Trabajo Práctico 1

Algoritmos de búsqueda

Autores

Nagelberg, Martin (56698) mnagelberg@itba.edu.ar

Grethe, Juan (57370) jgrethe@itba.edu.ar

Grabina, Martin (57360) mgrabina@itba.edu.ar

72.27 - Sistemas de Inteligencia Artificial

Prof. María Cristina Parpaglione

9 de septiembre de 2019

Introducción

El objetivo del trabajo fue la implementación de un engine que pudiera resolver problemas genéricos a partir de distintos algoritmos de búsqueda informada y desinformada y la implementación de el juego gridlock puzzle para la prueba del mismo. Se evaluaron distintos métodos para poder llegar a la solución, utilizando algoritmos de búsqueda informada y desinformada. Aquí se exponen tanto los resultados de los mismos, como las conclusiones extraídas a partir de ellos.

Problema

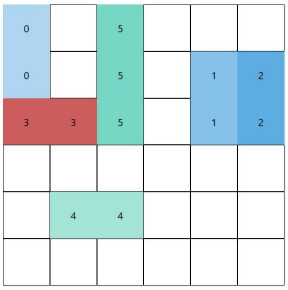
El gridlock es un puzzle en el cual debemos intentar mover una pieza principal hacia la salida, teniendo en cuenta que esta puede estar bloqueada por piezas secundarias. Las piezas se caracterizan por ser verticales (que se pueden mover de forma vertical) y horizontales (que se pueden mover de forma horizontal), y siempre se mueven de a una unidad.

Clasificación de casos estudiados

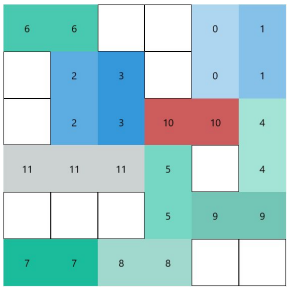
Para la realización de este trabajo práctico, tuvimos en cuenta tres tipos de tableros, ordenados por dificultad. Para determinar dicha dificultad, utilizamos una función sumamente aproximada

$$dificultad(tablero) = \#fichas(tablero) + \#fichas_{obstruyendo}(tablero)$$

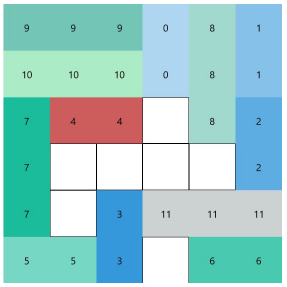
dif(t_facil) = 8



dif(t_medio) = 14



dif(t_dificll) = 15



Función de costo

La función de costo en este caso representa la cantidad de movimientos de una ficha en cualquier dirección. Cada vez que se realiza un movimiento, al costo se le suma una unidad.

Heurísticas

Aquí se presentan dos heurísticas admisibles, una básica y una media.

1) **Heurística básica**: se encarga de contar la distancia entre la ficha principal y la salida del puzzle. Tal como lo dice su nombre, es absolutamente básica, optimista y también admisible, pues siempre subestima a la óptima.
2) **Heurística media**: además de contar la distancia entre la ficha principal y la salida, se suman la cantidad de piezas que están bloqueando dicho camino. En este caso, también estamos hablando de una heurística admisible, pues sigue subestimando a la óptima. Podemos decir que esta es mejor que la primera, ya que al ser las dos admisibles, debemos tomar aquella que es mayor para que se acerque más a la realidad.
3) **Heurística avanzada**: además de contar lo que cuenta la anterior, en este caso, también estamos sumando la cantidad de movimientos que tienen que hacer las piezas que bloquean a la principal para desbloquearlas. A su vez, hicimos dos sub funciones, una que toma los casos optimistas (se queda con la menor cantidad de movimientos entre ir para abajo e ir para arriba) y otra que toma los pesimistas (máximo). La primera es admisible y encontramos que es superior a las otras dos, mientras que la segunda es inadmisible. En el tablero medio y difícil la le inadmisible nos dio la misma solución que la admisible, mientras que en el mas facil nos dio una solución con el mismo depth pero menos explosiones.

Métricas

Las métricas que tuvimos en cuenta a la hora de comparar los resultados de los distintos algoritmos fueron las siguientes:

- Cantidad de explosiones: Una menor cantidad de explosiones indica una menor cantidad de nodos analizados.
- Nodos frontera: Una menor cantidad de nodos en la frontera indica también una menor cantidad de nodos analizados.
- Costo de la solución: Quien sea la solución óptima deberá contar con el menor de los costos.
- Total elapsed time: Esta métrica es más bien interna, pero no ofrece ningún tipo de información muy relevante.

Algoritmos de búsqueda

BFS

En cualquiera de las dificultades, en caso de existir solución, este algoritmo nos da la solución óptima, pues debe expandir el árbol de posibilidades completo y lo va realizando por niveles. De esta forma la primera solución que se encuentre tendrá menor altura, y por tanto, menor cantidad de movimientos. Sin embargo, la cantidad de memoria utilizada es bastante elevada.

DFS

Este algoritmo fue el más rápido para determinar si el problema tenía solución o no. En ninguno de los casos la solución encontrada fue la mejor. Su consumo de memoria no fue tan elevado.

IDDFS

En este caso, se puede observar que la solución que se encuentra siempre es la óptima, pues su funcionamiento es similar a BFS en cuanto a forma de recorrer. Sin embargo, su consumo de memoria es inferior, pues no tuvo que armar todo el árbol hasta ese punto y mantenerlo guardado.

Greedy

Aquí ya entran mucho en juego las funciones heurísticas creadas. En todos los casos la búsqueda fue absolutamente rápida. Sin embargo, este algoritmo no asegura encontrar la solución óptima y de hecho, en un solo caso la encontró (tablero básico + heurística básica).

A*

En todos los casos encontramos que este algoritmo garantiza el camino de menor costo, siempre y cuando la heurística utilizada fuera admisible. Presenta una amplia mejora en nodos explorados frente al resto de los algoritmos, excepto frente a la técnica voraz. Sin embargo, esta última, como ya fue mencionado previamente, no garantiza el mejor costo. Tanto este algoritmo como el anterior, tienen un pequeño overhead para realizar el sort por f(x).

Conclusiones

La realización del trabajo fue sumamente interesante para obtener los siguientes resultados:

- Para determinar si un problema tiene o no solución, el algoritmo que resultó ser más adecuado fue DFS.
- Para encontrar el camino óptimo, encontramos que teniendo heurísticas buenas y admisibles, la mejor opción es A*. Sin embargo, en caso de no poder armar una función de estimación, la mejor opción sería IDDFS que explota menos nodos que BFS.
- Para encontrar una solución rápida, si podemos prescindir de la mejor, una gran opción es utilizar greedy dado que su velocidad es superior al resto.