Approximation algorithms for discrete stochastic optimization problems

David B. Shmoys
Cornell University

Stochastic Optimization

- Way of modeling uncertainty.
- Exact data is unavailable or expensive – data is uncertain, specified by a probability distribution.

  Want to make the best decisions given this uncertainty in the data.
- Dates back to 1950's and the work of Dantzig.
- Applications in logistics, transportation models, financial instruments, network design, production planning, …

Two-Stage Recourse Model

Given : Probability distribution over inputs.

Stage I : Make some advance decisions – plan ahead or hedge against uncertainty.

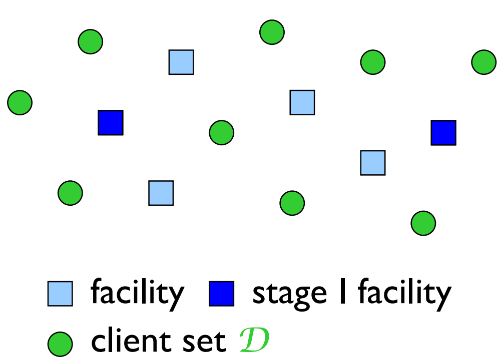
Observe the actual input scenario.

Stage II : Take recourse. Can augment earlier solution paying a recourse cost.

Choose stage I decisions to minimize

(stage I cost) + (expected stage II recourse cost).

2-Stage Stochastic Facility Location

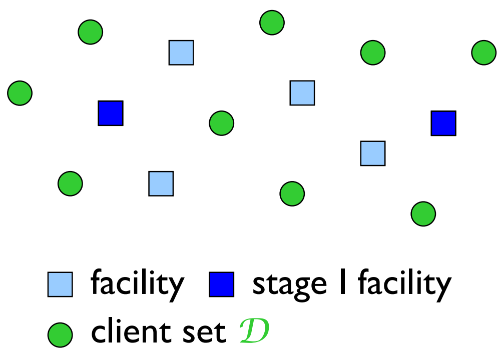


Distribution over clients gives the set of clients to serve.

Stage I: Open some facilities in advance; pay cost f_i for facility i.

Stage I cost = $\sum_{(i \text{ opened})} f_i$.

2-Stage Stochastic Facility Location



Distribution over clients gives the set of clients to serve.

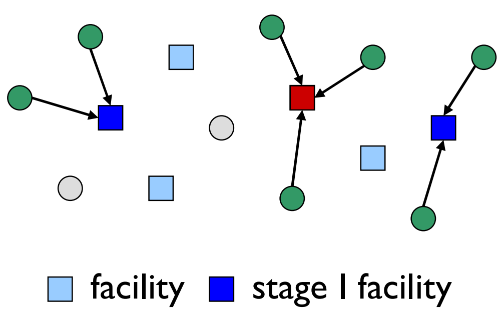
Stage I: Open some facilities in advance; pay cost f_i for facility i.

Stage I cost = $\sum_{(i \text{ opened})} f_i$.

How is the probability distribution on clients specified?

- A short (polynomial) list of possibile scenarios;
- Independent probabilities that each client exists;
- A black box that can be sampled.

2-Stage Stochastic Facility Location



Distribution over clients gives the set of clients to serve.

Stage I: Open some facilities in advance; pay cost f_i for facility i.

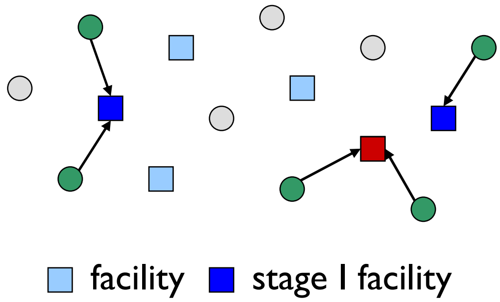
Stage I cost = $\sum_{(i \text{ opened})} f_i$.

Actual scenario A = { ● clients to serve}, materializes.

Stage II: Can open more facilities to serve clients in A; pay cost f_i^A to open facility i. Assign clients in A to facilities.

Stage II cost = $\sum_{\substack{i \text{ opened in} \\ \text{scenario } A}} f_i^A$ + (cost of serving clients in A).

2-Stage Stochastic Facility Location



Distribution over clients gives the set of clients to serve.

Stage I: Open some facilities in advance; pay cost f_i for facility i.

Stage I cost = $\sum_{(i \text{ opened})} f_i$.

Actual scenario A = { ● clients to serve}, materializes.

Stage II: Can open more facilities to serve clients in A; pay cost f_i^A to open facility i. Assign clients in A to facilities.

Stage II cost = $\sum_{i \text{ opened in scenario } A} f_i^A$ + (cost of serving clients in A).

Want to decide which facilities to open in stage I.

Goal: Minimize Total Cost =

$$(\text{stage I cost}) + \mathbf{E}_{A \subseteq \mathcal{D}}[\text{stage II cost for } A].$$

We want to prove a worst-case guarantee.

Give an algorithm that "works well" on any instance, and for any probability distribution.

A is an α-approximation algorithm if -

- A runs in polynomial time;
- $A(I) \leq \alpha.OPT(I)$ on all instances I.

α is called the approximation ratio of A.

Goals of this Tutorial

- Focus on techniques of approximation algorithm design
  - LP-rounding
  - Primal-dual algorithms and analysis
  - Random sampling
- Five illustrative problems
  - Set cover problem
  - Facility location problem
  - Steiner tree problem
  - Traveling salesman problem
  - Maximum-weight on-time scheduling

Stochastic Set Cover (SSC)

Universe $U = \{e_1, \ldots, e_n\}$, subsets $S_1, S_2, \ldots, S_m \subseteq U$, set S has weight ω_S.

Deterministic problem: Pick a minimum weight collection of sets that covers each element.

Stochastic version: Set of elements to be covered is given by a probability distribution.

- choose some sets initially paying ω_S for set S
- subset $A \subseteq U$ to be covered is revealed
- can pick additional sets paying W_S for set S.

Minimize (ω-cost of sets picked in stage I) +
$\mathbf{E}_{A \subseteq U}$ [W_S -cost of new sets picked for scenario A].

An LP formulation

p_A : probability of scenario $A \subseteq U$.
x_S : indicates if set S is picked in stage I.
$y_{A,S}$: indicates if set S is picked in scenario A.

Minimize $\sum_S \omega_S x_S + \sum_{A \subseteq U} p_A \sum_S W_S y_{A,S}$
subject to,

$$\sum_{S:e \in S} x_S + \sum_{S:e \in S} y_{A,S} \geq 1 \qquad \text{for each } A \subseteq U,\ e \in A$$

$$x_S,\ y_{A,S} \geq 0 \qquad \text{for each } S, A.$$

Exponential number of variables and exponential number of constraints.

A Rounding Theorem (S & Swamy)

Stochastic Problem: LP can be solved in polynomial time.

Example: polynomial scenario setting

Deterministic problem: α-approximation algorithm A with respect to the LP relaxation, $\mathcal{A}(I) \leq \alpha \cdot \text{LP-OPT}(I)$ for each I.

Example: "the greedy algorithm" for set cover is a $\log n$-approximation algorithm w.r.t. LP relaxation.

Theorem: Can use such an α-approx. algorithm to get a 2α-approximation algorithm for stochastic set cover.

Rounding the LP

Assume LP can be solved in polynomial time.

Suppose we have an α-approximation algorithm wrt. the LP relaxation for the deterministic problem.

Let (x,y) : optimal solution with cost LP-OPT.

$$\sum_{S:e\in S} x_S + \sum_{S:e\in S} y_{A,S} \geq 1 \qquad \text{for each } A \subseteq U,\ e\in A$$

$\Rightarrow$ for every element e, either

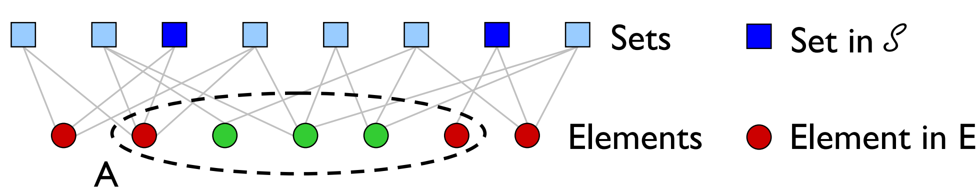
$\sum_{S:e\in S} x_S \geq ½$ OR in each scenario $A : e\in A$, $\sum_{S:e\in S} y_{A,S} \geq ½$.

Let $E = \{e : \sum_{S:e\in S} x_S \geq ½\}$.

So $(2x)$ is a fractional set cover for the set $E \Rightarrow$ can "round" to get an integer set cover $\mathcal{S}$ for E of cost $\sum_{S\in\mathcal{S}} \omega_S \leq \alpha(\sum_S 2\omega_S x_S)$.

$\mathcal{S}$ is the first stage decision.

Rounding (contd.)



Consider any scenario A. Elements in $A \cap E$ are covered.

For every $e \in A\backslash E$, it must be that $\sum_{S:e\in S} y_{A,S} \geq ½$.

So $(2y^A)$ is a fractional set cover for $A\backslash E \Rightarrow$ can round to get a set cover of W-cost $\leq \alpha(\sum_S 2W_S y_{A,S})$.

Using this to augment $\mathcal{S}$ in scenario A, expected cost

$$\leq \sum_{S\in\mathcal{S}} \omega_S + 2\alpha\cdot\sum_{A\subseteq U} p_A \left(\sum_S W_S y_{A,S}\right) \leq 2\alpha\cdot\text{LP-OPT}.$$

A Rounding Theorem

Stochastic Problem: LP can be solved in polynomial time.

Example: polynomial scenario setting

Deterministic problem: α-approximation algorithm A with respect to the LP relaxation, $\mathcal{A}(I) \leq \alpha \cdot \text{LP-OPT}(I)$ for each I.

Example: "the greedy algorithm" for set cover is a log n-approximation algorithm w.r.t. LP relaxation.

Theorem: Can use such an α-approx. algorithm to get a 2α-approximation algorithm for stochastic set cover.

A Rounding Technique

Assume LP can be solved in polynomial time.

Suppose we have an α-approximation algorithm w.r.t. the LP relaxation for the deterministic problem.

Let (x,y) : optimal solution with cost OPT.

$$\sum_{S:e\in S} x_S + \sum_{S:e\in S} y_{A,S} \geq 1 \qquad \text{for each } A \subseteq U,\ e\in A$$

$\Rightarrow$ for every element e, either

$\sum_{S:e\in S} x_S \geq ½$ OR in each scenario $A : e\in A$, $\sum_{S:e\in S} y_{A,S} \geq ½$.

Let $E = \{e : \sum_{S:e\in S} x_S \geq ½\}$.

So $(2x)$ is a fractional set cover for the set $E \Rightarrow$ can "round" to get an integer set cover $\mathcal{S}$ of cost $\sum_{S\in\mathcal{S}} \omega_S \leq \alpha(\sum_S 2\omega_S x_S)$.

$\mathcal{S}$ is the first stage decision.

A Compact Formulation

p_A : probability of scenario $A \subseteq U$.
x_S : indicates if set S is picked in stage I.

Minimize $h(x) = \sum_S \omega_S x_S + f(x)$ s.t. $x_S \geq 0$ for each S

where, $f(x) = \sum_{A \subseteq U} p_A f_A(x)$

and $f_A(x) = \min. \sum_S W_S y_{A,S}$

s.t. $\sum_{S:e \in S} y_{A,S} \geq 1 - \sum_{S:e \in S} x_S$ for each $e \in A$

$y_{A,S} \geq 0$ for each S.

Equivalent to earlier LP.

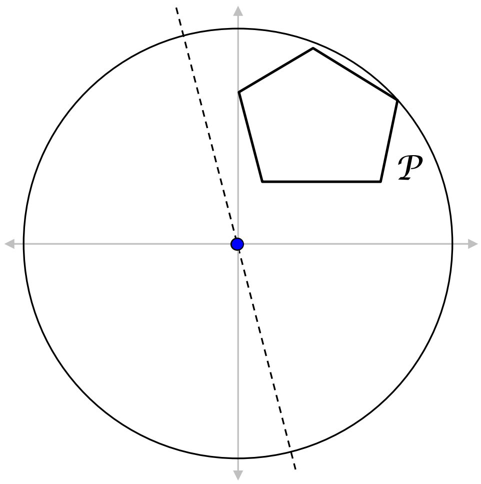
Each $f_A(x)$ is convex, so $f(x)$ and $h(x)$ are convex functions.

Solving the Stochastic LP?

- The LP has exponential number of variables and constraints, but can give other compact formulation as convex program that focuses on Stage I
- Can compute a fractional solution of cost at most $(1+\varepsilon)$LP-OPT with probability at least $1-\delta$ in time polynomial in input size and $\lambda = \max_S W_S/\omega_S$
- Many approaches are possible, including ellipsoid method and sample average approximation [S&Swamy, Nemirovski& Shapiro, Charikar, Chekuri, & Pál]

The Ellipsoid Method

Min c·x subject to x∈ $\mathcal{P}$.



Ellipsoid ≡ squashed sphere

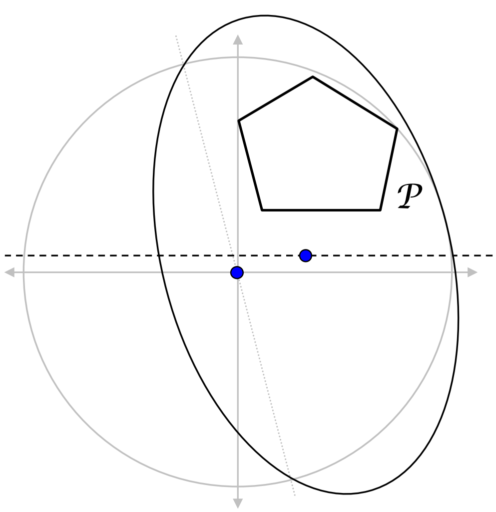
Start with ball containing polytope $\mathcal{P}$.

y_i = center of current ellipsoid.

If y_i is infeasible, use violated inequality to chop off infeasible half-ellipsoid.

The Ellipsoid Method

Min c·x subject to x∈ $\mathcal{P}$.



Ellipsoid ≡ squashed sphere

Start with ball containing polytope $\mathcal{P}$.

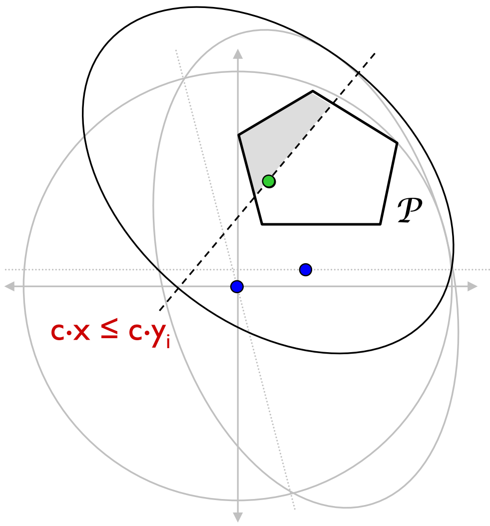
y_i = center of current ellipsoid.

If y_i is infeasible, use violated inequality to chop off infeasible half-ellipsoid.

New ellipsoid = min. volume ellipsoid containing “unchopped” half-ellipsoid.

The Ellipsoid Method

Min c·x subject to x∈ 𝒫.



Ellipsoid ≡ squashed sphere

Start with ball containing polytope 𝒫.

y_i = center of current ellipsoid.

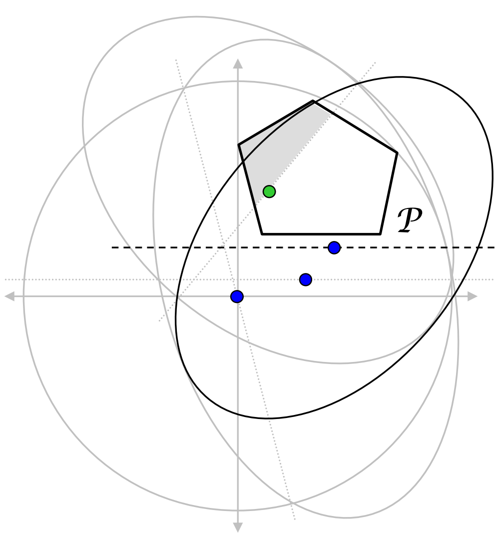
If y_i is infeasible, use violated inequality to chop off infeasible half-ellipsoid.

If $y_i \in$ 𝒫, use objective function cut $c \cdot x \le c \cdot y_i$ to chop off polytope, half-ellipsoid.

New ellipsoid = min. volume ellipsoid containing "unchopped" half-ellipsoid.

The Ellipsoid Method

Min c·x subject to x∈ 𝒫.



Ellipsoid ≡ squashed sphere

Start with ball containing polytope 𝒫.

y_i = center of current ellipsoid.

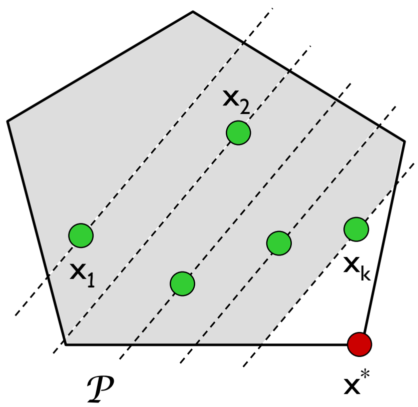
If y_i is infeasible, use violated inequality to chop off infeasible half-ellipsoid.

