VIII. Approximation Algorithms: MAX-CUT Problem (Outlook)

Thomas Sauerwald

Easter 2017

UNIVERSITY OF CAMBRIDGE

Outline

Simple Algorithms for MAX-CUT

A Solution based on Semidefinite Programming

Summary

Max-Cut

MAX-CUT Problem

Max-Cut

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$

Max-Cut

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Max-Cut

Weighted MAX-CUT: Every edge $e \in E$ has a non-negative weight $w(e)$

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Max-Cut

Weighted MAX-CUT: Every edge $e \in E$ has a non-negative weight $w(e)$

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Weighted MAX-CUT: Maximize the weights of edges crossing the cut, i.e., maximize $w(S) := \sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u, v\})$

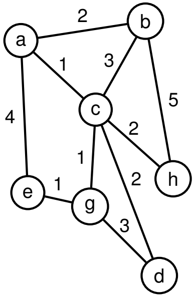
Max-Cut

Weighted MAX-CUT: Every edge $e \in E$ has a non-negative weight $w(e)$

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Weighted MAX-CUT: Maximize the weights of edges crossing the cut, i.e., maximize $w(S) := \sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u, v\})$



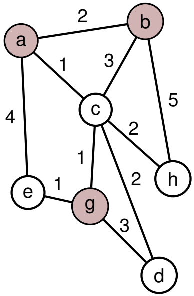
Max-Cut

Weighted MAX-CUT: Every edge $e \in E$ has a non-negative weight $w(e)$

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Weighted MAX-CUT: Maximize the weights of edges crossing the cut, i.e., maximize $w(S) := \sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u, v\})$



$S = \{a, b, g\}$

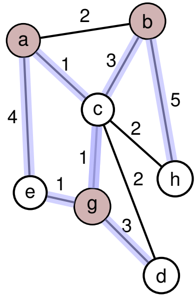
Max-Cut

Weighted MAX-CUT: Every edge $e \in E$ has a non-negative weight $w(e)$

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Weighted MAX-CUT: Maximize the weights of edges crossing the cut, i.e., maximize $w(S) := \sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u, v\})$



$S = \{a, b, g\}$

$w(S) = 18$

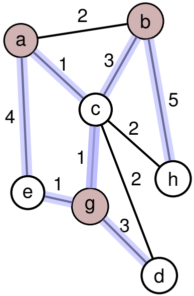
Max-Cut

Weighted MAX-CUT: Every edge $e \in E$ has a non-negative weight $w(e)$

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Weighted MAX-CUT: Maximize the weights of edges crossing the cut, i.e., maximize $w(S) := \sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u, v\})$



$S = \{a, b, g\}$

$w(S) = 18$

Applications:

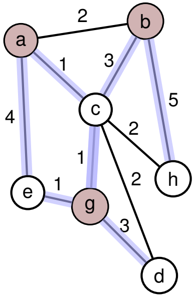
Max-Cut

Weighted MAX-CUT: Every edge $e \in E$ has a non-negative weight $w(e)$

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Weighted MAX-CUT: Maximize the weights of edges crossing the cut, i.e., maximize $w(S) := \sum_{\{u,v\} \in E(S, V\setminus S)} w(\{u, v\})$



$S = \{a, b, g\}$

$w(S) = 18$

Applications:

- cluster analysis

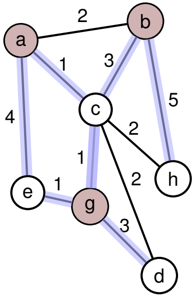
Max-Cut

Weighted MAX-CUT: Every edge $e \in E$ has a non-negative weight $w(e)$

MAX-CUT Problem

- Given: Undirected graph $G = (V, E)$
- Goal: Find a subset $S \subseteq V$ such that $|E(S, V \setminus S)|$ is maximized.

