Lecture 3: Minimum Spanning Trees

1 Minimum Spanning Trees

We are given a connected graph $G(V, E)$, and each edge $e$ has a cost (weight) of $c(e) > 0$. Let the cost of $G$ be $\sum_{e \in E} c(e)$.

**Question:** What is cheapest connected subgraph of $G$?

Clearly, the solution $G'$ is a tree. Otherwise, suppose $G'$ has at least one cycle. Pick an edge from that cycle and delete it. The resulting graph is still connected and has a lower cost, thus $G'$ could not be the cheapest connected subgraph.

**Definition:** For a connected graph $G(V, E)$, a spanning tree of $G$, $T(G)$ is a connected subgraph of $G$ that is a tree and has vertex set $V$. Given the cost function $c(\cdot)$, the minimum spanning tree of $G$, $MST(G)$, is the cheapest spanning tree of $G$.

To find the cheapest connected subgraph of $G$ - or equivalently $MST(G)$ - one way would be to enumerate all the spanning trees and find the cheapest one. However, the number of spanning trees can be exponentially large. We need to come up with a more efficient way to find $MST(G)$.

Joseph Kruskal proposed the following greedy algorithm to produce an $MST$. For simplicity assume that the costs of edges are unique. This assumption is not necessary, but simplifies the proof.

**Algorithm 1: The Greedy Algorithm, Greedy-MST**

1. Sort the edge weights (and relabel), so $c(e_1) < c(e_2) < \cdots < c(e_m)$.
2. Given graph $G(V, E)$, initialize $E(T) = \emptyset$ and $V(T) = V$.
3. While $|E(T)| < |V| - 1$, add the cheapest unused $e_k$ that does not create a cycle.

**Claim 1** Greedy-MST produces $T^* = MST(G)$.

**Proof:** Let $T$ be the tree that Greedy-MST outputs. Clearly, $e_1 \in E(T)$. We prove that $e_1 \in E(T^*)$; assume $e_1 \notin E(T^*)$, adding $e_1$ to $T^*$ results in creating a cycle in the new graph. Remove any edge from such cycle (except $e_1$); we have a new spanning tree whose weight is strictly smaller than that of $T^*$, which is contradicting with the assumption that $T^*$ is $MST(G)$.

Let $S = \{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$ be the set of the first $k$ edges that Greedy-MST picks and let $T^*$ be the MST that contains all the edges in $S$. Suppose $e_{i_{k+1}}$ is the first mistake the algorithm makes, i.e. $e_{i_{k+1}}$ does not belong to $T^*$. Adding $e_{i_{k+1}}$ to $T^*$ results in a graph with a cycle, $c$. If we show that $c$ contains at least one edge $e_l$ with $c(e_l) > c(e_{i_{k+1}})$ - then by removing $e_l$, we have a new spanning tree whose weight is smaller than or equal to the weight of $T^*$, which is contradicting with the assumption that $T^*$ is $MST(G)$.

Now we prove that cycle $c$ contains at least one edge with index $l > i_{k+1}$; suppose $c$ does not have such an edge, hence all edges of $c$ have indices smaller than $i_{k+1}$. Looking back at Greedy-MST algorithm we know that this can’t be true; if all edges in the cycle have indices smaller than $i_{k+1}$ then $e_{i_{k+1}}$ would create a cycle with the existing edges that have already been picked by the algorithm, however the algorithm does not select such an edge. So this is contradicting with $e_{k+1}$ being selected by the Greedy-MST algorithm.
2 Application: Clustering in Bio-informatics

Biologists commonly use DNA arrays to analyze gene function. This technique allows the measurement of “expression level” (amounts of mRNA produced) in genes. The result is a $n \times m$ matrix recording in each row the expression levels of a gene.

Clustering algorithms allow for genes with similar expression levels to be linked together in hope that genes placed in common clusters share common functions. To achieve this, an $n \times n$ distance matrix $d$ is constructed. $d(i, j)$ records how close, or similar, genes $i$ and $j$ are based on their expression levels.

Given $n$ nodes (genes), a distance function, and $l < n$, the clustering problem aims to partition the nodes into $l$ groups so that the nodes in the same group are “close” and the nodes in different groups are “far apart”. One way to formulate the problem and address these goals is as follows:

Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of nodes, and $d : V \mapsto \mathbb{R}^+$ be the corresponding distance function that satisfies the following conditions:

1. $d(v_i, v_i) = 0$.
2. $d(v_i, v_j) = d(v_j, v_i)$.

Let $C = \{C_1, C_2, \ldots, C_l\}$ be a partition of $V$ where $C_i \neq \emptyset$. We define $D(C_i, C_j)$ to be:

$$D(C_i, C_j) = \min_{v \in C_i, w \in C_j} d(v, w),$$

and $D^*(C) = \min_{i,j} D(C_i, C_j)$.

The problem of clusterings of maximum spacing is to find the partition that maximizes $D^*(C)$.

An efficient way to solve this problem is to consider the complete graph on $V$ and for $e = (i, j)$, let $c(e) = d(i, j)$; ignore all the self loops and run Greedy-MST; delete the $l - 1$ most expensive edges - or equivalently, the last $l - 1$ edges added in the run of the algorithm - more formally, let $\{i_1, i_2, \ldots, i_{n-2}\}$ be the indices of edges of the tree selected by Greedy-MST, $T$. It is easy to see that by deleting one edge of $T$ we will have two trees. Similarly, by deleting $\{i_{n-1}, \ldots, i_{n-1}\}$ we will have $l$ connected components, $C_1, C_2, \ldots, C_l$.

Claim 2 $C = \{C_1, C_2, \ldots, C_l\}$ is the solution of the $l$-clusterings of maximum spacing problem.

Proof: First note that $D^*(C)$ is $c(e_i)$ or the $(l-1)$st most expensive edge of $T$. We prove the claim by contradiction, assume that $C' = \{C'_1, C'_2, \ldots, C'_l\}$ is another partition in which all $C'_i$’s are non-empty and $D^*(C') > D^*(C)$. Since $C \neq C'$, there exists a cluster $C_r$ with nodes $v_i \in C_r$ and $v_j \in C_r$ that are in two different clusters of $C'$. Say $v_i \in C'_s$ and $v_j \in C'_t$. Since $v_i$ and $v_j$ belong to the connected component $C_r$ there must be a path $P$ between them in $T$. Moreover, since all the edges in $\{i_1, i_2, \ldots, i_{n-1}\}$ have cost smaller than the cost of $i_{n-1}$, the cost of any edge in $P$ is smaller than $D^*(C)$.

Consider path $P$ in $C'_r$, let $u$ be the first node of $P$ that does not belong to $C'_s$ and $w$ be the node comes just before $u$ in $P$. Suppose $u \in C'_q$, clearly $D^*(C') \leq D(C'_s, C'_q) \leq c(u, w) < D^*(C)$ which is contradicting with the assumption that $C'$ is the $l$-clusterings of maximum spacing.
We can also use MST to create a hierarchical clustering of a set. In biology, this tool is used to classify species, i.e. to build the “tree of life” or the phylogenetic tree. Solving this problem exactly is very hard but several approximation heuristics are used, one of which relies on minimum spanning trees. To build a “maximum likelihood” phylogenetic tree in this way, one can use the “closeness” of DNA sequences between different species as weights of a graph and build the MST of this graph. The phylogenetic tree can then be reconstructed by looking at where the nodes are located in the MST.

References
