Minimum spannig trees

Data Structures and Algorithms for Computational Linguistics III
(ISCL-BA-07)

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

Winter Semester 2022/23

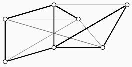
version: ec0d372 @2022-11-30

Spanning trees

reminder

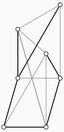
A spanning tree of a graph is

- A *spanning subgraph*: it includes all nodes
- It is a *tree*: it is *acyclic*, and *connected*



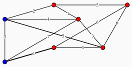
Minimum spanning trees

- A minimum spanning tree (MST) is a spanning tree of weighted graph with minimum total weigh
- MST is a fundamental problem with many applications, including
  - Network design (communication, transportation, electrical, ...)
  - Cluster analysis
  - Approximate solutions to traveling salesman problem
  - Object/network recognition in images
  - Avoiding cycles in broadcasting in communication networks
  - Dithering in images, audio, video
  - Error correction codes
  - DNA sequencing
  - ...



The 'cut property'

- A *cut* of a graph is a partition that divides its nodes into two disjoint (non-empty) sets
- Given any cut, the edge with the lowest weight across the cut is in the MST



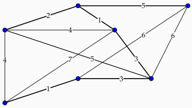
Prim-Jarník algorithm

intuition

- Prim-Jarník algorithm is a greedy algorithm for finding an MST for a weighted undirected graph
- Algorithm starts with a single 'start' node, and grows the MST greedily
- At each step we consider a cut between nodes visited and the rest of the nodes, and select the minimum edge across the cut
- Repeat the process until all nodes are visited

Prim-Jarník algorithm

demonstration



Prim-Jarník algorithm

analysis

- Two loops over number of nodes n, $O(n^2)$ if we need to search
- If we use a priority queue for Q, then complexity becomes $O(m \log m)$

```
pick any node s
C[s] ← 0
for each node v ≠ s do
    C[v] ← ∞
    E[v] ← None
T ← ∅
Q ← nodes
while Q is not empty do
    Retrieve v with min C[v] from Q
    Connect v to T
    for edge (v, w), where w is in Q do
        if cost(v, w) < C[w] then
            C[w] ← cost(v, w)
            E[w] ← v
```

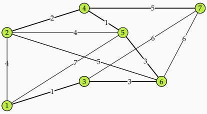

Kruskal's algorithm

intuition

- Another popular algorithm for finding MST on undirected graphs
- The main idea is starting with each node in its own partition
- At each iteration, we choose the edge with the minimum weight across any two clusters, and join them
- Algorithm terminates when there are no clusters to join

Kruskal's algorithm

demo



Kruskal's algorithm

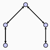
analysis

- Loop over edges, but beware of the sorting requirement
- With simple data structures complexity is $O(m \log m)$

```
T ← ∅
for each node v do
    create_cluster(v)
for (u,v) in edges sorted by weight do
    if cluster(u) ≠ cluster(v) then
        T ← T ∪ {(u, v)}
        union(cluster(u), cluster(v))
```


Directed trees

- Trees with directed edges come in few flavors
  - A *rooted directed tree* (arborescence) is an acyclic directed graph where all nodes are reachable from the root node through a single directed path (this is what computational linguists simply calls a *tree*)
  - An anti-arborescence is a rooted directed tree where all edges are reversed
  - A polytree (also called a directed tree) is a directed graph where undirected edges form a tree
- The equivalent of finding an MST in a directed graph is finding a rooted directed tree (arborescence)

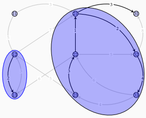


Chu–Liu/Edmonds algorithm

a sketch

- The MST for a directed graph has to start from a designated root node
  - If selected node has any incoming edges, remove them
  - It is also a common practice to introduce an artificial root node with equal-weight edges to all nodes
- For all non-root nodes, select the incoming edge with lowest weight, remove others
- If the resulting graph has no cycles, it is an MST
- If there are cycles break them
  - Consider the cycle as a single node
  - Select the incoming edge that yields the lowest cost if used for breaking the cycle
- Repeat until no cycles remain

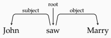
Chu–Liu/Edmonds algorithm
demonstration



Chu–Liu/Edmonds algorithm
analysis

- The algorithm is generally defined recursively: at each step, create a new graph with a contracted cycle call the procedure with the new graph
- At most n recursions: the cycle has to include more nodes at every step
- At each call, m steps for finding minimum incoming edge (also finding a cycle with $O(n)$, but $m \geqslant n$)
- The 'vanilla' algorithm runs in $O(mn)$
- There are improved versions

Chu–Liu/Edmonds algorithm in Computational Linguistics
dependency parsing



- In a dependency analysis, the structure of the sentence is represented by asymmetric binary relations between syntactic units
- Each relation defines one of the words as the head and the other as dependent
- Often an artificial root node is used for computational convenience
- The links (relations) may have labels (dependency types)
- A dependency analysis (parse) is simply a rooted directed tree

Chu–Liu/Edmonds for dependency parsing

- Begin with fully connected weighted graph, except the root node has no incoming edges
- Weights are estimated from a treebank, typically determined by a machine learning method trained on a treebank
- We often use probabilities rather than costs/distances, so, rather than minimizing, maximize the weight of the tree
- Given the fully connected graph, now the parsing becomes finding the MST
- This method is one of the most common (and successful) approaches to dependency parsing

Summary

- Minimum spanning trees have many applications
- An MST of a undirected graph can be found (efficiently) using Prim-Jarník or Kruskal's algorithms
- For directed graph, the corresponding problem can be solved using Chu–Liu/Edmonds algorithm (technically what we find is a rooted directed tree, or arborescence)
- MST also has quite a few applications in CL/NLP

Next:

- Maps and hashing
- Reading: **goodrich2013**

Acknowledgments, credits, references

blank

blank

blank

blank