Algoritmo de optimización de Grafos

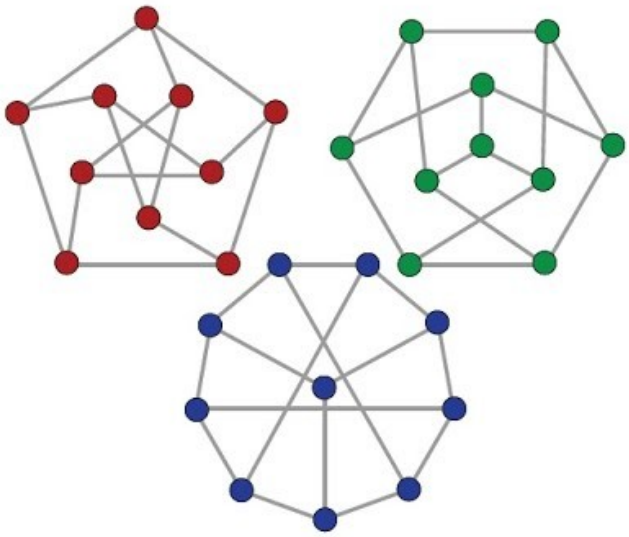
Jaime de Castro 14708: Código de los algoritmos

Guillermo Moreno 16289: elaboración y organización

Alejandro Pérez Vicente 16335: Toma de datos y análisis

Introducción: La teoría de optimización de grafos estudia el cálculo del camino optimo entre dos nodos de un grafo, con múltiples nodos organizados con distancias diferentes.

Es muy usada en las matemáticas y las ciencias de la computación que estudia las propiedades de los grafos, especialmente en el entorno de las redes.



Los grafos pueden ser de cualquier dimensión y geometría, por tanto suelen ser bastante complejos conceptualmente.

En nuestro caso práctico nos limitaremos a la obtención del camino óptimo de un punto del grafo a otro mediante el uso de dos algoritmos: uno recursivo que hemos desarrollado frente a una implementeacion adaptada del algoritmo de Dijkstra(muy popular).

Además tendremos en consideración el tamaño de los datos que utilicemos en los algoritmos ,es decir, dimensión del grafo, distancia mínima entre nodos, numero de iteraciones, etc.

La herramienta principal que vamos a usar para la evaluación de experimentos es el tiempo de procesamiento que se empleará para la resolución del algoritmo.

Hay que tener en cuenta que los resultados son únicamente de carácter comparativo, ya que el código no está extensamente optimizado(se mide el tiempo de ejecución del algoritmo completo, incluyendo su inicialización)

Análisis del experimento:

Para simplificar el tratamiento de los datos, consideramos la representación del grafo en una matriz de conectividad. Dicha matriz se caracteriza por:

- Reflejar la distancia de un nodo a otro.
- Tener los elementos de la diagonal principal nulos, puesto que no tiene sentido que haya distancia de un nodo al mismo nodo.
- Ser simétrica, ya es razonable que la distancia del nodo1(elemento[i][j]) al nodo2(elemento[j][i]) es la misma que del nodo2 al nodo1.

El parámetro básico que se mide es, como hemos dicho antes, el tiempo total de ejecución del algoritmo. Para ello utilizamos la librería de c++ <chrono> ,permitiéndonos tomar el intervalo de tiempo transcurrido durante la ejecución del algoritmo.

Las distancias entre nodos están representadas en la matriz mediante números del 0 al 9. Podemos suponer

por ejemplo que ese número son los metros de un nodo a otro, pero también se podria haber supuesto que son kilómetros u otra medida distinta. Luego el significado de los valores de la matriz depende de como queramos definirlo. También podriamos aumentar fácilmente el límite de estos valores para que puedan ser superiores a 9.

El numero de nodos esta definido en DIM (variable solo modificable al compilar el programa).

Cuantos mas ceros incluya la matriz (sin tener en cuenta los de la diagonal), mas dispersa es. Esto implica que hay menos conexiones entre nodos, es decir, no todos los caminos son posibles. La dispersión de la matriz se puede modificar a partir de X (variable solo modificable desde el editor de texto del programa).

Como ya hemos visto, el número de nodos y el numero de conexiones son los principales factores que pueden afectar a la resolución del grafo. Por tanto, probaremos los algoritmos desarrollados (el dijkstra y el recursivo) modificando dichas condiciones. El programa cuenta previamente con una generación de matrices aleatorias que cumplan con las caracteristicas citadas en la introducción, seguido de la solucion de esa matriz por ambos algoritmos. Esto permite realizar un experimento distinto a cada ejecución del programa.

El orden de complejidad de los algoritmos es de

$O(V^2)$ para Dijkstra ($O(V*logV +N)$ si se dan los nodos ordenados)

$O(V!)$ para el recursivo.

Donde V es el numero de vértices(enlaces entre nodos) y N el número de nodos

Esto nos permite estimar la relación entre tiempos de ejecución, queprobablemente sean mucho mas reducidos para Dijkstra para órdenes altos

La tabla con los experimentos probados y sus resultados se adjunta a continuación.

DIM	Dispersion	tiempo Dijkstra	tiempo recursivo
8	15	8817	2784
8	20	6640	2903
16	15	13010	7772
16	20	10534	7228
32	15	23680	26474
32	20	28015	25623
64	15	88052	87952
64	20	77852	90948
256	15	549226	668957
256	20	463852	689516

Conclusiones:

En primer lugar se cumplen las expectativas ya que el algoritmo de dijkstra es ligeramente superior en altos órdenes. No obstante nuestro algoritmo se comporta relativamente bien a bajos y medios ordenes.

Otro detalle interesante es que el algoritmo recursivo es mucho mas estable en el tiempo de ejecucion (desviación típica del 13% frente a un 28% de dijkstra)