Advanced Algorithms (I)

Chihao Zhang

Shanghai Jiao Tong University

Feb. 25, 2019

Advanced Algorithms

Advanced Algorithms

In the course, we will learn Approximation Algorithms

Advanced Algorithms

In the course, we will learn Approximation Algorithms

- linear programming, semi-definite programming
- spectral method
- random walks
- ...

Advanced Algorithms

In the course, we will learn Approximation Algorithms

- linear programming, semi-definite programming
- spectral method
- random walks
- ...

We will emphasize on

- tools for designing approximation algorithms
- rigorous analysis of algorithms

Course Info

Course Info

- Instructor: Chihao Zhang
- Course Homepage:
  `http://chihaozhang.com/teaching/AA2019spring/`
- Office Hour: every Monday, 7:00pm - 9:00pm

Course Info

- Instructor: Chihao Zhang
- Course Homepage:
  `http://chihaozhang.com/teaching/AA2019spring/`
- Office Hour: every Monday, 7:00pm - 9:00pm

Grading Policy

- Homework 30%
- Mid-term Exam 30%
- Course Project 40%

MaxSAT

MaxSAT

Given a CNF formula ϕ, is it satisfiable?

MaxSAT

Given a CNF formula ϕ, is it satisfiable?

$$\phi = (x_1 \vee x_3 \vee \bar{x}_{29}) \wedge (\bar{x}_3 \vee x_7) \wedge \cdots \wedge (\bar{x}_{33} \vee \bar{x}_{34} \vee x_{90} \vee x_{126})$$

MaxSAT

Given a CNF formula ϕ, is it satisfiable?

$$\phi = (x_1 \vee x_3 \vee \bar{x}_{29}) \wedge (\bar{x}_3 \vee x_7) \wedge \cdots \wedge (\bar{x}_{33} \vee \bar{x}_{34} \vee x_{90} \vee x_{126})$$

NP-hard, we look at its optimization version.

MaxSAT

Given a CNF formula ϕ, is it satisfiable?

$$\phi = (x_1 \vee x_3 \vee \bar{x}_{29}) \wedge (\bar{x}_3 \vee x_7) \wedge \cdots \wedge (\bar{x}_{33} \vee \bar{x}_{34} \vee x_{90} \vee x_{126})$$

NP-hard, we look at its optimization version.

MaxSAT	
Input:	A CNF formula $\phi = C_1 \wedge C_2 \cdots \wedge C_m$.
Problem:	Compute an assignment that satisfies maximum number of clauses.

MaxSAT

Given a CNF formula ϕ, is it satisfiable?

$$\phi = (x_1 \vee x_3 \vee \bar{x}_{29}) \wedge (\bar{x}_3 \vee x_7) \wedge \cdots \wedge (\bar{x}_{33} \vee \bar{x}_{34} \vee x_{90} \vee x_{126})$$

NP-hard, we look at its optimization version.

MaxSAT	
Input:	A CNF formula $\phi = C_1 \wedge C_2 \cdots \wedge C_m$.
Problem:	Compute an assignment that satisfies maximum number of clauses.

Harder than SAT, so we look for an approximate solution.

Tossing a Coin

Tossing a Coin

An instance ϕ

- The variable sets $V = \{x_1, x_2, \ldots, x_n\}$
- The set of clauses $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$
- Each clause C_i contains ℓ_i literals

Tossing a Coin

An instance ϕ

- The variable sets $V = \{x_1, x_2, \ldots, x_n\}$
- The set of clauses $C = \{C_1, C_2, \ldots, C_m\}$
- Each clause C_i contains ℓ_i literals

We first cosnider the following simple algorithm:

Tossing a Coin

An instance ϕ

- The variable sets $V = \{x_1, x_2, \ldots, x_n\}$
- The set of clauses $C = \{C_1, C_2, \ldots, C_m\}$
- Each clause C_i contains ℓ_i literals

We first cosnider the following simple algorithm:

- For each variable x_i, toss an independent fair coin.

Tossing a Coin

An instance ϕ

- The variable sets $V = \{x_1, x_2, \ldots, x_n\}$
- The set of clauses $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$
- Each clause C_i contains ℓ_i literals

We first cosnider the following simple algorithm:

- For each variable x_i, toss an independent fair coin.
- If the coin goes `HEAD`, we set x_i **true**, otherwise we set x_i **false**.

Analysis

Analysis

The outcome of the algorithm is random, we are interested in its expectation.

Analysis

The outcome of the algorithm is random, we are interested in its expectation.

For this particular algorithm, it can be derandomized.

Analysis

The outcome of the algorithm is random, we are interested in its expectation.

For this particular algorithm, it can be derandomized.

$$\mathbf{E}[X] = \sum_{i=1}^{m} \mathbf{Pr}[C_i \text{ is satisfied}]$$

Analysis

The outcome of the algorithm is random, we are interested in its expectation.

For this particular algorithm, it can be derandomized.

$$\mathbf{E}[X] = \sum_{i=1}^{m} \mathbf{Pr}[C_i \text{ is satisfied}] = \sum_{i=1}^{m} \left(1 - 2^{-\ell_i}\right) \geq \frac{m}{2}.$$

Analysis

The outcome of the algorithm is random, we are interested in its expectation.

For this particular algorithm, it can be derandomized.

$$\mathbf{E}[X] = \sum_{i=1}^{m} \mathbf{Pr}[C_i \text{ is satisfied}] = \sum_{i=1}^{m} \left(1 - 2^{-\ell_i}\right) \geq \frac{m}{2}.$$

On the otherhand,

$$\mathbf{OPT} \leq m.$$

Analysis

The outcome of the algorithm is random, we are interested in its expectation.

For this particular algorithm, it can be derandomized.

$$\mathbf{E}[X] = \sum_{i=1}^{m} \mathbf{Pr}[C_i \text{ is satisfied}] = \sum_{i=1}^{m} \left(1 - 2^{-\ell_i}\right) \geq \frac{m}{2}.$$

On the otherhand,

$$\mathbf{OPT} \leq m.$$

Therefore,

$$\mathbf{E}[X] \geq \frac{1}{2} \cdot \mathbf{OPT}.$$

Can we improve the previous algorithm?

Can we improve the previous algorithm?

Observations

- the worst case happens when for some singleton clause, i.e., $\ell_i = 1$;

Can we improve the previous algorithm?

Observations

- the worst case happens when for some singleton clause, i.e., $\ell_i = 1$;
- for a singleton $C = x$, if there is no $C' = \bar{x}$, then we can increase the probability of x to be true;

Can we improve the previous algorithm?

