Problema NP-completo

k-MST

- Roberto Juan Cayro Cuadros
- Gabriel Alexander Valdivia Medina
- Giulia Alexa Naval Fernández
- Rodrigo Alonso Torres Sotomayor

01

Introducción

02

Demostración

03

Algoritmo Fb

04

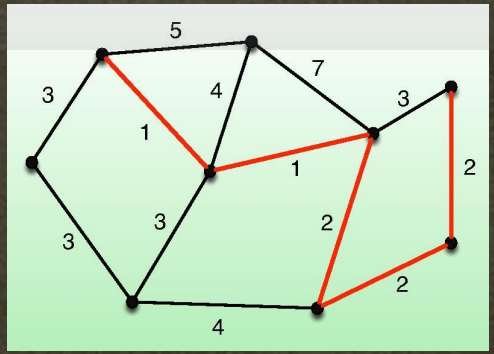
Algoritmo-aproximado

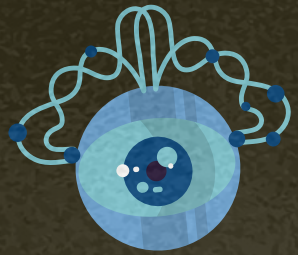
04

Aplicación

Introducción

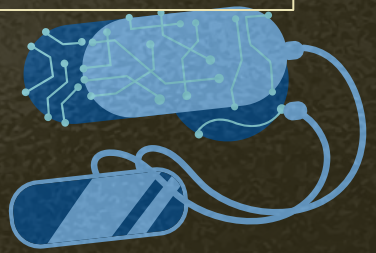
El k-MST o k-minnimum spanning tree problem, árbol de expansión de péso mínimo k en español es un problema computacional que pide un árbol de mínimo costo con exactamente k vértices que forme un subgrafo del grafo original.





Demostración

1. k-MST $\in$ NP
2. NP-completo α k-MST



1. k-MST $\in$ NP

- Desde t=0 hasta k, para cumplir con el número de vértices necesarios.
- Con la función ESCOGER seleccionaremos un vértice u.
- Si u no está en el árbol x lo añadimos.

Algoritmo no-determinístico

k-MST (G,k)

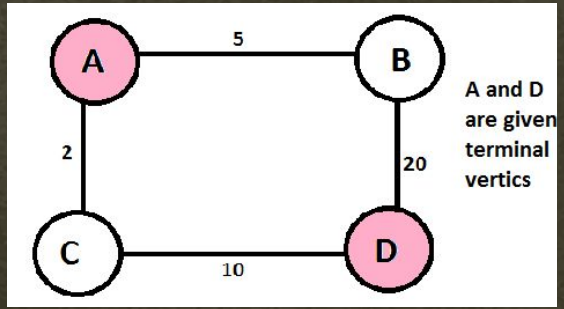
```
1. x ← árbol.
2. t ← 0
3. while t < k
4.       do u ←ESCOGER(G)
5.             if u is not in x
6.                   do x . add(u)
7.                      t++
```

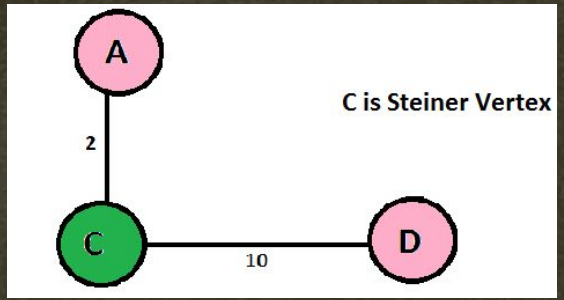
2. NP-completo α k-MST

Steiner-tree problem

(21 karp problems)

Dado un grafo no dirigido G y un sub-conjunto de vértices S del mismo grafo, se debe generar un árbol de el menor costo posible usando los vértices de S. Es posible construir este árbol usando vértices originarios de G, pero que no estén en S, llamados Steiner-vértices.





