CS675: Convex and Combinatorial Optimization
Spring 2022
Combinatorial Problems as Linear and Convex Programs

Instructor: Shaddin Dughmi

Outline

Combinatorial vs Convex Optimization

- In CS, discrete problems are traditionally viewed/analyzed using discrete mathematics and combinatorics
  - Algorithms are combinatorial in nature (greedy, dynamic programming, divide and conquor, etc)

Combinatorial vs Convex Optimization

- In CS, discrete problems are traditionally viewed/analyzed using discrete mathematics and combinatorics
  - Algorithms are combinatorial in nature (greedy, dynamic programming, divide and conquor, etc)
- In OR and optimization community, these problems are often expressed as continuous optimization problems
  - Usually linear programs, but increasingly more general convex programs

Combinatorial vs Convex Optimization

- In CS, discrete problems are traditionally viewed/analyzed using discrete mathematics and combinatorics
  - Algorithms are combinatorial in nature (greedy, dynamic programming, divide and conquor, etc)
- In OR and optimization community, these problems are often expressed as continuous optimization problems
  - Usually linear programs, but increasingly more general convex programs
- Increasingly in recent history, it is becoming clear that combining both viewpoints is the way to go
  - Better algorithms (runtime, approximation)
  - Structural insights (e.g. market clearing prices in matching markets)
  - Unifying theories and general results (Matroids, submodular optimization, constraint satisfaction)

Discrete Problems as Linear Programs

- The oldest examples of linear programs were discrete problems
  - Dantzig's original application was the problem of matching 70 people to 70 jobs!

Discrete Problems as Linear Programs

- The oldest examples of linear programs were discrete problems
  - Dantzig's original application was the problem of matching 70 people to 70 jobs!
- This is not surprising, since almost any finite family of discrete objects can be encoded as a finite subset of Euclidean space
  - Convex hull of that set is a polytope
  - E.g. spanning trees, paths, cuts, TSP tours, assignments...

Discrete Problems as Linear Programs

- LP algorithms typically require representation as a “small” family of inequalities,
  - Not possible in general (Say when problem is NP-hard, assuming $(P \neq NP)$)
  - Shown unconditionally impossible in some cases (e.g. TSP)

Discrete Problems as Linear Programs

- LP algorithms typically require representation as a “small” family of inequalities,
  - Not possible in general (Say when problem is NP-hard, assuming $(P \neq NP)$)
  - Shown unconditionally impossible in some cases (e.g. TSP)
- But, in many cases, polyhedra in inequality form can be shown to encode a combinatorial problems at the vertices

Discrete Problems as Linear Programs

- LP algorithms typically require representation as a “small” family of inequalities,
  - Not possible in general (Say when problem is NP-hard, assuming $(P \neq NP)$)
  - Shown unconditionally impossible in some cases (e.g. TSP)
- But, in many cases, polyhedra in inequality form can be shown to encode a combinatorial problems at the vertices

Next

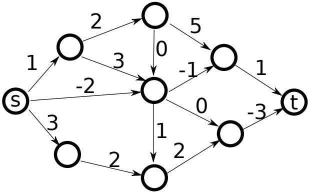
We examine some combinatorial problems through the lense of LP and convex optimization, starting with shortest path.

Outline

The Shortest Path Problem

Given a directed graph $G = (V, E)$ with cost $c_e \in \mathbb{R}$ on edge e, find the minimum cost path from s to t.

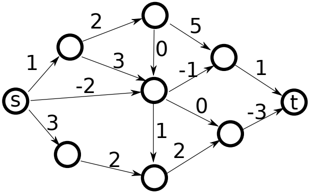
- We use n and m to denote $|V|$ and $|E|$, respectively.
- We allow costs to be negative, but assume no negative cycles
- We assume that there is some path from s to t (Check via BFS)



The Shortest Path Problem

Given a directed graph $G = (V, E)$ with cost $c_e \in \mathbb{R}$ on edge e, find the minimum cost path from s to t.

- We use n and m to denote $|V|$ and $|E|$, respectively.
- We allow costs to be negative, but assume no negative cycles
- We assume that there is some path from s to t (Check via BFS)



When costs are nonnegative, Dijkstra's algorithm finds the shortest path from s to every other node in time $O(m + n \log n)$.

Using primal/dual paradigm, we will design a polynomial-time algorithm that works when graph has negative edges but no negative cycles

Note: Negative Edges and Complexity

- When the graph has no negative cycles, there is a shortest path which is simple
- When the graph has negative cycles, there may not be a shortest path from s to t.
- In these cases, the algorithm we design can be modified to "fail gracefully" by detecting such a cycle
  - Can be used to detect arbitrage opportunities in currency exchange networks

Note: Negative Edges and Complexity

- When the graph has no negative cycles, there is a shortest path which is simple
- When the graph has negative cycles, there may not be a shortest path from s to t.
- In these cases, the algorithm we design can be modified to "fail gracefully" by detecting such a cycle
  - Can be used to detect arbitrage opportunities in currency exchange networks
- In the presence of negative cycles, finding the shortest simple path is NP-hard (by reduction from Hamiltonian cycle)

An LP Relaxation of Shortest Path

Consider the following LP

Primal Shortest Path LP

$$
\begin{aligned}
&\text{min } \textstyle\sum_{e \in E} c_e x_e \\
&\text{s.t.} \\
&\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, \quad \forall v \in V. \\
&x_e \geq 0, \qquad\qquad\qquad\qquad \forall e \in E.
\end{aligned}
$$

where $\delta_v = -1$ if $v = s$, 1 if $v = t$, and 0 otherwise.

An LP Relaxation of Shortest Path

Consider the following LP

Primal Shortest Path LP

$$
\begin{aligned}
&\min \textstyle\sum_{e \in E} c_e x_e \\
&\text{s.t.} \\
&\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, \quad && \forall v \in V. \\
&x_e \geq 0, && \forall e \in E.
\end{aligned}
$$

where $\delta_v = -1$ if $v = s$, 1 if $v = t$, and 0 otherwise.

- This is a relaxation of the shortest path problem
  - Indicator vector x_P of $s - t$ path P is a feasible solution, with cost as given by the objective
    - LP is feasible
  - Fractional feasible solutions may not correspond to paths

An LP Relaxation of Shortest Path

Consider the following LP

Primal Shortest Path LP

$$
\begin{aligned}
&\text{min } \textstyle\sum_{e \in E} c_e x_e \\
&\text{s.t.} \\
&\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, \quad && \forall v \in V. \\
&x_e \geq 0, && \forall e \in E.
\end{aligned}
$$

where $\delta_v = -1$ if $v = s$, 1 if $v = t$, and 0 otherwise.

- This is a relaxation of the shortest path problem
  - Indicator vector x_P of $s - t$ path P is a feasible solution, with cost as given by the objective
    - LP is feasible
  - Fractional feasible solutions may not correspond to paths
- A-priori, it is conceivable that optimal value of LP is less than length of shortest path.

Integrality of the Shortest Path Polyhedron

$$
\begin{aligned}
&\text{min } \textstyle\sum_{e \in E} c_e x_e \\
&\text{s.t.} \\
&\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, && \forall v \in V. \\
&x_e \geq 0, && \forall e \in E.
\end{aligned}
$$

We will show that above LP encodes the shortest path problem exactly

Claim

When c satisfies the no-negative-cycles property, the indicator vector of the shortest $s - t$ path is an optimal solution to the LP.

Dual LP

We will use the following LP dual

Primal LP

min $\sum_{e \in E} c_e x_e$
s.t.

$$\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, \quad \forall v \in V.$$
$$x_e \geq 0, \quad \forall e \in E.$$

Dual LP

max $y_t - y_s$
s.t.

$$y_v - y_u \leq c_e, \quad \forall (u, v) \in E.$$

- Interpretation of dual variables y_v: "height" or "potential"
- Relative potential of vertices constrained by length of edge between them (triangle inequality)
- Dual is trying to maximize relative potential of s and t,

Proof Using the Dual

Claim

When c satisfies the no-negative-cycles property, the indicator vector of the shortest $s - t$ path is an optimal solution to the LP.

Proof Using the Dual

Claim

When c satisfies the no-negative-cycles property, the indicator vector of the shortest $s - t$ path is an optimal solution to the LP.

Primal LP

$$\begin{aligned}
&\text{min } \textstyle\sum_{e \in E} c_e x_e \\
&\text{s.t.} \\
&\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, \quad && \forall v \in V. \\
&x_e \geq 0, && \forall e \in E.
\end{aligned}$$

Dual LP

$$\begin{aligned}
&\text{max } y_t - y_s \\
&\text{s.t.} \\
&y_v - y_u \leq c_e, \quad \forall (u, v) \in E.
\end{aligned}$$

Proof Using the Dual

Claim

When c satisfies the no-negative-cycles property, the indicator vector of the shortest $s-t$ path is an optimal solution to the LP.