Observations

- the worst case happens when for some singleton clause, i.e., $\ell_i = 1$;
- for a singleton $C = x$, if there is no $C' = \bar{x}$, then we can increase the probability of x to be true;
- otherwise, we can improve the upper bound for **OPT**! (x and $\bar{x}$ cannot be both satisfied)

Tossing a Biased Coin

Tossing a Biased Coin

Let the p-biased coin be with probability p to `HEAD`, $1 - p$ to `TAIL`.

Tossing a Biased Coin

Let the p-biased coin be with probability p to `HEAD`, $1 - p$ to `TAIL`.

Tossing a Biased Coin

Let the p-biased coin be with probability p to `HEAD`, $1 - p$ to `TAIL`.

- For each variable x_i, toss an independent p-biased coin.

Tossing a Biased Coin

Let the p-biased coin be with probability p to HEAD, $1 - p$ to TAIL.

- For each variable x_i, toss an independent p-biased coin.
- If the coin goes HEAD, we set x_i **true**, otherwise we set x_i **false**.

Analysis

Analysis

Assumption for Clauses of ϕ

- More positive singletons than negative singletons.

Analysis

Assumption for Clauses of ϕ

- More positive singletons than negative singletons.
- There are t pairs of x and $\bar{x}$.

Analysis

Assumption for Clauses of ϕ

- More positive singletons than negative singletons.
- There are t pairs of x and $\bar{x}$.
- $p \geq \frac{1}{2}$.

$$\mathbf{E}[X] \geq t + (m - 2t) \min\left\{p, 1 - p^2\right\}.$$

Analysis

Assumption for Clauses of ϕ

- More positive singletons than negative singletons.
- There are t pairs of x and $\bar{x}$.
- $p \geq \frac{1}{2}$.

$$\mathbf{E}[X] \geq t + (m - 2t) \min\left\{p, 1 - p^2\right\}.$$

The new upper bound for **OPT**:

$$\mathbf{OPT} \leq m - t$$

Analysis

Assumption for Clauses of ϕ

- More positive singletons than negative singletons.
- There are t pairs of x and $\bar{x}$.
- $p \geq \frac{1}{2}$.

$$\mathbf{E}[X] \geq t + (m - 2t) \min\left\{p, 1 - p^2\right\}.$$

The new upper bound for **OPT**:

$$\mathbf{OPT} \leq m - t$$

Combine them and obtain

$$\mathbf{E}[X] \geq \alpha \cdot \mathbf{OPT}$$

where $\alpha \approx 0.618$.

Can we further improve the algorithm?

Can we further improve the algorithm?

We can use different coins for each variable, e.g., in

$$\phi = x_1 \wedge (x_1 \vee \bar{x}_2),$$

we prefer to set x_1 to **true**.

Can we further improve the algorithm?

We can use different coins for each variable, e.g., in

$$\phi = x_1 \wedge (x_1 \vee \bar{x}_2),$$

we prefer to set x_1 to **true**.

How can we make use of the information?

Can we further improve the algorithm?

We can use different coins for each variable, e.g., in

$$\phi = x_1 \wedge (x_1 \vee \bar{x}_2),$$

we prefer to set x_1 to **true**.

How can we make use of the information?

Linear Programming helps.

Tossing Clever Coins

Tossing Clever Coins

We introduce the following variables.

Tossing Clever Coins

We introduce the following variables.

- for every $i \in [n]$, y_i indicates whether x_i is true;
- for every $j \in [m]$, z_j indicates whether C_j is satisfied.

Tossing Clever Coins

We introduce the following variables.

- for every $i \in [n]$, y_i indicates whether x_i is true;
- for every $j \in [m]$, z_j indicates whether C_j is satisfied.

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{m} z_j \\
\text{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{k \in N_j} (1 - y_k) \geq z_j, \quad \forall j \in [m] \text{ s.t. } C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{k \in N_j} \bar{x}_k \\
& z_j \in \{0, 1\}, \quad \forall j \in [m] \\
& y_i \in \{0, 1\}, \quad \forall i \in [n]
\end{aligned}
$$

Tossing Clever Coins

We introduce the following variables.

- for every $i \in [n]$, y_i indicates whether x_i is true;
- for every $j \in [m]$, z_j indicates whether C_j is satisfied.

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} z_j \\ \text{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{k \in N_j} (1 - y_k) \geq z_j, \quad \forall j \in [m] \text{ s.t. } C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{k \in N_j} \bar{x}_k \\ & z_j \in \{0, 1\}, \quad \forall j \in [m] \\ & y_i \in \{0, 1\}, \quad \forall i \in [n] \end{aligned}$$

This integer program is equivalent to MaxSAT.

Relaxation

Relaxation

There is no efficient algorithm for integer programming in general

Relaxation

There is no efficient algorithm for integer programming in general

Therefore, we relax its non-linear constriants

Relaxation

There is no efficient algorithm for integer programming in general

Therefore, we relax its non-linear constriants

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{m} z_j \\
\text{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{k \in P_j} (1 - y_k) \ge z_j, \quad \forall j \in [m] \text{ s.t. } C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{k \in N_j} \bar{x}_k \\
& 0 \le z_j \le 1, \quad \forall j \in [m] \\
& 0 \le y_i \le 1, \quad \forall i \in [n]
\end{aligned}
$$

Relaxation

There is no efficient algorithm for integer programming in general

Therefore, we relax its non-linear constriants

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{m} z_j \\
\text{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{k \in P_j} (1 - y_k) \geq z_j, \quad \forall j \in [m] \text{ s.t. } C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{k \in N_j} \bar{x}_k \\
& 0 \leq z_j \leq 1, \quad \forall j \in [m] \\
& 0 \leq y_i \leq 1, \quad \forall i \in [n]
\end{aligned}
$$

We can solve this LP in poly-time

Algorithm

Algorithm

Let $\left(\left\{y_i^*\right\}_{i\in[n]}, \left\{z_j^*\right\}_{j\in[m]}\right)$ be an optimal solution of the LP.

Algorithm

Let $\left(\left\{y_i^*\right\}_{i\in[n]}, \left\{z_j^*\right\}_{j\in[m]}\right)$ be an optimal solution of the LP.

Algorithm

Let $\left(\left\{y_i^*\right\}_{i\in[n]}, \left\{z_j^*\right\}_{j\in[m]}\right)$ be an optimal solution of the LP.

- For each variable x_i, toss an independent y_i^*-biased coin.

Algorithm

Let $\left(\left\{ y_i^* \right\}_{i \in [n]}, \left\{ z_j^* \right\}_{j \in [m]} \right)$ be an optimal solution of the LP.

