MST: Red Rule, Blue Rule

Some of these lecture slides are adapted from material in:
• Data Structures and Algorithms, R. E. Tarjan.
• Randomized Algorithms, R. Motwani and P. Raghavan.

Cycles and Cuts

Cycle.
- A cycle is a set of arcs of the form \{(a,b), (b,c), (c,d), \ldots, (z,a)\}.

Cut.
- The cut induced by a subset of nodes \(S\) is the set of all arcs with exactly one endpoint in \(S\).

Cycle-Cut Intersection

A cycle and a cut intersect in an even number of arcs.

Proof.

Spanning Tree

Spanning tree. Let \(T = (V,F)\) be a subgraph of \(G = (V,E)\). TFAE:
- \(T\) is a spanning tree of \(G\).
- \(T\) is acyclic and connected.
- \(T\) is connected and has \(|V|-1\) arcs.
- \(T\) is acyclic and has \(|V|-1\) arcs.
- \(T\) is minimally connected: removal of any arc disconnects it.
- \(T\) is maximally acyclic: addition of any arc creates a cycle.
- \(T\) has a unique simple path between every pair of vertices.
**Minimum Spanning Tree**

Minimum spanning tree. Given connected graph $G$ with real-valued arc weights $c_e$, an MST is a spanning tree of $G$ whose sum of arc weights is minimized.

Cayley’s Theorem (1889). There are $n^{n-2}$ spanning trees of $K_n$.
- $n = |V|$, $m = |E|$.
- Can’t solve MST by brute force.

**Applications**

MST is central combinatorial problem with diverse applications.
- Designing physical networks.
  - telephone, electrical, hydraulic, TV cable, computer, road
- Cluster analysis.
  - delete long edges leaves connected components
  - finding clusters of quasars and Seyfert galaxies
  - analyzing fungal spore spatial patterns
- Approximate solutions to NP-hard problems.
  - metric TSP, Steiner tree
- Indirect applications.
  - describing arrangements of nuclei in skin cells for cancer research
  - learning salient features for real-time face verification
  - modeling locality of particle interactions in turbulent fluid flow
  - reducing data storage in sequencing amino acids in a protein

**Optimal Message Passing**

Optimal message passing.
- Distribute message to $N$ agents.
- Each agent can communicate with some of the other agents, but their communication is (independently) detected with probability $p_{ij}$.
- Group leader wants to transmit message (e.g., Divx movie) to all agents so as to minimize the total probability that message is detected.

Objective.
- Find tree $T$ that minimizes: $1 - \prod_{(i,j) \in T} (1 - p_{ij})$
- Or equivalently, that maximizes: $\prod_{(i,j) \in T} (1 - p_{ij})$
- Or equivalently, that maximizes: $\sum_{(i,j) \in T} \log(1 - p_{ij})$
- Or equivalently, MST with weights $p_{ij}$.

**Fundamental Cycle**

Fundamental cycle.
- Adding any non-tree arc $e$ to $T$ forms unique cycle $C$.
- Deleting any arc $f \in C$ from $T \cup \{e\}$ results in new spanning tree.

Cycle optimality conditions: For every non-tree arc $e$, and for every tree arc $f$ in its fundamental cycle: $c_f \leq c_e$.
Observation: If $c_f > c_e$ then $T$ is not a MST.
**Fundamental Cut**

Deleting any tree arc $f$ from $T$ disconnects tree into two components with cut $D$.
- Adding back any arc $e \in D$ to $T - \{f\}$ results in a new spanning tree.

**Cut optimality conditions:** For every tree arc $f$, and for every non-tree arc $e$ in its fundamental cut: $c_e \geq c_f$.

**Observation:** If $c_e < c_f$ then $T$ is not a MST.

---

**MST: Cut Optimality Conditions**

**Theorem.** Cut optimality $\Rightarrow$ MST. (proof by contradiction)
- $T$ = spanning tree that satisfies cut optimality conditions.
- $T^*$ = MST that has as many arcs in common with $T$ as possible.
- If $T = T^*$, then we are done. Otherwise, let $f \in T$ s.t. $f \notin T^*$.
- Let $D$ be fundamental cut formed by deleting $f$ from $T$.
- Adding $f$ to $T^*$ creates a fund cycle $C$, which shares (at least) two arcs with cut $D$. One is $f$, let $e$ be another. Note: $e \notin T$.
- Cut optimality conditions $\Rightarrow c_e \leq c_f$.
- Thus, we can replace $e$ with $f$ in $T^*$ without increasing its cost.

---

**MST: Cycle Optimality Conditions**

**Theorem.** Cut optimality $\Rightarrow$ MST. (proof by contradiction)
- $T$ = spanning tree that satisfies cut optimality conditions.
- $T^*$ = MST that has as many arcs in common with $T$ as possible.
- If $T = T^*$, then we are done. Otherwise, let $f \in T$ s.t. $f \notin T^*$.
- Let $D$ be fundamental cut formed by deleting $f$ from $T$.
- Adding $f$ to $T^*$ creates a fund cycle $C$, which shares (at least) two arcs with cut $D$. One is $f$, let $e$ be another. Note: $e \notin T$.
- Cut optimality conditions $\Rightarrow c_e \leq c_f$.
- Thus, we can replace $e$ with $f$ in $T^*$ without increasing its cost.