Weighted MAX-CUT: Maximize the weights of edges crossing the cut, i.e., maximize $w(S) := \sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u,v\})$



$S = \{a, b, g\}$

$w(S) = 18$

Applications:

- cluster analysis
- VLSI design

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

$$\mathbf{E}\left[\, w(S, V \setminus S) \,\right]$$

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

$$\mathbf{E}\left[\, w(S, V \setminus S) \,\right]$$

$$= \mathbf{E}\left[\sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u, v\}) \right]$$

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

$$\mathbf{E}\left[\, w(S, V \setminus S) \,\right]$$

$$= \mathbf{E}\left[\sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u, v\}) \right]$$

$$= \sum_{\{u,v\} \in E} \mathbf{Pr}\left[\, \{u \in S \cap v \in (V \setminus S)\} \cup \{u \in (V \setminus S) \cap v \in S\} \,\right] \cdot w(\{u, v\})$$

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

$$\mathbf{E}\left[\, w(S, V \setminus S)\,\right]$$

$$= \mathbf{E}\left[\sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u,v\})\right]$$

$$= \sum_{\{u,v\} \in E} \mathbf{Pr}\left[\, \{u \in S \cap v \in (V \setminus S)\} \cup \{u \in (V \setminus S) \cap v \in S\}\,\right] \cdot w(\{u,v\})$$

$$= \sum_{\{u,v\} \in E} \left(\frac{1}{4} + \frac{1}{4}\right) \cdot w(\{u,v\})$$

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

$$\begin{aligned}
&\mathbf{E}\left[\, w(S, V \setminus S) \,\right] \\
&= \mathbf{E}\left[\sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u,v\}) \right] \\
&= \sum_{\{u,v\} \in E} \mathbf{Pr}\left[\, \{u \in S \cap v \in (V \setminus S)\} \cup \{u \in (V \setminus S) \cap v \in S\} \,\right] \cdot w(\{u,v\}) \\
&= \sum_{\{u,v\} \in E} \left(\frac{1}{4} + \frac{1}{4} \right) \cdot w(\{u,v\}) \\
&= \frac{1}{2} \sum_{\{u,v\} \in E} w(\{u,v\})
\end{aligned}$$

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

$$
\begin{aligned}
&\mathbf{E}\left[\, w(S, V \setminus S) \,\right] \\
&= \mathbf{E}\left[\sum_{\{u,v\} \in E(S, V\setminus S)} w(\{u,v\}) \right] \\
&= \sum_{\{u,v\}\in E} \mathbf{Pr}\left[\, \{u \in S \cap v \in (V \setminus S)\} \cup \{u \in (V \setminus S) \cap v \in S\} \,\right] \cdot w(\{u,v\}) \\
&= \sum_{\{u,v\}\in E} \left(\frac{1}{4} + \frac{1}{4} \right) \cdot w(\{u,v\}) \\
&= \frac{1}{2} \sum_{\{u,v\}\in E} w(\{u,v\}) \geq \frac{1}{2} w^*.
\end{aligned}
$$

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

$$
\begin{aligned}
&\mathbf{E}\left[w(S, V \setminus S) \right] \\
&= \mathbf{E}\left[\sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u,v\}) \right] \\
&= \sum_{\{u,v\} \in E} \mathbf{Pr}\left[\{u \in S \cap v \in (V \setminus S)\} \cup \{u \in (V \setminus S) \cap v \in S\} \right] \cdot w(\{u,v\}) \\
&= \sum_{\{u,v\} \in E} \left(\frac{1}{4} + \frac{1}{4} \right) \cdot w(\{u,v\}) \\
&= \frac{1}{2} \sum_{\{u,v\} \in E} w(\{u,v\}) \geq \frac{1}{2} w^*. \qquad \square
\end{aligned}
$$

Random Sampling

Ex 35.4-3

Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V \setminus S$ with probability $1/2$. Then this algorithm is a randomized 2-approximation algorithm.

We could employ the same derandomisation used for MAX-3-CNF.