- For each variable x_i, toss an independent y_i^*-biased coin.
- If the coin goes `HEAD`, we set x_i **true**, otherwise we set x_i **false**.

Analysis

Analysis

A typical upper bound of **OPT** for LP based algorithms is

$$\mathbf{OPT} \le \mathbf{OPT}(LP) = \sum_{j=1}^{m} z_j^*$$

Analysis

A typical upper bound of **OPT** for LP based algorithms is

$$\mathbf{OPT} \leq \mathbf{OPT}(LP) = \sum_{j=1}^{m} z_j^*$$

We can further establish

$$\mathbf{E}[X] \geq \left(1 - \frac{1}{e}\right) \sum_{j=1}^{m} z_j^*.$$

Analysis

A typical upper bound of **OPT** for LP based algorithms is

$$\mathbf{OPT} \leq \mathbf{OPT}(LP) = \sum_{j=1}^{m} z_j^*$$

We can further establish

$$\mathbf{E}[X] \geq \left(1 - \frac{1}{e}\right) \sum_{j=1}^{m} z_j^*.$$

Therefore, the LP rounding is a $\left(1 - \frac{1}{e}\right)$-approximation algorithm for MaxSAT

Advanced Algorithms (II)

Chihao Zhang

Shanghai Jiao Tong University

Mar. 4, 2019

MaxSAT

MaxSAT

Recall that we can obtain a $\frac{3}{4}$-approximation by choosing the better of the two.

MaxSAT

Recall that we can obtain a $\frac{3}{4}$-approximation by choosing the better of the two.

We show that this ratio can also be achieved via direct rounding.

MaxSAT

Recall that we can obtain a $\frac{3}{4}$-approximation by choosing the better of the two.

We show that this ratio can also be achieved via direct rounding.

Recall that we have the following linear programming relaxation.

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{m} z_j \\
\text{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{k \in N_j} (1 - y_k) \geq z_j, \quad \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{k \in N_j} \bar{x}_k \\
& z_j \in [0,1], \quad \forall j \in [m] \\
& y_i \in [0,1], \quad \forall i \in [n]
\end{aligned}
$$

Let $\left\{y_i^*\right\}_{i\in[n]}, \left\{z_j^*\right\}_{j\in[m]}$ be an optimal solution of the LP.

Let $\left\{y_i^*\right\}_{i\in[n]}, \left\{z_j^*\right\}_{j\in[m]}$ be an optimal solution of the LP.

Instead of tossing y_i^*-biased coins, we toss $f(y_i^*)$-biased coins for some function $f(\cdot) \in [0,1]$.

Let $\{y_i^*\}_{i\in[n]}, \left\{z_j^*\right\}_{j\in[m]}$ be an optimal solution of the LP.

Instead of tossing y_i^*-biased coins, we toss $f(y_i^*)$-biased coins for some function $f(\cdot) \in [0,1]$.

For $C_j = \bigvee_{i\in P_j} x_i \vee \bigvee_{k\in N_j} \bar{x}_k$,

$$\mathbf{Pr}\left[C_j \text{ is not satisfied }\right] = \prod_{i\in P_j} (1 - f(y_i^*)) \prod_{k\in N_j} f(y_k^*).$$

Let $\left\{y_i^*\right\}_{i\in[n]}, \left\{z_j^*\right\}_{j\in[m]}$ be an optimal solution of the LP.

Instead of tossing y_i^*-biased coins, we toss $f(y_i^*)$-biased coins for some function $f(\cdot) \in [0,1]$.

For $C_j = \bigvee_{i\in P_j} x_i \vee \bigvee_{k\in N_j} \bar{x}_k$,

$$\mathbf{Pr}\left[C_j \text{ is not satisfied }\right] = \prod_{i\in P_j}(1 - f(y_i^*)) \prod_{k\in N_j} f(y_k^*).$$

We can choose a suitable f to get $\frac{3}{4}$ approximation.

Integrality Gap

Integrality Gap

In most LP based approximation algorithms, the upper bound for the OPT is

$$\mathbf{OPT} \leq \mathbf{OPT}(LP).$$

Integrality Gap

In most LP based approximation algorithms, the upper bound for the OPT is

$$\mathbf{OPT} \leq \mathbf{OPT}(LP).$$

Then we establish

$$OPT^* \geq \alpha \cdot \mathbf{OPT}(LP) \geq \alpha \cdot \mathbf{OPT}.$$

Integrality Gap

In most LP based approximation algorithms, the upper bound for the OPT is

$$\mathbf{OPT} \leq \mathbf{OPT}(LP).$$

Then we establish

$$OPT^* \geq \alpha \cdot \mathbf{OPT}(LP) \geq \alpha \cdot \mathbf{OPT}.$$

If we already know

$$\mathbf{OPT} \leq \beta \cdot \mathbf{OPT}(LP),$$

Integrality Gap

In most LP based approximation algorithms, the upper bound for the OPT is

$$\mathbf{OPT} \le \mathbf{OPT}(LP).$$

Then we establish

$$OPT^* \ge \alpha \cdot \mathbf{OPT}(LP) \ge \alpha \cdot \mathbf{OPT}.$$

If we already know

$$\mathbf{OPT} \le \beta \cdot \mathbf{OPT}(LP),$$

then we cannot have $\alpha > \beta$!

Integrality Gap

In most LP based approximation algorithms, the upper bound for the OPT is

$$\mathbf{OPT} \leq \mathbf{OPT}(LP).$$

Then we establish

$$OPT^* \geq \alpha \cdot \mathbf{OPT}(LP) \geq \alpha \cdot \mathbf{OPT}.$$

If we already know

$$\mathbf{OPT} \leq \beta \cdot \mathbf{OPT}(LP),$$

then we cannot have $\alpha > \beta$!

The ratio β is called the integrality gap of the LP relaxation.

Integrality Gap for MaxSAT

Integrality Gap for MaxSAT

Consider the instance,

$$(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

Integrality Gap for MaxSAT

Consider the instance,

$$(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

$\mathbf{OPT} = 3$ and $\mathbf{OPT}(LP) = 4$.

Integrality Gap for MaxSAT

Consider the instance,

$$(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

$\mathbf{OPT} = 3$ and $\mathbf{OPT}(LP) = 4$.

The integrality gap of our LP is $\frac{3}{4}$.

Integrality Gap for MaxSAT

Consider the instance,

$$(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

$\mathbf{OPT} = 3$ and $\mathbf{OPT}(LP) = 4$.

The integrality gap of our LP is $\frac{3}{4}$.