If $y_i \in$ 𝒫, use objective function cut $c \cdot x \le c \cdot y_i$ to chop off polytope, half-ellipsoid.

New ellipsoid = min. volume ellipsoid containing "unchopped" half-ellipsoid.

The Ellipsoid Method

Min $c \cdot x$ subject to $x \in \mathcal{P}$.



Ellipsoid ≡ squashed sphere

Start with ball containing polytope $\mathcal{P}$.

y_i = center of current ellipsoid.

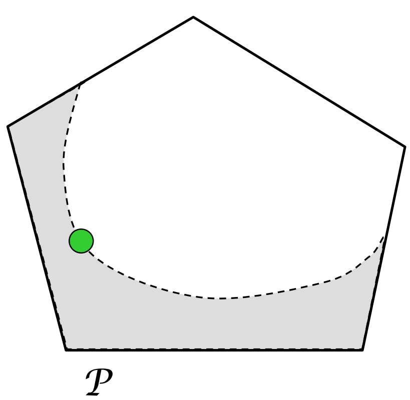
- If y_i is infeasible, use violated inequality to chop off infeasible half-ellipsoid.
- If $y_i \in \mathcal{P}$, use objective function cut $c \cdot x \leq c \cdot y_i$ to chop off polytope, half-ellipsoid.

New ellipsoid = min. volume ellipsoid containing "unchopped" half-ellipsoid.

$x_1, x_2, \ldots, x_k$: points lying in $\mathcal{P}$. $c \cdot x_k$ is a close to optimal value.

Ellipsoid for Convex Optimization

Min $h(x)$ subject to $x \in \mathcal{P}$.



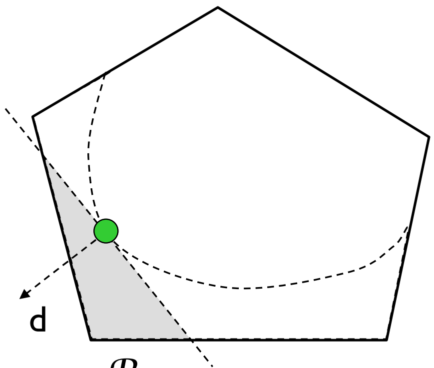
Start with ball containing polytope $\mathcal{P}$.

y_i = center of current ellipsoid.

- If y_i is infeasible, use violated inequality.
- If $y_i \in \mathcal{P}$ – how to make progress?
  add inequality $h(x) \leq h(y_i)$? Separation becomes difficult.

Ellipsoid for Convex Optimization

Min $h(x)$ subject to $x \in \mathcal{P}$.



$\mathcal{P}$

Start with ball containing polytope $\mathcal{P}$.
y_i = center of current ellipsoid.

If y_i is infeasible, use violated inequality.

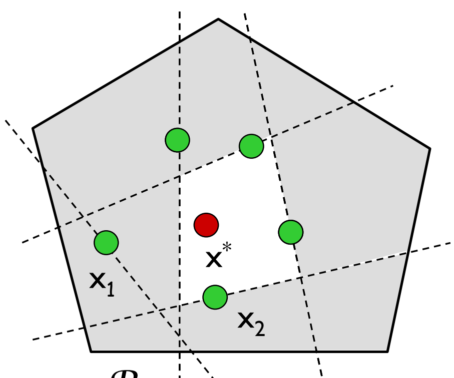
If $y_i \in \mathcal{P}$ – how to make progress?
add inequality $h(x) \leq h(y_i)$? Separation becomes difficult.
Let d = subgradient at y_i.
use subgradient cut $d \cdot (x - y_i) \leq 0$.

Generate new min. volume ellipsoid.

$d \in \Re^n$ is a subgradient of $h(.)$ at u, if for every v, $h(v)-h(u) \geq d \cdot (v-u)$.

Ellipsoid for Convex Optimization

Min $h(x)$ subject to $x \in \mathcal{P}$.



$\mathcal{P}$

Start with ball containing polytope $\mathcal{P}$.
y_i = center of current ellipsoid.

If y_i is infeasible, use violated inequality.

If $y_i \in \mathcal{P}$ – how to make progress?
add inequality $h(x) \leq h(y_i)$? Separation becomes difficult.
Let d = subgradient at y_i.
use subgradient cut $d \cdot (x - y_i) \leq 0$.

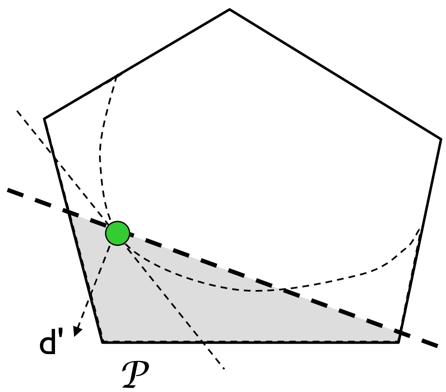
Generate new min. volume ellipsoid.

$d \in \Re^n$ is a subgradient of $h(.)$ at u, if for every v, $h(v)-h(u) \geq d \cdot (v-u)$.

$x_1, x_2, \ldots, x_k$: points in $\mathcal{P}$. Can show, $\min_{i=1\ldots k} h(x_i) \leq OPT+\rho$.

Ellipsoid for Convex Optimization

Min $h(x)$ subject to $x \in \mathcal{P}$.



Start with ball containing polytope $\mathcal{P}$.
y_i = center of current ellipsoid.

If y_i is infeasible, use violated inequality.

If $y_i \in \mathcal{P}$ – how to make progress?
add inequality $h(x) \le h(y_i)$? Separation becomes difficult.
subgradient is difficult to compute.
Let d' = ε-subgradient at y_i.
use ε-subgradient cut $d' \cdot (x - y_i) \le 0$.

$d' \in \Re^n$ is a ε-subgradient of $h(.)$ at u, if $\forall v \in \mathcal{P}$, $h(v)-h(u) \ge d' \cdot (v-u) - \varepsilon \cdot h(u)$.

$x_1, x_2, \ldots, x_k$: points in $\mathcal{P}$. Can show, $\min_{i=1\ldots k} h(x_i) \le OPT/(1-\varepsilon) + \rho$.

Subgradients and ε-subgradients

Vector d is a subgradient of $h(.)$ at u,
if for every v, $h(v) - h(u) \ge d \cdot (v-u)$.

Vector d' is an ε-subgradient of $h(.)$ at u,
if for every $v \in \mathcal{P}$, $h(v) - h(u) \ge d' \cdot (v-u) - \varepsilon \cdot h(u)$.

$\mathcal{P} = \{ x : 0 \le x_S \le 1$ for each set $S \}$.

$h(x) = \sum_S \omega_S x_S + \sum_{A \subseteq U} p_A f_A(x) = \omega \cdot x + \sum_{A \subseteq U} p_A f_A(x)$

Lemma: Let d be a subgradient at u, and d' be a vector such that $d_S - \varepsilon\omega_S \le d'_S \le d_S$ for each set S. Then, d' is an ε-subgradient at point u.

Getting a “nice” subgradient

$h(x) = \omega \cdot x + \sum_{A \subseteq U} p_A f_A(x)$

$f_A(x) =$ min. $\sum_S W_S y_{A,S}$

s.t. $\sum_{S: e \in S} y_{A,S} \geq 1 - \sum_{S: e \in S} x_S \quad \forall e \in A$

$y_{A,S} \geq 0 \quad \forall S$

Getting a “nice” subgradient

$h(x) = \omega \cdot x + \sum_{A \subseteq U} p_A f_A(x)$

$f_A(x) =$ min. $\sum_S W_S y_{A,S}$ = max. $\sum_{e \in A} (1 - \sum_{S: e \in S} x_S) z_{A,e}$

s.t. $\sum_{S: e \in S} y_{A,S} \geq 1 - \sum_{S: e \in S} x_S \quad \forall e \in A$ s.t. $\sum_{e \in A \cap S} z_{A,e} \leq W_S \quad \forall S$

$y_{A,S} \geq 0 \quad \forall S$ $z_{A,e} \geq 0 \quad \forall e \in A$

Getting a "nice" subgradient

$h(x) = \omega \cdot x + \sum_{A \subseteq U} p_A f_A(x)$

$f_A(x) = \min. \sum_S W_S y_{A,S} \quad = \quad \max. \sum_e (1 - \sum_{S: e \in S} x_S) z_{A,e}$

s.t. $\sum_{S: e \in S} y_{A,S} \geq 1 - \sum_{S: e \in S} x_S \quad \forall e \in A$ s.t. $\sum_{e \in S} z_{A,e} \leq W_S \quad \forall S$

$y_{A,S} \geq 0 \quad \forall S \qquad z_{A,e} = 0 \quad \forall e \notin A, \quad z_{A,e} \geq 0 \quad \forall e$

Consider point $u \in \Re^n$. Let $z_A \equiv$ optimal dual solution for A at u.

Lemma: For any point $v \in \Re^n$, we have $h(v) - h(u) \geq d \cdot (v-u)$ where $d_S = \omega_S - \sum_{A \subseteq U} p_A \sum_{e \in S} z_{A,e}$.

$\Rightarrow$ d is a subgradient of h(.) at point u.

Getting a "nice" subgradient

$h(x) = \omega \cdot x + \sum_{A \subseteq U} p_A f_A(x)$

$f_A(x) = \min. \sum_S W_S y_{A,S} \quad = \quad \max. \sum_e (1 - \sum_{S: e \in S} x_S) z_{A,e}$

s.t. $\sum_{S: e \in S} y_{A,S} \geq 1 - \sum_{S: e \in S} x_S \quad \forall e \in A$ s.t. $\sum_{e \in S} z_{A,e} \leq W_S \quad \forall S$

$y_{A,S} \geq 0 \quad \forall S \qquad z_{A,e} = 0 \quad \forall e \notin A, \quad z_{A,e} \geq 0 \quad \forall e$

Consider point $u \in \Re^n$. Let $z_A \equiv$ optimal dual solution for A at u. So

$$f_A(u) = \sum_e (1 - \sum_{S: e \in S} u_S) z_{A,e}.$$

For any other point v, z_A is a feasible dual solution for A. So

$$f_A(v) \geq \sum_e (1 - \sum_{S: e \in S} v_S) z_{A,e}.$$

Get that $h(v) - h(u) \geq \sum_S (\omega_S - \sum_{A \subseteq U} p_A \sum_{e \in S} z_{A,e})(v_S - u_S) = d \cdot (v-u)$ where $d_S = \omega_S - \sum_{A \subseteq U} p_A \sum_{e \in S} z_{A,e}$. So d is a subgradient of h(.) at point u.

Computing an ε-Subgradient

Given point $u \in \Re^n$. $z_A \equiv$ optimal dual solution for A at u.

Subgradient at u: $d_S = \omega_S - \sum_{A \subseteq U} p_A \sum_{e \in S} z_{A,e}$.

Want: d' such that $d_S - \varepsilon\omega_S \le d'_S \le d_S$ for each S.

For each S, $-W_S \le d_S \le \omega_S$. Let $\lambda = \max_S W_S / \omega_S$.

Sample once from black box to get random scenario A.

Compute X with $X_S = \omega_S - \sum_{e \in S} z_{A,e}$.

$\mathbf{E}[X_S] = d_S$ and $\mathbf{Var}[X_S] \le W_S^2$.

Sample $O(\lambda^2/\varepsilon^2 \cdot \log(n/\delta))$ times to compute d' such that

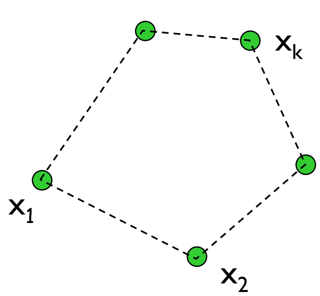
$\mathbf{Pr}[\forall S,\ d_S - \varepsilon\omega_S \le d'_S \le d_S] \ge 1\text{-}\delta$.

$\Rightarrow$ d' is an ε-subgradient at u with probability $\ge 1\text{-}\delta$.

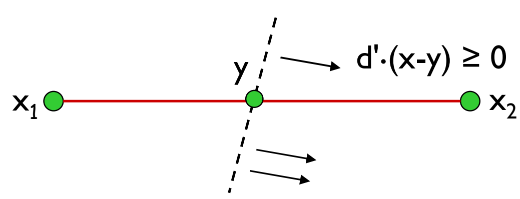
One last hurdle

Cannot evaluate $h(.)$ – how to compute $\bar{x} = \text{argmin}_{i=1\ldots k}\ h(x_i)$?

Will find point $\bar{x}$ in the convex hull of $x_1, \ldots, x_k$ such that $h(\bar{x})$ is close to $\min_{i=1\ldots k} h(x_i)$.



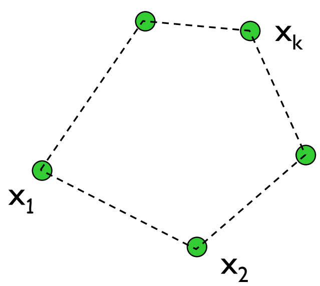
Take two points x_1 and x_2. Find point $\bar{x}$ on x_1–x_2 line segment with value close to $\min(h(x_1), h(x_2))$ using bisection search.



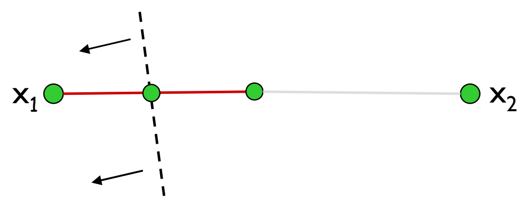
One last hurdle

Cannot evaluate $h(.)$ – how to compute $\bar{x} = \text{argmin}_{i=1\ldots k}\ h(x_i)$?

Will find point $\bar{x}$ in the convex hull of $x_1, \ldots, x_k$ such that $h(\bar{x})$ is close to $\min_{i=1\ldots k}\ h(x_i)$.



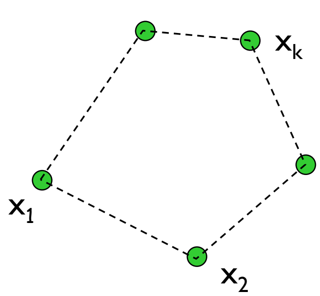
Take two points x_1 and x_2. Find point $\bar{x}$ on x_1–x_2 line segment with value close to $\min(h(x_1), h(x_2))$ using bisection search.



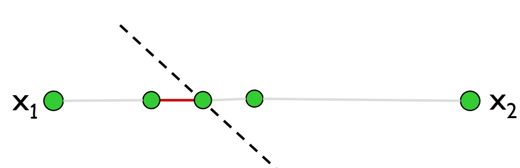
One last hurdle

Cannot evaluate $h(.)$ – how to compute $\bar{x} = \text{argmin}_{i=1\ldots k}\ h(x_i)$?

Will find point $\bar{x}$ in the convex hull of $x_1, \ldots, x_k$ such that $h(\bar{x})$ is close to $\min_{i=1\ldots k}\ h(x_i)$.



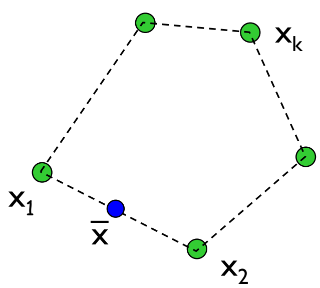
Take two points x_1 and x_2. Find point $\bar{x}$ on x_1–x_2 line segment with value close to $\min(h(x_1), h(x_2))$ using bisection search.



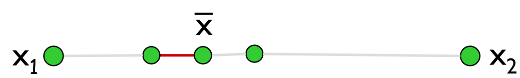
One last hurdle

Cannot evaluate $h(.)$ – how to compute $\bar{x} = \text{argmin}_{i=1\ldots k}\, h(x_i)$?

Will find point $\bar{x}$ in the convex hull of $x_1, \ldots, x_k$ such that $h(\bar{x})$ is close to $\min_{i=1\ldots k} h(x_i)$.



Take two points x_1 and x_2. Find point $\bar{x}$ on x_1–x_2 line segment with value close to $\min(h(x_1), h(x_2))$ using bisection search.

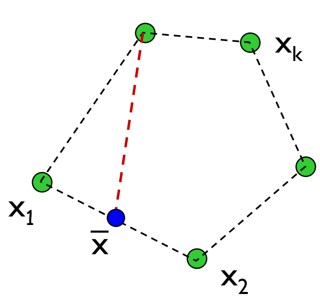


Stop when search interval is small enough.
Set $\bar{x}$ = either end point of remaining segment.

One last hurdle

Cannot evaluate $h(.)$ – how to compute $\bar{x} = \text{argmin}_{i=1\ldots k}\, h(x_i)$?

Will find point $\bar{x}$ in the convex hull of $x_1, \ldots, x_k$ such that $h(\bar{x})$ is close to $\min_{i=1\ldots k} h(x_i)$.



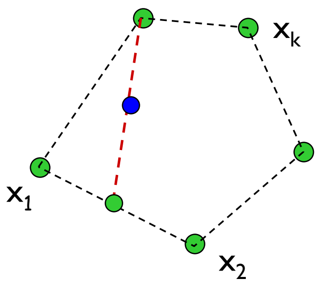
Take two points x_1 and x_2. Find point $\bar{x}$ on x_1–x_2 line segment with value close to $\min(h(x_1), h(x_2))$ using bisection search.

Iterate using $\bar{x}$ and $x_3, \ldots, x_k$ updating $\bar{x}$ along the way.

One last hurdle

Cannot evaluate $h(.)$ – how to compute $\bar{x} = \text{argmin}_{i=1\ldots k}\, h(x_i)$?