Primal LP

min $\sum_{e \in E} c_e x_e$
s.t.

$$\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, \quad \forall v \in V.$$

$$x_e \geq 0, \qquad \forall e \in E.$$

Dual LP

max $y_t - y_s$
s.t.

$$y_v - y_u \leq c_e, \quad \forall (u, v) \in E.$$

- Let x^* be indicator vector of shortest s-t path
  - Feasible for primal

Proof Using the Dual

Claim

When c satisfies the no-negative-cycles property, the indicator vector of the shortest $s-t$ path is an optimal solution to the LP.

Primal LP

min $\sum_{e\in E} c_e x_e$
s.t.

$$\sum_{e\to v} x_e - \sum_{v\to e} x_e = \delta_v, \quad \forall v \in V.$$
$$x_e \geq 0, \quad \forall e \in E.$$

Dual LP

max $y_t - y_s$
s.t.

$$y_v - y_u \leq c_e, \quad \forall (u,v) \in E.$$

- Let x^* be indicator vector of shortest s-t path
  - Feasible for primal
- Let y^*_v be shortest path distance from s to v
  - Feasible for dual (by triangle inequality)

Proof Using the Dual

Claim

When c satisfies the no-negative-cycles property, the indicator vector of the shortest $s-t$ path is an optimal solution to the LP.

Primal LP

min $\sum_{e \in E} c_e x_e$
s.t.

$$\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, \quad \forall v \in V.$$
$$x_e \geq 0, \quad \forall e \in E.$$

Dual LP

max $y_t - y_s$
s.t.

$$y_v - y_u \leq c_e, \quad \forall (u, v) \in E.$$

- Let x^* be indicator vector of shortest s-t path
  - Feasible for primal
- Let y_v^* be shortest path distance from s to v
  - Feasible for dual (by triangle inequality)
- $\sum_e c_e x_e^* = y_t^* - y_s^*$, so both x^* and y^* optimal.

Integrality of Polyhedra

A stronger statement is true:

Integrality of Shortest Path LP

The vertices of the polyhedral feasible region are precisely the indicator vectors of simple paths in G.

- Implies that there always exists a vertex optimal solution which is a path whenever LP is bounded
  - We will also show that LP is bounded precisely when c has no negative cycles.
- Reduces computing shortest path in graphs with no negative cycles to finding optimal vertex of LP

Integrality of Polyhedra

A stronger statement is true:

Integrality of Shortest Path LP

The vertices of the polyhedral feasible region are precisely the indicator vectors of simple paths in G.

Proof

1. LP is bounded iff c satisfies no-negative-cycles
   - $\leftarrow$: previous proof
   - $\rightarrow$: If c has a negative cycle, there are arbitrarily cheap "flows" along that cycle

Integrality of Polyhedra

A stronger statement is true:

Integrality of Shortest Path LP

The vertices of the polyhedral feasible region are precisely the indicator vectors of simple paths in G.

Proof

1. LP is bounded iff c satisfies no-negative-cycles
   - $\leftarrow$: previous proof
   - $\rightarrow$: If c has a negative cycle, there are arbitrarily cheap "flows" along that cycle
2. Fact: For every LP vertex x there is objective c such that x is unique optimal. (Prove it!)

Integrality of Polyhedra

A stronger statement is true:

Integrality of Shortest Path LP

The vertices of the polyhedral feasible region are precisely the indicator vectors of simple paths in G.

Proof

1. LP is bounded iff c satisfies no-negative-cycles
   - $\leftarrow$: previous proof
   - $\rightarrow$: If c has a negative cycle, there are arbitrarily cheap "flows" along that cycle
2. Fact: For every LP vertex x there is objective c such that x is unique optimal. (Prove it!)
3. Since such a c satisfies no-negative-cycles property, claim on previous slide shows that x is integral.

Integrality of Polyhedra

A stronger statement is true:

Integrality of Shortest Path LP

The vertices of the polyhedral feasible region are precisely the indicator vectors of simple paths in G.

In general, the approach we took applies in many contexts: To show a polytope's vertices integral, it suffices to show that there is an integral optimal for any objective which admits an optimal solution.

Outline

Ford's Algorithm

Primal LP

min $\sum_{e \in E} c_e x_e$
s.t.

$$\sum_{e \to v} x_e - \sum_{v \to e} x_e = \delta_v, \quad \forall v \in V.$$

$$x_e \geq 0, \quad \forall e \in E.$$

Dual LP

max $y_t - y_s$
s.t.

$$y_v - y_u \leq c_e, \quad \forall e = (u, v) \in E.$$

For convenience, add (s, v) of length ∞ when one doesn't exist.

Ford's Algorithm

1. $y_v = c_{(s,v)}$ and $pred(v) = s$ for $v \neq s$
2. $y_s = 0$, $pred(s) = null$.
3. While some dual constraint is violated, i.e. $y_v > y_u + c_e$ for some $e = (u, v)$
   - Set $pred(v) = u$ (To get from s to v, take shortcut through u)
   - Set $y_v = y_u + c_e$
4. Output the path $t, pred(t), pred(pred(t)), \ldots, s$.

Correctness

Lemma (Loop Invariant 1)

Assuming no negative cycles, $pred$ defines a path P from s to t, of length at most $y_t - y_s$. (Hence also $y_t - y_s \geq distance(s, t)$)

Interpretation

- Ford's algorithm maintains an (initially infeasible) dual y
- Also maintains feasible primal P of length $\leq$ dual objective $y_t - y_s$
- Iteratively "fixes" dual y, tending towards feasibility
- Once y is feasible, weak duality implies P optimal.

Correctness

Lemma (Loop Invariant 1)

Assuming no negative cycles, $pred$ defines a path P from s to t, of length at most $y_t - y_s$. (Hence also $y_t - y_s \geq distance(s,t)$)

Interpretation

- Ford's algorithm maintains an (initially infeasible) dual y
- Also maintains feasible primal P of length $\leq$ dual objective $y_t - y_s$
- Iteratively "fixes" dual y, tending towards feasibility
- Once y is feasible, weak duality implies P optimal.

Correctness follows from loop invariant 1 and termination condition.

Theorem (Correctness)

If Ford's algorithm terminates, then it outputs a shortest path from s to t

Correctness

Lemma (Loop Invariant 1)

Assuming no negative cycles, $pred$ defines a path P from s to t, of length at most $y_t - y_s$. (Hence also $y_t - y_s \geq distance(s,t)$)

Interpretation

- Ford's algorithm maintains an (initially infeasible) dual y
- Also maintains feasible primal P of length $\leq$ dual objective $y_t - y_s$
- Iteratively "fixes" dual y, tending towards feasibility
- Once y is feasible, weak duality implies P optimal.

Correctness follows from loop invariant 1 and termination condition.

Theorem (Correctness)

If Ford's algorithm terminates, then it outputs a shortest path from s to t

Algorithms of this form, that output a matching primal and dual solution, are called Primal-Dual Algorithms.

Termination

Lemma (Loop Invariant 2)

Assuming no negative cycles, y_v is the length of some simple path from s to v.

Termination

Lemma (Loop Invariant 2)

Assuming no negative cycles, y_v is the length of some simple path from s to v.

Theorem (Termination)

When the graph has no negative cycles, Ford's algorithm terminates in a finite number of steps.

Proof

- The graph has a finite number N of simple paths
- By loop invariant 2, every dual variable y_v is the length of some simple path.
- Dual variables are nonincreasing throughout algorithm, and one decreases each iteration.
- There can be at most nN iterations.

Observation: Single source shortest paths

Ford's Algorithm

1. $y_v = c_{(s,v)}$ and $pred(v) = s$ for $v \neq s$
2. $y_s = 0$, $pred(s) = null$.
3. While some dual constraint is violated, i.e. $y_v > y_u + c_e$ for some $e = (u, v)$
   - Set $pred(v) = u$ (To get from s to v, take shortcut through u)
   - Set $y_v = y_u + c_e$
4. Output the path $t, pred(t), pred(pred(t)), \ldots, s$.

Observation

Algorithm does not depend on t till very last step. So essentially solves the single-source shortest path problem. i.e. finds shortest paths from s to all other vertices v.

Loop Invariant 1

We prove Loop Invariant 1 through two Lemmas

Lemma (Loop Invariant 1a)

For every node w, we have $y_w - y_{pred(w)} \geq c_{pred(w),w}$

Proof

- Fix w
- Holds at first iteration
- Preserved by Induction on iterations
  - If neither y_w nor $y_{pred(w)}$ updated, nothing changes.
  - If y_w (and $pred(w)$) updated, then $y_w = y_{pred(w)} + c_{pred(w),w}$
  - $y_{pred(w)}$ updated, it only goes down, preserving inequality.

Loop Invariant 1

Lemma (Invariant 1b)

Assuming no negative cycles, pred forms a directed tree rooted out of s.

We denote this path from s to a node w by $P(s,w)$.

