Homework 3

Preguntas teóricas

Pregunta 3: Antes de resolver el problema, ¿sabes cuál es el factor de ramificación “b\_level1” del nivel de profundidad 1 del árbol de búsqueda? ¿y de los niveles de profundidad 2 y 3 (“b\_level2” y “b\_level3”)?

El factor de ramificación para el nivel inicial es tanto como ciudades tengamos - 1 porque ya nos encontramos en una y no podemos ir a ella, así que podemos acudir a N ciudades - 1. Por cada nivel que se profundice el nivel de profundidad de estos será 1 menos por cada nivel que hayamos profundizado, así para el nivel 2 sería N ciudades -3, y para el nivel 3 serían N ciudades -4, para el nivel 1 sería N ciudades - 2.

Pregunta 4: Antes de resolver el problema, ¿sabes cuál es la profundidad “d” del árbol de búsqueda?

La profundidad d o profundidad mínima viene establecida por lo mínimo que necesitaremos profundizar para poder acabar el problema. Inicialmente tenemos N ciudades, que en el XML son 7 y un origen y un destino. Nada más empezar tenemos que ya nos encontramos en el origen, sin habernos desplazado y quedan por delante 8 movimientos , luego la profundidad mínima sería de N ciudades -1.

Pregunta 5: Considera la búsqueda “Best First Search” para resolver el problema. ¿Qué heurística utilizarías? Justifica tu respuesta y garantiza que tu heurística es ADMISIBLE.w

La heurística nos tiene que permitir estimar el coste para llegar al estado final. El coste de llegar al estado final vendrá determinado por la distancia que recorramos entre puntos, una forma que tenemos de estimar dado que podemos suponer que un nodo más cerca del destino es mejor que uno que no lo está, como heurística podría ser admisible el coste de ir desde la posición actual al destino directamente, esto hace un calculo mínimo de la distancia para llegar luego es la más optimista lo que la vuelve admisible. Así distinguimos que este nodo es mejor que otros en igual profundidad, aunque en ocasiones pueda tomar decisiones incorrectas, le permite hacer una aproximación muy precisa con un cálculo bastante rápido para la mayoría de las ocasiones.

Análisis del rendimiento del sistema

Estudio de la susceptibilidad de tu sistema a la ordenación de los operadores en la lista de operadores de tu problema. Explica tus observaciones incluyendo lo robusta que es tu heurística respecto a la ordenación de los operadores. Sugerencia: puedes utilizar el método shuffle() de Collections para reordenar los operadores de forma aleatoria.

He establecido la reorganización de los operadores y esto genera que como el orden de los operadores es diferente, los algoritmos de Depth First y Breath First tengan distintos resultados de una ejecución a otra, esto se debe a que buscan el primer resultado que encuentren cada uno con su forma de buscar y al aplicar las operaciones en distinto orden va yendo a distintos destinos. Esto hace que el coste en tiempo de ejecución de cada uno no varíe mucho de uno a otro porque siempre se aplica similar a pesar del orden de los operadores, pero si varía el coste del algoritmo y el orden en el que recorre las ciudades

En cambio en el de Best First no varía el resultado final, pero el cambiar los operadores hace que se de una mayor fluctuación en los tiempos de ejecución, dado que el cambio de orden de los operadores da lugar a distintos destinos y puede facilitar o dificultar encontrar la mejor ruta aproximada.

Estudio de la calidad de la solución obtenida (de acuerdo a la medida de rendimiento descrita como parte de la 1.a entrega) y el tiempo utilizado (o complejidad computacional temporal) por tu sistema; identificando si es posible mejorar alguno de ellos sin comprometer el otro. Si es posible indica cómo.

He tratado de mejorar los tiempos de ejecución simplificando las comparaciones, clonaciones y el isApplicable, haciendo un return false; cuando sepa que ya no se puede ejecutar. En la comparación de ciudad he comparado las coordenadas X e Y de enteros en lugar de hacer la operación de comparar cadenas que siempre me parece que es peor y además no me parece tan claro, dos ciudades podrían llamarse igual y situarse en distintos puntos. Esto me ha permitido ganar algo de rendimiento, pero podría mejorarse más.

* Depth First: 6ms de media
* Best First: 749ms de media
* Breadth First: 6s 272ms de media

Podemos mejorar el rendimiento si la lista de operadores está ordenada por cercanía en función de inicio, esto permite al Breath FS sacar un coste mucho más óptimo, no en tiempo sino en resultado mejor.