Approximating Max-Cut through Semidefinite Programming

K. Subramani[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

April 15, 2014

Outline

Outline

The Max-Cut Problem

The Max-Cut Problem

Definition

The Max-Cut Problem

Definition

Given an undirected, weighted graph $G = \langle V, E, \mathbf{w} \rangle$, with $\mathbf{w}$ assigning non-negative integral weights to the edges of E,

The Max-Cut Problem

Definition

Given an undirected, weighted graph $G = \langle V, E, \mathbf{w} \rangle$, with $\mathbf{w}$ assigning non-negative integral weights to the edges of E, a *cut* S, $S \subseteq V$ partitions V into two sets S and $\bar{S}$.

The Max-Cut Problem

Definition

Given an undirected, weighted graph $G = \langle V, E, \mathbf{w} \rangle$, with $\mathbf{w}$ assigning non-negative integral weights to the edges of E, a *cut* S, $S \subseteq V$ partitions V into two sets S and $\bar{S}$.

An edge $(u, v) \in E$ belongs to the cut $(S, \bar{S})$ if and only if exactly one of u and v belongs to S.

The Max-Cut Problem

Definition

Given an undirected, weighted graph $G = \langle V, E, \mathbf{w} \rangle$, with $\mathbf{w}$ assigning non-negative integral weights to the edges of E, a *cut* S, $S \subseteq V$ partitions V into two sets S and $\bar{S}$.

An edge $(u, v) \in E$ belongs to the cut $(S, \bar{S})$ if and only if exactly one of u and v belongs to S.

The weight of a cut $w(S, \bar{S})$ is defined as:

The Max-Cut Problem

Definition

Given an undirected, weighted graph $G = \langle V, E, \mathbf{w} \rangle$, with $\mathbf{w}$ assigning non-negative integral weights to the edges of E, a *cut* S, $S \subseteq V$ partitions V into two sets S and $\bar{S}$.

An edge $(u, v) \in E$ belongs to the cut $(S, \bar{S})$ if and only if exactly one of u and v belongs to S.

The weight of a cut $w(S, \bar{S})$ is defined as:

$$w(S, \bar{S}) =$$

The Max-Cut Problem

Definition

Given an undirected, weighted graph $G = \langle V, E, \mathbf{w} \rangle$, with $\mathbf{w}$ assigning non-negative integral weights to the edges of E, a *cut* S, $S \subseteq V$ partitions V into two sets S and $\bar{S}$.

An edge $(u, v) \in E$ belongs to the cut $(S, \bar{S})$ if and only if exactly one of u and v belongs to S.

The weight of a cut $w(S, \bar{S})$ is defined as:

$$w(S, \bar{S}) = \sum_{u \in S, v \in \bar{S}} w(u, v).$$

The Max-Cut Problem

Definition

Given an undirected, weighted graph $G = \langle V, E, \mathbf{w} \rangle$, with $\mathbf{w}$ assigning non-negative integral weights to the edges of E, a *cut* S, $S \subseteq V$ partitions V into two sets S and $\bar{S}$.

An edge $(u, v) \in E$ belongs to the cut $(S, \bar{S})$ if and only if exactly one of u and v belongs to S.

The weight of a cut $w(S, \bar{S})$ is defined as:

$$w(S, \bar{S}) = \sum_{u \in S, v \in \bar{S}} w(u, v).$$

The Max-Cut problem is concerned with finding the cut of maximum weight in G.

The IP-LP Approach

The IP-LP Approach

IP formulation

The IP-LP Approach

IP formulation

1. Let y_{ij} be a variable associated with edge e_{ij}.

The IP-LP Approach

IP formulation

1. Let y_{ij} be a variable associated with edge e_{ij}.
   $y_{ij} = 1$ means that e_{ij} is in the cut.

The IP-LP Approach

IP formulation

1. Let y_{ij} be a variable associated with edge e_{ij}.
   $y_{ij} = 1$ means that e_{ij} is in the cut.
2. Let x_i be a variable associated with each vertex v_i.

The IP-LP Approach

IP formulation

1. Let y_{ij} be a variable associated with edge e_{ij}.
   $y_{ij} = 1$ means that e_{ij} is in the cut.
2. Let x_i be a variable associated with each vertex v_i.
3. Consider the following "natural" integer programming formulation for Max-Cut.