Will find point $\bar{x}$ in the convex hull of $x_1,\ldots, x_k$ such that $h(\bar{x})$ is close to $\min_{i=1\ldots k} h(x_i)$.



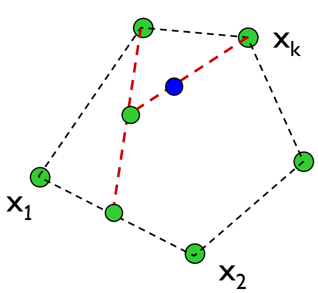
Take two points x_1 and x_2. Find point $\bar{x}$ on x_1–x_2 line segment with value close to $\min(h(x_1), h(x_2))$ using bisection search.

Iterate using $\bar{x}$ and $x_3,\ldots, x_k$ updating $\bar{x}$ along the way.

One last hurdle

Cannot evaluate $h(.)$ – how to compute $\bar{x} = \text{argmin}_{i=1\ldots k}\, h(x_i)$?

Will find point $\bar{x}$ in the convex hull of $x_1,\ldots, x_k$ such that $h(\bar{x})$ is close to $\min_{i=1\ldots k} h(x_i)$.



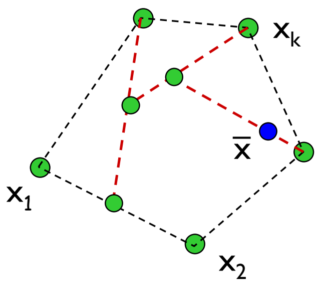
Take two points x_1 and x_2. Find point $\bar{x}$ on x_1–x_2 line segment with value close to $\min(h(x_1), h(x_2))$ using bisection search.

Iterate using $\bar{x}$ and $x_3,\ldots, x_k$ updating $\bar{x}$ along the way.

One last hurdle

Cannot evaluate $h(.)$ – how to compute $\bar{x} = \text{argmin}_{i=1\ldots k}\, h(x_i)$?

Will find point $\bar{x}$ in the convex hull of $x_1,\ldots, x_k$ such that $h(\bar{x})$ is close to $\min_{i=1\ldots k} h(x_i)$.



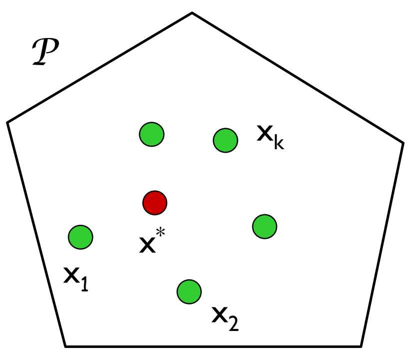
Take two points x_1 and x_2. Find point $\bar{x}$ on x_1–x_2 line segment with value close to $\min(h(x_1), h(x_2))$ using bisection search.

Iterate using $\bar{x}$ and $x_3,\ldots, x_k$ updating $\bar{x}$ along the way.

Can show that $h(\bar{x}) \le (\min_{i=1\ldots k} h(x_i) + k\cdot\rho)/(1-\varepsilon)^{kN}$.

Putting it all together

Min $h(x)$ subject to $x \in \mathcal{P}$.



✓ Can compute ε-subgradients.

Run ellipsoid algorithm.

Given y_i = center of current ellipsoid.

If y_i is infeasible, use violated inequality as a cut.

If $y_i \in \mathcal{P}$ use ε-subgradient cut.

Continue with smaller ellipsoid.

Generate points $x_1, x_2, \ldots, x_k$ in $\mathcal{P}$. Return $\bar{x} = \text{argmin}_{i=1\ldots k}\, h(x_i)$.

Get that $h(\bar{x}) \le \text{OPT}/(1-\varepsilon) + \rho$.

Finally,

Get solution x with $h(x)$ close to OPT.

Sample initially to detect if $\text{OPT} = \Omega(1/\lambda)$ – this allows one to get a $(1+\varepsilon)\cdot\text{OPT}$ guarantee.

Theorem: Compact convex program can be solved to within a factor of $(1+\varepsilon)$ in polynomial time, with high probability.

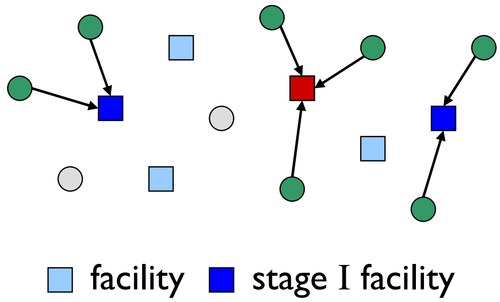
Gives a $(2\log n+\varepsilon)$-approximation algorithm for the stochastic set cover problem.

A Solvable Class of Stochastic LPs

$$
\begin{aligned}
\text{Minimize } h(x) &= w.x + \textstyle\sum_{A\subseteq U} p_A f_A(x) \\
\text{s.t.} \quad & x \in \Re^n,\ x \geq 0,\ x \in \mathcal{P} \\
\text{where} \quad f_A(x) &= \min.\ w^A.y_A + c^A.r_A \\
\text{s.t.} \quad & Br_A \geq j^A \\
& Dr_A + Ty_A \geq \ell^A - Tx \\
& y_A \in \Re^n,\ r_A \in \Re^m,\ y_A \geq 0,\ r_A \geq 0.
\end{aligned}
$$

Theorem: Can get a $(1+\varepsilon)$-optimal solution for this class of stochastic programs in polynomial time.

2-Stage Stochastic Facility Location



Distribution over clients gives the set of clients to serve.

Stage I: Open some facilities in advance; pay cost f_i for facility i.

Stage I cost = $\sum_{(i \text{ opened})} f_i$.

Actual scenario A = { ● clients to serve}, materializes.

Stage II: Can open more facilities to serve clients in A; pay cost f_i^A to open facility i. Assign clients in A to facilities.

Stage II cost = $\sum_{i \text{ opened in scenario } A} f_i^A$ + (cost of serving clients in A).

A Convex Program

p_A : probability of scenario $A \subseteq \mathcal{D}$.

y_i : indicates if facility i is opened in stage I.

$y_{A,i}$: indicates if facility i is opened in scenario A.

$x_{A,ij}$: whether client j is assigned to facility i in scenario A.

Minimize $h(y) = \sum_i f_i y_i + g(y)$ s.t. $y_i \geq 0$ for each i (SUFL-P)

where, $g(y) = \sum_{A \subseteq \mathcal{D}} p_A g_A(y)$

and $g_A(y) = \min. \sum_i F_i y_{A,i} + \sum_{j,i} c_{ij} x_{A,ij}$

$$\begin{aligned} \text{s.t.} \quad \sum_i x_{A,ij} &\geq 1 && \text{for each } j \in A \\ x_{A,ij} &\leq y_i + y_{A,i} && \text{for each } i,j \\ x_{A,ij}, y_{A,i} &\geq 0 && \text{for each } i,j. \end{aligned}$$

Lecture #2

A priori optimization (no recourse)

Given: Probability distribution over inputs.

In advance: Compute master plan.

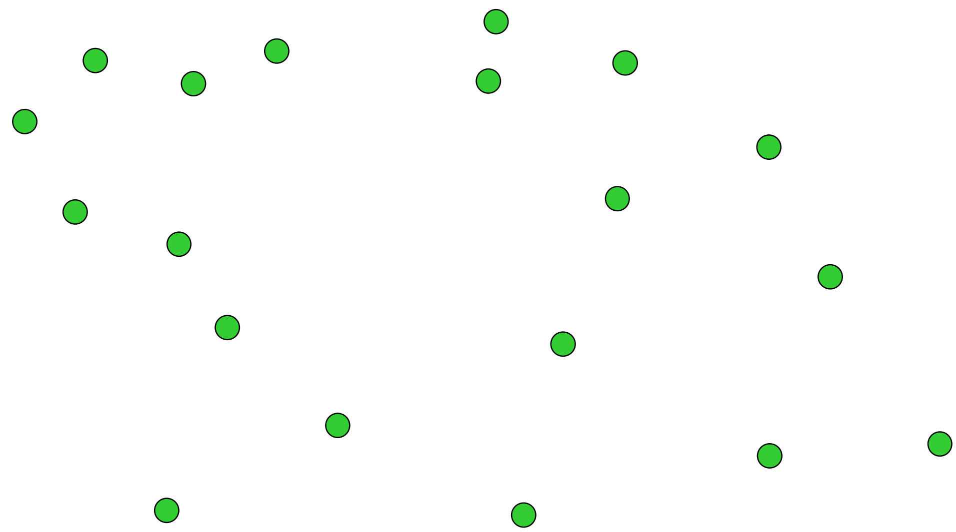
Observe the actual input scenario.

In real time: Adapt master plan to scenario.

Compute master plan to minimize

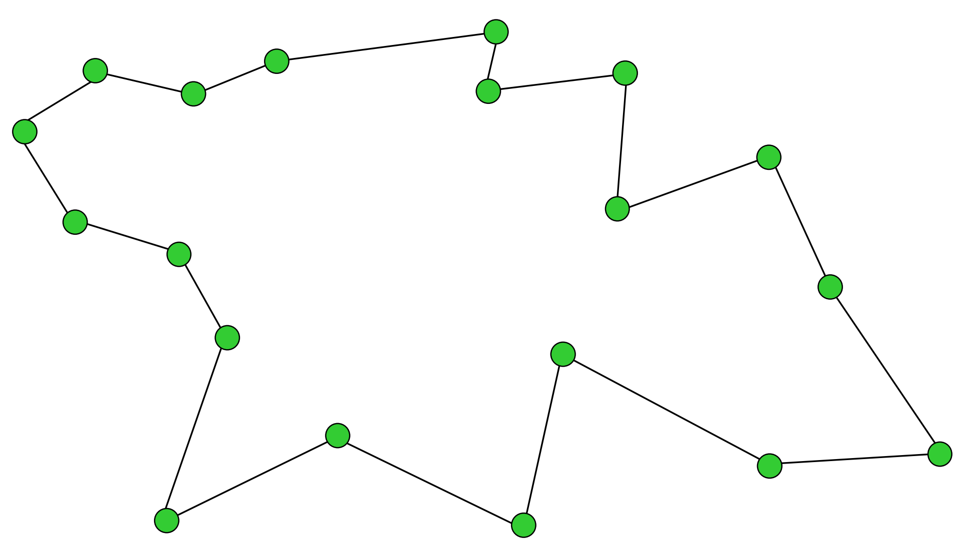
expected real time cost.

The Traveling Salesman Problem (TSP)



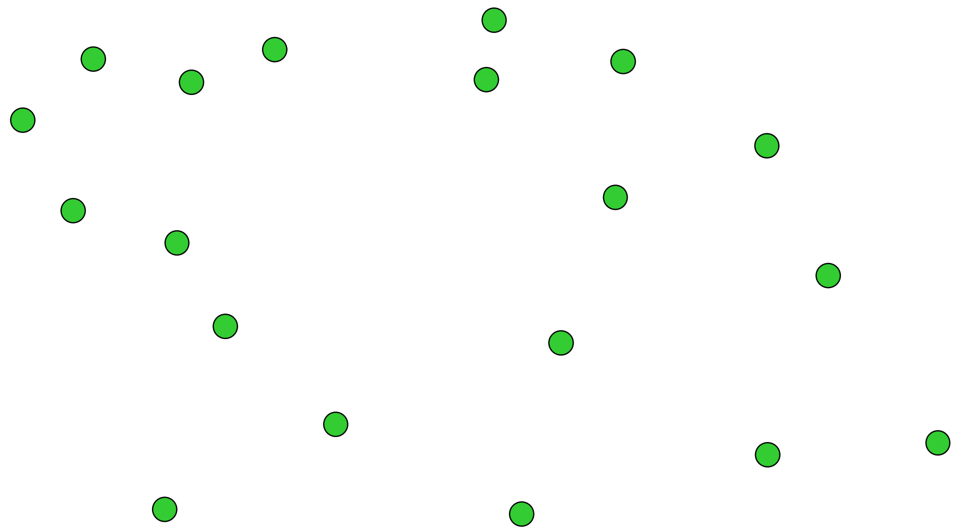
Given input points, compute tour τ to minimize total length $c(\tau)$

The Traveling Salesman Problem (TSP)



Given input points, compute tour τ to minimize total length $c(\tau)$

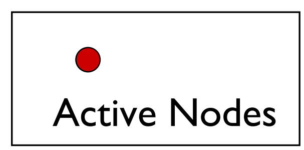
The *A Priori* TSP

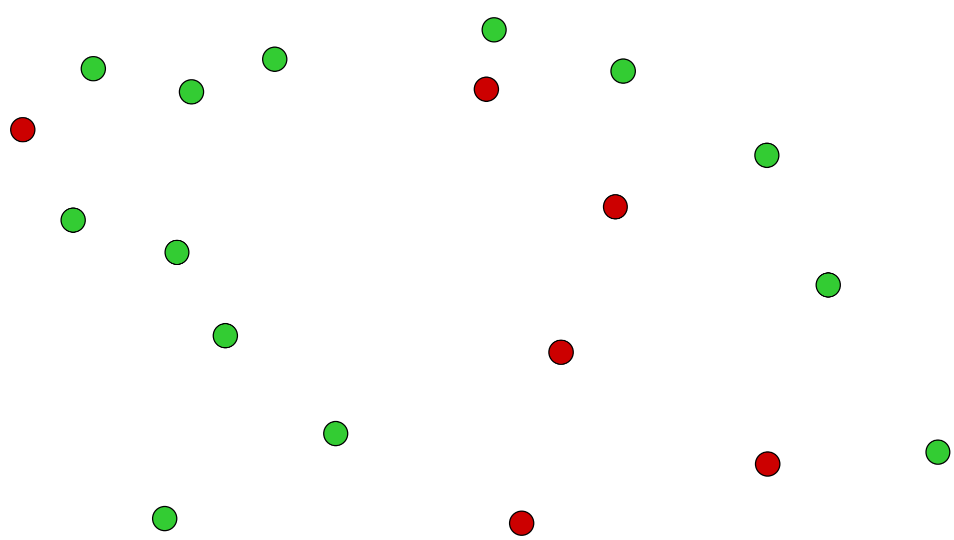


Given input points N and a distribution Π of active sets A 2 2^N

Need to specify the probability that a given set A is active

The *A Priori* TSP

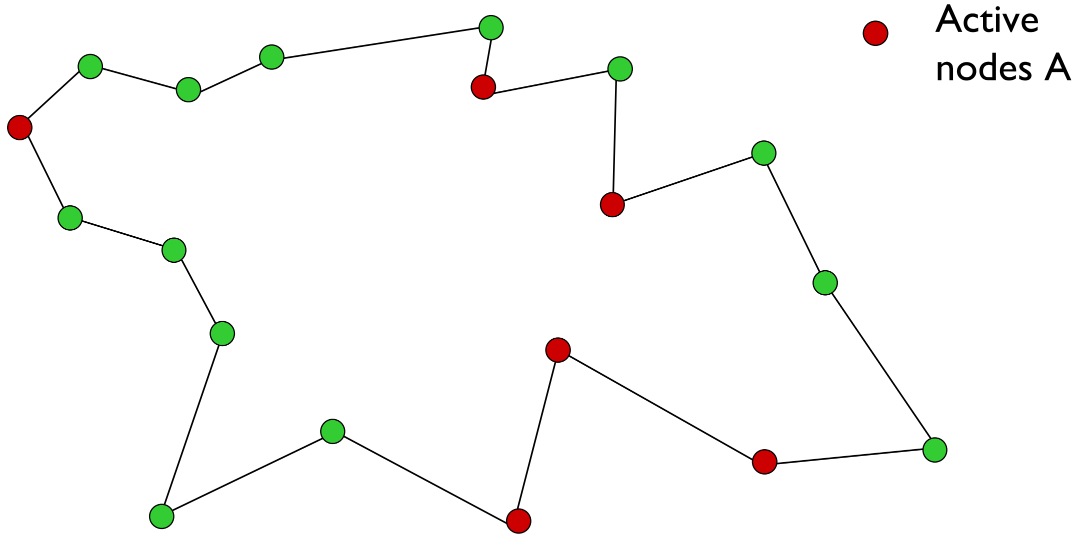




Given input points N and a distribution Π of active sets A 2 2^N

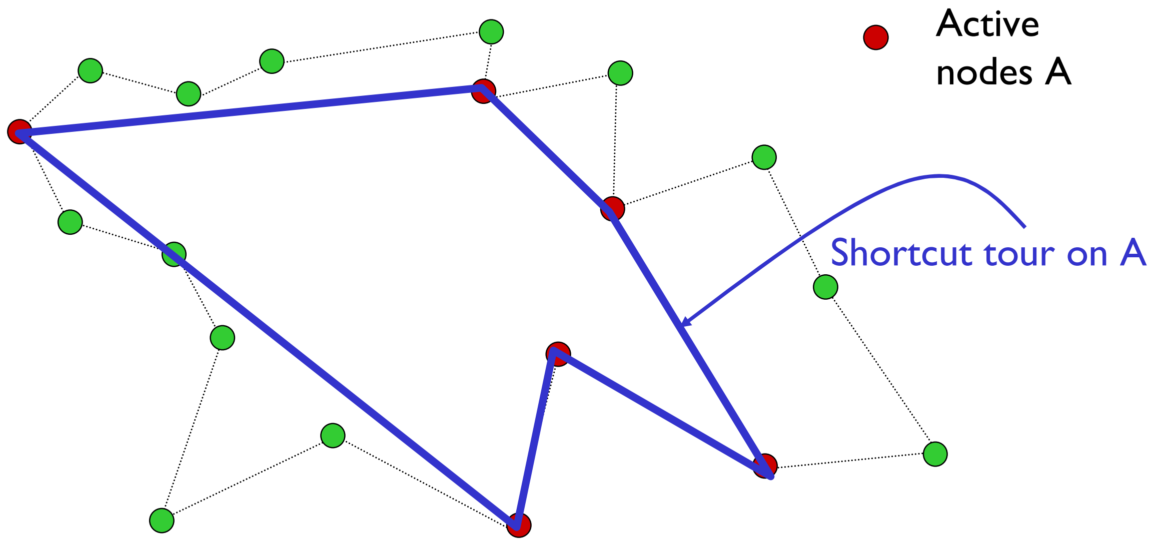
Need to specify the probability that a given set A is active