Corollary. We cannot beat $\frac{3}{4}$ if we use $\mathbf{OPT} \leq \mathbf{OPT}(LP)$ upper bound.

Minimum Label Cut

Minimum Label Cut

Minimum Label s-t Cut

Input: A graph $G = (V, E)$; a set of labels $[L] = \{1, 2, \ldots, L\}$ such that each $e \in E$ is labelled with one $\ell(e) \in [L]$; two vertices $s, t \in V$.

Problem: Compute a minimum set of labels $L' \subseteq [L]$ such that the removal of all edges with label in L' disconnects s and t.

Minimum Label Cut

Minimum Label s-t Cut

Input: A graph $G = (V, E)$; a set of labels $[L] = \{1, 2, \ldots, L\}$ such that each $e \in E$ is labelled with one $\ell(e) \in [L]$; two vertices $s, t \in V$.

Problem: Compute a minimum set of labels $L' \subseteq [L]$ such that the removal of all edges with label in L' disconnects s and t.

NP-hard, and even hard to approximate with any constant ratio (unless $\mathbf{NP} = \mathbf{P}$).

LP Relaxation

LP Relaxation

We introduce a variable z_j for each label $j \in [L]$. Let $\mathcal{P}_{s,t}$ be the collection of paths between s and t.

LP Relaxation

We introduce a variable z_j for each label $j \in [L]$. Let $\mathcal{P}_{s,t}$ be the collection of paths between s and t.

$$\begin{aligned} \min \quad & \sum_{j=1}^{L} z_j \\ \text{subject to} \quad & \sum_{e \in P} z_{\ell(e)} \geq 1, \quad \forall P \in \mathcal{P}_{s,t} \\ & z_j \in [0,1], \quad \forall j \in [L] \end{aligned}$$

LP Relaxation

We introduce a variable z_j for each label $j \in [L]$. Let $\mathcal{P}_{s,t}$ be the collection of paths between s and t.

$$\begin{aligned}
\min \quad & \sum_{j=1}^{L} z_j \\
\text{subject to} \quad & \sum_{e \in P} z_{\ell(e)} \geq 1, \quad \forall P \in \mathcal{P}_{s,t} \\
& z_j \in [0,1], \quad \forall j \in [L]
\end{aligned}$$

Q1: How to solve this LP efficiently?

LP Relaxation

We introduce a variable z_j for each label $j \in [L]$. Let $\mathcal{P}_{s,t}$ be the collection of paths between s and t.

$$\begin{aligned} \min \quad & \sum_{j=1}^{L} z_j \\ \text{subject to} \quad & \sum_{e \in P} z_{\ell(e)} \geq 1, \quad \forall P \in \mathcal{P}_{s,t} \\ & z_j \in [0,1], \quad \forall j \in [L] \end{aligned}$$

Q1: How to solve this LP efficiently?

Q2: What is the integrality gap of this LP?

LP Relaxation

We introduce a variable z_j for each label $j \in [L]$. Let $\mathcal{P}_{s,t}$ be the collection of paths between s and t.

$$\begin{aligned} \min \quad & \sum_{j=1}^{L} z_j \\ \text{subject to} \quad & \sum_{e \in P} z_{\ell(e)} \geq 1, \quad \forall P \in \mathcal{P}_{s,t} \\ & z_j \in [0,1], \quad \forall j \in [L] \end{aligned}$$

Q1: How to solve this LP efficiently?

Q2: What is the integrality gap of this LP? $\Omega(m)$.

Separation Oracle

Separation Oracle

We can solve the LP in poly-time using ellipsoid method provided a separation oracle:

Separation Oracle

We can solve the LP in poly-time using ellipsoid method provided a separation oracle:

Given a point, in PTIME either

- confirm it is a feasible solution; or
- find a violated constraint.

Separation Oracle

We can solve the LP in poly-time using ellipsoid method provided a separation oracle:

Given a point, in PTIME either

- confirm it is a feasible solution; or
- find a violated constraint.

Oracle here: shortest s-t path

Beat the Interality Gap

Beat the Interality Gap

- Obtain a partial solution via rounding;

Beat the Interality Gap

- Obtain a partial solution via rounding;
- Complement the solution by combinatorial construction.

Beat the Interality Gap

- Obtain a partial solution via rounding;
- Complement the solution by combinatorial construction.

Beat the Interality Gap

- Obtain a partial solution via rounding;
- Complement the solution by combinatorial construction.

Let $\left\{z_j^*\right\}_{j\in[L]}$ be an optimal solution of the LP. Let $\beta > 0$ be a parameter.

Beat the Interality Gap

- Obtain a partial solution via rounding;
- Complement the solution by combinatorial construction.

Let $\left\{z_j^*\right\}_{j\in[L]}$ be an optimal solution of the LP. Let $\beta > 0$ be a parameter.

1. Let $L_1 \triangleq \left\{j \in L \,:\, z_j^* \geq \beta\right\}$.

Beat the Interality Gap

- Obtain a partial solution via rounding;
- Complement the solution by combinatorial construction.

Let $\left\{z_j^*\right\}_{j\in[L]}$ be an optimal solution of the LP. Let $\beta > 0$ be a parameter.

1. Let $L_1 \triangleq \left\{j \in L \,:\, z_j^* \geq \beta\right\}$.
2. Let G' be the graph obtained from G by removing edges with label in L_1.

Beat the Interality Gap

- Obtain a partial solution via rounding;
- Complement the solution by combinatorial construction.

Let $\left\{z_j^*\right\}_{j\in[L]}$ be an optimal solution of the LP. Let $\beta > 0$ be a parameter.

1. Let $L_1 \triangleq \left\{j \in L \,:\, z_j^* \geq \beta\right\}$.
2. Let G' be the graph obtained from G by removing edges with label in L_1.
3. Let F be the minimum s-t cut of G', L_2 be the labels of edges in F.

Beat the Interality Gap

- Obtain a partial solution via rounding;
- Complement the solution by combinatorial construction.

Let $\left\{z_j^*\right\}_{j\in[L]}$ be an optimal solution of the LP. Let $\beta > 0$ be a parameter.

1. Let $L_1 \triangleq \left\{j \in L : z_j^* \geq \beta\right\}$.
2. Let G' be the graph obtained from G by removing edges with label in L_1.
3. Let F be the minimum s-t cut of G', L_2 be the labels of edges in F.
4. Return $L_1 \cup L_2$.