The IP-LP Approach

IP formulation

1. Let y_{ij} be a variable associated with edge e_{ij}.
   $y_{ij} = 1$ means that e_{ij} is in the cut.
2. Let x_i be a variable associated with each vertex v_i.
3. Consider the following "natural" integer programming formulation for Max-Cut.

$$\max \sum_{(i,j)\in E} w_{ij} \cdot y_{ij}$$

The IP-LP Approach

IP formulation

1. Let y_{ij} be a variable associated with edge e_{ij}.
   $y_{ij} = 1$ means that e_{ij} is in the cut.
2. Let x_i be a variable associated with each vertex v_i.
3. Consider the following "natural" integer programming formulation for Max-Cut.

$$\max \sum_{(i,j)\in E} w_{ij} \cdot y_{ij}$$

$$\text{subject to} \qquad y_{ij} \leq 1 - \frac{x_i + x_j}{2}, \qquad \text{for every edge } e_{ij}$$

The IP-LP Approach

IP formulation

1. Let y_{ij} be a variable associated with edge e_{ij}.
   $y_{ij} = 1$ means that e_{ij} is in the cut.
2. Let x_i be a variable associated with each vertex v_i.
3. Consider the following "natural" integer programming formulation for Max-Cut.

$$\begin{aligned} & \max \sum_{(i,j)\in E} w_{ij} \cdot y_{ij} \\ \text{subject to} \quad & y_{ij} \leq 1 - \frac{x_i + x_j}{2}, && \text{for every edge } e_{ij} \\ & y_{ij} \leq 1 + \frac{x_i + x_j}{2}, && \text{for every edge } e_{ij} \end{aligned}$$

The IP-LP Approach

IP formulation

1. Let y_{ij} be a variable associated with edge e_{ij}.
   $y_{ij} = 1$ means that e_{ij} is in the cut.
2. Let x_i be a variable associated with each vertex v_i.
3. Consider the following "natural" integer programming formulation for Max-Cut.

$$\begin{aligned} & \max \sum_{(i,j)\in E} w_{ij} \cdot y_{ij} & \\ \text{subject to} \quad & y_{ij} \leq 1 - \frac{x_i + x_j}{2}, & \text{for every edge } e_{ij} \\ & y_{ij} \leq 1 + \frac{x_i + x_j}{2}, & \text{for every edge } e_{ij} \\ & x_i \in \{-1, 1\}, & \text{for every vertex } v_i \end{aligned}$$

Analyzing the LP relaxation

Analyzing the LP relaxation

LP relaxation

Analyzing the LP relaxation

LP relaxation

1. In the LP relaxation, we set $-1 \leq x_i \leq 1$, for each x_i.

Analyzing the LP relaxation

LP relaxation

1. In the LP relaxation, we set $-1 \leq x_i \leq 1$, for each x_i.
2. However, in this case, the optimum is $|E|$, since we can simply set $x_i = 0$ for each vertex, which permits every edge to be selected!

Analyzing the LP relaxation

LP relaxation

1. In the LP relaxation, we set $-1 \leq x_i \leq 1$, for each x_i.
2. However, in this case, the optimum is $|E|$, since we can simply set $x_i = 0$ for each vertex, which permits every edge to be selected!
3. Thus, the relaxation is not useful, from the perspective of bounding the error of an approximate solution.

The Quadratically constrained programming approach

The Quadratically constrained programming approach

Modeling

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.
2. The partition $(S, \bar{S})$ is defined as follows:

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.
2. The partition $(S, \bar{S})$ is defined as follows: $S = \{v_i : y_i = 1\}$ and $\bar{S} = \{v_i : y_i = -1\}$.

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.
2. The partition $(S, \bar{S})$ is defined as follows: $S = \{v_i : y_i = 1\}$ and $\bar{S} = \{v_i : y_i = -1\}$.
3. The following quadratically constrained program captures Max-Cut:

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.
2. The partition $(S, \bar{S})$ is defined as follows: $S = \{v_i : y_i = 1\}$ and $\bar{S} = \{v_i : y_i = -1\}$.
3. The following quadratically constrained program captures Max-Cut:

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.
2. The partition $(S, \bar{S})$ is defined as follows: $S = \{v_i : y_i = 1\}$ and $\bar{S} = \{v_i : y_i = -1\}$.
3. The following quadratically constrained program captures Max-Cut:

$$\max \frac{1}{2} \cdot \sum_{1 \leq i < j \leq n} w_{ij} \cdot (1 - y_i \cdot y_j)$$

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.
2. The partition $(S, \bar{S})$ is defined as follows: $S = \{v_i : y_i = 1\}$ and $\bar{S} = \{v_i : y_i = -1\}$.
3. The following quadratically constrained program captures Max-Cut:

$$\begin{aligned} & \max \ \tfrac{1}{2} \cdot \textstyle\sum_{1 \le i < j \le n} w_{ij} \cdot (1 - y_i \cdot y_j) \\ \text{subject to} \quad & y_i^2 = 1, \qquad v_i \in V \end{aligned}$$