The *A Priori* TSP



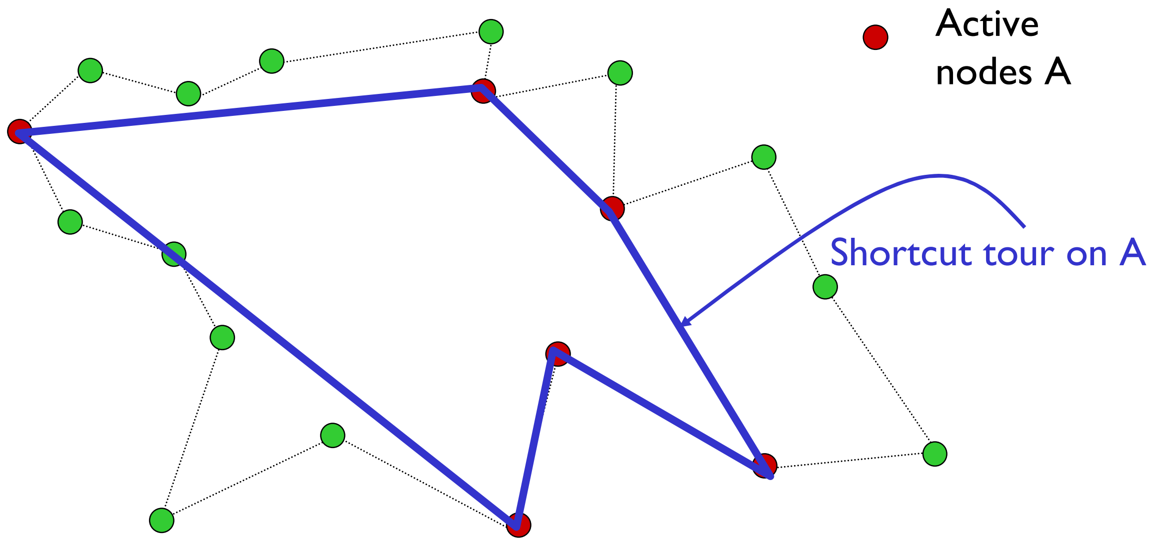
Given input points N and a distribution Π of active sets A 2 2^N, compute master tour τ to minimize expected length of the tour τ shortcut to serve only A

The *A Priori* TSP



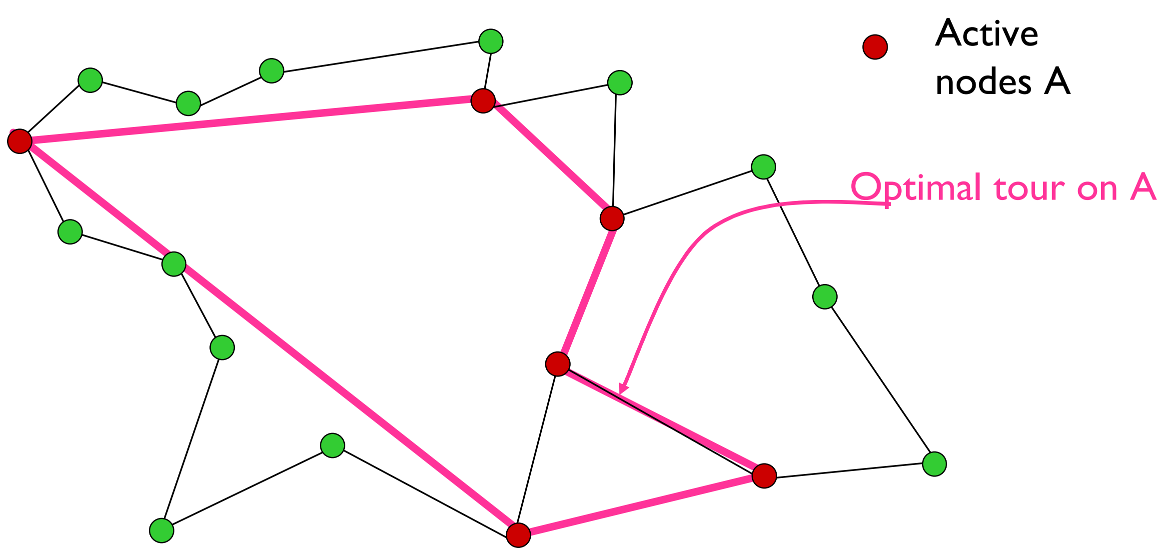
Given input points N and a distribution Π of active sets A 2 2^N, compute master tour τ to minimize expected length of the tour τ shortcut to serve only A

The *A Priori* TSP



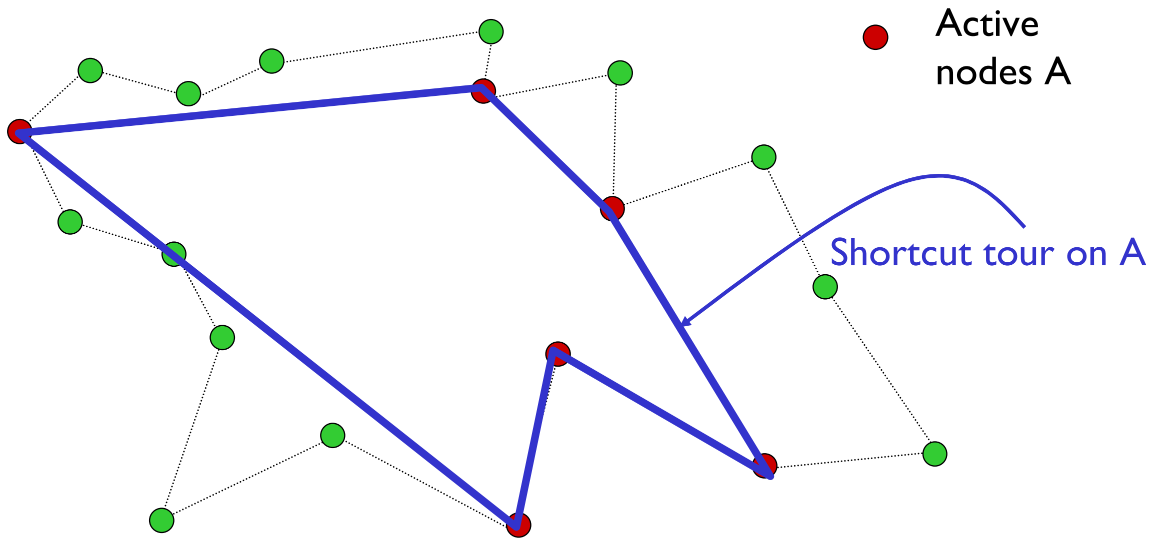
Given input points N and a distribution Π of active sets A 2 2^N, compute tour τ to minimize expected length E_A [c(τ_A)], where τ_A is the tour τ shortcut to serve only A

The *A Priori* TSP



Given input points N and a distribution Π of active sets A 2 2^N, compute tour τ to minimize expected length E_A [c(τ_A)], where τ_A is the tour τ shortcut to serve only A

The *A Priori* TSP



Given input points N and a distribution Π of active sets A 2 2^N, compute tour τ to minimize expected length E_A [c(τ_A)], where τ_A is the tour τ shortcut to serve only A

The *A Priori* TSP

Given input points N and a distribution Π of active sets A 2 2^N, compute tour ¿ to minimize expected length E_A [c(τ_A)], where τ_A is the tour τ shortcut to serve only A $\Rightarrow \tau^*$ (optimal solution)

Goal: Find tour τ such that E_A [c(τ_A)] $\leq$ ®E_A [c(τ^*_A)] $\Rightarrow$ ®OPT

(This is an ®-approximation algorithm for the *a priori* TSP.)

How is the probability distribution on active set specified?

- A short (polynomial) list of possibile scenarios;
- Independent probabilities that each point is active;
- A black box that can be sampled.

The *A Priori* TSP

Given input points N and a distribution Π of active sets A 2 2^N, compute tour ¿ to minimize expected length $E_A [c(\tau_A)]$, where τ_A is the tour τ shortcut to serve only A $\Rightarrow \tau^*$ (optimal solution)

Goal: Find tour τ such that $E_A [c(\tau_A)] \leq$ ®$E_A [c(\tau^*_A)] \Rightarrow$ ®OPT

(This is an ®-approximation algorithm for the *a priori* TSP.)

How is the probability distribution on active set specified?

- A short (polynomial) list of possibile scenarios;
- Independent probabilities that each point is active;
- A black box that can be sampled.

Some relevant history for *a priori* TSP

- Jaillet (1985, 1988), Bertsimas (1988), Jaillet, Bertsimas, & Odoni (1990) introduce problem – analyze with probabilistic assumptions on distances
- Schalekamp & S (2007) randomized O(log n)-approximation
- Maybecast problem Karger & Minkoff (2000)
- Rent-or-buy problem Gupta, Kumar, Pál, Roughgarden (2007)
- Stochastic Steiner Tree variants Gupta, Pál, Ravi, Sinha (2004)
  Gupta, Ravi, Sinha ('04), Hayraptian, Swamy, Tardos ('05)
  Garg, Gupta, Leonardi, Sankowski (2008)
- Universal TSP Bartholdi & Plazman (1989), Jia, Lin, Noubir, Rajaraman & Sundaram, (2005), Hajiaghayi, Kleinberg & Leighton (2006), Gupta, Hajiaghayi, Räcke (2006)

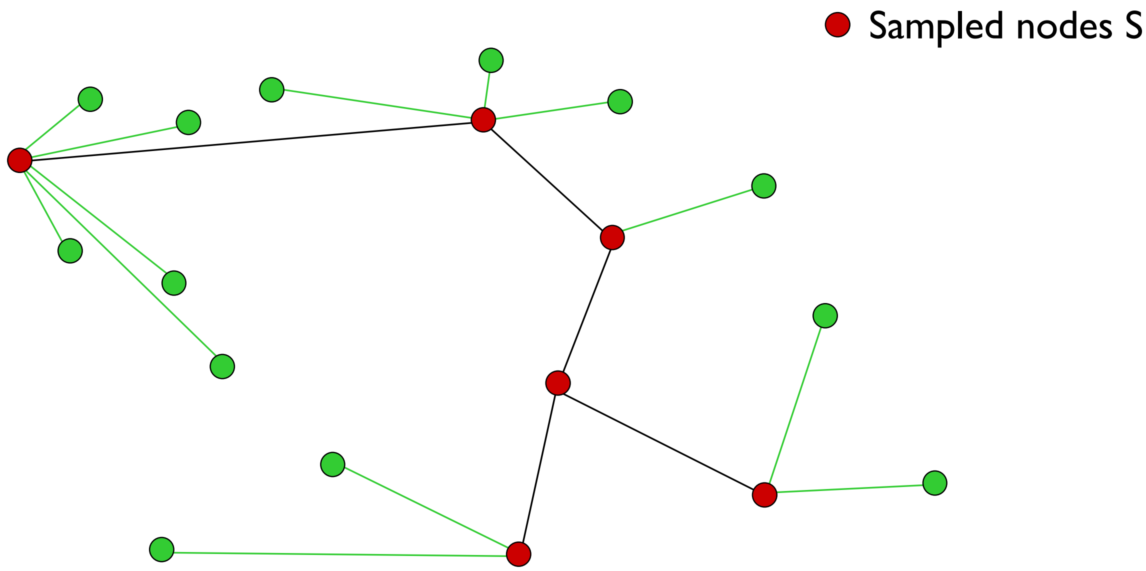
The One Random Sample Algorithm

1. Draw sample $S \subseteq N$ according to Π (i.e., pick each point j independently with probability p_j)
2. Build minimum spanning tree on S
3. For each $j \notin S$, connect j to its nearest neighbor in S
4. Build "double tree" tour of this tree $\Rightarrow \tau$

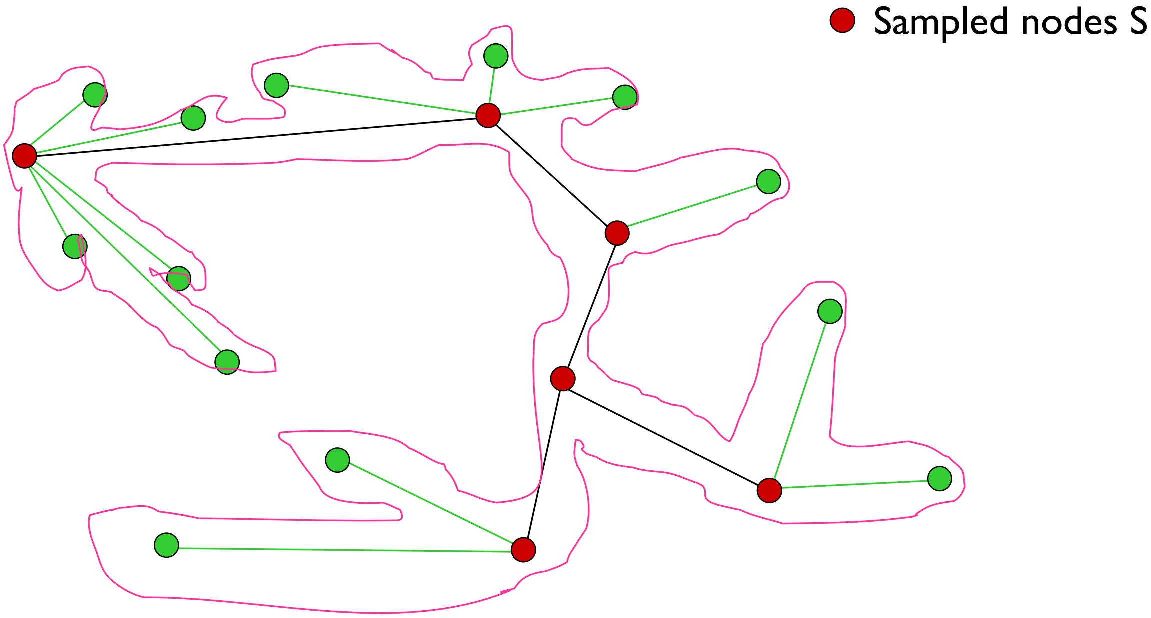
Simplifying Assumption: $\exists$ node r with $p_r = 1$ (wlog)

Theorem The one random sample algorithm is a 4-approximation algorithm for the *a priori* TSP.