Bounding L_1 and L_2

Bounding L_1 and L_2

It is clear that

$$|L_1| \leq \sum_{j \in [L]} \frac{1}{\beta} \cdot z_j^* = \frac{1}{\beta} \cdot \mathbf{OPT}(LP) \leq \frac{1}{\beta} \cdot \mathbf{OPT}.$$

Bounding L_1 and L_2

It is clear that

$$|L_1| \le \sum_{j \in [L]} \frac{1}{\beta} \cdot z_j^* = \frac{1}{\beta} \cdot \mathbf{OPT}(LP) \le \frac{1}{\beta} \cdot \mathbf{OPT}.$$

On the otherhand, there cannot be too many edge disjoint paths between s and t in G':

Bounding L_1 and L_2

It is clear that

$$|L_1| \leq \sum_{j \in [L]} \frac{1}{\beta} \cdot z_j^* = \frac{1}{\beta} \cdot \mathbf{OPT}(LP) \leq \frac{1}{\beta} \cdot \mathbf{OPT}.$$

On the otherhand, there cannot be too many edge disjoint paths between s and t in G':

- at least $\frac{1}{\beta}$ edges on each s-t path;
- at most $\frac{m-|L_1|}{1/\beta} = \beta(m - |L_1|)$ such paths;
- therefore $|L_2| \leq |F| \leq \beta(m - |L_1|)$ (Menger's theorem).

The Choice of β

The Choice of β

We already have

$$|L_1| + |L_2| \leq \frac{1}{\beta} \cdot \mathbf{OPT} + \beta(m - |L_1|) \leq \frac{1}{\beta} \cdot \mathbf{OPT} + \beta m.$$

Setting $\beta = \sqrt{\frac{\mathbf{OPT}}{m}}$ yields an $O\left(m^{\frac{1}{2}}\right)$ approximation.

The Choice of β

We already have

$$|L_1| + |L_2| \leq \frac{1}{\beta} \cdot \mathbf{OPT} + \beta(m - |L_1|) \leq \frac{1}{\beta} \cdot \mathbf{OPT} + \beta m.$$

Setting $\beta = \sqrt{\frac{\mathbf{OPT}}{m}}$ yields an $O\left(m^{\frac{1}{2}}\right)$ approximation.

Remark

Instead of using Menger's theorem, we can find a small cut by BFS from s.

The Choice of β

We already have

$$|L_1| + |L_2| \leq \frac{1}{\beta} \cdot \mathbf{OPT} + \beta(m - |L_1|) \leq \frac{1}{\beta} \cdot \mathbf{OPT} + \beta m.$$

Setting $\beta = \sqrt{\frac{\mathbf{OPT}}{m}}$ yields an $O\left(m^{\frac{1}{2}}\right)$ approximation.

Remark

Instead of using Menger's theorem, we can find a small cut by BFS from s.

Exercise. Find an $O\left(n^{\frac{2}{3}}\right)$-approx algorithm via rounding + BFS.

Advanced Algorithms (III)

Chihao Zhang

Shanghai Jiao Tong University

Mar. 11, 2019

MaxCut

MaxCut

MaxCut	
Input:	An undirected graph $G = (V, E)$.
Problem:	A set $S \subseteq V$ that maximizes $\left\|E(S, \bar{S})\right\|$.

MaxCut

MaxCut	
Input:	An undirected graph $G = (V, E)$.
Problem:	A set $S \subseteq V$ that maximizes $\left\|E(S, \overline{S})\right\|$.

NP-hard

MaxCut

MaxCut	
Input:	An undirected graph $G = (V, E)$.
Problem:	A set $S \subseteq V$ that maximizes $\left\lvert E(S, \overline{S}) \right\rvert$.

NP-hard

Similar to Max2SAT, tossing a fair coin yields an $\frac{1}{2}$-approximation. (Exercise)

MaxCut

MaxCut	
Input:	An undirected graph $G = (V, E)$.
Problem:	A set $S \subseteq V$ that maximizes $\left\|E(S, \bar{S})\right\|$.

NP-hard

Similar to Max2SAT, tossing a fair coin yields an $\frac{1}{2}$-approximation. (Exercise)

Can we find clever coins via LP relaxation...?

LP for MaxCut

LP for MaxCut

- introduce a vairable $x_u \in \{0, 1\}$ for every $u \in V$.

LP for MaxCut

- introduce a vairable $x_u \in \{0, 1\}$ for every $u \in V$.
- introduce a variable $y_{u,v} \in \{0, 1\}$ for every edge $e = \{u, v\}$.

LP for MaxCut

- introduce a vairable $x_u \in \{0, 1\}$ for every $u \in V$.
- introduce a variable $y_{u,v} \in \{0, 1\}$ for every edge $e = \{u, v\}$.

The cost function is $\sum_{e=\{u,v\}\in E} y_{u,v}$.

LP for MaxCut

- introduce a vairable $x_u \in \{0,1\}$ for every $u \in V$.
- introduce a variable $y_{u,v} \in \{0,1\}$ for every edge $e = \{u,v\}$.

The cost function is $\sum_{e=\{u,v\}\in E} y_{u,v}$.

How to write linear constraints for a cut?

LP for MaxCut

- introduce a vairable $x_u \in \{0,1\}$ for every $u \in V$.
- introduce a variable $y_{u,v} \in \{0,1\}$ for every edge $e = \{u,v\}$.

The cost function is $\sum_{e=\{u,v\}\in E} y_{u,v}$.

How to write linear constraints for a cut?

idea: Let $F = \left\{\{u,v\} \in E \,:\, y_{u,v} = 1\right\}$, we view $(S, \overline{S}, F)$ as a bipartite subgraph of G.

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u, v\} \in \binom{V}{2}$.

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u, v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u, v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.
- introduce constraints to rule out odd cycles.

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u, v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.
- introduce constraints to rule out odd cycles.

$$\max \quad \sum_{\{u,v\} \in E} y_{u,v}$$

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u, v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.
- introduce constraints to rule out odd cycles.

$$
\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} y_{u,v} \\
\text{s.t.} \quad & y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u, v, w \in V
\end{aligned}
$$

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u, v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.
- introduce constraints to rule out odd cycles.

$$
\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} y_{u,v} \\
\text{s.t.} \quad & y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u, v, w \in V \\
& \sum_{e=\{u,v\}\in C} y_{u,v} \le |C| - 1, \quad \forall \text{odd cycle } C
\end{aligned}
$$

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u,v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.
- introduce constraints to rule out odd cycles.

$$
\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} y_{u,v} \\
\text{s.t.} \quad & y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u,v,w \in V \\
& y_{u,v} + y_{u,w} + y_{w,v} \le 2, \quad \forall u,v,w, \in V
\end{aligned}
$$

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u, v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.
- introduce constraints to rule out odd cycles.