The Quadratically constrained programming approach

Modeling

1. Let y_i be an indicator variable for vertex v_i; $y_i \in \{+1, -1\}$.
2. The partition $(S, \bar{S})$ is defined as follows: $S = \{v_i : y_i = 1\}$ and $\bar{S} = \{v_i : y_i = -1\}$.
3. The following quadratically constrained program captures Max-Cut:

$$\begin{aligned} & \max \ \tfrac{1}{2} \cdot \textstyle\sum_{1 \le i < j \le n} w_{ij} \cdot (1 - y_i \cdot y_j) & \\ \text{subject to} \quad & y_i^2 = 1, & v_i \in V \\ & y_i \in \mathbf{Z}, & v_i \in V \end{aligned}$$

The vector program relaxation

The vector program relaxation

Definition

The vector program relaxation

Definition

Let $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ denote n vector variables in $\Re^n$.

The vector program relaxation

Definition

Let $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ denote n vector variables in $\Re^n$.

A vector program is the problem of minimizing or maximizing a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, subject to linear constraints on these inner products.

The vector program relaxation

Definition

Let $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ denote n vector variables in $\Re^n$.

A vector program is the problem of minimizing or maximizing a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, subject to linear constraints on these inner products.

Vector program relaxation

The vector program relaxation

Definition

Let $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ denote n vector variables in $\Re^n$.

A vector program is the problem of minimizing or maximizing a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, subject to linear constraints on these inner products.

Vector program relaxation

$$\max \ \frac{1}{2} \cdot \sum_{1 \leq i < j \leq n} w_{ij} \cdot (1 - \mathbf{v_i} \cdot \mathbf{v_j})$$

The vector program relaxation

Definition

Let $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ denote n vector variables in $\Re^n$.

A vector program is the problem of minimizing or maximizing a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, subject to linear constraints on these inner products.

Vector program relaxation

$$\begin{aligned} & \max \ \tfrac{1}{2} \cdot \textstyle\sum_{1 \le i < j \le n} w_{ij} \cdot (1 - \mathbf{v_i} \cdot \mathbf{v_j}) \\ \text{subject to} \quad & \mathbf{v_i} \cdot \mathbf{v_i} = 1, \qquad v_i \in V \end{aligned}$$

The vector program relaxation

Definition

Let $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ denote n vector variables in $\Re^n$.

A vector program is the problem of minimizing or maximizing a linear function of the inner products $\mathbf{v_i} \cdot \mathbf{v_j}$, subject to linear constraints on these inner products.

Vector program relaxation

$$\begin{aligned} & \max \ \tfrac{1}{2} \cdot \textstyle\sum_{1 \le i < j \le n} w_{ij} \cdot (1 - \mathbf{v_i} \cdot \mathbf{v_j}) & \\ \text{subject to} \quad & \mathbf{v_i} \cdot \mathbf{v_i} = 1, & v_i \in V \\ & \mathbf{v_i} \in \Re^n, & v_i \in V \end{aligned}$$

Notes on the vector program relaxation

Notes on the vector program relaxation

Note

Notes on the vector program relaxation

Note

1. *All the vectors* $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ *are constrained to lie on the n-dimensional sphere* S_{n-1}.

Notes on the vector program relaxation

Note

1. *All the vectors* $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ *are constrained to lie on the n-dimensional sphere* S_{n-1}.
2. *Any feasible solution to the quadratically constrained quadratic program yields a solution to the vector program relaxation, having the same objective function value, by setting* $\mathbf{v_i} = (y_i, 0, \ldots, 0)$.

Notes on the vector program relaxation

Note

1. *All the vectors* $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ *are constrained to lie on the n-dimensional sphere* S_{n-1}.
2. *Any feasible solution to the quadratically constrained quadratic program yields a solution to the vector program relaxation, having the same objective function value, by setting* $\mathbf{v_i} = (y_i, 0, \ldots, 0)$.
3. *Therefore, the vector program is a relaxation of the quadratically constrained quadratic program.*

Notes on the vector program relaxation

Note

1. *All the vectors* $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_n}$ *are constrained to lie on the n-dimensional sphere* S_{n-1}.
2. *Any feasible solution to the quadratically constrained quadratic program yields a solution to the vector program relaxation, having the same objective function value, by setting* $\mathbf{v_i} = (y_i, 0, \ldots, 0)$.
3. *Therefore, the vector program is a relaxation of the quadratically constrained quadratic program.*
4. *Vector programs are approximable to any desired level of accuracy in polynomial time and thus the vector program relaxation provides an upper bound on OPT for Max-Cut.*