Proof

- Holds at first iteration
- For a contradiction, consider iteration of first violation
  - v and u with $y_v > y_u + c_{u,v}$
- $P(s,u)$ passes through v
  - Otherwise tree property preserved by setting $pred(v) = u$
- Let $P(v,u)$ be the portion of $P(s,u)$ starting at v.
- By Invariant 1a, and telescoping sum, length of $P(v,u)$ is at most $y_u - y_v$.
- Length of cycle $\{P(v,u),(u,v)\}$ at most $y_u - y_v + c_{u,v} < 0$.

Summarizing Loop Invariant 1

Lemma (Invariant 1a)

For every node w, we have $y_w - y_{pred(w)} \geq c_{pred(w),w}$.

- By telescoping sum, can bound $y_w - y_s$ when pred leads back to s

Lemma (Invariant 1b)

Assuming no negative cycles, pred forms a directed tree rooted out of s.

- Implies that following $pred$ always leads back to s, and that y_s remains 0.

Corollary (Loop Invariant 1)

Assuming no negative cycles, $pred$ defines a path $P(s, w)$ from s to each node w, of length at most $y_w - y_s = y_w$. (Hence $y_w \geq distance(s, w)$)

Loop Invariant 2

Lemma (Loop Invariant 2)

Assuming no negative cycles, y_w is the length of some simple path $Q(s, w)$ from s to w, for all w.

Proof is technical, by induction, so we will skip. Instead, we will modify Ford's algorithm to guarantee polynomial time termination.

Bellman-Ford Algorithm

The following algorithm fixes an (arbitrary) order on edges E

Bellman-Ford Algorithm

1. $y_v = c_{(s,v)}$ and $pred(v) = s$ for $v \neq s$
2. $y_s = 0$, $pred(s) = null$.
3. While y is infeasible for the dual
   - For $e = (u, v)$ in order, if $y_v > y_u + c_e$ then
     - Set $pred(v) = u$ (To get from s to v, take shortcut through u)
     - Set $y_v = y_u + c_e$
4. Output the path $t, pred(t), pred(pred(t)), \ldots, s$.

Bellman-Ford Algorithm

The following algorithm fixes an (arbitrary) order on edges E

Bellman-Ford Algorithm

1. $y_v = c_{(s,v)}$ and $pred(v) = s$ for $v \neq s$
2. $y_s = 0$, $pred(s) = null$.
3. While y is infeasible for the dual
   - For $e = (u, v)$ in order, if $y_v > y_u + c_e$ then
     - Set $pred(v) = u$ (To get from s to v, take shortcut through u)
     - Set $y_v = y_u + c_e$
4. Output the path $t, pred(t), pred(pred(t)), \ldots, s$.

Note

Correctness follows from the correctness of Ford's Algorithm.

Runtime

Theorem

Bellman-Ford terminates after $n - 1$ scans through E, for a total runtime of $O(nm)$.

Runtime

Theorem

Bellman-Ford terminates after $n-1$ scans through E, for a total runtime of $O(nm)$.

Follows immediately from the following Lemma

Lemma

After k scans through E, vertices v with a shortest $s-v$ path consisting of $\leq k$ edges are correctly labeled. (i.e., $y_v = distance(s, v)$)

Proof

Lemma

After k scans through E, vertices v with a shortest $s - v$ path consisting of $\leq k$ edges are correctly labeled. (i.e., $y_v = distance(s, v)$)

Proof

- Holds for $k = 0$
- By induction on k.
  - Assume it holds for $k - 1$.
  - Let v be a node with a shortest path P from s with k edges.
  - $P = \{Q, e\}$, for some $e = (u, v)$ and $s - u$ path Q, where Q is a shortest $s - u$ path and Q has $k - 1$ edges.
  - By inductive hypothesis, u is correctly labeled before e is scanned for kth time – i.e. $y_u = distance(s, u)$.
  - Therefore, v is correctly labeled $y_v = y_u + c_{u,v} = distance(s, v)$ after e is scanned for kth time

A Note on Negative Cycles

Question

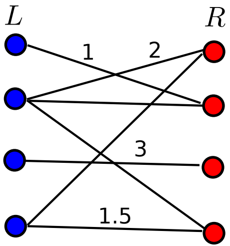
What if there are negative cycles? What does that say about LP? What about Ford's algorithm?

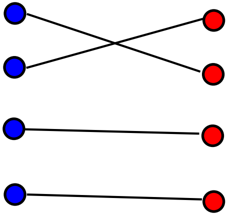
Outline

The Max-Weight Bipartite Matching Problem

Given a bipartite graph $G = (V, E)$, with $V = L \bigcup R$, and weights w_e on edges e, find a maximum weight matching.

- Matching: a set of edges covering each node at most once
- We use n and m to denote $|V|$ and $|E|$, respectively.
- Equivalent to maximum weight / minimum cost perfect matching.

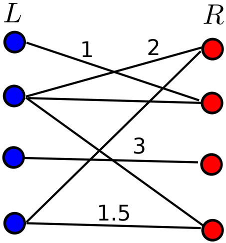


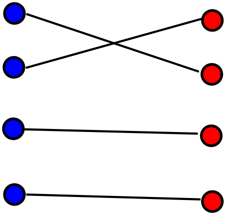


The Max-Weight Bipartite Matching Problem

Given a bipartite graph $G = (V, E)$, with $V = L \bigcup R$, and weights w_e on edges e, find a maximum weight matching.

- Matching: a set of edges covering each node at most once
- We use n and m to denote $|V|$ and $|E|$, respectively.
- Equivalent to maximum weight / minimum cost perfect matching.





Our focus will be less on algorithms, and more on using polyhedral interpretation to gain insights about a combinatorial problem.

An LP Relaxation of Bipartite Matching

Bipartite Matching LP

$$
\begin{aligned}
&\text{max } \textstyle\sum_{e\in E} w_e x_e \\
&\text{s.t.} \\
&\sum_{e\in\delta(v)} x_e \le 1, && \forall v \in V. \\
&x_e \ge 0, && \forall e \in E.
\end{aligned}
$$

An LP Relaxation of Bipartite Matching

Bipartite Matching LP

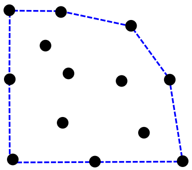
$$
\begin{aligned}
&\max \textstyle\sum_{e \in E} w_e x_e \\
&\text{s.t.} \\
&\sum_{e \in \delta(v)} x_e \leq 1, \qquad \forall v \in V. \\
&x_e \geq 0, \qquad \forall e \in E.
\end{aligned}
$$

- Feasible region is a polytope $\mathcal{P}$ (i.e. a bounded polyhedron)
- This is a relaxation of the bipartite matching problem
  - Integer points in $\mathcal{P}$ are the indicator vectors of matchings.

$$\mathcal{P} \cap \mathbb{Z}^m = \{x_M : M \text{ is a matching}\}$$

Integrality of the Bipartite Matching Polytope

$$\sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in V.$$
$$x_e \geq 0, \qquad \forall e \in E.$$



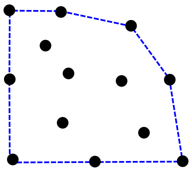
Theorem

The feasible region of the matching LP is the convex hull of indicator vectors of matchings.

$$\mathcal{P} = \text{convexhull}\{x_M : M \text{ is a matching}\}$$

Integrality of the Bipartite Matching Polytope

$$\sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in V.$$
$$x_e \geq 0, \quad \forall e \in E.$$



Theorem

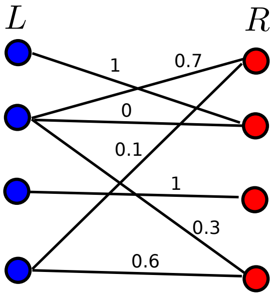
The feasible region of the matching LP is the convex hull of indicator vectors of matchings.

$$\mathcal{P} = \text{convexhull}\left\{x_M : M \text{ is a matching}\right\}$$

Note

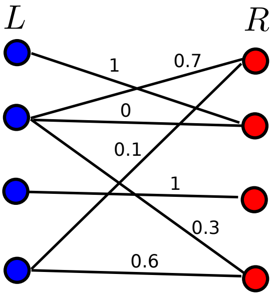
- This is the strongest guarantee you could hope for of an LP relaxation of a combinatorial problem
- Solving LP is equivalent to solving the combinatorial problem
- Stronger guarantee than shortest path LP from last time

Proof



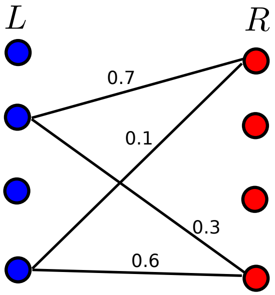
- Suffices to show that all vertices are integral (why?)

Proof



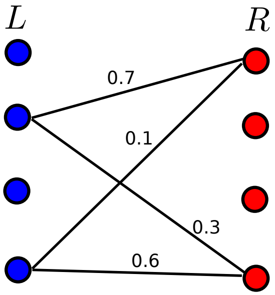
- Suffices to show that all vertices are integral (why?)
- Consider $x \in \mathcal{P}$ non-integral, we will show that x is not a vertex.

