontrara el camino de menor costo hacia el objetivo, gracias a que las heurísticas son admisibles.

- A estrella con MultiSurvivorProblem y survivorHeuristic: El algoritmo A* aplicado a rescate de múltiples sobrevivientes el espacio de estados crece notoriamente según el número de sobrevivientes k, por lo que tiene complejidad O(b^k), siendo b el factor de ramificación. En este caso este algoritmo es completo, ya que el espacio es finito y los costos son positivos, y tarde o temprano encontrara un camino que rescate todos los sobrevivientes, si existe. Además, es óptimo porque garantiza el mínimo costo total para rescatar a los sobrevivientes gracias a que la heurística es consistente debido a que combina la distancia Manhattan al sobreviviente más cercano con el costo del árbol de expansión mínima sobre los sobrevivientes restantes sin sobreestimar el costo real restante.

b. Reflexión sobre uso de IA

Durante el desarrollo del taller se hicieron primero implementaciones propias de los algoritmos y de la heurística. Sin embargo, al momento de probarlas se identificaron algunos errores por lo que se utilizó la IA Claude en funciones puntuales para revisarlos y mejorar la eficiencia, especialmente en Uniform Cost Search, el cálculo del MST y la heurística de rescate múltiple. En todos los casos ya existía una versión inicial, no obstante, con el apoyo y uso de la IA se logró entender mejor la lógica y llegar a optimizaciones y correcciones útiles para el programa.

Uno de los principales aprendizajes fue entender con mayor claridad el manejo del costo acumulado en UCS y la necesidad de estructuras como best_cost para evitar ciclos y expansiones de búsqueda redundantes. También se identificaron errores conceptuales que no eran evidentes al inicio, como el uso incorrecto de isGoalState y una lógica de visitados invertida que impedía encontrar la solución en mapas grandes y escenarios más complejos.

En la survivorHeuristic, la versión inicial solo consideraba la distancia al sobreviviente más cercano. Con ayuda de la IA se comprendió mejor cómo incorporar el costo del MST de los sobrevivientes restantes, lo que permitió entender que una heurística debe estimar el costo total del problema y no solo el siguiente paso. Además, se corrigió el cálculo del MST para que funcionara correctamente y pudiera integrarse de forma adecuada en la heurística.

El uso de la IA resultó especialmente útil para aclarar la lógica de los algoritmos y detectar pequeños errores que impedían obtener el resultado esperado. La IA funcionó principalmente como una herramienta de verificación y mejora más que como

generadora de soluciones. Consideramos que su uso para refinar el código es valioso y útil ya que facilita identificar detalles que inicialmente no eran evidentes y permite llegar a una mejor comprensión del problema y por ende a mejores soluciones.