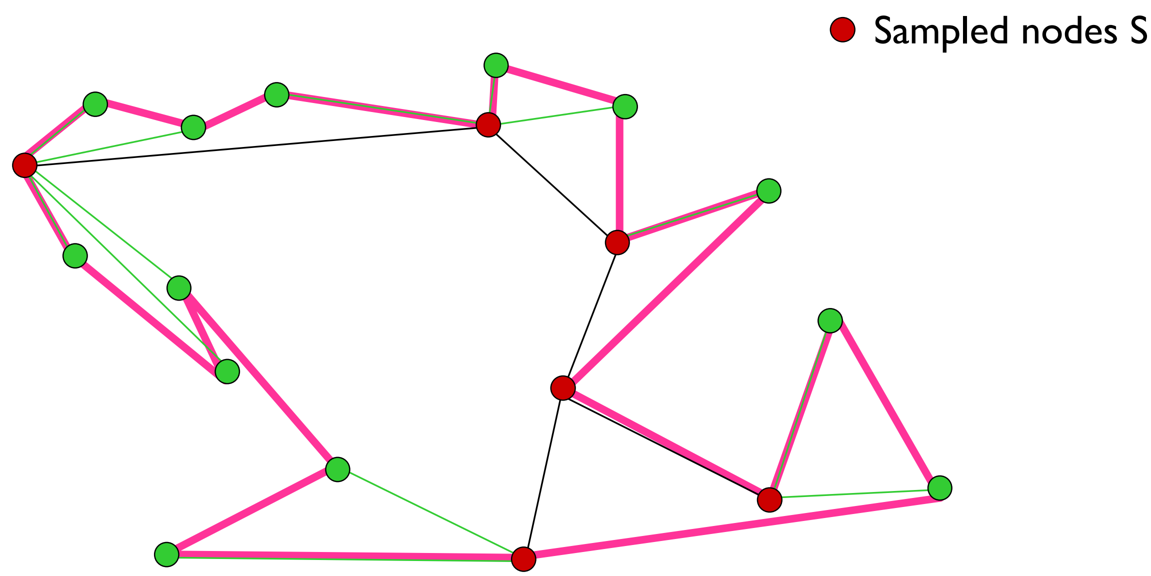
Running the Algorithm



Running the Algorithm



Running the Algorithm



Analyzing the Algorithm

Let $D_j(S)$ be the distance from j to its nearest neighbor in S-{j}

Let MST(S) be the length of the minimum spanning tree on S

Goal: Analyze $E_S [E_A [c(\tau_A)]]$

Fact 1. $E_S [D_j(S)] = E_S [D_j | j \notin S] = E_S [D_j | j \in S] = E_A [D_j(A) | j \in A]$

Why? Choice of S-{j} is independent of whether j 2 S, and
S and A are independent draws from same distribution

Fact 2. MST(A) ≤ c(¿ $^*_A$) for each A µ N

Why? Tour ¿* shortcut to A still contains spanning tree

Fact 3. $\sum_{j \neq r} \mathbb{1}(j \in A) D_j(A) \leq c(\tau^*_A)$ for all A

Why? Any tour on A "leaves" each node i by some edge

Let $D_j(S)$ be the distance from j to its nearest neighbor in S-{j}

Let MST(S) be the length of the minimum spanning tree on S

Goal: analyze $E_S [E_A [c(\tau_A)]]$

Fact 1. $E_S [D_j(S)] = E_S [D_j | j \notin S] = E_S [D_j | j \in S] = E_A [D_j(A) | j \in A]$

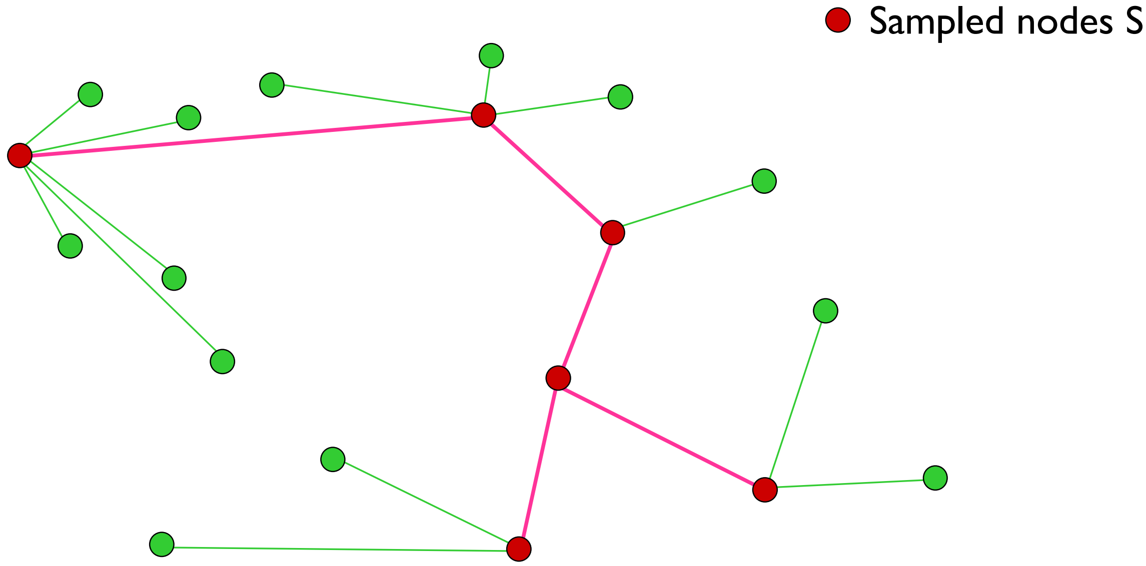
Fact 2. $MST(A) \leq c(\tau^*_A)$ for all A

Fact 3. $\sum_{j \neq r} \mathbb{1}(j \in A) D_j(A) \leq c(\tau^*_A)$ for all A

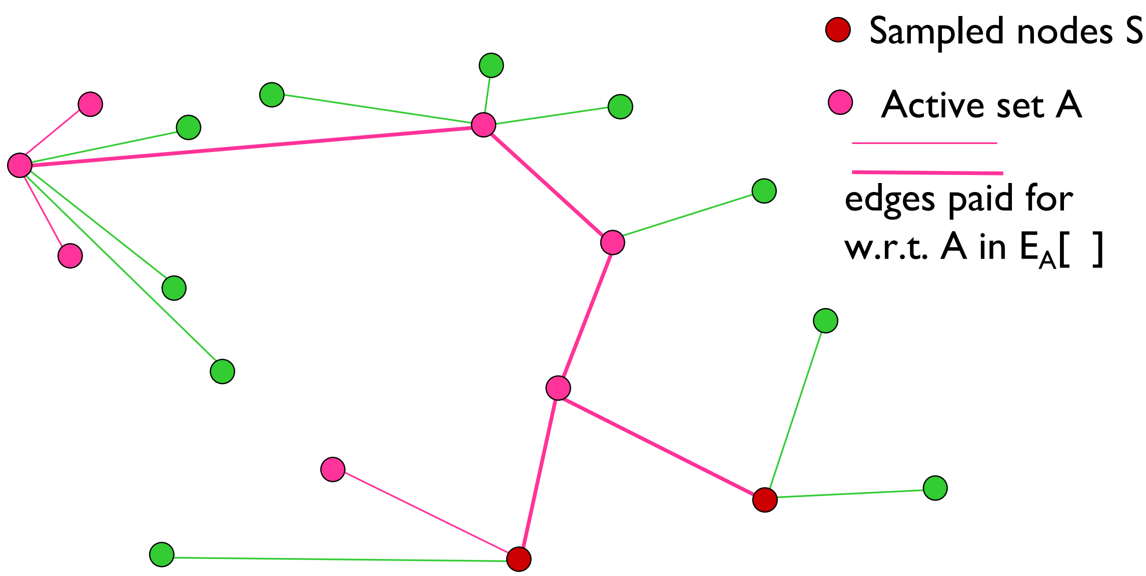
Key Idea: always pay for backbone built on S (for any active A)

$$
\begin{aligned}
E_S[E_A[c(\tau_A)]] &\leq E_S[2MST(S)]+E_S [E_A [\textstyle\sum_{j \neq r} \mathbb{1}(j \in A) \mathbb{1}(j \notin S) 2D_j(S)]] \\
&= E_S [2MST(S)] + \textstyle\sum_{j\neq r} E_{S,A} [\mathbb{1}(j \in A)\mathbb{1}(j \notin S) 2D_j (S)] \\
&= E_S [2MST(S)] + 2\textstyle\sum_{j \neq r} p_j (1-p_j) E_S[D_j(S)] \\
&\leq 2(E_S [MST(S)] + \textstyle\sum_{j \neq r} p_j E_S[D_j(S)]) \\
&\leq 2 (OPT+OPT) \\
&= 4OPT
\end{aligned}
$$

Analyzing the Algorithm



Analyzing the Algorithm



Always pay for all of backbone and just those attached leaves you need

Cost of shortcut tour for A is at most twice the cost of these edges

Let $D_j(S)$ be the distance from j to its nearest neighbor in S-{j}
Let MST(S) be the length of the minimum spanning tree on S

Goal: analyze $E_S [E_A [c(\tau_A)]]$

Fact 1. $E_S [D_j(S)] = E_S [D_j | j \notin S] = E_S [D_j | j \in S] = E_A [D_j(A) | j \in A]$

Fact 2. $MST(A) \leq c(\tau^*_A)$ for all A

Fact 3. $\sum_{j \neq r} \mathbb{1}(j \in A) D_j(A) \leq c(\tau^*_A)$ for all A

Key Idea: always pay for backbone built on S (for any active A)

$$
\begin{aligned}
E_S[E_A[c(\tau_A)]] &\leq E_S[2MST(S)]+E_S [E_A [\textstyle\sum_{j \neq r} \mathbb{1}(j \in A) \mathbb{1}(j \notin S) 2D_j(S)]] \\
&= E_S [2MST(S)] + \textstyle\sum_{j \neq r} E_{S,A} [\mathbb{1}(j \in A)\mathbb{1}(j \notin S) 2D_j (S)] \\
&= E_S [2MST(S)] + 2\textstyle\sum_{j \neq r} p_j (1-p_j) E_S[D_j(S)] \\
&\leq 2(E_S [MST(S)] + \textstyle\sum_{j \neq r} p_j E_S[D_j(S)]) \\
&\leq 2 (OPT+OPT) \\
&= 4OPT
\end{aligned}
$$

Let $D_j(S)$ be the distance from j to its nearest neighbor in S-{j}
Let MST(S) be the length of the minimum spanning tree on S

Goal: analyze $E_S [E_A [c(\tau_A)]]$

Fact 1. $E_S [D_j(S)] = E_S [D_j | j \notin S] = E_S [D_j | j \in S] = E_A [D_j(A) | j \in A]$

Fact 2. $MST(A) \leq c(\tau^*_A)$) $E_A[MST(A)] \cdot OPT$

Fact 3. $\sum_{j \neq r} \mathbb{1}(j \in A) D_j(A) \leq c(\tau^*_A)$) $\sum_{j \neq r} p_j E_A[D_j(A)] \cdot OPT$

Key Idea: always pay for backbone built on S (for any active A)

$$
\begin{aligned}
E_S[E_A[c(\tau_A)]] &\leq E_S[2MST(S)]+E_S [E_A [\textstyle\sum_{j \neq r} \mathbb{1}(j \in A) \mathbb{1}(j \notin S) 2D_j(S)]] \\
&= E_S [2MST(S)] + \textstyle\sum_{j \neq r} E_{S,A} [\mathbb{1}(j \in A)\mathbb{1}(j \notin S) 2D_j (S)] \\
&= E_S [2MST(S)] + 2\textstyle\sum_{j \neq r} p_j (1-p_j) E_S[D_j(S)] \\
&\leq 2(E_S [MST(S)] + \textstyle\sum_{j \neq r} p_j E_S[D_j(S)]) \\
&\leq 2 (OPT+OPT) \\
&= 4OPT
\end{aligned}
$$

The One Random Sample Algorithm

1. Draw sample $S \subseteq N$ according to Π (i.e., pick each point j independently with probability p_j)
2. Build minimum spanning tree on S
3. For each $j \notin S$, connect j to its nearest neighbor in S
4. Build "double tree" tour of this tree $\Rightarrow \tau$

Simplifying Assumption: $\exists$ node r with $p_r = 1$ (wlog)

Theorem (S & Talwar) The one random sample algorithm is a 4-approximation algorithm for the *a priori* TSP.

Two Footnotes

Can be derandomized -

Let $D_j(S)$ be the distance from j to its nearest neighbor in S-{j}

Let MST(S) be the length of the minimum spanning tree on S

Goal: analyze $E_S [E_A [c(\tau_A)]]$

Fact 1. $E_S [D_j(S)] = E_S [D_j | j \notin S] = E_S [D_j | j \in S] = E_A [D_j(A) | j \in A]$

Fact 2. $MST(A) \leq c(\tau^*_A)$ for all A

Fact 3. $\sum_{j \neq r} \mathbb{1}(j \in A) D_j(A) \leq c(\tau^*_A)$ for all A

Key Idea: always pay for backbone built on S (for any active A)

$$
\begin{aligned}
E_S[E_A[c(\tau_A)]] &\leq E_S[2MST(S)]+E_S [E_A [\textstyle\sum_{j \neq r} \mathbb{1}(j \in A) \mathbb{1}(j \notin S) 2D_j(S)]] \\
&= E_S [2MST(S)] + \textstyle\sum_{j\neq r} E_{S,A} [\mathbb{1}(j \in A)\mathbb{1}(j \notin S) 2D_j (S)] \\
&= E_S [2MST(S)] + 2\textstyle\sum_{j \neq r} p_j (1-p_j) E_S[D_j(S)] \\
&\leq 2(E_S [MST(S)] + \textstyle\sum_{j \neq r} p_j E_S[D_j(S)]) \\
&\leq 2 (OPT+OPT) \\
&= 4OPT
\end{aligned}
$$

Two Footnotes

Can be derandomized – Williamson & van Zuylen (2007) show how to deterministically achieve twice guarantee for rent-or-buy/connected facility location problem by the method of conditional probabilities (by an LP estimate)

Assumption that $p_r = 1$ is not needed;

Need only that $D_j(S)$ is well defined.

Modify Π to condition on that each set has cardinality ≥ 2

Can sample according to this new distribution also, and this just rescales things (any tour has cost 0 restricted to 0 or 1 points) but must be careful about dependence

Theorem (S & Talwar) There is a deterministic 8-approximation algorithm for the a priori TSP in the independent activation model

What about the black box model?

Recent work of Gorodezky, R. Kleinberg, S, & Spencer shows that for a (slightly) restricted class of algorithms can embed a universal computation in an a priori one, and thereby show a non-constant lower bound on performance guarantees possible with a polynomial number of samples

---

2-Stage Steiner Tree Problem

Given a set of points N (with root) in a metric space, integer inflation factor λ, and distribution over 2^N

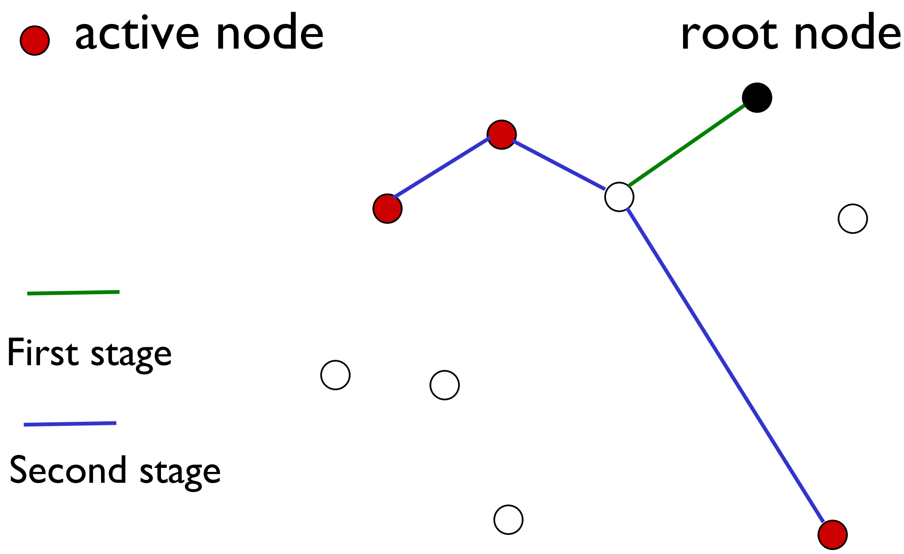
Stage I: install edges A_I – cost of e is c_e

Set of active terminals $T \subseteq N$ is selected (including root)

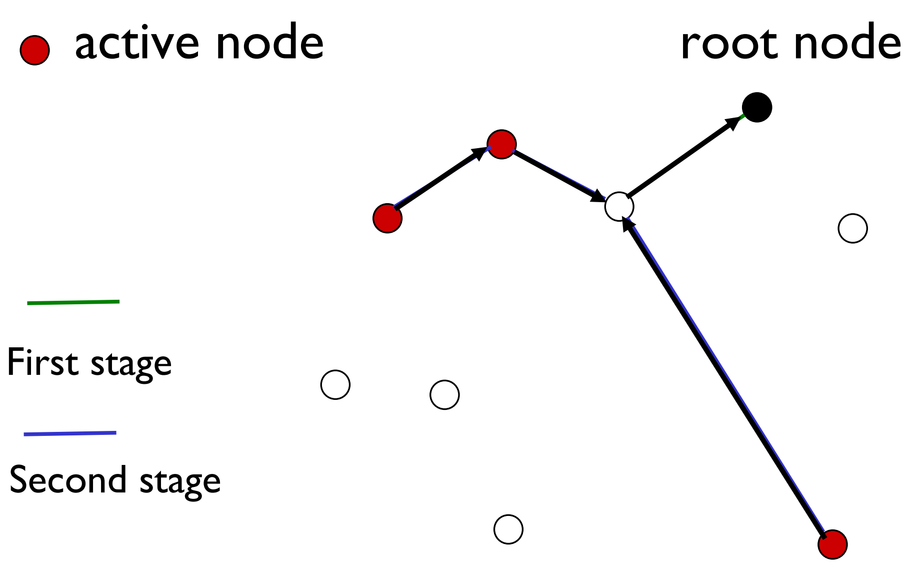
Stage II: install edges A_{II} s.t. $A_I \cup A_{II}$ is Steiner tree on T - cost of edge e is λc_e

Goal: Minimize
(cost of edges installed in stage I) +
$\lambda \, \mathbf{E}_{T \subseteq N}$ [cost of edges installed for scenario T].