$$
\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} y_{u,v} \\
\text{s.t.} \quad & y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u, v, w \in V \\
& y_{u,v} + y_{u,w} + y_{w,v} \le 2, \quad \forall u, v, w, \in V \\
& y_{u,v} = y_{v,u}, \quad \forall u, v \in V
\end{aligned}
$$

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u,v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.
- introduce constraints to rule out odd cycles.

$$
\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} y_{u,v} \\
\text{s.t.} \quad & y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u,v,w \in V \\
& y_{u,v} + y_{u,w} + y_{w,v} \le 2, \quad \forall u,v,w, \in V \\
& y_{u,v} = y_{v,u}, \quad \forall u,v \in V \\
& y_{u,v} \in \{0,1\}, \quad \forall u,v \in V
\end{aligned}
$$

- introduce $y_{u,v}$ and $y_{v,u}$ for every $\{u, v\} \in \binom{V}{2}$.
- introduce constaints for being bipartite graph metric.
- introduce constraints to rule out odd cycles.

$$
\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} y_{u,v} \\
\text{s.t.} \quad & y_{u,v} \le y_{u,w} + y_{w,v}, \quad \forall u, v, w \in V \\
& y_{u,v} + y_{u,w} + y_{w,v} \le 2, \quad \forall u, v, w, \in V \\
& y_{u,v} = y_{v,u}, \quad \forall u, v \in V \\
& y_{u,v} \in [0, 1], \quad \forall u, v \in V
\end{aligned}
$$

Interality Gap

Interality Gap

Theorem

For every $\varepsilon > 0$, there exists a graph G such that

$$\frac{LP(G)}{\textsc{MaxCut}(G)} \geq 2 - \varepsilon$$

Interality Gap

Theorem

For every $\varepsilon > 0$, there exists a graph G such that

$$\frac{LP(G)}{\textsc{MaxCut}(G)} \geq 2 - \varepsilon$$

Random graph $\mathcal{G}(n, p)$ for proper p...

LP in Matrix Form

LP in Matrix Form

$$
\begin{aligned}
\max \quad & 2x - 3y \\
\text{s.t.} \quad & x + y \leq 2 \\
& 3x - y \leq 1 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

LP in Matrix Form

$$
\begin{aligned}
\max \quad & 2x - 3y \\
\text{s.t.} \quad & x + y \le 2 \\
& 3x - y \le 1 \\
& x \ge 0 \\
& y \ge 0
\end{aligned}
$$

$$
\begin{aligned}
\max \quad & \begin{bmatrix} 2 & 0 \\ 0 & -3 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & -y \end{bmatrix} \\
\text{s.t.} \quad & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \le 2 \\
& \begin{bmatrix} 3 & 0 \\ 0 & -1 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \le 1 \\
& \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \succeq 0
\end{aligned}
$$

LP in Matrix Form

$$
\begin{aligned}
\max \quad & 2x - 3y \\
\text{s.t.} \quad & x + y \le 2 \\
& 3x - y \le 1 \\
& {\color{blue} x \ge 0} \\
& {\color{blue} y \ge 0}
\end{aligned}
$$

$$
\begin{aligned}
\max \quad & \begin{bmatrix} 2 & 0 \\ 0 & -3 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & -y \end{bmatrix} \\
\text{s.t.} \quad & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \le 2 \\
& \begin{bmatrix} 3 & 0 \\ 0 & -1 \end{bmatrix} \bullet \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \le 1 \\
& {\color{red} \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix} \succeq 0}
\end{aligned}
$$

Hadamard Product

$$A \bullet B \triangleq \sum_{1 \le i,j \le n} a_{ij} \cdot b_{ij}.$$

Positive Semi-definite Matrix

Definition

An $n \times n$ symmetric matrix A is positive semi-definite if $x^T A x \geq 0$ for every vector x. We write it as

$$A \succeq 0.$$

Positive Semi-definite Matrix

Definition

An $n \times n$ symmetric matrix A is positive semi-definite if $x^T A x \geq 0$ for every vector x. We write it as

$$A \succeq 0.$$

Linear Programming

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & a_i^T x \leq b_i, \quad \forall i \in [m] \\ & x_j \geq 0, \quad \forall j \in [n] \end{aligned}$$

Positive Semi-definite Matrix

Definition

An $n \times n$ symmetric matrix A is positive semi-definite if $x^T A x \geq 0$ for every vector x. We write it as

$$A \succeq 0.$$

Linear Programming

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & a_i^T x \leq b_i, \quad \forall i \in [m] \\ & x_j \geq 0, \quad \forall j \in [n] \end{aligned}$$

PSD Programming

$$\begin{aligned} \max \quad & C \bullet X \\ \text{s.t.} \quad & A_i \bullet X \leq b_i, \quad \forall i \in [m] \\ & X \succeq 0 \end{aligned}$$

Property of PSD Matrix

Property of PSD Matrix

Theorem

For an $n \times n$ symmetric matrix, the followings are equivalent

1. $A \succeq 0$;
2. A has n non-negative eigenvalues;
3. $A = V^T V$ for some $n \times n$ matrix $V = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}$.

Property of PSD Matrix

Theorem

For an $n \times n$ symmetric matrix, the followings are equivalent

1. $A \succeq 0$;
2. A has n non-negative eigenvalues;
3. $A = V^T V$ for some $n \times n$ matrix $V = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}$.

We now prove the theorem using spectral theorem for symmetric matrices.

Vector Programming

Vector Programming

If we write $X = V^T V$ for some $V = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}$, then

PSD Programming

$$\begin{aligned} \max \quad & C \bullet X \\ \text{s.t.} \quad & A_k \bullet X \le b_k, \quad \forall k \in [m] \\ & X \succeq 0 \end{aligned}$$

Vector Programming

$$\begin{aligned} \max \quad & \sum_{1 \le i,j \le n} c(i,j) \cdot \mathbf{v}_i^T \mathbf{v}_j \\ \text{s.t.} \quad & \sum_{1 \le i,j \le n} a_k(i,j) \cdot \mathbf{v}_i^T \mathbf{v}_j \le b_k, \quad \forall k \in [m] \\ & \mathbf{v}_i \in \mathbb{R}^n, \quad \forall i \in [n] \end{aligned}$$

Back to MaxCut

Back to MaxCut

It is easy to model MaxCut as the following quadratic programming.

Back to MaxCut

It is easy to model MaxCut as the following quadratic programming.

$$
\begin{aligned}
\max \quad & \frac{1}{2} \sum_{e=\{u,v\} \in E} (1 - x_u x_v) \\
\text{s.t.} \quad & x_u \in \{-1, 1\}, \quad \forall u \in V
\end{aligned}
$$

Back to MaxCut

It is easy to model MaxCut as the following quadratic programming.