Proof



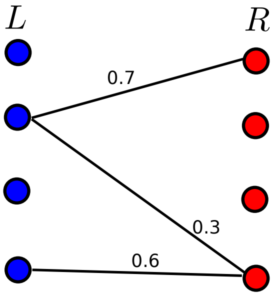
- Suffices to show that all vertices are integral (why?)
- Consider $x \in \mathcal{P}$ non-integral, we will show that x is not a vertex.
- Let H be the subgraph formed by edges with $x_e \in (0, 1)$

Proof



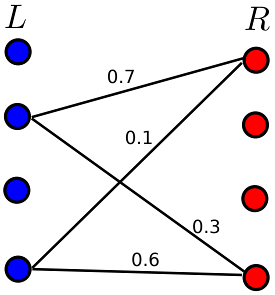
- Suffices to show that all vertices are integral (why?)
- Consider $x \in \mathcal{P}$ non-integral, we will show that x is not a vertex.
- Let H be the subgraph formed by edges with $x_e \in (0, 1)$
- H either contains a cycle, or else a maximal path which is simple.

Proof



- Suffices to show that all vertices are integral (why?)
- Consider $x \in \mathcal{P}$ non-integral, we will show that x is not a vertex.
- Let H be the subgraph formed by edges with $x_e \in (0, 1)$
- H either contains a cycle, or else a maximal path which is simple.

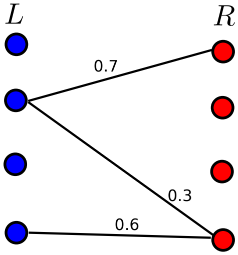
Proof



Case 1: Cycle C

- Let $C = (e_1, \ldots, e_k)$, with k even
- There is $\epsilon > 0$ such that adding $\pm\epsilon(+1, -1, \ldots, +1, -1)$ to x_C preserves feasibility
- x is the midpoint of $x + \epsilon(+1, -1, ..., +1, -1)_C$ and $x - \epsilon(+1, -1, \ldots, +1, -1)_C$, so x is not a vertex.

Proof



Case 2: Maximal Path P

- Let $P = (e_1, \ldots, e_k)$, going through vertices $v_0, v_1, \ldots, v_k$
- By maximality, e_1 is the only edge of v_0 with non-zero x-weight
  - Similarly for e_k and v_k.
- There is $\epsilon > 0$ such that adding $\pm\epsilon(+1, -1, \ldots, ?1)$ to x_P preserves feasibility
- x is the midpoint of $x + \epsilon(+1, -1, ..., ?1)_P$ and $x - \epsilon(+1, -1, \ldots, ?1)_P$, so x is not a vertex.

Related Fact: Birkhoff Von-Neumann Theorem

$$\begin{aligned} \sum_{e \in \delta(v)} x_e &= 1, \quad && \forall v \in V. \\ x_e &\geq 0, && \forall e \in E. \end{aligned}$$

- The analogous statement holds for the perfect matching LP above, by an essentially identical proof.

Related Fact: Birkhoff Von-Neumann Theorem

$$\begin{aligned} \sum_{e \in \delta(v)} x_e &= 1, \quad && \forall v \in V. \\ x_e &\geq 0, && \forall e \in E. \end{aligned}$$

- The analogous statement holds for the perfect matching LP above, by an essentially identical proof.
- When bipartite graph is complete and has the same # of nodes on either side, can be equivalently phrased as a property of matrices.

Related Fact: Birkhoff Von-Neumann Theorem

$$\sum_{e \in \delta(v)} x_e = 1, \quad \forall v \in V.$$
$$x_e \geq 0, \qquad \forall e \in E.$$

- The analogous statement holds for the perfect matching LP above, by an essentially identical proof.
- When bipartite graph is complete and has the same # of nodes on either side, can be equivalently phrased as a property of matrices.

Birkhoff Von-Neumann Theorem

The set of $n \times n$ doubly stochastic matrices is the convex hull of $n \times n$ permutation matrices.

$$\begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = 0.5 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + 0.5 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Related Fact: Birkhoff Von-Neumann Theorem

$$\sum_{e \in \delta(v)} x_e = 1, \quad \forall v \in V.$$
$$x_e \geq 0, \qquad \forall e \in E.$$

- The analogous statement holds for the perfect matching LP above, by an essentially identical proof.
- When bipartite graph is complete and has the same # of nodes on either side, can be equivalently phrased as a property of matrices.

Birkhoff Von-Neumann Theorem

The set of $n \times n$ doubly stochastic matrices is the convex hull of $n \times n$ permutation matrices.

$$\begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = 0.5 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + 0.5 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

By Caratheodory's theorem, we can express every doubly stochastic matrix as a convex combination of $n^2 + 1$ permutation matrices.

We will see later: this decomposition can be computed efficiently!

Outline

Total Unimodularity

We could have proved integrality of the bipartite matching LP using a more general tool

Definition

A matrix A is Totally Unimodular if every square submatrix has determinant 0, $+1$ or -1.

Theorem

If $A \in \mathbb{R}^{m \times n}$ is totally unimodular, and b is an integer vector, then $\{x : Ax \leq b, x \geq 0\}$ has integer vertices.

Total Unimodularity

We could have proved integrality of the bipartite matching LP using a more general tool

Definition

A matrix A is Totally Unimodular if every square submatrix has determinant 0, $+1$ or -1.

Theorem

If $A \in \mathbb{R}^{m \times n}$ is totally unimodular, and b is an integer vector, then $\{x : Ax \leq b, x \geq 0\}$ has integer vertices.

Proof

- Non-zero entries of vertex x are solution of $A'x' = b'$ for some nonsingular square submatrix A' and corresponding sub-vector b'
- Cramer's rule:

$$x_i' = \frac{\det(A_i' | b')}{\det A'}$$

Total Unimodularity of Bipartite Matching

$$\sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in V.$$

Claim

The constraint matrix of the bipartite matching LP is totally unimodular.

Total Unimodularity of Bipartite Matching

$$\sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in V.$$

Claim

The constraint matrix of the bipartite matching LP is totally unimodular.

Proof

- $A_{ve} = 1$ if e incident on v, and 0 otherwise.
- By induction on size of submatrix A'. Trivial for base case $k = 1$.

Total Unimodularity of Bipartite Matching

$$\sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in V.$$

Claim

The constraint matrix of the bipartite matching LP is totally unimodular.

Proof

- $A_{ve} = 1$ if e incident on v, and 0 otherwise.
- By induction on size of submatrix A'. Trivial for base case $k = 1$.
- If A' has all-zero column, then $\det A' = 0$

Total Unimodularity of Bipartite Matching

$$\sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in V.$$

Claim

The constraint matrix of the bipartite matching LP is totally unimodular.

Proof

- $A_{ve} = 1$ if e incident on v, and 0 otherwise.
- By induction on size of submatrix A'. Trivial for base case $k = 1$.
- If A' has all-zero column, then $\det A' = 0$
- If A' has column with single 1, then holds by induction.

Total Unimodularity of Bipartite Matching

$$\sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in V.$$

Claim

The constraint matrix of the bipartite matching LP is totally unimodular.

Proof

- $A_{ve} = 1$ if e incident on v, and 0 otherwise.
- By induction on size of submatrix A'. Trivial for base case $k = 1$.
- If A' has all-zero column, then $\det A' = 0$
- If A' has column with single 1, then holds by induction.
- If all columns of A' have two 1's,
  - Partition rows (vertices) into L and R
  - Sum of rows L is $(1, 1, \ldots, 1)$, similarly for R
  - A' is singular, so $\det A' = 0$.

Outline

Primal and Dual LPs

Primal LP

$$\max \sum_{e \in E} w_e x_e$$
s.t.
$$\sum_{e \in \delta(v)} x_e \leq 1, \qquad \forall v \in V.$$
$$x_e \geq 0, \qquad \forall e \in E.$$

Dual LP

$$\min \sum_{v \in V} y_v$$
s.t.
$$y_u + y_v \geq w_e, \quad \forall e = (u, v) \in E.$$
$$y_v \succeq 0, \qquad \forall v \in V.$$

- Primal interpertation: Player 1 looking to build a set of projects
  - Each edge e is a project generating “profit” w_e
  - Each project $e = (u, v)$ needs two resources, u and v
  - Each resource can be used by at most one project at a time
  - Must choose a profit-maximizing set of projects

Primal and Dual LPs

Primal LP

max $\sum_{e \in E} w_e x_e$
s.t.

$$\sum_{e \in \delta(v)} x_e \le 1, \qquad \forall v \in V.$$

$$x_e \ge 0, \qquad \forall e \in E.$$

Dual LP

min $\sum_{v \in V} y_v$
s.t.

$$y_u + y_v \ge w_e, \quad \forall e = (u, v) \in E.$$

$$y_v \succeq 0, \qquad \forall v \in V.$$

- Primal interpretation: Player 1 looking to build a set of projects
  - Each edge e is a project generating "profit" w_e
  - Each project $e = (u, v)$ needs two resources, u and v
  - Each resource can be used by at most one project at a time
  - Must choose a profit-maximizing set of projects
- Dual interpretation: Player 2 looking to buy resources
  - Offer a price y_v for each resource.
  - Prices should incentivize player 1 to sell resources
  - Want to pay as little as possible.

