2.2 Optimal cost spanning trees

Spanning trees have a number of applications:

- network design (communication, electrical, ...)
- IP network protocols
- compact memory storage (DNA)
- ...
2.2.1 Minimum cost spanning tree problem

Example

Design a communication network so as to connect \( n \) cities at minimum total cost.

Model: Graph \( G = (N, E) \) with \( n = |N| \), \( m = |E| \) and a cost function \( c : E \rightarrow c_e \in \mathbb{R} \), with \( e = [v, w] \in E \).
Required properties:

1) Each pair of cities must communicate ⇒ connected subgraph containing all the nodes.

2) Minimum total cost ⇒ subgraph with no cycles.

Problem

Given an undirected graph $G = (N, E)$ and a cost function, find a spanning tree of minimum total cost

$$\min_{T \in X} \sum_{e \in T} c_e$$

where $X$ is the set of all spanning trees of $G$
Some feasible solutions:

\[ G = (N, E) \]

\[ c(T_1) = 16 \]
\[ c(T_2) = 9 \]
\[ c(T_3) = 11 \]
Theorem (A. Cayley 1889)

A complete graph with \( n \) nodes \( (n \geq 1) \) has \( n^{n-2} \) spanning trees.

Examples: \( K_3 \) (\( n=3 \), \( m=3 \) edges) has 3 spanning trees

\[ K_5 \] (\( n=5 \), \( m=10 \)) has 125 spanning trees

Recall: A tree with \( n \) nodes has \( n - 1 \) edges.
2.2.2 Prim’s algorithm

**Idea:** Iteratively build a spanning tree.

At each step, add to the current partial tree an edge of minimum cost among those which connect a node from the partial tree to another node that does not belong to it.
Given $G = (N,E)$ with edge costs

Procedure:

$S = \{1\}$

$T = \emptyset$

$S = \{1, 2\}$

$T = \{[1,2]\}$
$S = \{1, 2\}$

$S = \{1, 2, 3, 5\}$

$S^* = \{1, 2, 5\}$

$S = \{1, 2, 5\}$

$c(T) = 9$
Pseudocode of Prim’s algorithm

**Input**
Connected \( G = (N, E) \) with edge costs

**Output**
Subset of edges \( T \subseteq E \) such that \( G_T = (N, T) \) is a spanning tree of \( G \)

BEGIN

\[ T := \emptyset; \quad S := \{1\}; \]

WHILE \( |T| < n-1 \) DO /* a tree with \( n \) nodes has \( n-1 \) edges */

Select an edge \([v, h] \in \delta(S)\) of minimum cost (\( v \in S \) and \( h \in N \setminus S \));

\[ T := T \cup \{[v, h]\}; \]

\[ S := S \cup \{h\}; \]

END-WHILE

END

If all edges are scanned at each iteration, complexity: \( O(nm) \)
2.2.3 Exactness of Prim’s algorithm

**Definition**: An algorithm is *exact* if it provides an optimal solution for every instance, otherwise it is *heuristic*.

**Proposition**: Prim’s algorithm is exact.

We show that each selected edge belongs to a minimum spanning tree.

As we shall see, exactness does not depend on the choice of the first node or of the edge of minimum cost in \( \delta(S) \).
Let $F$ be a partial tree (spanning nodes in $S \subseteq N$) contained in an optimal tree of $G$. Consider $e = [v, h] \in \delta(S)$ of minimum cost, then there exists a minimum cost spanning tree of $G$ containing $e$.

**Proof**

By contradiction:

Let $T^* \subseteq E$ be a minimum cost spanning tree with $F \subseteq T^*$ and $e \notin T^*$.

Adding edge $e$ creates the cycle $C$.

Let $f \in \delta(S) \cap C$.

If $c_e = c_f$ then $T^* \cup \{e\} \setminus \{f\}$ is (also) optimal since it has same cost of $T^*$.

If $c_e < c_f$ then $c(T^* \cup \{e\} \setminus \{f\}) < c(T^*)$, hence $T^*$ is not optimal.
**Definition:** A *greedy algorithm* constructs a feasible solution iteratively by making at each step a “locally optimal” choice, without reconsidering previous choices.