---

**Towards a Generic MST Algorithm**

If all arc weights are distinct:
- MST is unique.
- Arc with largest weight in cycle $C$ is not in MST.
- Arc with smallest weight in cutset $D$ is in MST.

---
Generic MST Algorithm

Red rule.
- Let \( C \) be a cycle with no red arcs. Select an uncolored arc of \( C \) of max weight and color it \textcolor{red}{red}.

Blue rule.
- Let \( D \) be a cut with no blue arcs. Select an uncolored arc in \( D \) of min weight and color it \textcolor{blue}{blue}.

Greedy algorithm.
- Apply the red and blue rules (non-deterministically!) until all arcs are colored. The blue arcs form a MST.
- Note: can stop once \( n-1 \) arcs colored blue.

Greedy Algorithm: Proof of Correctness

\textbf{Theorem.} The greedy algorithm terminates. Blue edges form a MST.

\textbf{Proof.} (by induction on number of iterations)

\textbf{Color Invariant:} There exists a MST \( T^* \) containing all the blue arcs and none of the red ones.

- \textbf{Base case:} no arcs colored \( \Rightarrow \) every MST satisfies invariant.
- \textbf{Induction step:} suppose color invariant true before blue rule.
  - let \( D \) be chosen cut, and let \( f \) be arc colored blue
  - if \( f \in T^* \), \( T^* \) still satisfies invariant
  - o/w, consider fundamental cycle \( C \) by adding \( f \) to \( T^* \)
  - let \( e \in C \) be another arc in \( D \)
  - \( e \) is uncolored and \( c_e \geq c_f \) since
    - \( e \in T^* \Rightarrow \) not red
    - blue rule \( \Rightarrow \) not blue, \( c_e \geq c_f \)
  - \( T^* \cup \{ f \} - \{ e \} \) satisfies invariant

Proof (continued).
- \textbf{Induction step:} suppose color invariant true before red rule.
  - cut-and-paste

- Either the red or blue rule (or both) applies.
  - suppose arc \( e \) is left uncolored
  - blue edges form a forest
Special Case: Prim’s Algorithm

Prim’s algorithm. (Jarník 1930, Dijkstra 1957, Prim 1959)
- **S** = vertices in tree connected by blue arcs.
- Initialize **S** = any vertex.
- Apply blue rule to cut induced by **S**.

Dijkstra’s Shortest Path Algorithm

Dijkstra’s **Algorithm**

\[
Q \leftarrow PQinit()
\]

for each \( v \in V \)
- \( key(v) \leftarrow \infty \)
- \( pred(v) \leftarrow \text{nil} \)
- \( PQinsert(v, Q) \)

\[ key(s) \leftarrow 0 \]

while (!PQisempty(Q))
- \( v = PQdelmin(Q) \)
  - for each \( w \in Q \) s.t \( \{v, w\} \in E \)
    - if \( key(w) > c(v, w) \)
      - \( PQdekey(w, c(v, w)) \)
      - \( pred(w) \leftarrow v \)

O(m + n log n) Fib. heap
O(n^2) array

Special Case: Kruskal’s Algorithm

Kruskal’s algorithm (1956).
- Consider arcs in ascending order of weight.
  - if both endpoints of \( e \) in same blue tree, color **red** by applying red rule to unique cycle
  - else color \( e \) **blue** by applying blue rule to cut consisting of all vertices in blue tree of one endpoint

Implementing Prim’s Algorithm

**Prim’s Algorithm**

\[
Q \leftarrow PQinit()
\]

for each \( v \in V \)
- \( key(v) \leftarrow \infty \)
- \( pred(v) \leftarrow \text{nil} \)
- \( PQinsert(v, Q) \)

\[ key(s) \leftarrow 0 \]

while (!PQisempty(Q))
- \( v = PQdelmin(Q) \)
  - for each \( w \in Q \) s.t \( \{v, w\} \in E \)
    - if \( key(w) > c(v, w) \)
      - \( PQdekey(w, c(v, w)) \)
      - \( pred(w) \leftarrow v \)

O(m + n log n) Fib. heap
O(n^2) array
Implementing Kruskal’s Algorithm

Kruskal’s Algorithm
Sort edges weights in ascending order $c_1 \leq c_2 \leq \ldots \leq c_m$.

$S = \emptyset$
for each $v \in V$
  UFmake-set($v$)
for $i = 1$ to $m$
  $(v,w) = e_i$
  if ($UF\text{find-set}(v) \neq UF\text{find-set}(w)$)
    $S \leftarrow S \cup \{i\}$
    UFunion($v$, $w$)

$O(n \log n)$ $O(m \alpha (m, n))$
sorting union-find

Implementing Boruvka’s Algorithm

Boruvka implementation.
- Contract blue trees, deleting loops and parallel arcs.
- Remember which edges were contracted in each super-node.