Vertex Cover Interpretation

Primal LP

$$\max \sum_{e \in E} x_e$$
s.t.
$$\sum_{e \in \delta(v)} x_e \leq 1, \qquad \forall v \in V.$$
$$x_e \geq 0, \qquad \forall e \in E.$$

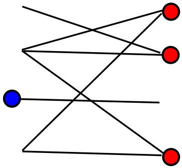
Dual LP

$$\min \sum_{v \in V} y_v$$
s.t.
$$y_u + y_v \geq 1, \qquad \forall e = (u, v) \in E.$$
$$y_v \succeq 0, \qquad \forall v \in V.$$

When edge weights are 1, binary solutions to dual are vertex covers

Definition

$C \subseteq V$ is a vertex cover if every $e \in E$ has at least one endpoint in C



Vertex Cover Interpretation

Primal LP

max $\sum_{e \in E} x_e$
s.t.

$$\sum_{e \in \delta(v)} x_e \leq 1, \qquad \forall v \in V.$$
$$x_e \geq 0, \qquad \forall e \in E.$$

Dual LP

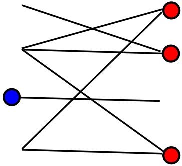
min $\sum_{v \in V} y_v$
s.t.

$$y_u + y_v \geq 1, \qquad \forall e = (u, v) \in E.$$
$$y_v \succeq 0, \qquad \forall v \in V.$$

When edge weights are 1, binary solutions to dual are vertex covers

Definition

$C \subseteq V$ is a vertex cover if every $e \in E$ has at least one endpoint in C



- Dual is a relaxation of the minimum vertex cover problem for bipartite graphs.
- By weak duality: min-vertex-cover $\geq$ max-cardinality-matching

König's Theorem

Primal LP

max $\sum_{e\in E} x_e$
s.t.

$$\begin{aligned} \sum_{e\in\delta(v)} x_e &\le 1, \qquad &\forall v\in V.\\ x_e &\ge 0, &\forall e\in E. \end{aligned}$$

Dual LP

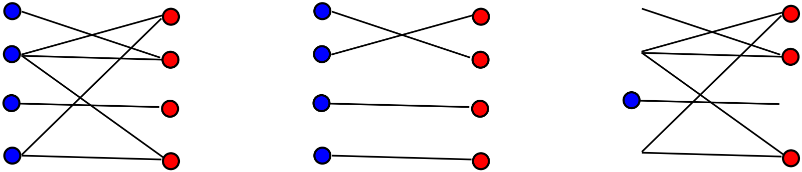
min $\sum_{v\in V} y_v$
s.t.

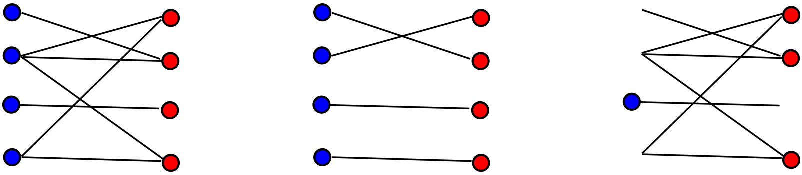
$$\begin{aligned} y_u + y_v &\ge 1, \qquad &\forall e=(u,v)\in E.\\ y_v &\succeq 0, &\forall v\in V. \end{aligned}$$

König's Theorem

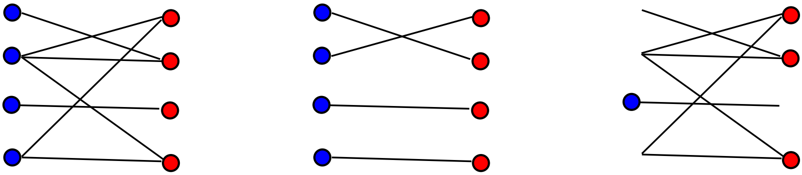
In a bipartite graph, the cardinality of the maximum matching is equal to the cardinality of the minimum vertex cover.

i.e. the dual LP has an integral optimal solution

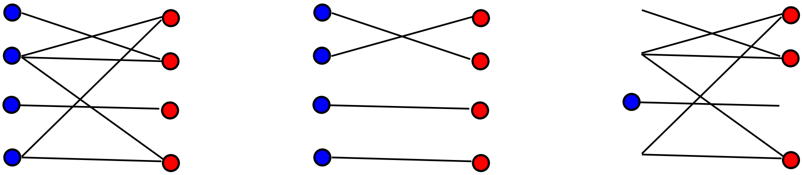




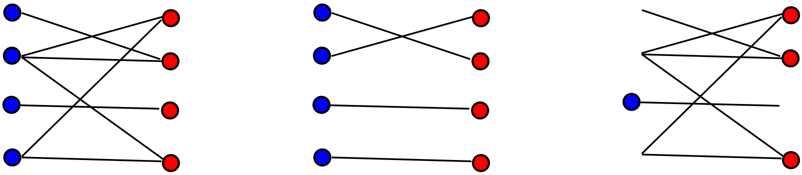
- Let $M(G)$ be a max cardinality of a matching in G
- Let $C(G)$ be min cardinality of a vertex cover in G
- We already proved that $M(G) \leq C(G)$
- We will prove $C(G) \leq M(G)$ by induction on number of nodes in G.



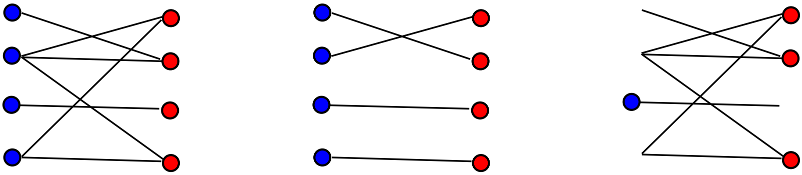
- Let y be an optimal dual, and v a vertex with $y_v > 0$



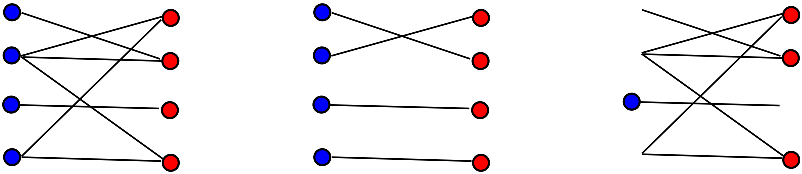
- Let y be an optimal dual, and v a vertex with $y_v > 0$
- By integrality of matching LP, and complementary slackness, every maximum cardinality matching must match v.



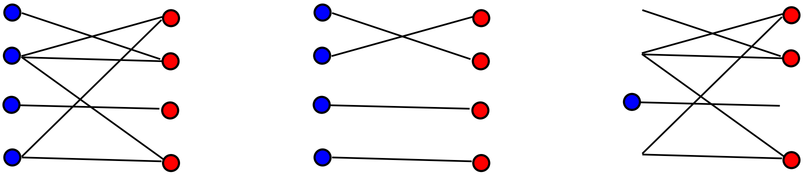
- Let y be an optimal dual, and v a vertex with $y_v > 0$
- By integrality of matching LP, and complementary slackness, every maximum cardinality matching must match v.
  - $M(G \setminus v) = M(G) - 1$



- Let y be an optimal dual, and v a vertex with $y_v > 0$
- By integrality of matching LP, and complementary slackness, every maximum cardinality matching must match v.
  - $M(G \setminus v) = M(G) - 1$
- By inductive hypothesis, $C(G \setminus v) = M(G \setminus v) = M(G) - 1$



- Let y be an optimal dual, and v a vertex with $y_v > 0$
- By integrality of matching LP, and complementary slackness, every maximum cardinality matching must match v.
  - $M(G \setminus v) = M(G) - 1$
- By inductive hypothesis, $C(G \setminus v) = M(G \setminus v) = M(G) - 1$
- $C(G) \leq C(G \setminus v) + 1 = M(G)$.



- Let y be an optimal dual, and v a vertex with $y_v > 0$
- By integrality of matching LP, and complementary slackness, every maximum cardinality matching must match v.
  - $M(G \setminus v) = M(G) - 1$
- By inductive hypothesis, $C(G \setminus v) = M(G \setminus v) = M(G) - 1$
- $C(G) \leq C(G \setminus v) + 1 = M(G)$.

Note: Could have proved the same using total unimodularity

Consequences of König's Theorem

- Vertex covers can serve as a certificate of optimality for bipartite matchings, and vice versa

Consequences of König's Theorem

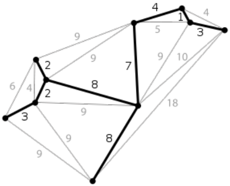
- Vertex covers can serve as a certificate of optimality for bipartite matchings, and vice versa
- Like maximum cardinality matching, minimum cardinality vertex cover in bipartite graphs can be formulated as an LP, and solved in polynomial time

Consequences of König's Theorem

- Vertex covers can serve as a certificate of optimality for bipartite matchings, and vice versa
- Like maximum cardinality matching, minimum cardinality vertex cover in bipartite graphs can be formulated as an LP, and solved in polynomial time
- The same is true for the maximum independent set problem in bipartite graphs.
  - C is a vertex cover iff $V \setminus C$ is an independent set.