**Observation:** Prim’s algorithm is a greedy algorithm.

At each step a minimum cost edge is selected among those in the cut $\delta(S)$ induced by the current set of nodes $S$.

**N.B.** For most discrete optimization problems greedy-type algorithms yield a feasible solution with no guarantee of optimality.
Various greedy algorithms for the minimum cost spanning tree problem are based on the cut property:

- Boruvka (1926)
- Kruskal (1956) -- Exercise 2.2
- Prim (1957)
- ...
2.2.4 \( O(n^2) \) implementation

Data structure:

- \( k \) = number of edges selected so far
- Subset \( T \subseteq E \) of selected edges
- Subset \( S \subseteq N \) of nodes incident to the selected edges

.....
• \( C[j] = \min \{c_{ij} : i \in S\} \) for \( j \notin S \) -- if \([i,j] \notin E\), \( c_{ij} = +\infty \)

\[
\text{closest}[j] = \begin{cases} 
\text{argmin} \{c_{ij} : i \in S\}, & \text{for } j \notin S \\
\text{“predecessor” of } j \text{ in the min spanning tree}, & \text{for } j \in S 
\end{cases}
\]
Example

Iteration 1:  \( T = \{[1,2]\}, \ C = (+\infty, 1, 4, 6, 2), \ \text{closest} = (1, 1, 1, 1, 1) \)

Iteration 2:  \[ \text{cut } \delta(S) = \{[1,3], [1,4], [1,5], [2,3], [2,5]\} \]

\[ S = \{1, 2\} \]

\begin{align*}
C[3] &:= c_{23} = 3 \ (\text{since } c_{23} < c_{13}) \\
\text{closest}[3] &:= 2 \\
C[5] &:= c_{15} = 2 \ (\text{since } c_{15} = c_{25}) \\
\text{closest}[5] &:= 1
\end{align*}

\[ C[4] := c_{14} = 6 \ (\text{since } [2,4] \text{ does not exist}) \]
\[ \text{closest}[4] := 1 \]

Since  \( C[5] = c_{15} \leq C[3] = c_{23} \) and  \( C[5] = c_{15} \leq C[4] = c_{14} \), then

\[ h := 5, \ \nu = \text{closest}[h] := 1, \ S := S \cup \{5\}, \ T := T \cup \{[1,5]\}\]
$O(n^2)$ version of Prim's algorithm

BEGIN

T := \emptyset; S := \{1\}; /* initialization */

FOR j:=2 TO n DO /* \forall nodes j \notin S */

C[j] := c_{1j};
closest[j] := 1;

END-FOR

FOR k:=1 TO n-1 DO /* select \(n - 1\) edges of the tree */

min := +\infty;

FOR j:=2 TO n DO /* select minimum edge in \(\delta(S)\) */

IF j \notin S AND (C[j] < min) THEN

min := C[j]; h := j; END-IF

END-FOR

S := S \cup \{h\}; T := T \cup \{\text{closest}[h], h\}; /* extend S and T */

FOR j:=2 TO n DO /* update C[j] e closest[j] \(\forall j \notin S\) */

IF j \notin S AND (c_{hj} < C[j]) THEN

C[j] := c_{hj}; closest[j] := h; END-IF

END-FOR

END-FOR

END
Example

\[ T = \emptyset \]
\[ C = (+\infty, 1, 4, 6, 2) \]
\[ \text{closest} = (1, 1, 1, 1, 1) \]

\[ S = \{1\} \]
\[ C = (+\infty, 1, 4, 6, 2) \]
\[ \text{closest} = (1, 1, 1, 1, 1) \]

\[ T = \{[1, 2]\} \]
\[ C = (+\infty, 1, 3, 6, 2) \]
\[ \text{closest} = (1, 1, 2, 1, 1) \]

\[ S = \{1, 2\} \]
\[ C = (+\infty, 1, 3, 6, 2) \]
\[ \text{closest} = (1, 1, 2, 1, 1) \]

\[ S = \{1, 2, 5\} \]
\[ C = (+\infty, 1, 2, 4, 2) \]
\[ \text{closest} = (1, 1, 5, 5, 1) \]

\[ T = \{[1, 2], [1, 5]\} \]
\[ C = (+\infty, 1, 2, 4, 2) \]
\[ \text{closest} = (1, 1, 5, 5, 1) \]

\[ \text{etc...} \]

How to retrieve the spanning tree from closest?