The semidefinite program for Max-Cut

Semidefinite program relaxation

The semidefinite program for Max-Cut

Semidefinite program relaxation

$$\max \ \frac{1}{2} \cdot \sum_{1 \leq i < j \leq n} w_{ij} \cdot (1 - y_i \cdot y_j)$$

The semidefinite program for Max-Cut

Semidefinite program relaxation

$$\begin{aligned} \max \ & \tfrac{1}{2} \cdot \textstyle\sum_{1 \le i < j \le n} w_{ij} \cdot (1 - y_i \cdot y_j) \\ \text{subject to} \quad & y_i^2 = 1, \qquad v_i \in V \end{aligned}$$

The semidefinite program for Max-Cut

Semidefinite program relaxation

$$\begin{aligned} & \max \ \tfrac{1}{2} \cdot \textstyle\sum_{1 \le i < j \le n} w_{ij} \cdot (1 - y_i \cdot y_j) & \\ \text{subject to} \quad & y_i^2 = 1, & v_i \in V \\ & \mathbf{Y} \succeq \mathbf{0}, & \end{aligned}$$

The semidefinite program for Max-Cut

Semidefinite program relaxation

$$\begin{aligned} & \max \ \tfrac{1}{2} \cdot \textstyle\sum_{1 \le i < j \le n} w_{ij} \cdot (1 - y_i \cdot y_j) & \\ \text{subject to} \quad & y_i^2 = 1, & v_i \in V \\ & \mathbf{Y} \succeq \mathbf{0}, & \\ & \mathbf{Y} \in \mathbf{M_n} & \end{aligned}$$

The Semidefinite programming algorithm

The Max-Cut Algorithm

The Max-Cut Algorithm

Randomized Rounding Algorithm

The Max-Cut Algorithm

Randomized Rounding Algorithm

1. Solve the semidefinite program corresponding to the vector program relaxation of the Max-Cut QCQIP optimally.

The Max-Cut Algorithm

Randomized Rounding Algorithm

1. Solve the semidefinite program corresponding to the vector program relaxation of the Max-Cut QCQIP optimally.
2. Convert the solution into a solution for the corresponding vector program.

The Max-Cut Algorithm

Randomized Rounding Algorithm

1. Solve the semidefinite program corresponding to the vector program relaxation of the Max-Cut QCQIP optimally.
2. Convert the solution into a solution for the corresponding vector program. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution.

The Max-Cut Algorithm

Randomized Rounding Algorithm

1. Solve the semidefinite program corresponding to the vector program relaxation of the Max-Cut QCQIP optimally.
2. Convert the solution into a solution for the corresponding vector program. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution.
3. Pick $\mathbf{r}$ to be a uniformly distributed vector on the unit sphere S_{n-1}.

The Max-Cut Algorithm

Randomized Rounding Algorithm

1. Solve the semidefinite program corresponding to the vector program relaxation of the Max-Cut QCQIP optimally.
2. Convert the solution into a solution for the corresponding vector program. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution.
3. Pick $\mathbf{r}$ to be a uniformly distributed vector on the unit sphere S_{n-1}.
4. Let $S = \{v_i \ : \ \mathbf{a_i} \cdot \mathbf{r} \geq 0\}$.

The Max-Cut Algorithm

Randomized Rounding Algorithm

1. Solve the semidefinite program corresponding to the vector program relaxation of the Max-Cut QCQIP optimally.
2. Convert the solution into a solution for the corresponding vector program. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution.
3. Pick $\mathbf{r}$ to be a uniformly distributed vector on the unit sphere S_{n-1}.
4. Let $S = \{v_i \ : \ \mathbf{a_i} \cdot \mathbf{r} \geq 0\}$.
5. $\bar{S} = V - S$.

Analysis

Analysis

Bounding the error

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_V denote the optimal value.

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_V denote the optimal value.
2. Let θ_{ij} denote the angle between vectors $\mathbf{a_i}$ and $\mathbf{a_j}$.

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_V denote the optimal value.
2. Let θ_{ij} denote the angle between vectors $\mathbf{a_i}$ and $\mathbf{a_j}$.
3. The contribution of this pair of vectors to OPT_V is:

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_v denote the optimal value.
2. Let θ_{ij} denote the angle between vectors $\mathbf{a_i}$ and $\mathbf{a_j}$.
3. The contribution of this pair of vectors to OPT_v is: $\frac{w_{ij}}{2} \cdot (1 - \cos\theta_{ij})$.

