MST-based Dependency Parsing

Pawan Goyal

CSE, IIT Kharagpur

Week 6, Lecture 4

Maximum Spanning Tree Based

Maximum Spanning Tree Based

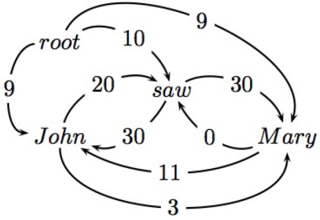
Basic Idea

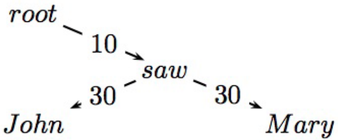
Starting from all possible connections, find the maximum spanning tree.

Maximum Spanning Tree Based

Basic Idea

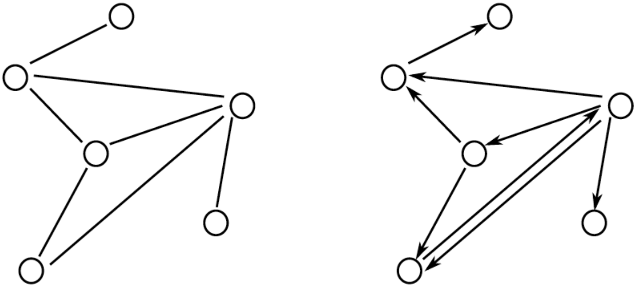
Starting from all possible connections, find the maximum spanning tree.





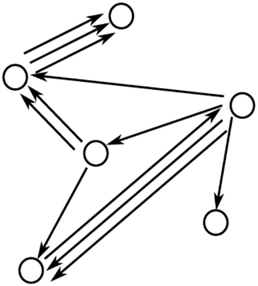
Some Graph Theory Reminders

- A graph $G = (V, A)$ is a set of vertices V and arcs $(i,j) \in A$ where $i,j \in V$.
- Undirected graphs: $(i,j) \in A \Leftrightarrow (j,i) \in A$
- Directed graphs (digraphs) : $(i,j) \in A \nRightarrow (j,i) \in A$



Multi-Digraphs

- A multi-digraph is a digraph where multiple arcs between vertices are possible
- $(i,j,k) \in A$ represents the k^{th} arc from vertex i to vertex j.

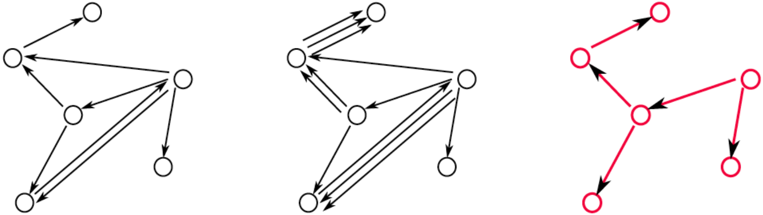


Directed Spanning Trees

- A directed spanning tree of a (multi-)digraph $G = (V,A)$ is a subgraph $G' = (V',A')$ such that:
  - $V' = V$
  - $A' \subseteq A$, and $|A'| = |V'| - 1$
  - G' is a tree (acyclic)

Directed Spanning Trees

- A directed spanning tree of a (multi-)digraph $G = (V, A)$ is a subgraph $G' = (V', A')$ such that:
  - $V' = V$
  - $A' \subseteq A$, and $|A'| = |V'| - 1$
  - G' is a tree (acyclic)
- A spanning tree of the following (multi-)digraphs



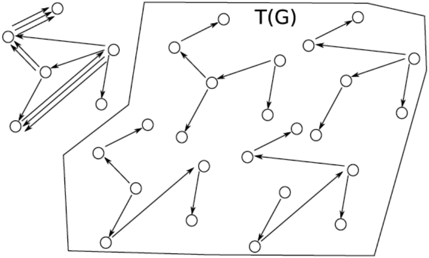
Weighted Directed Spanning Trees

- Assume we have a weight function for each arc in a multi-digraph $G = (V, A)$.
- Define ${w_{ij}}^k \geq 0$ to be the weight of $(i, j, k) \in A$ for a multi-digraph
- Define the weight of directed spanning tree G' of graph G as

$$w(G') = \sum_{(i,j,k) \in G'} {w_{ij}}^k$$

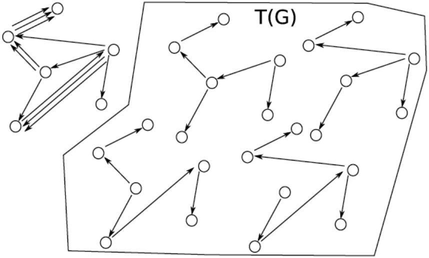
Maximum Spanning Trees (MST)

Let $T(G)$ be the set of all spanning trees for graph G



Maximum Spanning Trees (MST)

Let $T(G)$ be the set of all spanning trees for graph G



The MST problem

Find the spanning tree G' of the graph G that has the highest weight

$$G' = \arg\max_{G' \in T(G)} w(G') = \arg\max_{G' \in T(G)} \sum_{(i,j,k) \in G'} w_{ij}{}^{k}$$

