Sollin's Algorithm for Minimum Spanning Trees

Rebecca Robinson

May 5, 2020

Definitions

Definition (Spanning Tree)

A **spanning tree** is a connected, acyclic subgraph that spans all nodes.

Definition (Minimum Spanning Tree)

A **minimum spanning tree** is a spanning tree with smallest total cost.

Applications

- Cluster analysis
  - We have a network and want to split into k clusters with total cost of clusters minimized
  - Take minimum spanning tree from algorithm and delete the $k-1$ arcs with the highest cost

History of Sollin's Algorithm

- Originally published by Boruvka in 1926
- Designed as a way of constructing an efficient electricity network in Moravia
- He published two papers with this algorithm
  - One was 22 pages and has been viewed as unnecessarily complicated
  - The other was one page
- Independently rediscovered by Choquet in 1938
- Again in 1951 by Florek, Lukasiewicz, Perkal, Steinhaus, and Zubrzycki
- And again in 1962 by Sollin

Boruvka's Paper

5.x.26. O. Borůvka.

ZVLÁŠTNÍ OTISK Z ČASOPISU „ELEKTROTECHNICKÝ OBZOR"

Roč. 15. Čís. 10. Praha III., Cihelná 102. 5. března 1926.

Dr. OTAKAR BORŮVKA:

Příspěvek k řešení otázky ekonomické stavby elektrovodných sítí.

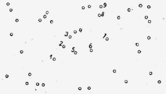
Ve své práci „O jistém problému minimálním"*) odvodil jsem obecnou větu, jíž jest ve zvláštním případě řešena tato úloha:

V rovině (v prostoru) jest dáno n bodů, jejichž vzájemné vzdálenosti jsou veskrz různé. Jest je spojiti sítí tak, aby:

1. každé dva body byly spojeny buď přímo anebo prostřednictvím jiných,
2. celková délka sítě byla co nejmenší.

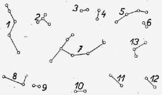
Jest zřejmé, že řešení této úlohy může míti v elektrotechnické praksi při návrzích plánů elektrovodných sítí jistou důležitost; z toho důvodu je zde stručně na příkladě vyložím. Čtenáře, jenž by se o věc blíže zajímal, odkazuji na citované pojednání.

Řešení úlohy provedu v případě 40 bodů daných v obr. 1.



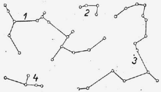
Obr. 1.

Každý z daných bodů spojím s bodem nejbližším. Tedy na př. bod 1 s bodem 2, bod 2 s bodem 3, bod 3 s bodem 4 (bod 4 s bodem 3), bod 5 s bodem 2, bod 6 s bodem 5, bod 7 s bodem 6, bod 8 s bodem 9, (bod 9 s bodem 8) atd. Obdržím řadu polygonálních tahů 1, 2,.... 13 (obr. 2).



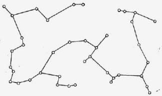
Obr. 2.

Každý z nich spojím nejkratším způsobem s tahem nejbližším. Tedy na př. 1 s tahem 2, (tah 2 s tahem 1), tah 3 s tahem 4, (tah 4 s tahem 3) atd. Obdržím řadu polygonálních tahů 1, 2,.... 4 (obr. 3).



Obr. 3.

Každý z nich spojím nejkratším způsobem s tahem nejbližším. Tedy tah 1 s tahem 3, tah 2 s tahem 3 (tah 3 s tahem 1), tah 4 s tahem 1. Obdržím konečně jediný polygonální tah (obr. 4), jenž řeší danou úlohu.



Obr. 4.

Orazec převrácen.

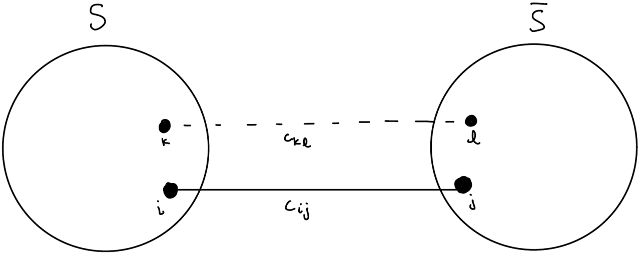
*) Vyjde v nejbližší době v Pracích Moravské přírodovědecké společnosti.