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_v denote the optimal value.
2. Let θ_{ij} denote the angle between vectors $\mathbf{a_i}$ and $\mathbf{a_j}$.
3. The contribution of this pair of vectors to OPT_v is: $\frac{w_{ij}}{2} \cdot (1 - \cos\theta_{ij})$.
4. $\mathbf{Pr}[\mathbf{a_i}$ is separated from $\mathbf{a_j}] =$

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_V denote the optimal value.
2. Let θ_{ij} denote the angle between vectors $\mathbf{a_i}$ and $\mathbf{a_j}$.
3. The contribution of this pair of vectors to OPT_V is: $\frac{w_{ij}}{2} \cdot (1 - \cos\theta_{ij})$.
4. $\mathbf{Pr}[\mathbf{a_i}$ is separated from $\mathbf{a_j}] = \frac{\theta_{ij}}{\pi}$.

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_V denote the optimal value.
2. Let θ_{ij} denote the angle between vectors $\mathbf{a_i}$ and $\mathbf{a_j}$.
3. The contribution of this pair of vectors to OPT_V is: $\frac{w_{ij}}{2} \cdot (1 - \cos\theta_{ij})$.
4. $\mathbf{Pr}[\mathbf{a_i}$ is separated from $\mathbf{a_j}] = \frac{\theta_{ij}}{\pi}$.
5. Let W be the random variable denoting the weight of the edges in the cut.

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_V denote the optimal value.
2. Let θ_{ij} denote the angle between vectors $\mathbf{a_i}$ and $\mathbf{a_j}$.
3. The contribution of this pair of vectors to OPT_V is: $\frac{w_{ij}}{2} \cdot (1 - \cos\theta_{ij})$.
4. $\mathbf{Pr}[\mathbf{a_i}$ is separated from $\mathbf{a_j}] = \frac{\theta_{ij}}{\pi}$.
5. Let W be the random variable denoting the weight of the edges in the cut.
6. Let $\alpha = \frac{2}{\pi} \min_{0<\theta\leq\pi} \frac{\theta}{1-\cos\theta}$.

Analysis

Bounding the error

1. Let $\mathbf{a_1}, \mathbf{a_2}, \ldots \mathbf{a_n}$ denote the optimal solution and let OPT_V denote the optimal value.
2. Let θ_{ij} denote the angle between vectors $\mathbf{a_i}$ and $\mathbf{a_j}$.
3. The contribution of this pair of vectors to OPT_V is: $\frac{w_{ij}}{2} \cdot (1 - \cos\theta_{ij})$.
4. $\mathbf{Pr}[\mathbf{a_i}$ is separated from $\mathbf{a_j}] = \frac{\theta_{ij}}{\pi}$.
5. Let W be the random variable denoting the weight of the edges in the cut.
6. Let $\alpha = \frac{2}{\pi} \min_{0<\theta\leq\pi} \frac{\theta}{1-\cos\theta}$. It is not hard to see that $\alpha > 0.87856$.