Finding MST

Directed Graph

For each sentence x, define the directed graph $G_x = (V_x, E_x)$ given by

$$V_x = \{x_0 = root, x_1, \ldots, x_n\}$$

$$E_x = \{(i,j) : i \neq j, (i,j) \in [0:n] \times [1:n]\}$$

Finding MST

Directed Graph

For each sentence x, define the directed graph $G_x = (V_x, E_x)$ given by

$$V_x = \{x_0 = root, x_1, \ldots, x_n\}$$

$$E_x = \{(i,j) : i \neq j, (i,j) \in [0:n] \times [1:n]\}$$

G_x is a graph with

- the sentence words and the dummy root symbol as vertices and
- a directed edge between every pair of distinct words and
- a directed edge from the root symbol to every word

Chu-Liu-Edmonds Algorithm

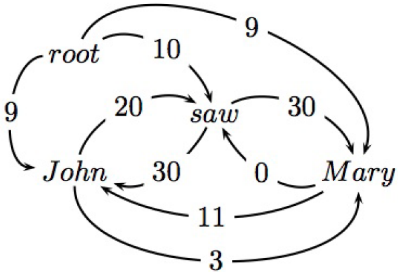
Chu-Liu-Edmonds Algorithm

- Each vertex in the graph greedily selects the incoming edge with the highest weight.
- If a tree results, it must be a maximum spanning tree.
- If not, there must be a cycle.
  - Identify the cycle and contract it into a single vertex.
  - Recalculate edge weights going into and out of the cycle.

Chu-Liu-Edmonds Algorithm

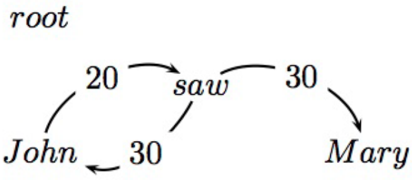
x = John saw Mary

- Build the directed graph



Chu-Liu-Edmonds Algorithm

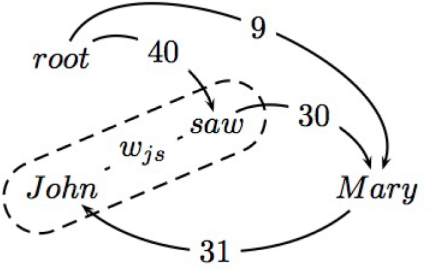
- Find the highest scoring incoming arc for each vertex



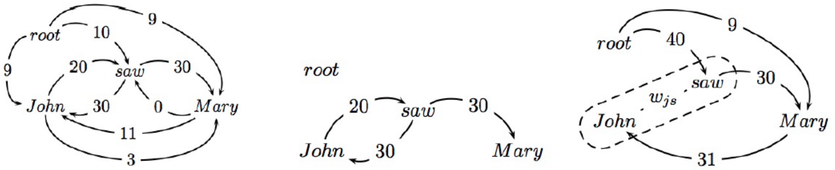
- If this is a tree, then we have found MST.

Chu-Liu-Edmonds Algorithm

- If not a tree, identify cycle and contract
- Recalculate arc weights into and out-of cycle



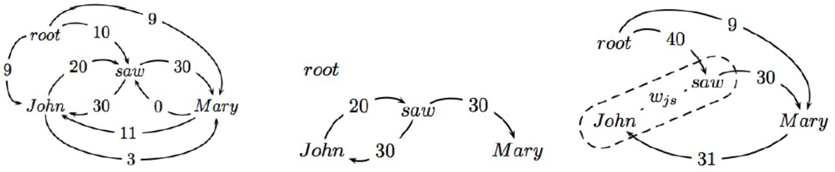
Chu-Liu-Edmonds Algorithm



Outgoing arc weights

- Equal to the max of outgoing arc over all vertices in cycle
- e.g., John → Mary is 3 and saw → Mary is 30.

Chu-Liu-Edmonds Algorithm

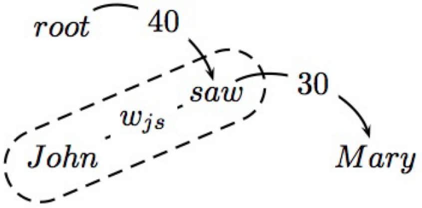


Incoming arc weights

- Equal to the weight of best spanning tree that includes head of incoming arc and all nodes in cycle
- root → saw → John is 40
- root → John → saw is 29

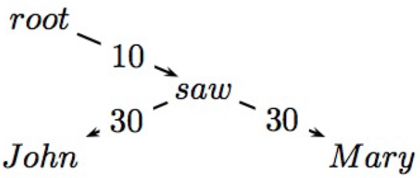
Chu-Liu-Edmonds Algorithm

Calling the algorithm again on the contracted graph:



- This is a tree and the MST for the contracted graph
- Go back up the recursive call and reconstruct final graph

Chu-Liu-Edmonds Algorithm



- The edge from w_{js} to *Mary* was from *saw*
- The edge from *root* to w_{js} represented a tree from *root* to *saw* to *John*.