Longest and shortest paths algorithm

1. Johnson's algorithm

Este se centra en la ruta más corta de todos los pares. Este toma un gráfico con vértices y aristas con el fin de generar la ruta más corta entre cada par de vértices del gráfico. Además, este funciona en gráficos dirigidos y ponderados. Cabe mencionar que dicho gráfico soporta peso negativo debido a las subrutinas.

Este algoritmo posee 3 pasos:

1. Se agrega un nuevo vértice al gráfico y se conecta a todos los bordes con 0 peso a todos los demás vértices del gráfico.
2. Todos los bordes pasan un proceso de ponderación que elimina los bordes de peso negativo.
3. El vértice agregado del paso a se elimina y se corre el algoritmo de Dijkstra’s en cada nodo de la gráfica.

2ventajas:

1. Permite que algunos de “peso” de números sean negativos pero no pueden existir ciclos de “peso” negativo.
2. Podría ser más rápido que Floyd-Warshall en gráficos dispersos.

Aplicaciones:

Se podría utilizar en un taller de flujo de dos máquinas.

Pseudocódigo:

Paso #1:

Crear G’ donde G’.V(vértice) = G(grafo).V, +{s}

G’. E =G.E +((s,u) for u in G.V) y weight(s,u) = 0 for u in G.V

Paso#2:

If algoritmo == FALSE

Return “Ciclo ponderado negative”

Else:

Para vértice v in G’.V:

F(v) = distance(s,v)

For edge(u,v) in G’.E:

Weight’(u,v) = weight(u,v) +h(u) -h(v)

Paso #3:

D = nueva matriz f distances sin fin

For vértice v2 in G.V:

(algoritmo Dijkstra)

For each vértice v in G.V:

D\_(u,v) = distancia(u,v) +h(v) – h(u)

Return D

1. A\* search algorithm

Este algoritmo crea un árbol de ruta de menor “costo” desde el nodo inicio al nodo destino. Para cada nodo, este algoritmo utiliza una función que proporciona una estimación del costo total de la ruta que utiliza ese mismo nodo. Cabe mencionar que pasa los nodos en el orden del estimado heurístico.

1. Bellman-Ford algorithm

El algoritmo Bellman-Ford es una búsqueda para gráficos que encuentra el camino más corto entre vértice fuente y todos los demás vértices en la gráfica. Este puede ser utilizado tanto en gráficos ponderados como no ponderados. Dicho algoritmo opera en un gráfico de entrada con vértices y bordes. Adicionalmente, no se necesita proporcionar un vértice destino debido a que se calcula la distancia más corta a todos los vértices en la gráfica del vértice fuente.

El mismo algoritmo utiliza el principio d relajación para encontrar la longitud de las rutas cada vez más precisas.

2 ventajas:

1. Se puede minimizar el costo cuando se construye una red. Esto debido a que Bellman Ford encuentra el peso de ruta más corto desde un nodo fuente a otro. Así pues, no se necesita construir mucho enrautador para construir la ruta de un nodo a otro.
2. Puede maximizar el rendimiento del sistema. Este puede encontrar el mínimo de peso de la ruta. Este peso puede evitar demoras de propagación para un sistema dado.

Aplicaciones:

1. Ahorro de recursos de red
2. Routing Information Protocol

Pseudocódigo:

For v in V:

v.distance = sin fin

v.p = M¿None

source.distance = 0

for a from 1 to V:

for(u,v) in E(edges):

1. Viterbi algorithm

Este es un algoritmo de programación dinámica que busca encontrar la secuencia más probable de distintos estados escondidos que produce una secuencia observada de sucesos. Entre los pasos que incluye dicho algoritmo están: generar los datos para ser transmitidos, codificar, mapa de símbolos, entre otros. Además, el ruido hace que el algoritmo vaya cambiando y el algoritmo de Viterbi debe descifrar lo que fue cambiando.

1. Floyd-Warshalll algorithm

Este es capaz de calcular las distancias más cortas entre cada par de vértices en el gráfico de entrada. El algoritmo busca proveer la distancia óptima entre cada par de los vértices. Dicho es un ejemplo de programación dinámica, ya que el problema lo divide en subproblemas que combina las respuestas de los subproblemas para resolver el problema inicial. Como resultado, es más eficaz en gestionar múltiples paradas en la ruta porque es capaz e calcular las rutas más cortas entre todos los nodos más relevantes.

Referencias

<https://brilliant.org/wiki/a-star-search/#implementation>

<https://prezi.com/ume4etvnq4me/algoritmo-de-viterbi/>

<https://brilliant.org/wiki/bellman-ford-algorithm/#overview>

<https://brilliant.org/wiki/floyd-warshall-algorithm/#overview>

<https://brilliant.org/wiki/johnsons-algorithm/#algorithm-pseudocode>

<https://www.ijaiem.org/volume2issue7/IJAIEM-2013-07-23-079.pdf>