Proof of separation probability

Proof of separation probability

Proof

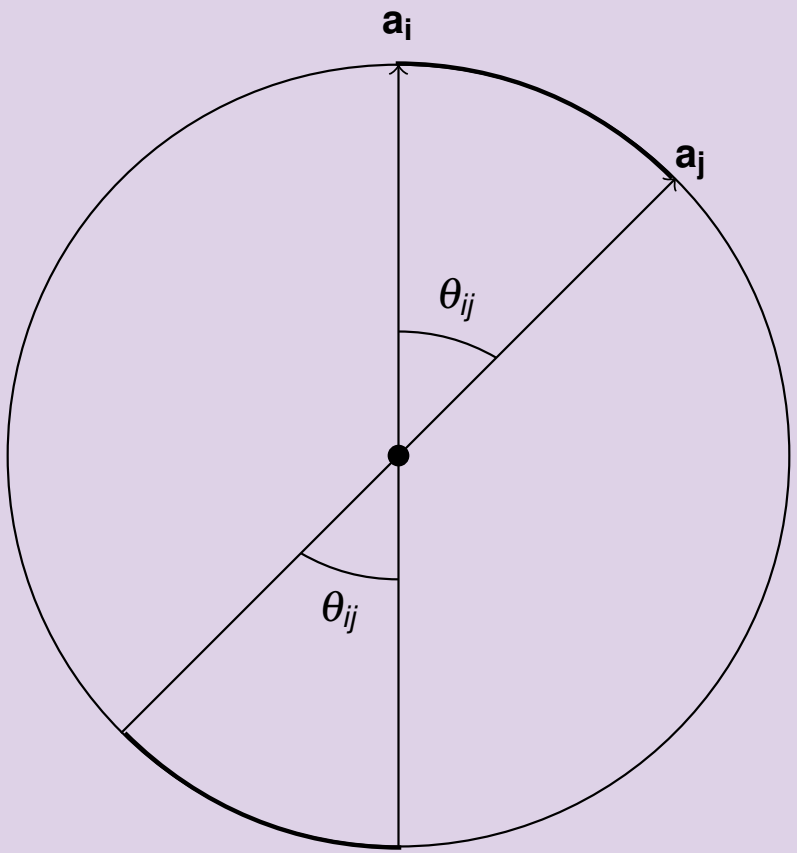

Figure: Proof by picture

Proof of separation probability

Proof

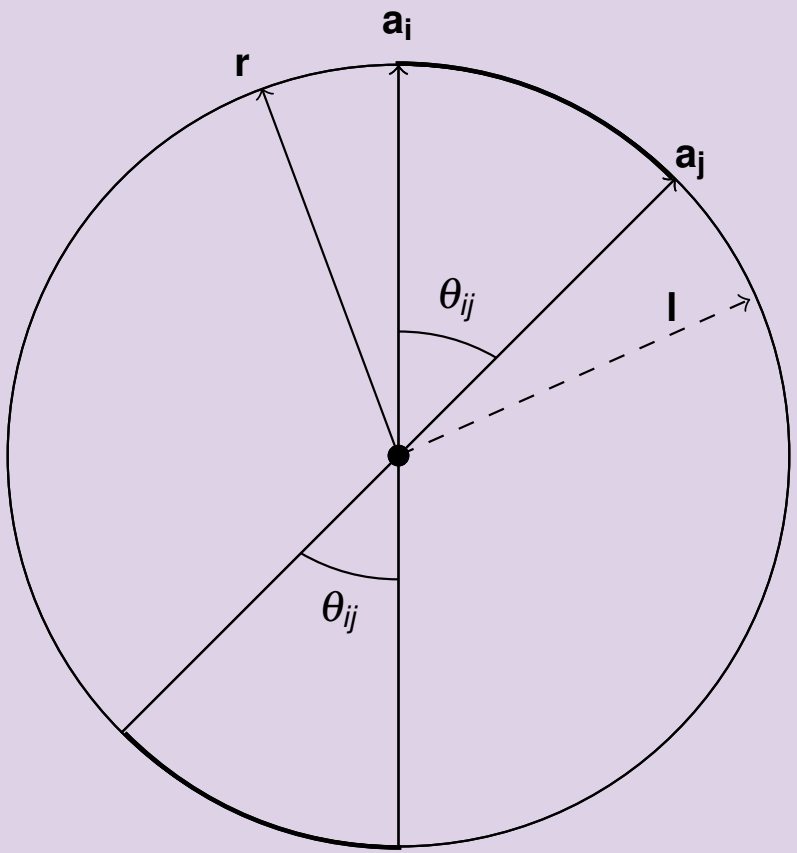

Figure: Proof by picture

Analysis (contd.)

Analysis (contd.)

Lemma

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have,

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\mathbf{E}[W] \quad =$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\mathbf{E}[W] \quad = \quad \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }]$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned} \mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\ &= \end{aligned}$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned} \mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\ &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[\mathbf{a_i} \text{ and } \mathbf{a_j} \text{ are separated by } \mathbf{r}] \end{aligned}$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned}
\mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[\mathbf{a_i} \text{ and } \mathbf{a_j} \text{ are separated by } \mathbf{r}] \\
&=
\end{aligned}$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned}
\mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[\mathbf{a_i} \text{ and } \mathbf{a_j} \text{ are separated by } \mathbf{r}] \\
&=
\end{aligned}$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned}
\mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[\mathbf{a_i} \text{ and } \mathbf{a_j} \text{ are separated by } \mathbf{r}] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \frac{\theta_{ij}}{\pi}
\end{aligned}$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned}
\mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[\mathbf{a_i} \text{ and } \mathbf{a_j} \text{ are separated by } \mathbf{r}] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \frac{\theta_{ij}}{\pi} \\
&\geq
\end{aligned}$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned}
\mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[\mathbf{a_i} \text{ and } \mathbf{a_j} \text{ are separated by } \mathbf{r}] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \frac{\theta_{ij}}{\pi} \\
&\geq \alpha \cdot \sum_{1 \leq i < j \leq n} \frac{1}{2} \cdot w_{ij} \cdot (1 - \cos\theta_{ij})
\end{aligned}$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned}
\mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[\mathbf{a_i} \text{ and } \mathbf{a_j} \text{ are separated by } \mathbf{r}] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \frac{\theta_{ij}}{\pi} \\
&\geq \alpha \cdot \sum_{1 \leq i < j \leq n} \frac{1}{2} \cdot w_{ij} \cdot (1 - \cos\theta_{ij}) \\
&= \alpha \cdot OPT_v
\end{aligned}$$

Analysis (contd.)