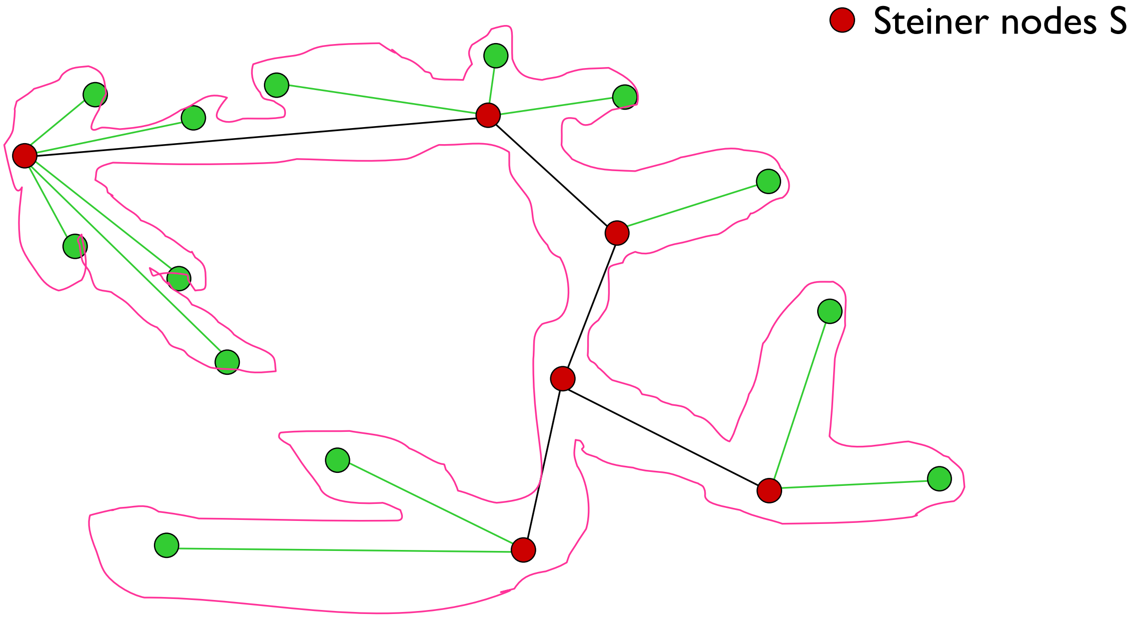
An Example



An Example

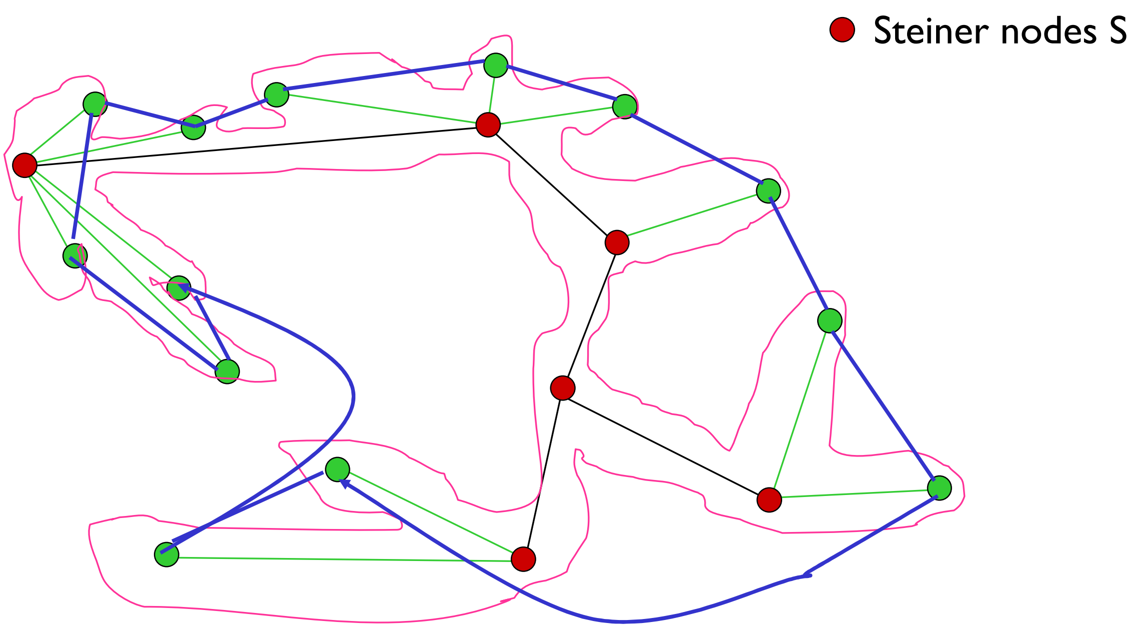


Deterministic Steiner Tree



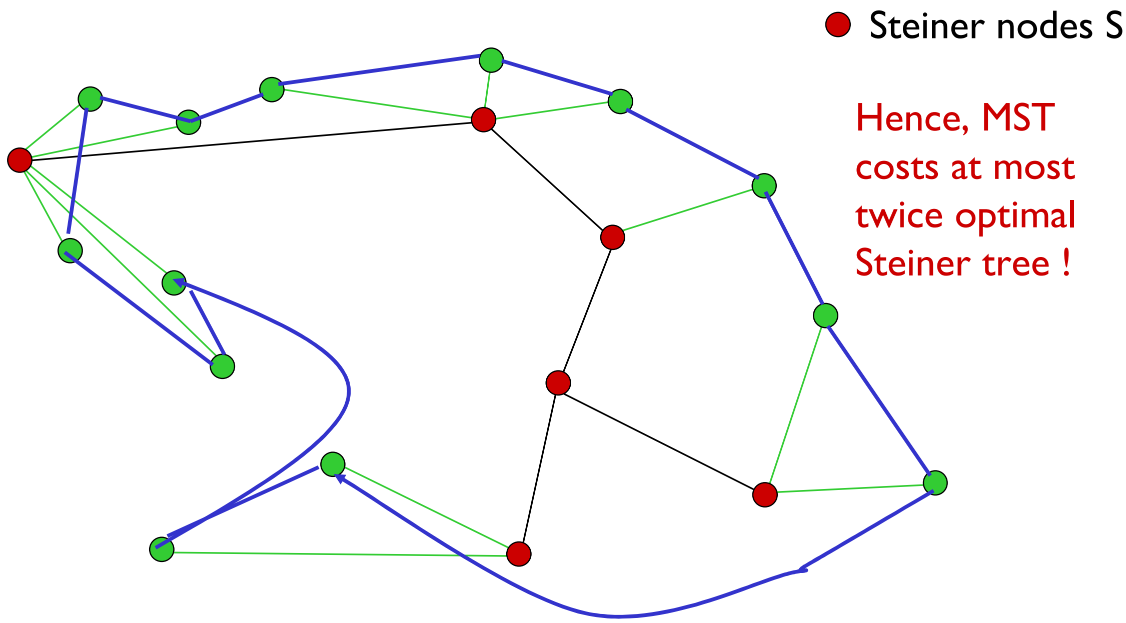
Walking “around” optimal Steiner tree gives connected graph, so can view as connected graph on just terminal nodes (by shortcuts)

Deterministic Steiner Tree



Walking “around” optimal Steiner tree gives connected graph, so can view as connected graph on just terminal nodes (by shortcuts)

Deterministic Steiner Tree



Walking “around” optimal Steiner tree gives connected graph, so can view as connected graph on just terminal nodes (by shortcuts)

Boosted Sampling Algorithm
(Gupta, Pál, Ravi, Sinha)

- Draw λ independent samples $S_1, S_2, \ldots, S_\lambda \rightarrow S$
- First stage decision: compute minimum spanning tree (MST) for S (including root), and install those edges $\rightarrow \text{Alg}_I$
- Observe scenario T (independently drawn from same dist)
- Compute (rooted) minimum spanning tree on $S \cup T$,
  (but make cost of edges Alg_I all 0)
  and let e[j] be edge from j to its parent
- Let $\text{Alg}_{II} \leftarrow \{ e[j] : j \in T \}$

First Stage Cost

Optimal cost $Z^* = c(Opt_I) + \lambda\, E_{T \subseteq N} [\, c(Opt_{II}(T))\,]$

We compute MST on $S \leftarrow S_1 \cup \ldots \cup S_\lambda$ for Stage I

(this is 2-approximation for S)

How expensive is it to connect S? Could use

$$Opt_I \cup Opt_{II}(S_1) \cup \ldots \cup Opt_{II}(S_\lambda)$$

Each S_i is identical random T so its expected cost is

$$c(Opt_I) + \lambda\, E_{T \subseteq N} [\, c(Opt_{II}(T))] \rightarrow Z^*$$

Since MST is 2-approximation algorithm $\Rightarrow$

expected Stage I cost is at most $2Z^*$

Cost sharing role of parental edge

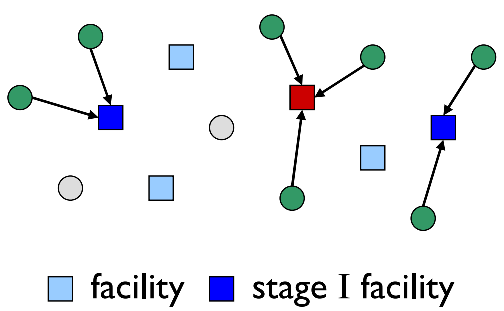
- Build a MST on a set $S \cup T$ (plus root)
- Focus on parental edge e[j] for each $j \in S \cup T$
- Total edge cost is $\sum_{j \in S \cup T} c_{e[j]}$
- But this is $\leq$ twice cost of optimal Steiner tree on $S \cup T$
- Attribute share $c_{e[j]}/2$ of optimal cost to j
- Total share cost is $\leq$ optimal Steiner tree cost

Second Stage Cost

- Algorithm computes Steiner tree for $S_I \cup \ldots \cup S_\lambda \cup T$
- Consider $\mathcal{T} \leftarrow Opt_I \cup Opt_{II}(S_I) \cup \ldots \cup Opt_{II}(S_\lambda) \cup Opt_{II}(T)$
- Role of $\lambda+1$ sets, $S_I, \ldots, S_\lambda$, T is symmetric
- $E[c(\mathcal{T})] \leq c(Opt_I) + (\lambda+1)\, E[c(Opt_{II}(S_i))] \leq (\lambda+1)/\lambda\, Z^*$
- Form $D_1, \ldots, D_\lambda$ by deleting nodes in multiple sets
- $\sum_{j \in T-S} c_{e[j]} + \sum_i \sum_{j \in D_i} c_{e[j]} \leq 2c(\mathcal{T})$
- By symmetry, $E[\sum_{j \in T-S} c_{e[j]}] \leq 2c(\mathcal{T})/(\lambda+1)$
- Hence, $E[\sum_{j \in T-S} c_{e[j]}] \leq 2Z^* / \lambda \Rightarrow$ Stage II cost $\leq 2Z^*$!

$\Rightarrow$ Boosted Sampling is 4-approximation algorithm

2-Stage Stochastic Facility Location



Distribution over clients gives the set of clients to serve.

Stage I: Open some facilities in advance; pay cost f_i for facility i.

Stage I cost = $\sum_{(i \text{ opened})} f_i$.

Actual scenario A = { clients to serve}, materializes.

Stage II: Can open more facilities to serve clients in A; pay cost f_i^A to open facility i. Assign clients in A to facilities.

Stage II cost = $\sum_{i \text{ opened in scenario } A} f_i^A$ + (cost of serving clients in A).

Deterministic Facility Location

Minimize $\sum_i f_i y_i + \sum_{j,i} d_j c_{ij} x_{ij}$

subject to $\sum_i x_{ij} \geq 1 \quad \forall j$

$x_{ij} \leq y_i \quad \forall i, j$

$x_{ij}, y_i \geq 0 \quad \forall i, j$

y_i : indicates if facility i is open.

x_{ij} : indicates if client j is assigned to facility i.

d_j is the demand at client j

A Convex Program

p_A : probability of scenario $A \subseteq \mathcal{D}$.

y_i : indicates if facility i is opened in stage I.

$y_{A,i}$: indicates if facility i is opened in scenario A.

$x_{A,ij}$: whether client j is assigned to facility i in scenario A.

Minimize $h(y) = \sum_i f_i y_i + g(y)$ s.t. $y_i \geq 0$ for each i (SUFL-P)

where, $g(y) = \sum_{A \subseteq \mathcal{D}} p_A g_A(y)$

and $g_A(y) = \min. \sum_i F_i y_{A,i} + \sum_{j,i} c_{ij} x_{A,ij}$

s.t. $\sum_i x_{A,ij} \geq 1$ for each $j \in A$

$x_{A,ij} \leq y_i + y_{A,i}$ for each i,j

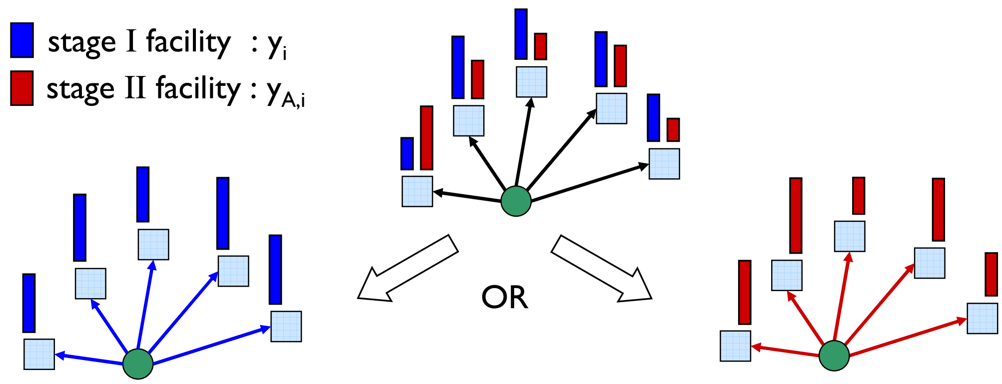
$x_{A,ij}, y_{A,i} \geq 0$ for each i,j.

Rounding (SUFL-P)

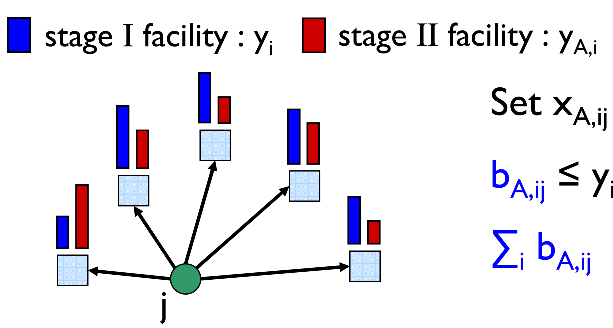
Let y : optimal solution with cost OPT.
(x_A, y_A) : optimal solution for scenario A.

Goal: Decouple stage I and the stage II scenarios.

Assign $j \in A$ exclusively to stage I facilities, or to stage II facilities.



Rounding (contd.)

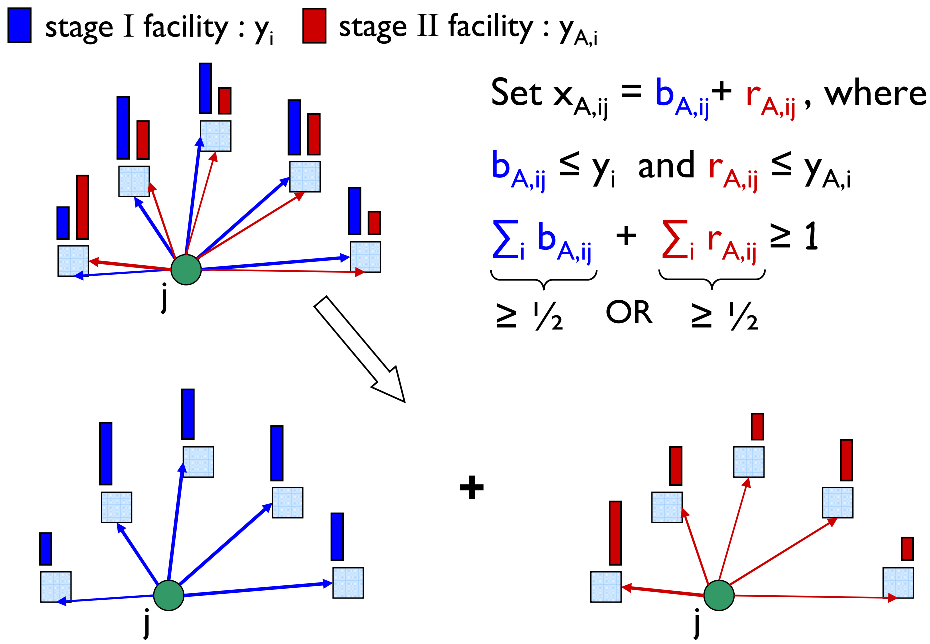


Set $x_{A,ij} = b_{A,ij} + r_{A,ij}$, where

$b_{A,ij} \le y_i$ and $r_{A,ij} \le y_{A,i}$

$\sum_i b_{A,ij} + \sum_i r_{A,ij} \ge 1$

Rounding (contd.)



Set $x_{A,ij} = b_{A,ij} + r_{A,ij}$, where

$b_{A,ij} \leq y_i$ and $r_{A,ij} \leq y_{A,i}$

$\underbrace{\sum_i b_{A,ij}}_{\geq 1/2} + \underbrace{\sum_i r_{A,ij}}_{\geq 1/2} \geq 1$

$\geq ½$ OR $\geq ½$

Rounding (contd.)

$\sum_i b_{A,ij} \geq ½ \Rightarrow$ $(2b_{A,ij})$ is a feasible assignment for j with facility variables $2y_i$.

$\sum_i r_{A,ij} \geq ½ \Rightarrow$ $(2r_{A,ij})$ is a feasible assignment for j with facility variables $2y_{A,i}$.

Have an α-approx. algorithm for UFL wrt. LP relaxation.

Stage I

- Solve UFL instance: facility set $\mathcal{F}$ with costs f_i, client set $D = \{(j,A) : \sum_i b_{A,ij} \geq ½\}$, (j,A) has demand p_A.
- $(\{2b_{A,ij}\}_{(j,A)\in D}, 2y)$ is a feasible fractional solution.
- Obtain integer solution: gives facilities to open in stage I.
- Takes care of client j in each scenario A where $\sum_i b_{A,ij} \geq ½$.

Rounding (contd.)

Stage II, scenario A

- Assign $j \in A$ such that $\sum_i b_{A,ij} \geq ½$ to stage I facility.
  $\Rightarrow$ Only need to assign remaining clients with $\sum_i r_{A,ij} \geq ½$.
- Solve UFL instance: facility set $\mathcal{F}$ with costs F_i , client set $D_A = \{ j \in A : \sum_i r_{A,ij} \geq ½ \}$.
- $(\{2r_{A,ij}\}_{j \in D_A}, 2y_A)$ is a feasible fractional solution.
- "Round" to get an integer solution
  – determines what other facilities to open in scenario A,
  – how to assign clients in D_A.