Proof: We express the expected weight of the random cut $(S, V \setminus S)$ as:

$$\begin{aligned}
&\mathbf{E}\left[\, w(S, V \setminus S) \,\right] \\
&= \mathbf{E}\left[\sum_{\{u,v\} \in E(S, V \setminus S)} w(\{u,v\}) \right] \\
&= \sum_{\{u,v\} \in E} \mathbf{Pr}\left[\, \{u \in S \cap v \in (V \setminus S)\} \cup \{u \in (V \setminus S) \cap v \in S\} \,\right] \cdot w(\{u,v\}) \\
&= \sum_{\{u,v\} \in E} \left(\frac{1}{4} + \frac{1}{4} \right) \cdot w(\{u,v\}) \\
&= \frac{1}{2} \sum_{\{u,v\} \in E} w(\{u,v\}) \geq \frac{1}{2} w^{*}. \qquad \square
\end{aligned}$$

Local Search

Local Search: Switch side of a vertex if it increases the cut.

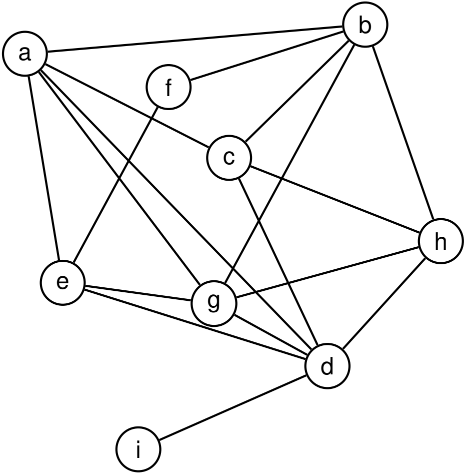
Local Search

Local Search: Switch side of a vertex if it increases the cut.

LOCAL SEARCH(G, w)

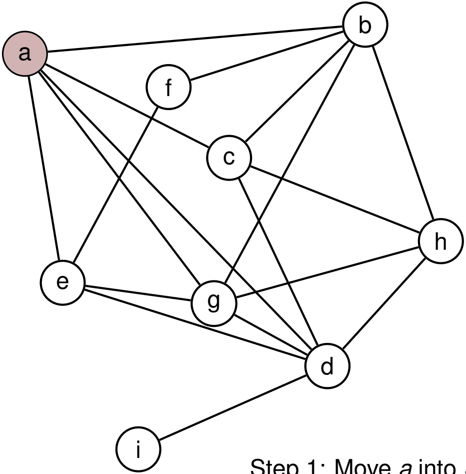
Let S be an arbitrary subset of V
do
    $flag = 0$
    if $\exists u \in S$ with $w(S \setminus \{u\}, (V \setminus S) \cup \{u\}) \geq w(S, V \setminus S)$ then
        $S = S \setminus \{u\}$
        $flag = 1$
    end if
    if $\exists u \in V \setminus S$ with $w(S \cup \{u\}, (V \setminus S) \setminus \{u\}) \geq w(S, V \setminus S)$ then
        $S = S \cup \{u\}$
        $flag = 1$
    end if
while $flag = 1$
return S