Lemma

$\mathbf{E}[W] \geq \alpha \cdot OPT_v$ *and hence* $\mathbf{E}[W] \geq \alpha \cdot OPT$.

Proof.

1. Observe that, for any θ, $0 < \theta \leq \pi$, we must have, $\frac{\theta}{\pi} \geq \alpha \cdot (\frac{1-\cos\theta}{2})$.
2. It follows that,

$$\begin{aligned}
\mathbf{E}[W] &= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[v_i \text{ and } v_j \text{ are separated }] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \mathbf{Pr}[\mathbf{a_i} \text{ and } \mathbf{a_j} \text{ are separated by } \mathbf{r}] \\
&= \sum_{1 \leq i < j \leq n} w_{ij} \cdot \frac{\theta_{ij}}{\pi} \\
&\geq \alpha \cdot \sum_{1 \leq i < j \leq n} \frac{1}{2} \cdot w_{ij} \cdot (1 - \cos\theta_{ij}) \\
&= \alpha \cdot OPT_v \geq \alpha \cdot OPT.
\end{aligned}$$

□

High Probability Guarantee

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$.

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1 - \varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have,

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that,

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq$

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] =$

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq$

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq \alpha \cdot OPT_v \geq$

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq \alpha \cdot OPT_v \geq \alpha \cdot OPT \geq$

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq \alpha \cdot OPT_v \geq \alpha \cdot OPT \geq \frac{\alpha \cdot T}{2}$.

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq \alpha \cdot OPT_v \geq \alpha \cdot OPT \geq \frac{\alpha \cdot T}{2}$.
7. Therefore,

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq \alpha \cdot OPT_v \geq \alpha \cdot OPT \geq \frac{\alpha \cdot T}{2}$.
7. Therefore, $\frac{\alpha}{2} \leq a \leq 1$.

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a \cdot \varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq \alpha \cdot OPT_v \geq \alpha \cdot OPT \geq \frac{\alpha \cdot T}{2}$.
7. Therefore, $\frac{\alpha}{2} \leq a \leq 1$.
8. Hence,

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a\cdot\varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq \alpha \cdot OPT_v \geq \alpha \cdot OPT \geq \frac{\alpha \cdot T}{2}$.
7. Therefore, $\frac{\alpha}{2} \leq a \leq 1$.
8. Hence, $p \leq 1 - \frac{\varepsilon \cdot \frac{\alpha}{2}}{1+\varepsilon-\frac{\alpha}{2}}$.

High Probability Guarantee

Theorem

There is a randomized approximation algorithm for Max-Cut that achieves the approximation factor of 0.87856*.*

Proof.

1. Let T denote the sum of weights of all edges in G.
2. Define a so that $\mathbf{E}[W] = a \cdot T$. ($a$ will be decided later.)
3. Let $p = \mathbf{Pr}[W < (1-\varepsilon) \cdot a \cdot T]$. ($\varepsilon$ will be chosen later.)
4. Since $W \leq T$, we must have, $a \cdot T \leq p \cdot (1-\varepsilon) \cdot a \cdot T + (1-p) \cdot T$.
5. It follows that, $p \leq \frac{1-a}{1-a+a\cdot\varepsilon}$.
6. Now observe that,
   $T \geq \mathbf{E}[W] = a \cdot T \geq \alpha \cdot OPT_v \geq \alpha \cdot OPT \geq \frac{\alpha \cdot T}{2}$.
7. Therefore, $\frac{\alpha}{2} \leq a \leq 1$.
8. Hence, $p \leq 1 - \frac{\varepsilon \cdot \frac{\alpha}{2}}{1+\varepsilon-\frac{\alpha}{2}}$.
9. Let $c = \frac{\varepsilon \cdot \frac{\alpha}{2}}{1+\varepsilon-\frac{\alpha}{2}}$.