The minimum spanning tree found by Prim’s algorithm consists of the $n-1$ edges: $[ \text{closest}[j], j ]$ with $j = 2, ..., n$.

Example: Since $\text{closest} = (1,1,5,5,1)$ a spanning tree consists of the edges: $[1,2]$, $[5,3]$, $[5,4]$ and $[1,5]$.

total cost: 9
Complexity

BEGIN
  <initialization>
  FOR j:=2 TO n DO
    (...) END-FOR
  FOR k:=1 TO n-1 DO
    FOR j:=2 TO n DO
      (...) END-FOR
    END-FOR
    FOR j:=2 TO n DO
      (...) END-FOR
  END-FOR
END

1. initialization requires $O(n)$
2. They are executed $n - 1$ times in the external cycle
3. The two internal FOR cycles require $O(n)$ each

Overall complexity: $O(n^2)$
For sparse graphs, where $m \ll n(n-1)/2$, a more sophisticated data structure leads to an $O(m \log n)$ complexity.
2.2.5 Optimality condition

Definition: Given a spanning tree \( T \), an edge \( e \not\in T \) is cost decreasing if when \( e \) is added to \( T \) it creates a cycle \( C \) with \( C \subseteq T \cup \{e\} \) and \( \exists \) an edge \( f \in C \setminus \{e\} \) such that \( c_e < c_f \).

\[
c( T \cup \{e\} \setminus \{f\} ) < c(T) = \sum_{e' \in T} c_{e'},
\]

A tree $T$ is of minimum total cost if and only if no cost-decreasing edge exists.

**Proof** \((\Rightarrow)\) If a cost-decreasing edge exists, $T$ is not of minimum total cost.

Because the cost of $T$ could be decreased by exchanging the cost-decreasing edge $e$ with any $f$ of $C$ with $c_e < c_f$. 
\( \iff \) If no cost-decreasing edge exists, then \( T \) is of minimum total cost.

Let \( T^* \) be a minimum cost spanning tree found by Prim’s algorithm.

It can be verified that, by exchanging one edge at a time, \( T^* \) can be iteratively transformed into \( T \) without modifying the total cost.

Thus \( T \) is also optimal.
The optimality condition allows us to verify whether a given spanning tree $G_T$ is optimum:

It suffices to check that each $e \in E \setminus T$ is not a cost-decreasing edge.
Given a communication network $G = (N, E)$, we want to broadcast a secret message to all the nodes so that it is not intercepted along any edge.

Let $p_{ij}$, $0 \leq p_{ij} \leq 1$, be the probability the message is intercepted along edge $[i,j] \in E$.

**Problem**

How to broadcast the message to all the nodes of $G$ so as to minimize the probability of interception along any edge?
Minimize the probability of interception (along any edge)

\[ \Updownarrow \]

Maximize the probability of non-interception

\[
\max \prod_{[i,j] \in T} (1 - p_{ij})
\]

\( T \) is a spanning tree

\[ \begin{cases} 
\bullet \text{ Broadcasting to all nodes } \Rightarrow \text{ connected} \\
\bullet \text{ acyclic to avoid redundancy and a higher probability of interception} 
\end{cases} \]
By applying a monotone increasing function like $\log(.)$, the optimal solutions remain unchanged (only the solution values change)

\[
\max \log \left( \prod_{[i,j] \in T} (1 - p_{ij}) \right) \equiv \max \sum_{[i,j] \in T} \log(1 - p_{ij})
\]

Solved by a straightforward adaptation of any minimum cost spanning tree algorithm