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{e=\{u,v\}\in E} (1 - x_u x_v) \\ \text{s.t.} \quad & x_u \in \{-1, 1\}, \quad \forall u \in V \end{aligned}$$

We relax it to a vector programming, which is equivalent to an SDP.

Back to MaxCut

It is easy to model MaxCut as the following quadratic programming.

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{e=\{u,v\}\in E} (1 - x_u x_v) \\ \text{s.t.} \quad & x_u \in \{-1, 1\}, \quad \forall u \in V \end{aligned}$$

We relax it to a vector programming, which is equivalent to an SDP.

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{e=\{u,v\}\in E} \left(1 - \mathbf{w}_u^T \mathbf{w}_v\right) \\ \text{s.t.} \quad & \mathbf{w}_u \in \mathbb{R}^n, \quad \forall u \in V \\ & \|\mathbf{w}_u\|_2 = 1, \quad \forall u \in V \end{aligned}$$

Back to MaxCut

It is easy to model MaxCut as the following quadratic programming.

$$
\begin{aligned}
\max \quad & \frac{1}{2} \sum_{e=\{u,v\} \in E} (1 - x_u x_v) \\
\text{s.t.} \quad & x_u \in \{-1, 1\}, \quad \forall u \in V
\end{aligned}
$$

We relax it to a vector programming, which is equivalent to an SDP.

$$
\begin{aligned}
\max \quad & \frac{1}{2} \sum_{e=\{u,v\} \in E} \left(1 - \mathbf{w}_u^T \mathbf{w}_v\right) \\
\text{s.t.} \quad & \mathbf{w}_u \in \mathbb{R}^n, \quad \forall u \in V \\
& \|\mathbf{w}_u\|_2 = 1, \quad \forall u \in V
\end{aligned}
$$

Rounding

Let $\{\mathbf{w}_u\}$ be an optimal solution of the SDP.

Rounding

Let $\{\mathbf{w}_u\}$ be an optimal solution of the SDP.

Choose a random hyperplane to divide the vectors into two classes...

Rounding

Let $\{\mathbf{w}_u\}$ be an optimal solution of the SDP.

Choose a random hyperplane to divide the vectors into two classes...

Next week: How to implement the rounding? How to analyze the performance?

Advanced Algorithms (IV)

Chihao Zhang

Shanghai Jiao Tong University

Mar. 18, 2019

Review

Review

MaxCut

Input: An undirected graph $G = (V, E)$.

Problem: A set $S \subseteq V$ that maximizes $\left|E(S, \overline{S})\right|$.

Review

MaxCut

Input: An undirected graph $G = (V, E)$.

Problem: A set $S \subseteq V$ that maximizes $\left|E(S, \overline{S})\right|$.

Integer Program

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{e=\{u,v\}\in E} (1 - x_u x_v) \\ \text{s.t.} \quad & x_u \in \{-1, 1\}, \quad \forall u \in V \end{aligned}$$

Review

MaxCut

Input: An undirected graph $G = (V, E)$.
Problem: A set $S \subseteq V$ that maximizes $\left|E(S, \bar{S})\right|$.

Integer Program

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{e=\{u,v\}\in E} (1 - x_u x_v) \\ \text{s.t.} \quad & x_u \in \{-1, 1\}, \quad \forall u \in V \end{aligned}$$

Vector Program

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{e=\{u,v\}\in E} \left(1 - \mathbf{w}_u^T \mathbf{w}_v\right) \\ \text{s.t.} \quad & \mathbf{w}_u \in \mathbb{R}^n, \quad \forall u \in V \\ & \mathbf{w}_u^T \mathbf{w}_u = 1, \quad \forall u \in V \end{aligned}$$

$$
\begin{aligned}
\max \quad & \frac{1}{2}\sum_{e=\{u,v\}\in E}\left(1-\mathbf{w}_u^T\mathbf{w}_v\right)\\
\text{s.t.} \quad & \mathbf{w}_u \in \mathbb{R}^n, \quad \forall u \in V\\
& \mathbf{w}_u^T\mathbf{w}_u = 1, \quad \forall u \in V
\end{aligned}
$$

is equivalent to a positive semi-definite programming (solvable in polynomial-time)

$$
\begin{aligned}
\max \quad & \frac{1}{2} \sum_{e=\{u,v\}\in E} \left(1 - \mathbf{w}_u^T \mathbf{w}_v\right) \\
\text{s.t.} \quad & \mathbf{w}_u \in \mathbb{R}^n, \quad \forall u \in V \\
& \mathbf{w}_u^T \mathbf{w}_u = 1, \quad \forall u \in V
\end{aligned}
$$

is equivalent to a positive semi-definite programming (solvable in polynomial-time)

Let $\{\widehat{\mathbf{w}_v}\}_{v\in V}$ be an optimal solution.

$$
\begin{aligned}
\max \quad & \frac{1}{2} \sum_{e=\{u,v\}\in E} \left(1 - \mathbf{w}_u^T \mathbf{w}_v\right) \\
\text{s.t.} \quad & \mathbf{w}_u \in \mathbb{R}^n, \quad \forall u \in V \\
& \mathbf{w}_u^T \mathbf{w}_u = 1, \quad \forall u \in V
\end{aligned}
$$

is equivalent to a positive semi-definite programming (solvable in polynomial-time)

Let $\{\widehat{\mathbf{w}_v}\}_{v\in V}$ be an optimal solution.

Task: Round $\{\widehat{\mathbf{w}_v}\}_{v\in V}$ to a cut

Goemans–Williamson Rounding

Goemans–Williamson Rounding

1. Pick a random hyperplane crossing the origin;
2. The plane separates V into two sets.

Goemans–Williamson Rounding

1. Pick a random hyperplane crossing the origin;
2. The plane separates V into two sets.

Implementation

1. Choose a vector $\mathbf{r} = (r_1, \ldots, r_n)$ where each $r_i \sim \mathcal{N}(0, 1)$ i.i.d.
2. Let $S \triangleq \left\{u \in V : \mathbf{r}^T \widehat{\mathbf{w}_u} \geq 0\right\}$.

Analysis

Analysis

Proposition

$\frac{\mathbf{r}}{\|r\|}$ is a point on S^{n-1} uniformly at random.

Analysis

Proposition

$\frac{\mathbf{r}}{\|r\|}$ is a point on S^{n-1} uniformly at random.

Proposition

An edge $\{u, v\} \in E$ is separated with probability $\frac{1}{\pi}\arccos(\widehat{\mathbf{w}_u}^T\widehat{\mathbf{w}_v})$.