Reducción

Steiner-tree

ENTRADA

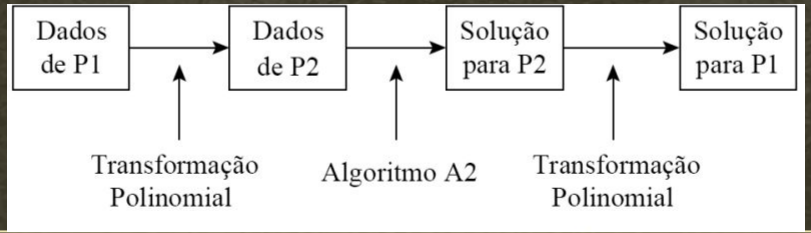
- Grafo no-dirigido G
- Sub-set de vértices S
- Número M

k-MST

ENTRADA

- Grafo no-dirigido G
- Número k

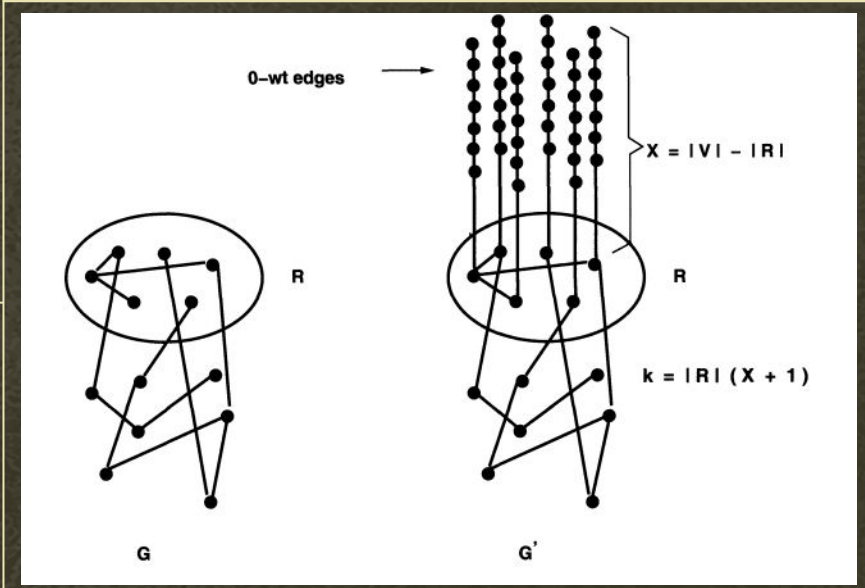
Es necesario modificar la entrada de Steiner hacia k-MST, para poder probar la reducción, a su vez que debe cumplirse.



Reducción

1. Añadir árboles con aristas de peso 0 a cada vértice de R.
2. Siendo |R| = al número de vértices en R.
3. Cada uno de estos nuevos árboles tendrá X vértices, donde X= V(G) - R
4. Ahora transformaremos el número k, k=R(X+1).

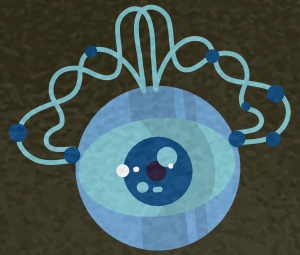
Transformación



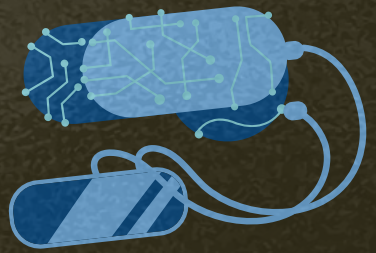
Reducción

Con esta transformación aseguramos:

1. La generación de un árbol en G´, dado un k modificado.
2. Puesto que los árboles añadidos a los vértices de S, son de peso 0, serán parte de la solución
3. Y dado que k será igual a el tamaño de los vértices añadidos +1, por la cantidad de los vértices en S. Los vértices de S, necesariamente deben ser parte de la solución de k-MST



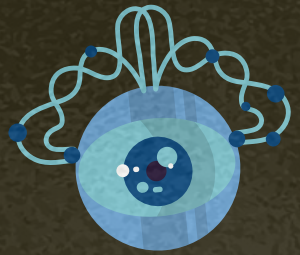
Algoritmo Fb



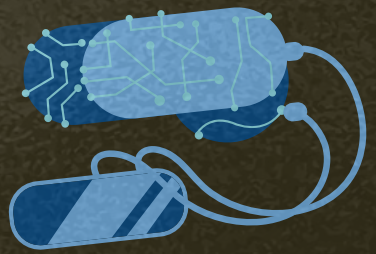
Prim modificado

```
int PrimsMST(int sourceNode, vector<adyacencias>& graph, int K)
{
    //Guardar detalles del nodo.
    priority_queue<valNode, vector<valNode>, greater<valNode>> k;
    int count = 0;
    vector<int> aux = { 0,sourceNode };
    k.push(aux);
    bool* nodesAdded = new bool[graph.size()];
    memset(nodesAdded, false, sizeof(bool) * graph.size());
    int mst_tree_cost = 0;

        while (count!=K) {
        // nodo mÃ-nimo
        valNode itemNode;
        itemNode = k.top();
        k.pop();
        int Node = itemNode[1];
        int Cost = itemNode[0];
        if (!nodesAdded[Node]) {
            mst_tree_cost += Cost;
            count++;
            if (count==K)
              break;
            nodesAdded[Node] = true;
            // iterar sobre nodos que se sacaron de la pq
            // se agregan los no aÃ±adidos
            for (auto &node_cost : graph[Node]) {
                int adjacency_node = node_cost[1];
                if (nodesAdded[adjacency_node] == false) {
                    k.push(node_cost);
                }
            }
        }
    }
    delete[] nodesAdded;
    return mst_tree_cost;
}
```