Illustration of Local Search



Cut $= 0$

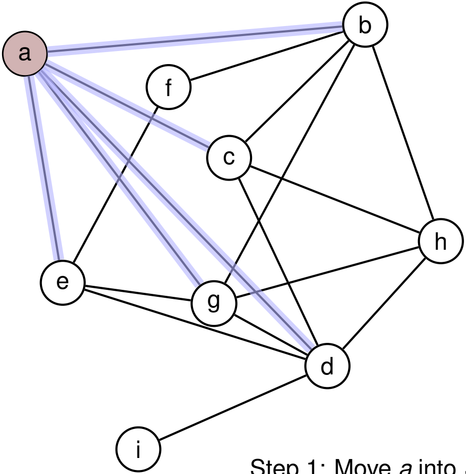
Illustration of Local Search



Step 1: Move a into S

$\text{Cut} = 0$

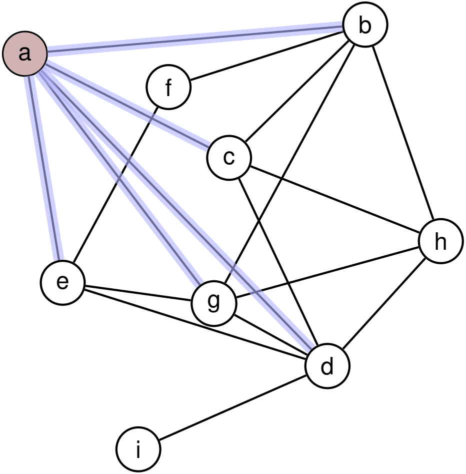
Illustration of Local Search



Step 1: Move *a* into *S*

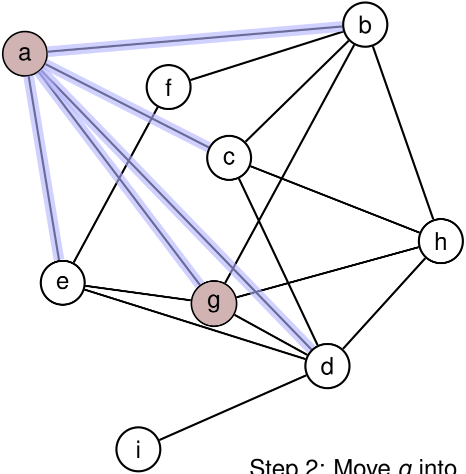
$\text{Cut} = \cancel{0}$

Illustration of Local Search



Cut $= 5$

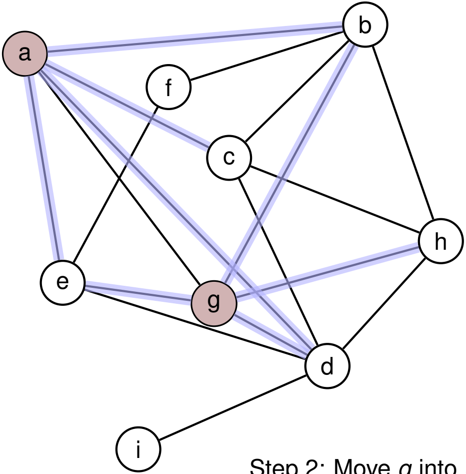
Illustration of Local Search



Step 2: Move g into S

$\text{Cut} = 5$

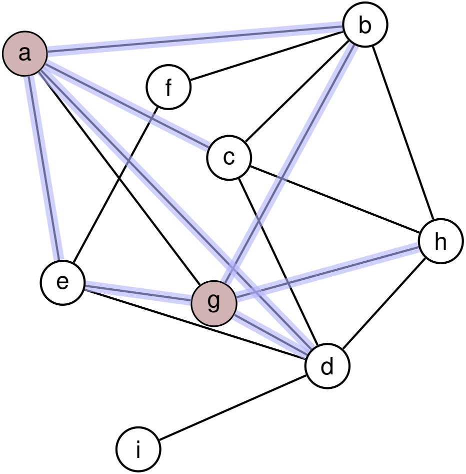
Illustration of Local Search



Step 2: Move g into S

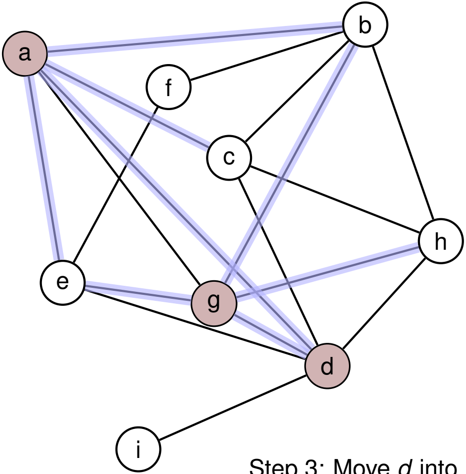