Outline

The Minimum Cost Spanning Tree Problem



Given a connected undirected graph $G = (V, E)$, and costs c_e on edges e, find a minimum cost spanning tree of G.

- Spanning Tree: an acyclic set of edges connecting every pair of nodes
- When graph is disconnected, can search for min-cost spanning forest instead
- We use n and m to denote $|V|$ and $|E|$, respectively.

Kruskal's Algorithm

The minimum spanning tree problem can be solved efficiently by a simple greedy algorithm

Kruskal's algorithm

1. $T = \emptyset$
2. Sort edges in increasing order of cost
3. For each edge e in order
    - if $T \bigcup e$ is acyclic, add e to T.

Kruskal's Algorithm

The minimum spanning tree problem can be solved efficiently by a simple greedy algorithm

Kruskal's algorithm

1. $T = \emptyset$
2. Sort edges in increasing order of cost
3. For each edge e in order
    - if $T \bigcup e$ is acyclic, add e to T.

- Proof of correctness is via a simple exchange argument.
- Generalizes to Matroids

MST Linear Program

MST LP

$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \displaystyle\sum_{e \in E} x_e = n - 1 & \\
 & \displaystyle\sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for } X \subset V. \\
 & x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

MST Linear Program

MST LP

$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \displaystyle\sum_{e \in E} x_e = n - 1 & \\
& \displaystyle\sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for } X \subset V. \\
& x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

Theorem

The feasible region of the above LP is the convex hull of spanning trees.

MST Linear Program

MST LP

$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \displaystyle\sum_{e \in E} x_e = n - 1 & \\
& \displaystyle\sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for } X \subset V. \\
& x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

Theorem

The feasible region of the above LP is the convex hull of spanning trees.

- Proof by finding a dual solution with cost matching the output of Kruskal's algorithm (See KV book)

MST Linear Program

MST LP

$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \sum_{e \in E} x_e = n - 1 & \\
 & \sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for } X \subset V. \\
 & x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

Theorem

The feasible region of the above LP is the convex hull of spanning trees.

- Proof by finding a dual solution with cost matching the output of Kruskal's algorithm (See KV book)
- Generalizes to Matroids

MST Linear Program

MST LP

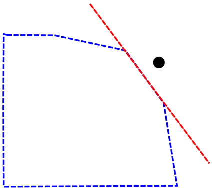
$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \displaystyle\sum_{e \in E} x_e = n - 1 & \\
& \displaystyle\sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for } X \subset V. \\
& x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

Theorem

The feasible region of the above LP is the convex hull of spanning trees.

- Proof by finding a dual solution with cost matching the output of Kruskal's algorithm (See KV book)
- Generalizes to Matroids
- Note: this LP has an exponential (in n) number of constraints

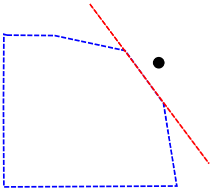
Solving the MST Linear Program



Definition

A separation oracle for a linear program with feasible set $\mathcal{P} \subseteq \mathbb{R}^m$ is an algorithm which takes as input $x \in \mathbb{R}^m$, and either certifies that $x \in \mathcal{P}$ or identifies a violated constraint.

Solving the MST Linear Program



Definition

A separation oracle for a linear program with feasible set $\mathcal{P} \subseteq \mathbb{R}^m$ is an algorithm which takes as input $x \in \mathbb{R}^m$, and either certifies that $x \in \mathcal{P}$ or identifies a violated constraint.

Theorem

A linear program with a polynomial number of variables is solvable in polynomial time if and only if it admits a polynomial time separation oracle (modulo some technicalities)

Follows from the ellipsoid method, which we will see next week.

Solving the MST Linear Program

Primal LP

$$\begin{array}{lll} \text{minimize} & \sum_{e \in E} c_e x_e & \\ \text{subject to} & \displaystyle\sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for nonempty } X \subset V. \\ & \displaystyle\sum_{e \in E} x_e = n - 1 & \\ & x_e \geq 0, & \text{for } e \in E. \end{array}$$

- Given $x \in \mathbb{R}^m$, separation oracle must find a violated constraint if one exists

Solving the MST Linear Program

Primal LP

$$\begin{array}{lll} \text{minimize} & \sum_{e \in E} c_e x_e & \\ \text{subject to} & \sum\limits_{e \subseteq X} x_e \leq |X| - 1, & \text{for nonempty } X \subset V. \\ & \sum\limits_{e \in E} x_e = n - 1 & \\ & x_e \geq 0, & \text{for } e \in E. \end{array}$$

- Given $x \in \mathbb{R}^m$, separation oracle must find a violated constraint if one exists
- Reduces to finding nonempty $X \subset V$ with $\sum_{e \subseteq X} x_e > |X| - 1$, if one exists
  - Equivalently $|X| - \sum_{e \subseteq X} x_e < 1$

Solving the MST Linear Program

Primal LP

$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \sum\limits_{e \subseteq X} x_e \leq |X| - 1, & \text{for nonempty } X \subset V. \\
& \sum\limits_{e \in E} x_e = n - 1 & \\
& x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

- Given $x \in \mathbb{R}^m$, separation oracle must find a violated constraint if one exists
- Reduces to finding nonempty $X \subset V$ with $\sum_{e \subseteq X} x_e > |X| - 1$, if one exists
  - Equivalently $|X| - \sum_{e \subseteq X} x_e < 1$
- In turn, this reduces to minimizing $|X| - \sum_{e \subseteq X} x_e$ over $X \subset V$

Solving the MST Linear Program

Primal LP

$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for nonempty } X \subset V. \\
& \sum_{e \in E} x_e = n - 1 & \\
& x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

- Given $x \in \mathbb{R}^m$, separation oracle must find a violated constraint if one exists
- Reduces to finding nonempty $X \subset V$ with $\sum_{e \subseteq X} x_e > |X| - 1$, if one exists
  - Equivalently $|X| - \sum_{e \subseteq X} x_e < 1$
- In turn, this reduces to minimizing $|X| - \sum_{e \subseteq X} x_e$ over $X \subset V$

We will see how to do this efficiently later in the class, using submodular minimization

Application of Fractional Spanning Trees

- The LP formulation of spanning trees has many applications
- We will look at one contrived yet simple application that shows the flexibility enabled by polyhedral formulation

Fault-Tolerant MST

- Your tree is an overlay network on the internet used to transmit data
- A hacker is looking to attack your tree, by knocking off one of the edges of the graph
- You can foil the hacker by choosing a random tree
- The hacker knows the algorithm you use, but not your random coins

Fault-tolerant MST LP

$$\begin{array}{lll} \text{minimize} & \sum_{e \in E} c_e x_e & \\ \text{subject to} & \displaystyle\sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for } X \subset V. \\ & \displaystyle\sum_{e \in E} x_e = n - 1 & \\ & x_e \leq p, & \text{for } e \in E. \\ & x_e \geq 0, & \text{for } e \in E. \end{array}$$

- Above LP can be solved efficiently
- If feasible, can interpret resulting fractional spanning tree x as a recipe for a probability distribution over trees T
  - $e \in T$ with probability x_e
  - Since $x_e \leq p$, no edge is in the tree with probability more than p.

Fault-tolerant MST LP

$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \displaystyle\sum_{e \subseteq X} x_e \le |X| - 1, & \text{for } X \subset V. \\
& \displaystyle\sum_{e \in E} x_e = n - 1 & \\
& x_e \le p, & \text{for } e \in E. \\
& x_e \ge 0, & \text{for } e \in E.
\end{array}
$$

- Given feasible solution x, such a probability distribution exists!

Fault-tolerant MST LP

$$
\begin{array}{lll}
\text{minimize} & \sum_{e \in E} c_e x_e & \\
\text{subject to} & \displaystyle\sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for } X \subset V. \\
& \displaystyle\sum_{e \in E} x_e = n - 1 & \\
& x_e \leq p, & \text{for } e \in E. \\
& x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

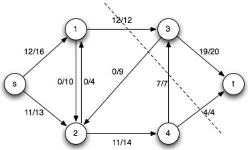
- Given feasible solution x, such a probability distribution exists!
  - x is in the (original) MST polytope
  - Caratheodory's theorem: x is a convex combination of $m + 1$ vertices of MST polytope
  - By integrality of MST polytope: x is the “expectation” of a probability distribution over spanning trees.