Advanced MST Algorithms

Deterministic comparison based algorithms.
- $O(m \log n)$ Jarník, Prim, Dijkstra, Kruskal, Boruvka
- $O(m \log n)$. Cheriton-Tarjan (1976), Yao (1975)
- $O(m \beta(m, n))$. Fredman-Tarjan (1987)
- $O(m \log \beta(m, n))$. Gabow-Galil-Spencer-Tarjan (1986)
- $O(m \alpha (m, n))$. Chazelle (2000)
- $O(m)$. Holy grail.

Worth noting.
- $O(m)$ verification. Dixon-Rauch-Tarjan (1992)
Linear Expected Time MST

Random sampling algorithm. \((Karger, \text{Klein, Tarjan, 1995})\)
- If lots of nodes, use Boruvka.
  - decreases number of nodes by factor of 2
- If lots of edges, delete useless ones.
  - use random sampling to decrease by factor of 2
- Expected running time is \(O(m + n)\).

Filtering Out F-Heavy Edges

Definition. Given graph \(G\) and forest \(F\), an edge \(e\) is F-heavy if both endpoints lie in the same component and \(c_e > c_f\) for all edges \(f\) on fundamental cycle.
- Cycle optimality conditions: \(T^*\) is MST \(\iff\) no \(T^*\)-heavy edges.
- If \(e\) is F-heavy for any forest \(F\), then safe to discard \(e\).
  - apply red rule to fundamental cycles

Verification subroutine. \((\text{Dixon-Rauch-Tarjan, 1992})\).
- Given graph \(G\) and forest \(F\), is \(F\) is a MSF?
- In \(O(m + n)\) time, either answers (i) YES or (ii) NO and output all F-heavy edges.

Random Sampling

Random sampling.
- Obtain \(G(p)\) by independently including each edge with \(p = 1/2\).
- Let \(F\) be MSF in \(G(p)\).
- Compute F-heavy edges in \(G\).
- Delete F-heavy edges from \(G\).
Random Sampling

Random sampling.
- Obtain $G(p)$ by independently including each edge with $p = 1/2$.
- Let $F$ be MSF in $G(p)$.
- Compute $F$-heavy edges in $G$.
- Delete $F$-heavy edges from $G$.

$G(1/2)$

MSF $F$ in $G(1/2)$

Random Sampling Lemma

Random sampling lemma. Given graph $G$, let $F$ be a MSF in $G(p)$. Then the expected number of $F$-light edges is $\leq n/p$.

Proof.
- WMA $c_1 \leq c_2 \leq \ldots \leq c_m$, and that $G(p)$ is constructed by flipping coin $m$ times and including edge $e_i$ if $i^{th}$ coin flip is heads.
- Construct MSF $F$ at same time using Kruskal's algorithm.
  - edge $e_i$ added to $F \iff e_i$ is $F$-light
  - $F$-lightness of edge $e_i$ depends only on first $i-1$ coin flips and does not change after phase $i$
- Phase $k = \text{period between when } |F| = k-1 \text{ and } |F| = k$.
  - $F$-light edge has probability $p$ of being added to $F$
  - # $F$-light edges in phase $k \sim \text{Geometric}(p)$
- Total # $F$-light edges $\sim \text{NegativeBinomial}(n, p)$. 
**Random Sampling Algorithm**

**Random Sampling Algorithm(G, m, n)**

Run 3 phases of Boruvka’s algorithm on G. Let G₁ be resulting graph, and let C be set of contracted edges.

IF G₁ has no edges RETURN F ← C

G₂ ← G₁(1/2)
Compute MSF F₂ of G₂ recursively.

Compute all F₂-heavy edges in G₁, remove these edges from G₁, and let G’ be resulting graph.

Compute MSF F’ of G’ recursively.

Return F ← C ∪ F’

---

**Analysis of Random Sampling Algorithm**

**Theorem.** The algorithm computes an MST in \(O(m+n)\) expected time.

**Proof.**

- **Correctness:** red-rule, blue-rule.
- Let \(T(m, n)\) denote expected running time to find MST on graph with \(n\) vertices and \(m\) arcs.
  - \(G₁\) has \(\leq m\) arcs and \(\leq n/8\) vertices.
    - each Boruvka phase decreases \(n\) by factor of 2
  - \(G₂\) has \(\leq n/8\) vertices and expected # arcs \(\leq m/2\)
    - each edge deleted with probability 1/2
  - \(G’\) has \(\leq n/8\) vertices and expected # arcs \(\leq n/4\)
    - random sampling lemma

\[
T(m, n) \leq \begin{cases} 
  c(m+n) & \text{if } m \leq 1 \text{ or } n \leq 1 \\
  T(m/2, n/8) + T(n/4, n/8) + c(m+n) & \text{otherwise}\\
\end{cases}
\]

\[
\Rightarrow T(m, n) \leq 2c(m+n)
\]