Analysis

Proposition

$\frac{\mathbf{r}}{\|r\|}$ is a point on S^{n-1} uniformly at random.

Proposition

An edge $\{u, v\} \in E$ is separated with probability $\frac{1}{\pi}\arccos(\widehat{\mathbf{w}_u}^T\widehat{\mathbf{w}_v})$.

Proposition

Random hyperplane rounding is a 0.878-approximation of MaxCut.

Quadratic Program

Quadratic Program

We try to apply Goemans–Williamson rounding to general quadratic programs.

Quadratic Program

We try to apply Goemans–Williamson rounding to general quadratic programs.

Quadratic Program

$$
\begin{aligned}
\max \quad & \sum_{1 \le i,j \le n} a_{i,j} x_i x_j \\
\text{s.t.} \quad & x_i \in \{-1, +1\}, \quad i = 1, \ldots, n.
\end{aligned}
$$

Quadratic Program

We try to apply Goemans–Williamson rounding to general quadratic programs.

Quadratic Program

$$\begin{aligned} \max \quad & \sum_{1 \le i,j \le n} a_{i,j} x_i x_j \\ \text{s.t.} \quad & x_i \in \{-1, +1\}, \quad i = 1, \dots, n. \end{aligned}$$

We assume $A = (a_{i,j})_{1 \le i,j \le n}$ is positive semi-definite.

Rounding

We simply follow G-W...

Rounding

We simply follow G-W...

Vector Program

$$\begin{aligned} \max \quad & \sum_{1 \le i,j \le n} a_{i,j} \mathbf{v}_i^T \mathbf{v}_j \\ \text{s.t.} \quad & \mathbf{v}_i \in \mathbb{R}^n, \quad i = 1, \ldots, n. \end{aligned}$$

Rounding

We simply follow G-W...

Vector Program

$$\begin{aligned} \max \quad & \sum_{1 \le i,j \le n} a_{i,j} \mathbf{v}_i^T \mathbf{v}_j \\ \text{s.t.} \quad & \mathbf{v}_i \in \mathbb{R}^n, \quad i = 1, \ldots, n. \end{aligned}$$

1. Compute $\{\widehat{\mathbf{v}}_i\}_{1 \le i \le n}$.
2. Pick a vector $\mathbf{r}$ u.a.r on S^{n-1}.
3. $\hat{x}_i = 1$ if $\widehat{\mathbf{v}}_i{}^T \mathbf{r} \ge 0$; $\hat{x}_i = -1$ otherwise.

Analysis

Analysis

Proposition

$$\mathbf{E}\left[\hat{x}_i \hat{x}_j\right] = \frac{2}{\pi}\arcsin(\hat{v}_i^T \cdot \hat{v}_j).$$

Analysis

Proposition

$$\mathbf{E}\left[\hat{x}_i\hat{x}_j\right] = \frac{2}{\pi}\arcsin(\hat{v}_i^T \cdot \hat{v}_j).$$

Proposition

Random hypergraph rounding is a $\frac{2}{\pi}$-approximation of QP.

Analysis

Proposition

$$\mathbf{E}\left[\hat{x}_i\hat{x}_j\right] = \frac{2}{\pi}\arcsin(\hat{v}_i^T \cdot \hat{v}_j).$$

Proposition

Random hypergraph rounding is a $\frac{2}{\pi}$-approximation of QP.

Proof.

Use Schur producet theorem. □

Correlation Clustering

Correlation Clustering

- Given a undirected graph $G = (V, E)$ in which each $e \in E$ has two weights $w_e^+, w_e^- \geq 0$.

Correlation Clustering

- Given a undirected graph $G = (V, E)$ in which each $e \in E$ has two weights $w_e^+, w_e^- \geq 0$.
- Find a partition $\mathcal{S} = (S_1, \ldots, S_k)$ of V.

Correlation Clustering

- Given a undirected graph $G = (V, E)$ in which each $e \in E$ has two weights $w_e^+, w_e^- \geq 0$.
- Find a partition $\mathcal{S} = (S_1, \ldots, S_k)$ of V.
- $E^+(\mathcal{S}) \triangleq$ edge in a cluster; $E^-(\mathcal{S}) \triangleq$ edges between clusters.

Correlation Clustering

- Given a undirected graph $G = (V, E)$ in which each $e \in E$ has two weights $w_e^+, w_e^- \geq 0$.
- Find a partition $\mathcal{S} = (S_1, \ldots, S_k)$ of V.
- $E^+(\mathcal{S}) \triangleq$ edge in a cluster; $E^-(\mathcal{S}) \triangleq$ edges between clusters.
- The goal is to maximize

$$\sum_{e \in E^+(\mathcal{S})} w_e^+ + \sum_{e \in E^-(\mathcal{S})} w_e^-.$$

Vector Program

Vector Program

For $1 \leq k \leq n$, let e_k be the k-th unit vector.

Vector Program

For $1 \le k \le n$, let e_k be the k-th unit vector.

$$\begin{aligned}
\max \quad & \sum_{\{u,v\}\in E} \left(w^+_{u,v}(x_u^T x_v) + w^-_{u,v}(1 - x_u^T x_v) \right) \\
\text{s.t.} \quad & x_u \in \{e_1, \ldots, e_n\}, \quad \forall u \in V.
\end{aligned}$$

Vector Program

For $1 \leq k \leq n$, let e_k be the k-th unit vector.

$$\begin{aligned} \max \quad & \sum_{\{u,v\} \in E} \left(w^+_{u,v} (x_u^T x_v) + w^-_{u,v} (1 - x_u^T x_v) \right) \\ \text{s.t.} \quad & x_u \in \{e_1, \ldots, e_n\}, \quad \forall u \in V. \end{aligned}$$

Relaxation

$$\begin{aligned} \max \quad & \sum_{\{u,v\} \in E} \left(w^+_{u,v} (x_u^T x_v) + w^-_{u,v} (1 - x_u^T x_v) \right) \\ \text{s.t.} \quad & x_v^T x_v = 1, \quad \forall v \in V, \\ & x_u^T x_v \geq 0, \quad \forall u, v \in V, \\ & x_u \in \mathbb{R}^n, \quad \forall u \in V. \end{aligned}$$

Rounding

Rounding

Follow G-W and choose two hyperplanes..

Rounding

Follow G-W and choose two hyperplanes..

We always obtain at most four clusters.

Rounding

Follow G-W and choose two hyperplanes..

We always obtain at most four clusters.

Proposition

Two random hyperplane rounding is a $\frac{3}{4}$-approximation for correlation clustering.