Fault-tolerant MST LP

$$\begin{array}{lll} \text{minimize} & \sum_{e \in E} c_e x_e & \\ \text{subject to} & \displaystyle\sum_{e \subseteq X} x_e \leq |X| - 1, & \text{for } X \subset V. \\ & \displaystyle\sum_{e \in E} x_e = n - 1 & \\ & x_e \leq p, & \text{for } e \in E. \\ & x_e \geq 0, & \text{for } e \in E. \end{array}$$

- Given feasible solution x, such a probability distribution exists!
  - x is in the (original) MST polytope
  - Caratheodory's theorem: x is a convex combination of $m + 1$ vertices of MST polytope
  - By integrality of MST polytope: x is the "expectation" of a probability distribution over spanning trees.
- Consequence of Ellipsoid algorithm: can compute such a decomposition of x efficiently!

Outline



The Maximum Flow Problem

Given a directed graph $G = (V, E)$ with capacities u_e on edges e, a source node s, and a sink node t, find a maximum flow from s to t respecting the capacities.

$$
\begin{array}{lll}
\text{maximize} & \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e & \\
\text{subject to} & \sum_{e \in \delta^-(v)} x_e = \sum_{e \in \delta^+(v)} x_e, & \text{for } v \in V \setminus \{s, t\}. \\
& x_e \le u_e, & \text{for } e \in E. \\
& x_e \ge 0, & \text{for } e \in E.
\end{array}
$$

Can be computed either by solving the LP, or by a combinatorial algorithm such as Ford Fulkerson.

Primal LP

$$\max \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e$$

s.t.

$$\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, \qquad \forall v \in V \setminus \{s,t\}$$

$$x_e \le u_e, \qquad \forall e \in E.$$

$$x_e \ge 0, \qquad \forall e \in E.$$

Dual LP (Simplified)

$$\min \sum_{e\in E} u_e z_e$$

s.t.

$$y_v - y_u \le z_e, \qquad \forall e = (u,v) \in E.$$

$$y_s = 0$$

$$y_t = 1$$

$$z_e \ge 0, \qquad \forall e \in E.$$

- Dual solution describes fraction z_e of each edge to fractionally cut

Primal LP

$$\max \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e$$

s.t.

$$\sum_{e \in \delta^-(v)} x_e = \sum_{e \in \delta^+(v)} x_e, \qquad \forall v \in V \setminus \{s,t\}$$

$$x_e \leq u_e, \qquad \forall e \in E.$$

$$x_e \geq 0, \qquad \forall e \in E.$$

Dual LP (Simplified)

$$\min \textstyle\sum_{e \in E} u_e z_e$$

s.t.

$$y_v - y_u \leq z_e, \qquad \forall e = (u,v) \in E.$$

$$y_s = 0$$

$$y_t = 1$$

$$z_e \geq 0, \qquad \forall e \in E.$$

- Dual solution describes fraction z_e of each edge to fractionally cut
- Dual constraints require that at least 1 edge is cut on every path from s to t.
  - $\sum_{(u,v) \in P} z_{uv} \geq \sum_{(u,v) \in P} y_v - y_u = y_t - y_s = 1$

Primal LP

$$\max \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e$$

s.t.

$$\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, \qquad \forall v \in V \setminus \{s,t\}$$

$$x_e \le u_e, \qquad \forall e \in E.$$

$$x_e \ge 0, \qquad \forall e \in E.$$

Dual LP (Simplified)

$$\min \sum_{e\in E} u_e z_e$$

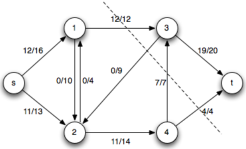
s.t.

$$y_v - y_u \le z_e, \qquad \forall e = (u,v) \in E.$$

$$y_s = 0$$

$$y_t = 1$$

$$z_e \ge 0, \qquad \forall e \in E.$$



- Every integral $s-t$ cut is feasible.

Primal LP

$$\max \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e$$

s.t.

$$\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, \qquad \forall v \in V \setminus \{s,t\}$$

$$x_e \le u_e, \qquad \forall e \in E.$$

$$x_e \ge 0, \qquad \forall e \in E.$$

Dual LP (Simplified)

$$\min \sum_{e\in E} u_e z_e$$

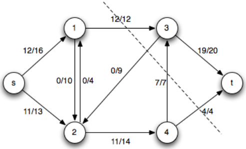
s.t.

$$y_v - y_u \le z_e, \qquad \forall e = (u,v) \in E.$$

$$y_s = 0$$

$$y_t = 1$$

$$z_e \ge 0, \qquad \forall e \in E.$$



- Every integral $s-t$ cut is feasible.
- By weak duality: max flow $\le$ minimum cut

Primal LP

$$\max \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e$$

s.t.

$$\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, \qquad \forall v \in V \setminus \{s,t\}$$

$$x_e \le u_e, \qquad \forall e \in E.$$

$$x_e \ge 0, \qquad \forall e \in E.$$

Dual LP (Simplified)

$$\min \textstyle\sum_{e\in E} u_e z_e$$

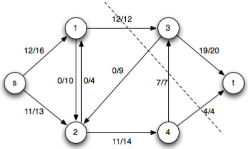
s.t.

$$y_v - y_u \le z_e, \qquad \forall e = (u,v) \in E.$$

$$y_s = 0$$

$$y_t = 1$$

$$z_e \ge 0, \qquad \forall e \in E.$$



- Every integral $s-t$ cut is feasible.
- By weak duality: max flow $\le$ minimum cut
- Ford-Fulkerson shows that max flow $=$ min cut
  - i.e. dual has integer optimal

Primal LP

$$\max \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e$$

s.t.

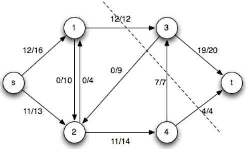
$$\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, \qquad \forall v \in V \setminus \{s,t\}$$
$$x_e \leq u_e, \qquad \forall e \in E.$$
$$x_e \geq 0, \qquad \forall e \in E.$$

Dual LP (Simplified)

$$\min \textstyle\sum_{e\in E} u_e z_e$$

s.t.

$$y_v - y_u \leq z_e, \qquad \forall e = (u,v) \in E.$$
$$y_s = 0$$
$$y_t = 1$$
$$z_e \geq 0, \qquad \forall e \in E.$$



- Every integral $s-t$ cut is feasible.
- By weak duality: max flow $\leq$ minimum cut
- Ford-Fulkerson shows that max flow $=$ min cut
  - i.e. dual has integer optimal
- Ford-Fulkerson also shows that there is an integral optimal flow when capacities are integer.

Generalizations of Max Flow

$$\begin{aligned}
&\text{max} \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e \\
&\text{s.t.} \\
&\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, && \forall v \in V \setminus \{s,t\}. \\
&x_e \le u_e, && \forall e \in E. \\
&x_e \ge 0, && \forall e \in E.
\end{aligned}$$

Writing as an LP shows that many generalizations are also tractable

Generalizations of Max Flow

$$\max \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e$$

s.t.

$$\begin{aligned} \sum_{e\in\delta^-(v)} x_e &= \sum_{e\in\delta^+(v)} x_e, & \forall v \in V \setminus \{s,t\}. \\ x_e &\le u_e, & \forall e \in E. \\ x_e &\ge 0, & \forall e \in E. \end{aligned}$$

Writing as an LP shows that many generalizations are also tractable

- Lower and upper bound constraints on flow: $\ell_e \le x_e \le u_e$

Generalizations of Max Flow

$$\max \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e$$

s.t.

$$\begin{aligned}
&\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, && \forall v \in V \setminus \{s,t\}. \\
&x_e \le u_e, && \forall e \in E. \\
&x_e \ge 0, && \forall e \in E.
\end{aligned}$$

Writing as an LP shows that many generalizations are also tractable

- Lower and upper bound constraints on flow: $\ell_e \le x_e \le u_e$
- minimum cost flow of a certain amount r
  - Objective $\min \sum_e c_e x_e$
  - Additional constraint: $\sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e = r$

Generalizations of Max Flow

$$\max \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e$$

s.t.

$$\sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, \qquad \forall v \in V \setminus \{s,t\}.$$

$$x_e \le u_e, \qquad \forall e \in E.$$

$$x_e \ge 0, \qquad \forall e \in E.$$

Writing as an LP shows that many generalizations are also tractable

- Lower and upper bound constraints on flow: $\ell_e \le x_e \le u_e$
- minimum cost flow of a certain amount r
    - Objective $\min \sum_e c_e x_e$
    - Additional constraint: $\sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e = r$
- Multiple commodities sharing the network

Generalizations of Max Flow

$$
\begin{aligned}
\text{max} \quad & \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e \\
\text{s.t.} \quad & \\
& \sum_{e \in \delta^-(v)} x_e = \sum_{e \in \delta^+(v)} x_e, && \forall v \in V \setminus \{s,t\}. \\
& x_e \le u_e, && \forall e \in E. \\
& x_e \ge 0, && \forall e \in E.
\end{aligned}
$$

Writing as an LP shows that many generalizations are also tractable

- Lower and upper bound constraints on flow: $\ell_e \le x_e \le u_e$
- minimum cost flow of a certain amount r
  - Objective $\min \sum_e c_e x_e$
  - Additional constraint: $\sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e = r$
- Multiple commodities sharing the network
- …

Minimum Congestion Flow

You are given a directed graph $G = (V, E)$ with congestion functions $c_e(.)$ on edges e, a source node s, a sink node t, and a desired flow amount r. Find a minimum average congestion flow from s to t.

$$
\begin{array}{lll}
\text{minimize} & \sum_e x_e c_e(x_e) & \\
\text{subject to} & \sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e = r & \\
 & \sum_{e\in\delta^-(v)} x_e = \sum_{e\in\delta^+(v)} x_e, & \text{for } v \in V \setminus \{s,t\}\,. \\
 & x_e \geq 0, & \text{for } e \in E.
\end{array}
$$

When $c_e(.)$ are polynomials with nonnegative co-efficients, e.g. $c_e(x) = a_e x^2 + b_e x + c_e$ with $a_e, b_e, c_e \geq 0$, this is a (non-linear) convex program.