□

Completing the analysis

Completing the analysis

Final steps

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\mathbf{Pr}[W' \geq (1-\varepsilon) \cdot a \cdot T]$$

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\mathbf{Pr}[W' \geq (1-\varepsilon) \cdot a \cdot T] \quad \geq$$

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\mathbf{Pr}[W' \geq (1-\varepsilon) \cdot a \cdot T] \quad \geq \quad 1-(1-c)^{\frac{1}{c}}$$

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\begin{aligned} \mathbf{Pr}[W' \geq (1-\varepsilon)\cdot a \cdot T] &\geq 1-(1-c)^{\frac{1}{c}} \\ &\geq \end{aligned}$$

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\begin{aligned} \mathbf{Pr}[W' \geq (1-\varepsilon)\cdot a \cdot T] &\geq 1-(1-c)^{\frac{1}{c}} \\ &\geq 1-\frac{1}{e}. \end{aligned}$$

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\begin{aligned} \mathbf{Pr}[W' \geq (1-\varepsilon)\cdot a \cdot T] &\geq 1-(1-c)^{\frac{1}{c}} \\ &\geq 1-\frac{1}{e}. \end{aligned}$$

4. Since $a \cdot T \geq$

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\begin{aligned} \mathbf{Pr}[W' \geq (1-\varepsilon)\cdot a \cdot T] &\geq 1-(1-c)^{\frac{1}{c}} \\ &\geq 1-\frac{1}{e}. \end{aligned}$$

4. Since $a \cdot T \geq \alpha \cdot OPT >$

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\begin{aligned} \mathbf{Pr}[W' \geq (1-\varepsilon)\cdot a \cdot T] &\geq 1-(1-c)^{\frac{1}{c}} \\ &\geq 1-\frac{1}{e}. \end{aligned}$$

4. Since $a \cdot T \geq \alpha \cdot OPT > 0.87856 \cdot OPT$,

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\begin{aligned} \mathbf{Pr}[W' \geq (1-\varepsilon)\cdot a \cdot T] &\geq 1-(1-c)^{\frac{1}{c}} \\ &\geq 1-\frac{1}{e}. \end{aligned}$$

4. Since $a \cdot T \geq \alpha \cdot OPT > 0.87856 \cdot OPT$, we can pick a value of $\varepsilon > 0$, so that $(1-\varepsilon)\cdot a \cdot T \geq$

Completing the analysis

Final steps

1. Run the semidefinite algorithm and perform the randomized rounding $\frac{1}{c}$ times and output the heaviest cut.
2. Let W' be the weight of this cut.
3. It follows that,

$$\begin{aligned} \mathbf{Pr}[W' \geq (1-\varepsilon)\cdot a \cdot T] &\geq 1-(1-c)^{\frac{1}{c}} \\ &\geq 1-\frac{1}{e}. \end{aligned}$$

4. Since $a \cdot T \geq \alpha \cdot OPT > 0.87856 \cdot OPT$, we can pick a value of $\varepsilon > 0$, so that $(1-\varepsilon)\cdot a \cdot T \geq 0.87856 \cdot OPT$.

Summary

Summary

Step by step procedure

Summary

Step by step procedure

1. Formulate the problem as a Quadratically Constrained Quadratic Integer Program (QCQIP).

Summary

Step by step procedure

1. Formulate the problem as a Quadratically Constrained Quadratic Integer Program (QCQIP).
2. Relax the QCQIP to a vector program (VP).

Summary

Step by step procedure

1. Formulate the problem as a Quadratically Constrained Quadratic Integer Program (QCQIP).
2. Relax the QCQIP to a vector program (VP).
3. Replace VP with an equivalent semidefinite program (SDP).

Summary

Step by step procedure

1. Formulate the problem as a Quadratically Constrained Quadratic Integer Program (QCQIP).
2. Relax the QCQIP to a vector program (VP).
3. Replace VP with an equivalent semidefinite program (SDP).
4. Solve the SDP optimally

Summary

Step by step procedure

1. Formulate the problem as a Quadratically Constrained Quadratic Integer Program (QCQIP).
2. Relax the QCQIP to a vector program (VP).
3. Replace VP with an equivalent semidefinite program (SDP).
4. Solve the SDP optimally (Ellipsoid Method).

Summary

Step by step procedure

1. Formulate the problem as a Quadratically Constrained Quadratic Integer Program (QCQIP).
2. Relax the QCQIP to a vector program (VP).
3. Replace VP with an equivalent semidefinite program (SDP).
4. Solve the SDP optimally (Ellipsoid Method).
5. Perform randomized rounding to obtain a solution.

Summary

Step by step procedure

1. Formulate the problem as a Quadratically Constrained Quadratic Integer Program (QCQIP).
2. Relax the QCQIP to a vector program (VP).
3. Replace VP with an equivalent semidefinite program (SDP).
4. Solve the SDP optimally (Ellipsoid Method).
5. Perform randomized rounding to obtain a solution.
6. If needed, improve the guarantee of the approximation, by running the algorithm an appropriate number of times.