Matematický ústav Masarykovy university v Brně, v lednu 1926.

Optimality Conditions

Theorem (Cut Optimality Conditions)

A spanning tree T^* is a minimum spanning tree if and only if it satisfies the following cut optimality conditions: For every arc $(i,j) \in T^*$, $c_{ij} \leq c_{kl}$ for every arc (k,l) contained in the cut formed by deleting arc (i,j) from T^*.



Sollin's Algorithm

- Sollin's algorithm is a hybrid of Kruskal's and Prim's algorithms
  - Maintains collections of nodes $N_1, N_2, \ldots$ and adds arcs to the collection, a technique borrowed from Kruskal
  - Adds minimum cost arcs at every iteration, a technique borrowed from Prim
- Requires arc costs to be distinct
  - Boruvka quoted saying "If we measure distances, we can assume that they are all different. Whether distance from Brno to Breclaw is 50 km or 50 km and 1 cm is a matter of conjecture".

The Two Basic Operations

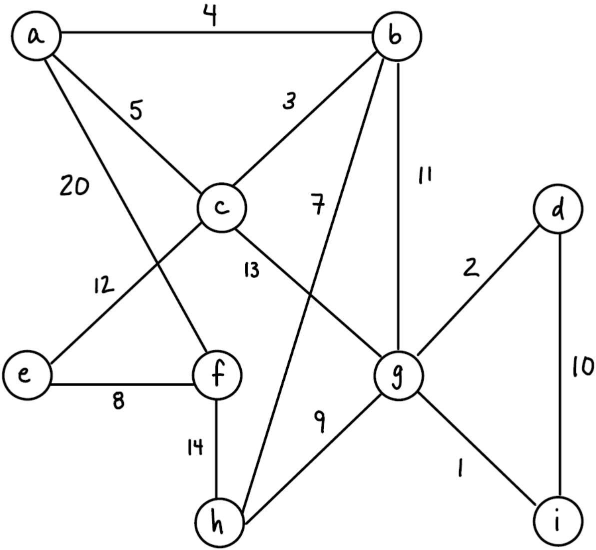
Sollin's algorithm uses two basic operations

- *nearest-neighbor*(N_k, i_k, j_k): Given a tree spanning the nodes N_k, this operation determines an arc (i_k, j_k) that is a minimum cost arc emanating from N_k.
- *merge*(i_k, j_k): Given two nodes i_k and j_k, if the two nodes belong to two different trees, we merge the two trees into a single tree

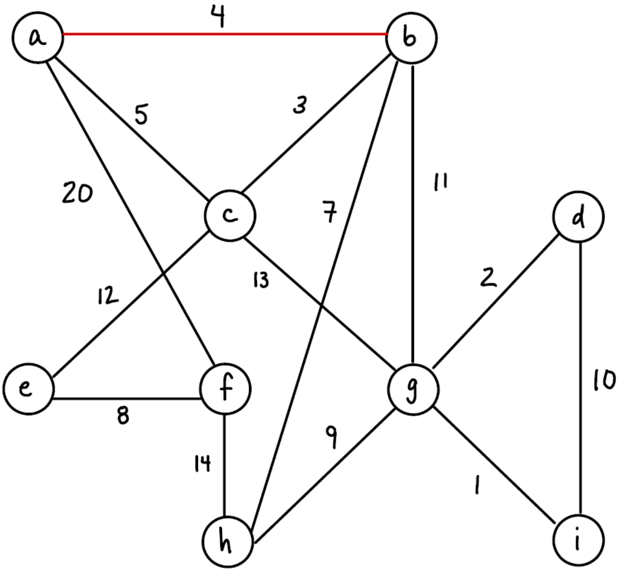
Pseudocode

```
algorithm Sollin;
begin
    for each i ∈ N do N_i := {i};
    T* := ∅;
    while |T*| < (n − 1) do
    begin
        for each tree N_k do  nearest-neighbor(N_k, i_k, j_k);
        for each N_k do
            if nodes i_k and j_k belong to different trees then
                merge (i_k, j_k) and update T* := T* ∪ {(i_k, j_k)};
    end;
end;
```