Outline

The Max Cut Problem

Given an undirected graph $G = (V, E)$, find a partition of V into $(S, V \setminus S)$ maximizing number of edges with exactly one end in S.

$$\begin{array}{ll} \text{maximize} & \sum_{(i,j)\in E} \frac{1 - x_i x_j}{2} \\ \text{subject to} & x_i \in \{-1, 1\}, \qquad \text{for } i \in V. \end{array}$$

The Max Cut Problem

Given an undirected graph $G = (V, E)$, find a partition of V into $(S, V \setminus S)$ maximizing number of edges with exactly one end in S.

$$\begin{array}{lll} \text{maximize} & \sum_{(i,j)\in E} \frac{1-x_i x_j}{2} & \\ \text{subject to} & x_i \in \{-1, 1\}, & \text{for } i \in V. \end{array}$$

Instead of requiring x_i to be on the 1 dimensional sphere, we relax and permit it to be in the n-dimensional sphere, where $n = |V|$.

Vector Program relaxation

$$\begin{array}{lll} \text{maximize} & \sum_{(i,j)\in E} \frac{1-\vec{v}_i\cdot\vec{v}_j}{2} & \\ \text{subject to} & ||\vec{v}_i||_2 = 1, & \text{for } i \in V. \\ & \vec{v}_i \in \mathbb{R}^n, & \text{for } i \in V. \end{array}$$

SDP Relaxation

- Recall: A symmetric $n \times n$ matrix Y is PSD iff $Y = V^T V$ for $n \times n$ matrix V
- Equivalently: PSD matrices encode pairwise dot products of columns of V
- When diagonal entries of Y are 1, V has unit length columns
- Recall: Y and V can be recovered from each other efficiently

SDP Relaxation

- Recall: A symmetric $n \times n$ matrix Y is PSD iff $Y = V^T V$ for $n \times n$ matrix V
- Equivalently: PSD matrices encode pairwise dot products of columns of V
- When diagonal entries of Y are 1, V has unit length columns
- Recall: Y and V can be recovered from each other efficiently

Vector Program relaxation

$$\begin{array}{lll} \text{maximize} & \sum_{(i,j)\in E} \frac{1-\vec{v}_i\cdot\vec{v}_j}{2} & \\ \text{subject to} & ||\vec{v}_i||_2 = 1, & \text{for } i \in V. \\ & \vec{v}_i \in \mathbb{R}^n, & \text{for } i \in V. \end{array}$$

SDP Relaxation

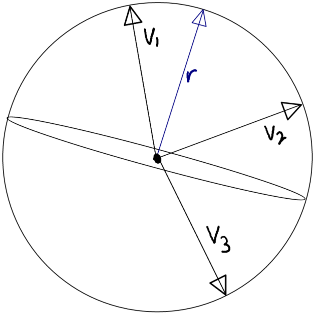
$$\begin{array}{lll} \text{maximize} & \sum_{(i,j)\in E} \frac{1-Y_{ij}}{2} & \\ \text{subject to} & Y_{ii} = 1, & \text{for } i \in V. \\ & Y \in S^n_+ & \end{array}$$

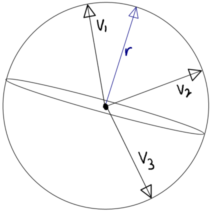
Goemans Williamson Algorithm for Max Cut

1. Solve the SDP to get $Y \succeq 0$
2. Decompose Y to VV^T
3. Draw random vector r on unit sphere
4. Place nodes i with $v_i \cdot r \geq 0$ on one side of cut, the rest on the other side

SDP Relaxation

$$\begin{array}{ll} \text{maximize} & \sum_{(i,j)\in E} \frac{1-Y_{ij}}{2} \\ \text{subject to} & Y_{ii} = 1 \; \forall i \\ & Y \in S^n_+ \end{array}$$

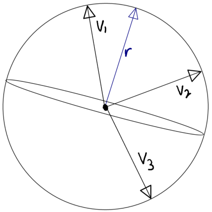




We will prove the following Lemma

Lemma

The random hyperplane cuts each edge (i, j) with probability at least $0.878\frac{1-Y_{ij}}{2}$



We will prove the following Lemma

Lemma

The random hyperplane cuts each edge (i,j) with probability at least $0.878\frac{1-Y_{ij}}{2}$

Therefore, by linearity of expectations, and the fact that $OPT_{SDP} \geq OPT$ (i.e. relaxation).

Theorem

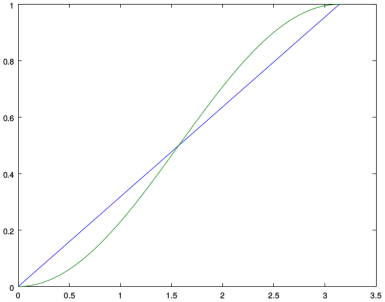
The Goemans Williamson algorithm outputs a random cut of expected size at least $0.878\,OPT$.

We use the following fact

Fact

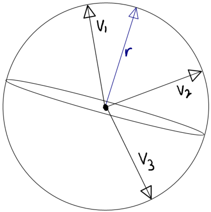
For all angles $\theta \in [0, \pi]$,

$$\frac{\theta}{\pi} \geq 0.878 \cdot \frac{1 - cos(\theta)}{2}$$



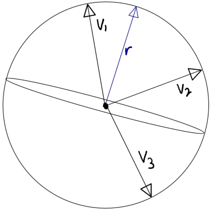
Lemma

The random hyperplane cuts each edge (i, j) with probability at least $0.878\frac{1-Y_{ij}}{2}$



Lemma

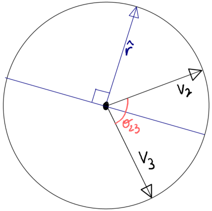
The random hyperplane cuts each edge (i, j) with probability at least $0.878\frac{1-Y_{ij}}{2}$



- (i, j) is cut iff $sign(r \cdot v_i) \neq sign(r \cdot v_j)$

Lemma

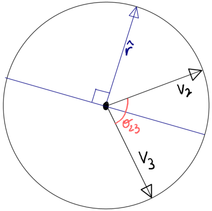
The random hyperplane cuts each edge (i, j) with probability at least $0.878\frac{1-Y_{ij}}{2}$



- (i, j) is cut iff $sign(r \cdot v_i) \neq sign(r \cdot v_j)$
- Can zoom in on the 2-d plane which includes v_i and v_j
  - Discard component r perpendicular to that plane, leaving $\widehat{r}$
  - Direction of $\widehat{r}$ is uniform in the plane

Lemma

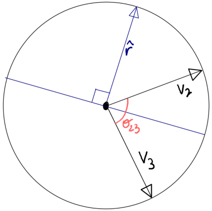
The random hyperplane cuts each edge (i, j) with probability at least $0.878\frac{1-Y_{ij}}{2}$



- (i, j) is cut iff $sign(r \cdot v_i) \neq sign(r \cdot v_j)$
- Can zoom in on the 2-d plane which includes v_i and v_j
  - Discard component r perpendicular to that plane, leaving $\widehat{r}$
  - Direction of $\widehat{r}$ is uniform in the plane
- Let θ_{ij} be angle between v_i and v_j. Note $Y_{ij} = v_i \cdot v_j = \cos(\theta_{ij})$

Lemma

The random hyperplane cuts each edge (i, j) with probability at least $0.878\frac{1-Y_{ij}}{2}$



- (i, j) is cut iff $sign(r \cdot v_i) \neq sign(r \cdot v_j)$
- Can zoom in on the 2-d plane which includes v_i and v_j
  - Discard component r perpendicular to that plane, leaving $\widehat{r}$
  - Direction of $\widehat{r}$ is uniform in the plane
- Let θ_{ij} be angle between v_i and v_j. Note $Y_{ij} = v_i \cdot v_j = \cos(\theta_{ij})$
- $\widehat{r}$ cuts (i, j) w.p.

$$\frac{2\theta_{ij}}{2\pi} = \frac{\theta_{ij}}{\pi} \geq 0.878\frac{1 - \cos\theta_{ij}}{2} = 0.878\frac{1 - Y_{ij}}{2}$$