Algoritmo aproximado



K-MST ∈ NP-Hard

1. El problema de K-MST es NP-Hard y, por tanto no se puede (o muy difícilmente) resolver en tiempo polinomial. Además de que el número de posibles soluciones crece de manera exponencial.

2. Un algoritmo que provea una solución “lo suficientemente buena” es necesario.

Branch and Bound

Algoritmo en que se enumeran las soluciones y se descarta la mayoría de acuerdo a ciertos límites establecidos.
Princ. Ramificación: permite efectuar una partición de un subconjunto dado, de acuerdo a cierta propiedad detectar si una de las partes está vacía o no hay solución.
Princ. Acotación: La condición que debe cumplirse y dependerá de cada subconjunto.

Upper Bound

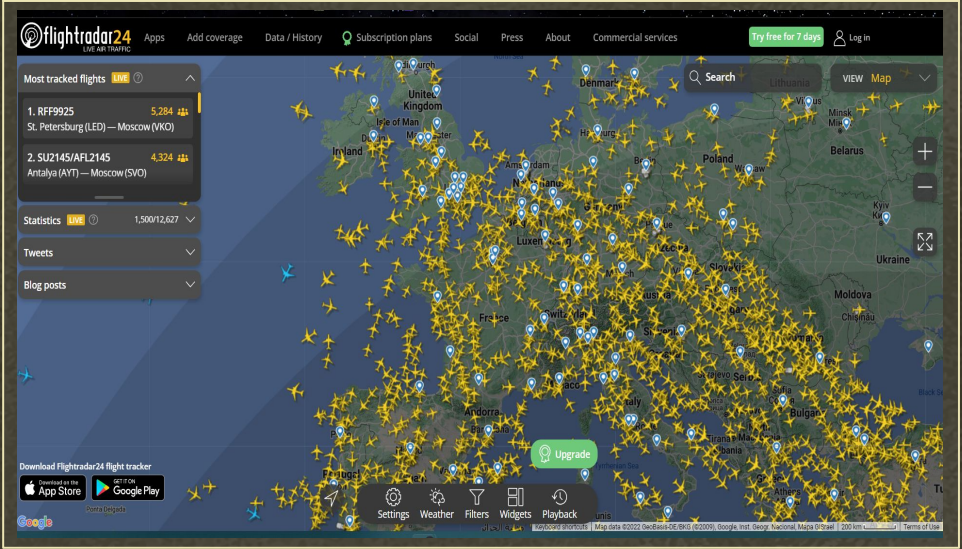
$$2* \mid V(G)|$$

Lower Bound

$$k-1-E(G)$$

Enumeration edge/weight ratio

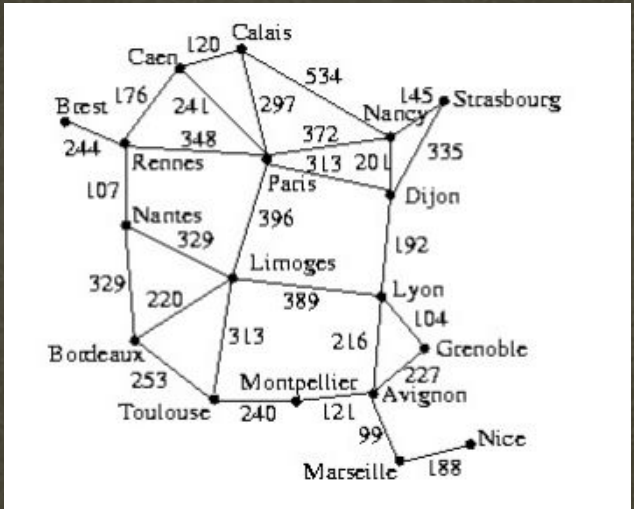
Aplicación en la sociedad



Realizar todas las escalas

Ahorrar combustible

Aplicación en la sociedad



Problemas de conexión

Construir red de menor costo

Gracias