Shows a 2α integrality gap for stochastic UFL-LP. Modify slightly to get a 3.23-approximation algorithm for stochastic UFL.

Lecture #3

Stochastic Set Cover (SSC)

Universe $U = \{e_1, \ldots, e_n\}$, subsets $S_1, S_2, \ldots, S_m \subseteq U$, set S has weight ω_S.

Deterministic problem: Pick a minimum weight collection of sets that covers each element.

Stochastic version: Set of elements to be covered is given by a probability distribution.

- choose some sets initially paying ω_S for set S
- subset $A \subseteq U$ to be covered is revealed
- can pick additional sets paying W_S for set S.

Minimize (ω-cost of sets picked in stage I) +
$\mathbf{E}_{A \subseteq U}$ [W_S -cost of new sets picked for scenario A].

Stochastic Set Covering LP

p_A : probability of scenario $A \subseteq U$.
x_S : indicates if set S is picked in stage I.
$y_{A,S}$: indicates if set S is picked in scenario A.

Minimize $\sum_S \omega_S x_S + \sum_{A \subseteq U} p_A \sum_S W_S y_{A,S}$
subject to,

$$\sum_{S: e \in S} x_S + \sum_{S: e \in S} y_{A,S} \geq 1 \qquad \text{for each } A \subseteq U, e \in A$$
$$x_S, y_{A,S} \geq 0 \qquad \text{for each } S, A.$$

Exponential number of variables and exponential number of constraints.

Inflation factor ¸ = $\max_S W_S/\omega_S$

Sample Average Approximation

Sample Average Approximation (SAA) method:

- Sample initially N times from scenario distribution
- Solve 2-stage problem estimating p_A with frequency of occurrence of scenario A

How large should N be?

Kleywegt, Shapiro & Homem De-Mello 01: bound N by variance of a certain quantity – need not be polynomially bounded even for our class of programs.

SwamyS 05: show using ε-subgradients that for our class, N can be poly-bounded.

Nemirovskii & Shapiro: show that for stochastic set cover LP with non-scenario dependent costs, KSH01 gives polynomial bound on N for (preprocessing + SAA) algorithm. Later also without preprocessing.

Charikar, Chekuri, & Pal 05: give elegant "Chernoff"-based proof that an α-approximation for polynomial-scenario setting yields (1+ε)α-approximation for black box setting

A Compact Formulation

p_A : probability of scenario $A \subseteq U$.

x_S : indicates if set S is picked in stage I.

Minimize $h(x) = \sum_S \omega_S x_S + f(x)$ s.t. $x_S \geq 0$ for each S

where, $f(x) = \sum_{A \subseteq U} p_A f_A(x)$

and $f_A(x) = \min. \sum_S W_S y_{A,S}$

s.t. $\sum_{S: e \in S} y_{A,S} \geq 1 - \sum_{S: e \in S} x_S$ for each $e \in A$

$y_{A,S} \geq 0$ for each S.

Equivalent to earlier LP.

Each $f_A(x)$ is convex, so $f(x)$ and $h(x)$ are convex functions.

Sample Average Approximation

Sample Average Approximation (SAA) method:

- Sample N times from distribution
- Estimate p_A by q_A = frequency of occurrence of scenario A

(P) $\min_{x\in \mathcal{P}} \left(h(x) = \omega\cdot x + \sum_{A\subseteq U} p_A f_A(x)\right)$

(SAA-P) $\min_{x\in \mathcal{P}} \left(h'(x) = \omega\cdot x + \sum_{A\subseteq U} q_A f_A(x)\right)$

To show: With poly-bounded N, if $\bar{x}$ solves (SAA-P) then $h(\bar{x}) \approx OPT$.

Let $z_A \equiv$ optimal dual solution for scenario A at point $u \in \Re^m$.

$\Rightarrow d_u$ with $d_{u,S} = \omega_S - \sum_{A\subseteq U} q_A \sum_{e\in S} z_{A,e}$ is a subgradient of $h'(.)$ at u.

Lemma: With high probability, for "many" points u in $\mathcal{P}$,

d_u is a subgradient of $h'(.)$ at u,

d_u is an approximate subgradient of $h(.)$ at u.

Establishes "closeness" of $h(.)$ and $h'(.)$ and suffices to prove result.

Intuition: Can run ellipsoid on both (P) and (SAA-P) using the same vector d_u at feasible point u.

Sample Average Approximation

Sample Average Approximation (SAA) method:

- Sample initially N times from scenario distribution
- Solve 2-stage problem estimating p_A with frequency of occurrence of scenario A

How large should N be?

Kleywegt, Shapiro & Homem De-Mello 01: bound N by variance of a certain quantity – need not be polynomially bounded even for our class of programs.

SwamyS 05: show using ε-subgradients that for our class, N can be poly-bounded.

Nemirovskii & Shapiro: show that for stochastic set cover LP with non-scenario dependent costs, KSH01 gives polynomial bound on N for (preprocessing + SAA) algorithm. Later also without preprocessing.

Charikar, Chekuri, & Pal 05: give elegant "Chernoff"-based proof that an α-approximation for polynomial-scenario setting yields $(1+\varepsilon)\alpha$-approximation for black box setting

Sample Average Approximation

(Charikar, Chekuri, and Pál)

Sample Average Approximation (SAA) method:

- Sample N times from distribution
- Estimate p_A by q_A = frequency of occurrence of scenario A

(P) $\min_{x\in\mathcal{P}}$ (h(x) = ω.x_+ $\sum_{A\subseteq U} p_A f_A(x)$) _

(SAA-P) $\min_{x\in\mathcal{P}}$ (h'(x) = ω.x + $\sum_{A\subseteq U} q_A f_A(x)$)

Prove just a weak version – let x' be a minimizer of h' – we want to show that for N polynomial, then x' is also of objective function value within a factor of 1+², with probability 1-∂

Chernoff Bound – Let X_k 2 [0,1], k=1,…,M, be ind. r.v.s & let X =$\sum_k X_k$. Then, for any ² > 0

Pr[|X-E[X]| > ²M] · 2 exp(-²² M)

We'll take N = c ¸² n log (1/∂) (1/²⁴)

Sample Average Approximation (SAA) method:

- Sample N times from distribution
- Estimate p_A by q_A = freq. of occurrence of scenario A

(P) OPT = $\min_{x\in\mathcal{P}}$ (h(x) = ω.x + $\sum_{A\subseteq U} p_A f_A(x)$ =ωx + f(x))

(SAA-P) $\min_{x\in\mathcal{P}}$ (h'(x) =_ω.x + $\sum_{A\subseteq U} q_A f_A(x)$_=ωx + f'(x))

Divide scenarios into high and low: say A is high if $f_A(0)$ ¸ ¸OPT/²

By def'n of ¸: for each x & A, $f_A(0)$ · ¸ωx + $f_A(x)$

Lemma: Let p be probability that A is high; then p · (1/¸)²/(1-²).

Proof: OPT = ωx* + $E_A[f_A(x^*)]$ for optimal x*.

¸ p $E_A[f_A(x^*)$ | A high] ¸ p $E_A[$ $f_A(0)$-¸ωx*| A high]

¸ p [¸ OPT/² - ¸ OPT]=p OPT ¸ (1-²)/² qed

Three Key Properties

$f(x) = \omega x + \sum_{A \mu U} p_A \, f_A(x)$

$f_{hi}(x) = \sum_{A\ high} p_A \, f_A(x)$ & $f_{lo}(x) = f(x) - f_{hi}(x) - \omega x$

and analogous for f'

- For each x, $|f_{lo}(x) - f'_{lo}(x)| \cdot {}^2OPT$ w.h.p.
- For each x, $f'_{hi}(0) - f'_{hi}(x) \cdot 2^2\omega x$ w.h.p.
- For each x, $f_{hi}(0) - f_{hi}(x) \cdot 2^2\omega x$.

Three Key Properties

$f(x) = \omega x + \sum_{A \mu U} p_A \, f_A(x)$ & f' replace p by q

$f_{hi}(x) = \sum_{A\ high} p_A \, f_A(x)$ & $f_{lo}(x) = f(x) - f_{hi}(x) - \omega x$

and analogous for f'

- For each x, $|f_{lo}(x) - f'_{lo}(x)| \cdot {}^2OPT$ w.h.p.
- For each x, $f'_{hi}(0) - f'_{hi}(x) \cdot 2^2\omega x$ w.h.p.
- For each x, $f_{hi}(0) - f_{hi}(x) \cdot 2^2\omega x$.

For SAA minimizer x': (by above + $f_{hi}(x) \cdot f_{hi}(0)$)

$$f(x') - f'(x') \cdot {}^2OPT + 2^2\omega x' + f_{hi}(0) - f'_{hi}(0)$$

$$+ \quad f'(x^*) - f(x^*) \cdot {}^2OPT + 2^2\omega x^* + f'_{hi}(0) - f_{hi}(0)\ )$$

$$f(x') - 2^2\omega x' \cdot f(x^*) + 2^2\omega x^* + 2^2OPT\ )$$

$$(1-2^2)f(x') \cdot (1+4^2)OPT \quad !!$$

Three Key Properties

- For each x, $|f_{lo}(x) – f'_{lo}(x)| \cdot {}^{2}OPT$ w.h.p.
- For each x, $f'_{hi}(0) – f'_{hi}(x) \cdot 2^{2}\omega x$ w.h.p.
- For each x, $f_{hi}(0) – f_{hi}(x) \cdot 2^{2}\omega x$.

Lemma. With probability $1-\partial$, the fraction of high scenarios is at most $2^{2}/_{¸}$ (by Chernoff)

This yields 2nd and 3rd properties directly.

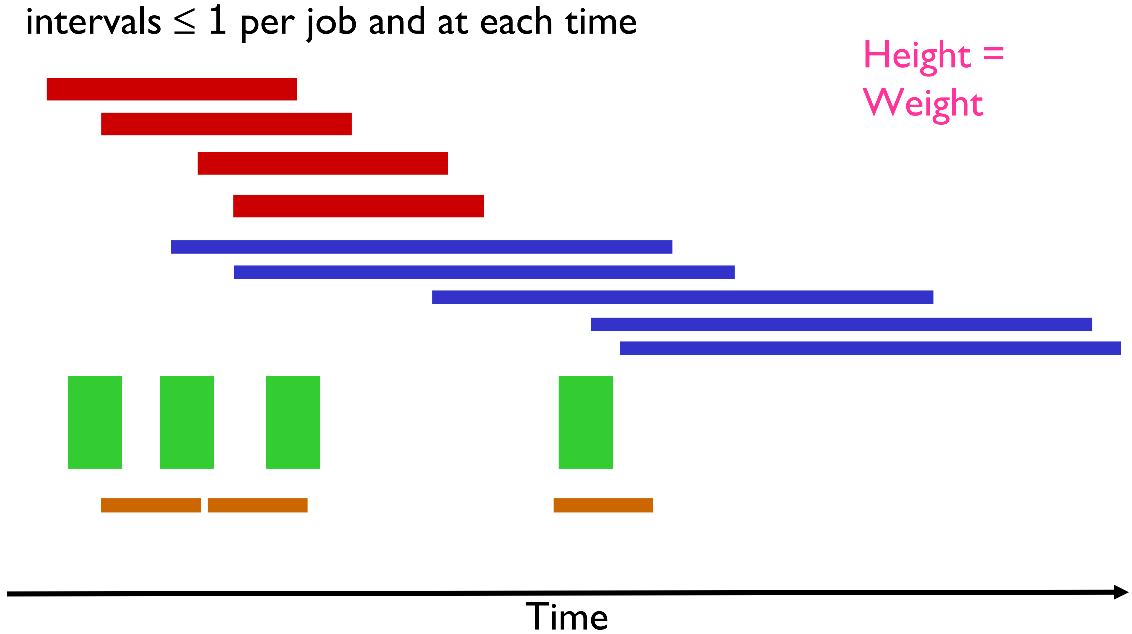
For 1st, view $f'_{lo}(x)$ as the mean of N independent random variable F_i that is $f_A(x)$ if A is low, but 0 otherwise

Apply Chernoff bound to $(1/N)\sum_i F_i/[¸\, OPT/^{2}]$ to get 1st property

Maximum-weight on-time set

Jobs N = {1,2,…,n} - job j has set of allowed time intervals $S_j = \{[s_{1j},e_{1j}),\ldots,[s_{kj},e_{kj})\}$ with corresponding weights w_{ij}

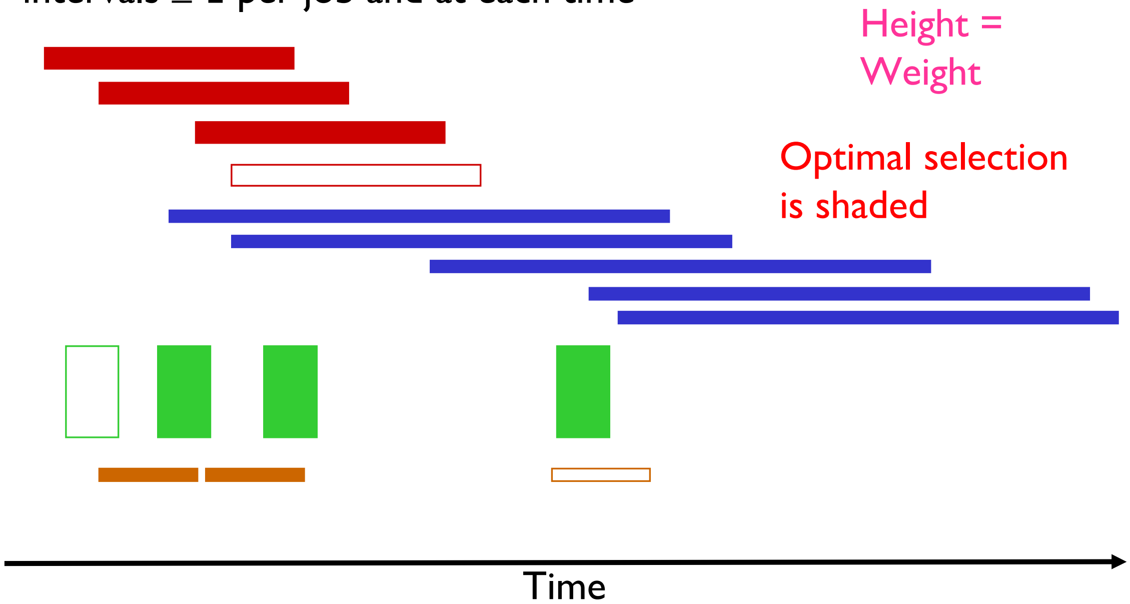
Deterministic problem: Pick a maximum-weight collection of intervals ≤ 1 per job and at each time



Maximum-weight on-time set

Jobs $N = \{1,2,\ldots,n\}$ - job j has set of allowed time intervals
$S_j = \{[s_{1j},e_{1j}),\ldots,[s_{kj},e_{kj})\}$ with corresponding weights w_{ij}

Deterministic problem: Pick a maximum-weight collection of intervals ≤ 1 per job and at each time



Maximum-weight on-time set

Jobs $N = \{1,2,\ldots,n\}$ - job j has set of allowed time intervals
$S_j = \{[s_{1j},e_{1j}),\ldots,[s_{kj},e_{kj})\}$ with corresponding weights w_{ij}

Deterministic problem: Pick a maximum-weight collection of intervals ≤ 1 per job and at each time

Linear Programming Relaxation

Let T_t be the set of intervals (for all jobs) containing time t : (i,j)

x_{ij} : indicates whether $[s_{ij},e_{ij})$ selected for job j

Maximize $\sum_{i,j} w_{ij}\, x_{ij}$

Subject to	$\sum_i x_{ij} \leq 1,$	for each j=1,…,n
	$\sum_{(i,j) \in T_t} x_{ij} \leq 1$	for each t
	$x_{ij} \geq 0$	for each i,j

Theorem [Bar-Noy, Bar-Yehuda, Freund, Naor, & Schieber]
Primal-dual 2-approximation algorithm for max-weight schedule

2-Stage Stochastic Variant

Scenario $A \subseteq N$ of active jobs occurs with probability p_A

Stage I: Choose set $D \subseteq N$ of jobs to defer to subcontractor and receive small weight ω_j for each $j \in D$

Stage II: Given realized scenario A, make selection T_A where $(i,j) \in T_A \Rightarrow j \in A\text{-}D$ and has weight W_{ij}

Goal: Maximize the total expected weight scheduled (where expectation is with respect to active subset probabilities)

A Primal-Dual Theorem

Theorem: [S & Sozio] We can (adapt the 2-approximation algorithm for deterministic setting to) obtain a 2-approximation algorithm for stochastic maximum-weight on-time scheduling.

Note: it is trivial to obtain a 4-approximation algorithm (flip a coin and either decide to either put all of your eggs in Stage I or Stage II) and almost as simple to obtain a 3-approximation algorithm

We focus first on the polynomial-scenario model

Maximum-weight on-time set

Jobs $N = \{1,2,\ldots,n\}$ - job j has set of allowed time intervals

$S_j = \{[s_{1j},e_{1j}),\ldots,[s_{kj},e_{kj})\}$ with corresponding weights w_{ij}

Deterministic problem: Pick a maximum-weight collection of intervals ≤ 1 per job and at each time

Linear Programming Relaxation

Let T_t be the set of intervals (for all jobs) containing time t : (i,j)

x_{ij} : indicates whether $[s_{ij},e_{ij})$ selected for job j

Maximize $\sum_{i,j} w_{ij} x_{ij}$

Subject to

$\sum_i x_{ij} \leq 1$	for each j=1,…,n
$\sum_{(i,j) \in T_t} x_{ij} \leq 1$	for each t
$x_{ij} \geq 0$	for each i,j

Theorem [Bar-Noy, Bar-Yehuda, Freund, Naor, & Schieber]
Primal-dual 2-approximation algorithm for max-weight schedule

Dual Linear Program

Let T_t be the set of intervals (for all jobs) containing time t

Minimize $\sum_j u_j + \sum_t v_t$
Subject to

$$u_j + \sum_{t:\,(i,j) \in T_t} v_t \geq w_{ij} \text{ for each } (i,j)$$

$$u_j,\ v_t \geq 0$$

The primal-dual algorithm has two phases:

•first it constucts a feasible dual solution, while building a stack of possible pairs (i,j) to be selected;

•next it pops the stack, selecting any pair that doesn't conflict with those already selected;

•amortization shows dual cost is at most twice the value of the primal.