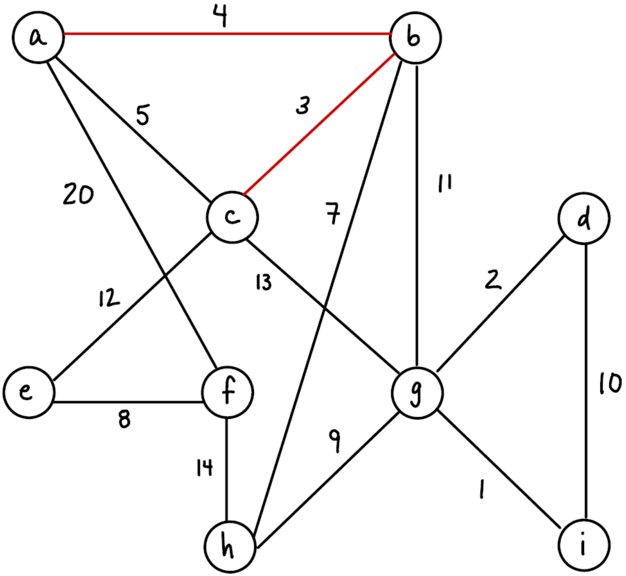
Example



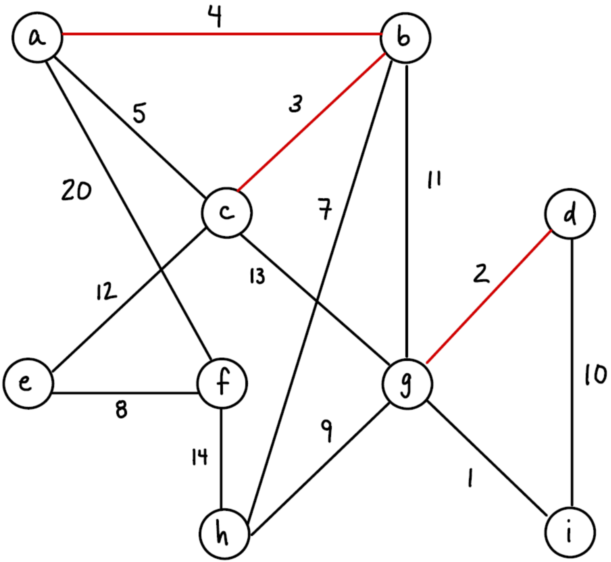
Example



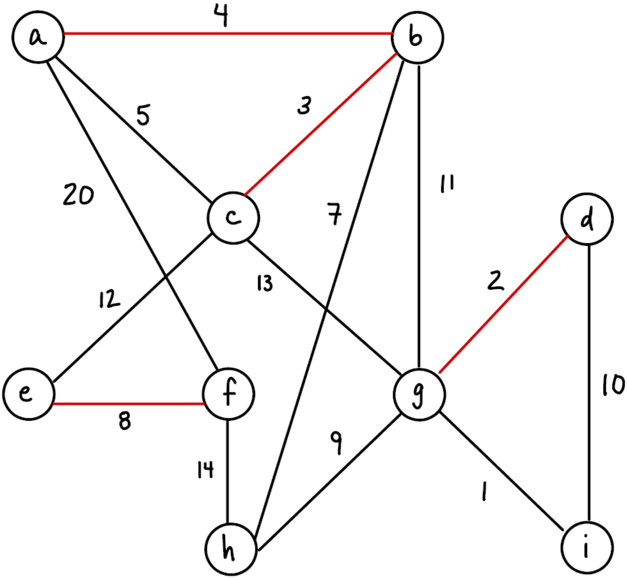
Example



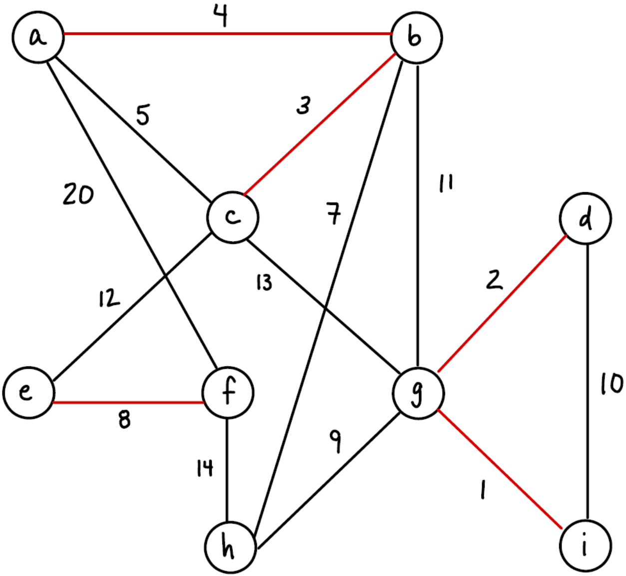
Example



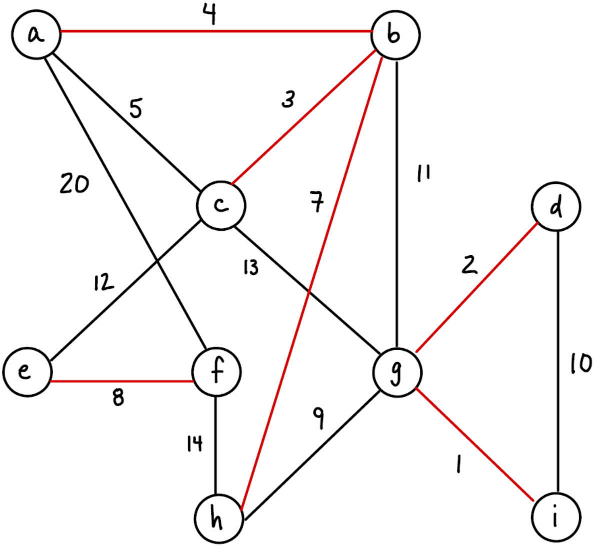
Example



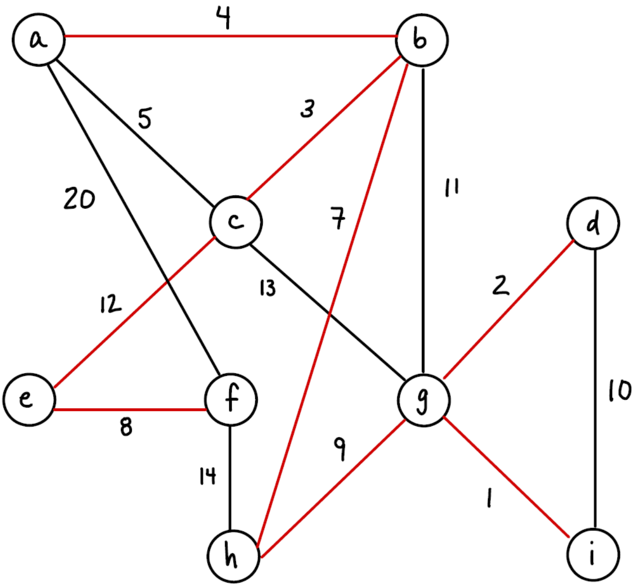
Example



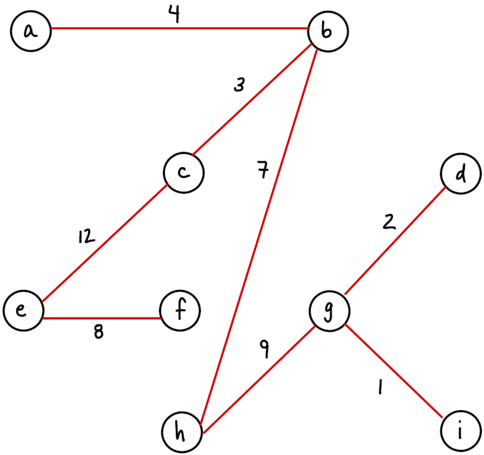
Example



Example



Example



Minimum spanning tree with cost 46.

Complexity of the Algorithm

We look at the running time of each of the two major operations in Sollins algorithm.

- Nearest-neighbor
  - Scans over arcs emanating from each tree
  - $O(\sum_{i \in N} |A(i)|) = O(m)$
- Merge
  - Scans over nodes to see if in the same tree
  - $O(n)$
- Every iteration lowers the number of trees by at least a factor of 2, so we run the while loop $O(\log n)$ times
- This implies total running time of $O(m \log n)$.

Correctness of Algorithm

Cut optimality conditions can be restated as the following: for any cut, the minimum cost edge in the cut must be in the minimum spanning tree. At each iteration of the algorithm, we pick the minimum cost edge emanating from each tree, which correspond to the minimum cost arc for every cut.