Dual Linear Program

Let T_t be the set of intervals (for all jobs) containing time t

Minimize $\sum_t u_j + \sum_t v_t$

Subject to

$u_j + \sum_{t: (i,j) \in T_t} v_t \geq w_{ij}$ for each (i,j) ← For dual solution (u,v) call (i,j) covered if this constraint is satisfied

$u_j, v_t \geq 0$

The primal-dual algorithm has two phases:

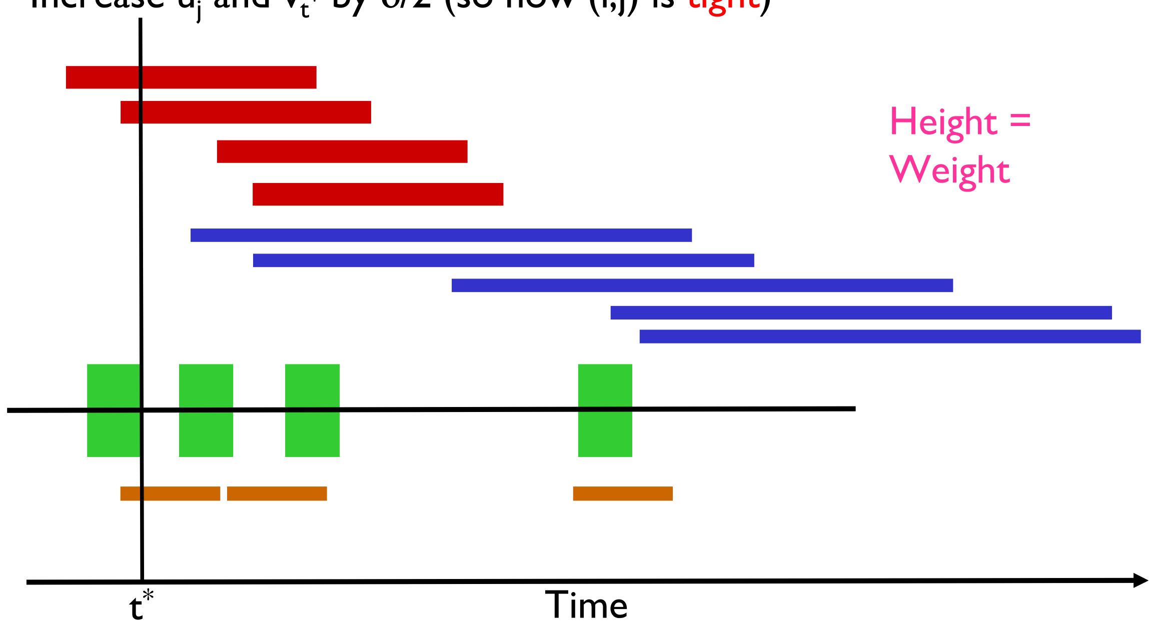
•first it constucts a feasible dual solution, while building a stack of possible pairs (i,j) to be selected;

•next it pops the stack, selecting any pair that doesn't conflict with those already selected;

•amortization shows dual cost is at most twice the cost of the primal.

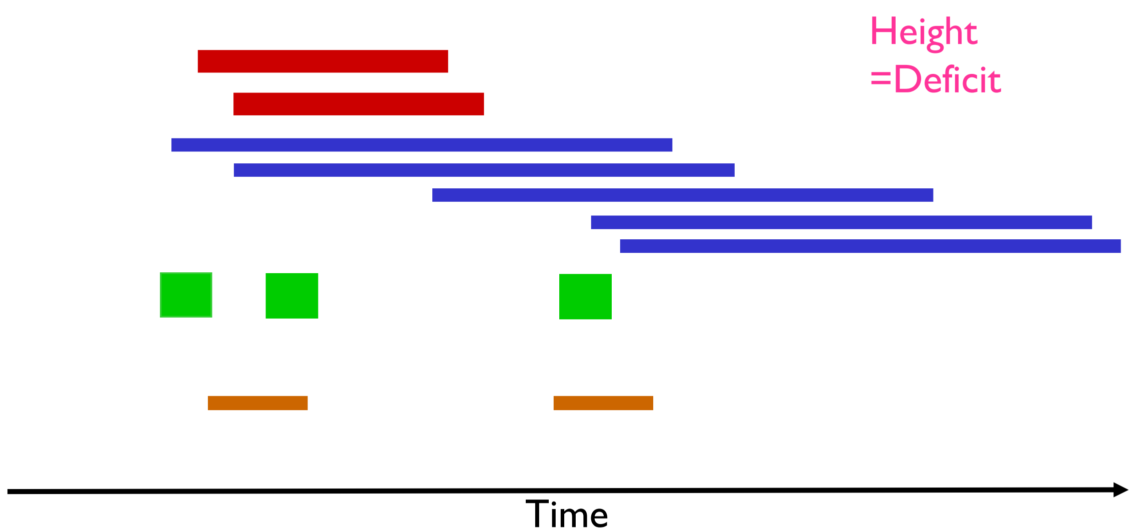
The Primal-Dual Algorithm of Bar-Noy et al.

- Pick the uncovered interval (i,j) with the earliest ending point t^*
- Compute its deficit $\delta = w_{ij} - u_j - \sum_{t: (i,j) \in T_t} v_t$
- Increase u_j and v_{t^*} by $\delta/2$ (so now (i,j) is tight)



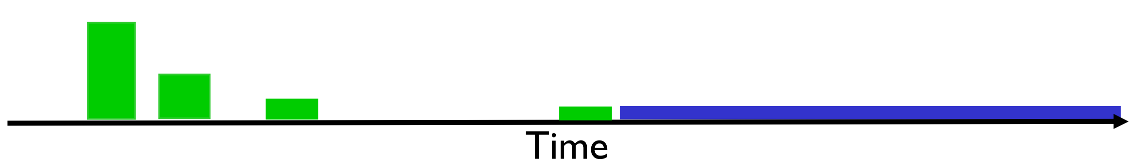
The Primal-Dual Algorithm of Bar-Noy et al.

- Pick the uncovered interval (i,j) with the earliest ending point t^*
- Compute its deficit $\delta = w_{ij} - u_j - \sum_{t:\,(i,j) \in T_t} v_t$
- Increase u_j and v_{t^*} by $\delta/2$ (so now (i,j) is tight)



The Primal-Dual Algorithm of Bar-Noy et al.

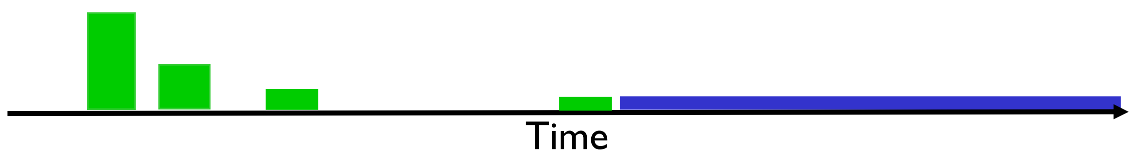
- Pick the uncovered interval (i,j) with the earliest ending point t^*
- Compute its deficit $\delta = w_{ij} - u_j - \sum_{t:\,(i,j) \in T_t} v_t$
- Increase u_j and v_{t^*} by $\delta/2$ (so now (i,j) is tight)
- Keep "stack" of tight (i,j)'s
- Pop them off an add to selection if they don't conflict with ones chosen already



The Primal-Dual Algorithm of Bar-Noy et al.

- Pick the uncovered interval (i,j) with the earliest ending point t^*
- Compute its deficit $\delta = w_{ij} - u_j - \sum_{t:\,(i,j) \in T_t} v_t$
- Increase u_j and v_{t^*} by $\delta/2$ (so now (i,j) is tight)
- Keep "stack" of tight (i,j)'s
- Pop them off an add to selection if they don't conflict with ones chosen already

•Analysis: every selected interval is tight; every iteration adds δ to dual objective and contributes at least $\delta/2$ to "paying for" selected (i,j)'s; hence, dual objective is at most twice amount paid!!



Linear Programming Relaxation for 2-Stage Problem

Let T_t be the set of intervals (for all jobs) containing time t

x_j : indicates whether job j is deferred in stage I

$y_{ij}(S)$: indicates whether $[s_{ij}, e_{ij})$ selected for job j in stage II for scenario S

Maximize $\sum_j \omega_j x_j + \sum_{i,j,S} p_S W_{ij}\, y_{ij}(S)$

Subject to $x_j + \sum_i y_{ij}(S) \leq 1,$ for each j,S

$\sum_{(i,j) \in T_t} y_{ij}(S) \leq 1,$ for each t,S

$x_j,\ y_{ij}(S) \geq 0$ for each i,j,S

DUAL Minimize $\sum_{j,S} u_j(S) + \sum_{t,S} v_t(S)$

Subject to $\sum_S u_j(S) \geq \omega_j$ for each j

$u_j(S) + \sum_{t:\,(i,j) \in T_t} v_t(S) \geq p(S)\, W_{ij}$ for each (i,j), S

$u_j(S),\ v_t(S) \geq 0$

A Simple 2-Stage Algorithm

For each scenario $A \subseteq N$ with probability $p_A > 0$

run the deterministic algorithm with job set A where weight of job j for $[s_{ij}, e_{ij})$ is $p_A W_{ij}$

let $u_j(A)$ denote the dual values constucted by the algorithm

Stage I: Let D be the set of jobs j for which

$$\omega_j > \Sigma_A \; u_j(A)$$

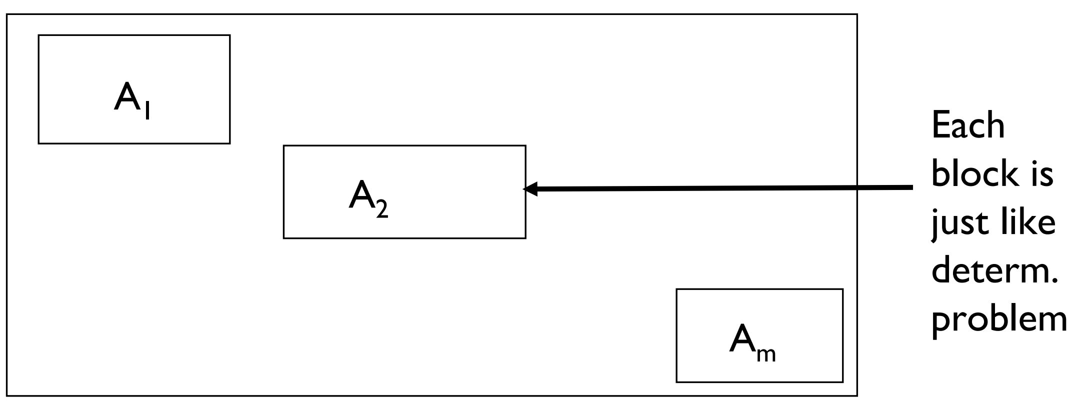
Stage II: Given realized scenario A,

recompute first phase of algorithm (to get duals)

but in second phase never select (i,j) for $j \in D$

Main Idea of Analysis

What is 2-stage dual? Block-structured by scenario A with additional linking constraints:



$$\Sigma_A \; u_j(S) \qquad \geq \omega_j \; \forall \; j$$

So we can adapt the scenario-by-scenario constuction as building a feasible dual solution for the 2-stage linear relaxation

What about black box model?

Just use sampling to estimate the deferral rule! - use M samples

For each sampled scenario $A \subseteq N$ run deterministic algorithm with job set A where weight of job j for $[s_{ij}, e_{ij})$ is W_{ij} to obtain dual values $u_j(A)$ – let A_k be k^{th} sample

Stage I: Let D be the set of jobs j for which

$$(1+\varepsilon)\ \omega_j > (1/M) \sum_k u_j(A_k)$$

Stage II: Given realized scenario A, compute T_A and then recompute first phase of algorithm (to get duals) but in second phase never select (i,j) for $j \in D$

Number of samples needed is polynomial in n, $1/\varepsilon$, and $\lambda = \max_j W_j/\omega_j$

Similar to "sample average approximation" results of [Swamy & S, Shapiro & Nemirovski, and Charikar, Chekuri, & Pál]

Some Additional Details

Previously used profits equal to $p_A W_{ij}$ for i^{th} interval of job j in scenario A – what now?

Ignore p_A – call rescaled duals $u_j^*(A)$ where

$$u_j(A) = p_A\ u_j^*(A)$$

Had used $r = \sum_A p_A\ u_j(A)$ as threshold

Now use $\mathbf{r}^* = \sum_K (1/M)\ u_j^*(A_k)$ instead

Use Chernoff bounds to prove $\mathbf{r}^*$ ¼ r w/high prob.

Chernoff – Let X_k 2 [0,1] , k=1,…,M be ind. r.v.s & let $X = \sum_k X_k$. Then, for any ² > 0

Pr[|X-E[X]| > ² M] · 2 exp(-²² M)

What are the [0,1] random variables?

What about black box model?

Just use sampling to estimate the deferral rule! - use M samples

For each sampled scenario $A \subseteq N$ run deterministic algorithm with job set A where weight of job j for $[s_{ij},e_{ij})$ is W_{ij} to obtain dual values $u_j(A)$ – let A_k be k^{th} sample

Stage I: Let D be the set of jobs j for which

$$(1+\varepsilon)\,\omega_j > (1/M)\sum_k u_j(A_k)$$

Stage II: Given realized scenario A, compute T_A and then recompute first phase of algorithm (to get duals) but in second phase never select (i,j) for $j \in D$

Number of samples needed is polynomial in n, $1/\varepsilon$, and $\lambda = \max_j W_j/\omega_j$

Similar to "sample average approximation" results of [Swamy & S, Shapiro & Nemirovski, and Charikar, Chekuri, & Pál]

Applying the Chernoff Bound

Let $X_k = u_j^*(A_k) /(¸\ \omega_j)$

Why is X_k 2 [0,1]?

There is some i such that

$u_j^*(A_k)$ · W_{ij} (perhaps i that became tight)

and so $u_j^*(A_k)$ · W_{ij} · ¸ ω_j

Now take M= £((¸²/²²) log (n/°)) to get

Pr[|X-E[X]| > ²M/¸] ·exp(-²²M/ ¸²) ,

Pr[| **r*** - r | > ² ω_j] · °/n

And apply "union bound" to get failure for all jobs j occurs with probability at most °

Another 2-Stage Stochastic Variant

Scenario $A \subseteq N$ of active jobs occurs with probability p_A

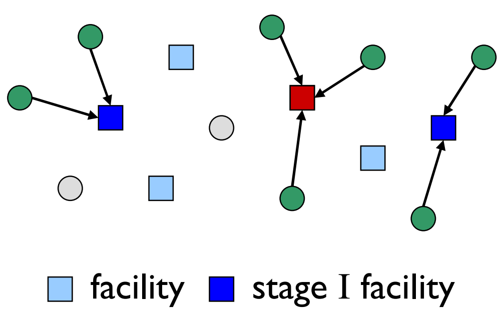
Stage I: Choose set $C \subseteq N$ of jobs j to commit to service and get weight ω_j

Stage II: Given realized scenario A, make selection T_A where $\exists\ (i,j) \in T_A$ for each $j \in C$ plus some additional ones

Goal: Maximize the total expected weight scheduled

Bad News: There is an approximation-preserving reduction from the deterministic maximum independent set problem, and hence no "reasonable" performance guarantee can be proved (unless P=NP).

2-Stage Stochastic Facility Location



Distribution over clients gives the set of clients to serve.

Stage I: Open some facilities in advance; pay cost f_i for facility i.

Stage I cost = $\sum_{(i \text{ opened})} f_i$.

Actual scenario A = { ● clients to serve}, materializes.

Stage II: Can open more facilities to serve clients in A; pay cost f_i^A to open facility i. Assign clients in A to facilities.

Stage II cost = $\sum_{i \text{ opened in scenario } A} f_i^A$ + (cost of serving clients in A).

Several Ways to Skin The Cat Facility Location Yet Again

- Can apply LP-rounding approach as done for set covering [S & Swamy]
- Can apply the boosted sampling approach if the second stage costs are proportional
- In polynomial scenario setting can adapt primal-dual algorithm of Jain & Vazirani for deterministic version to get 3-approximation algorithm [Mahdian]
- Can then apply result Sample Average Approximation result of [Charikar, Chekuri, & Pál] to extend to "black box" model

Discrete Stochastic Optimization and Approximation Algorithms

- Area of emerging importance
- Rich source of algorithmic questions
- Can one prove a strong result for approximate stochastic dynamic programming? [Levi Roundy & S] [Halman, Klabjan, Mostagir, Orlin & Simchi-Levi]
- When is sampling information good enough to derive near-optimal solutions?
- Reconsider some well-studied problems but now in "black box" model, not just specific distributions
- Expectation is not enough

Thank You.