Cut $= \cancel{5}$

Illustration of Local Search



Cut = 8

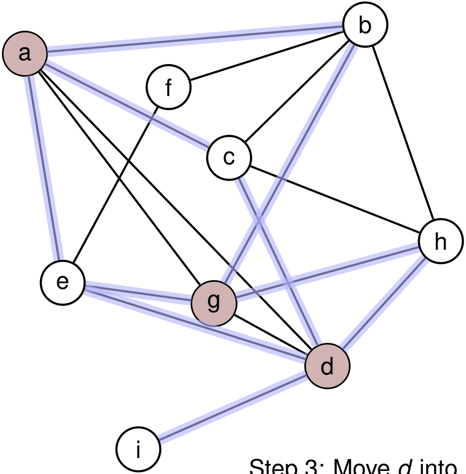
Illustration of Local Search



Step 3: Move d into S

$\text{Cut} = 8$

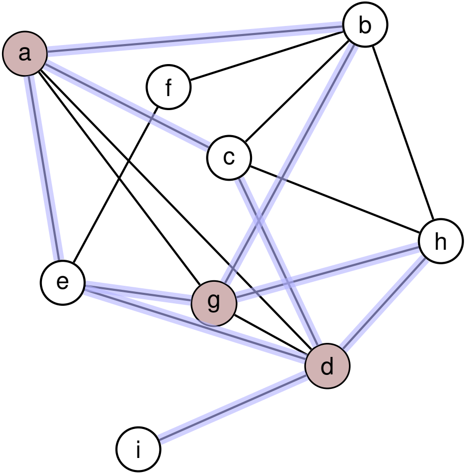
Illustration of Local Search



Step 3: Move d into S

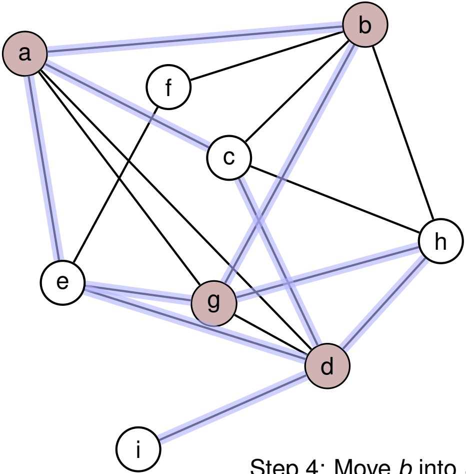
Cut $= \cancel{8}$

Illustration of Local Search



Cut $= 10$

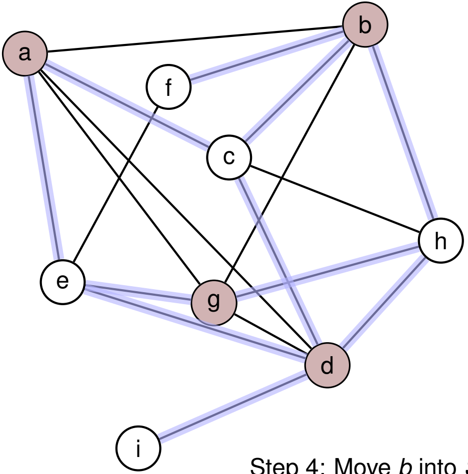
Illustration of Local Search



Step 4: Move *b* into *S*

Cut $= 10$

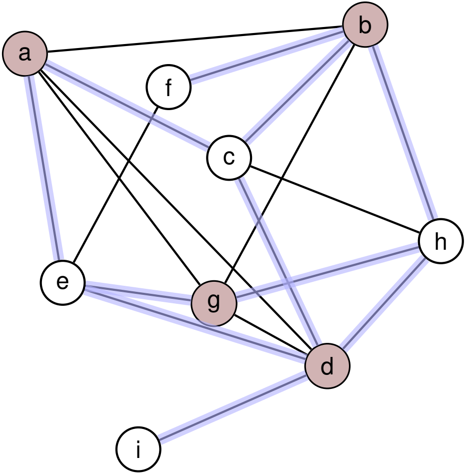
Illustration of Local Search



Step 4: Move *b* into *S*

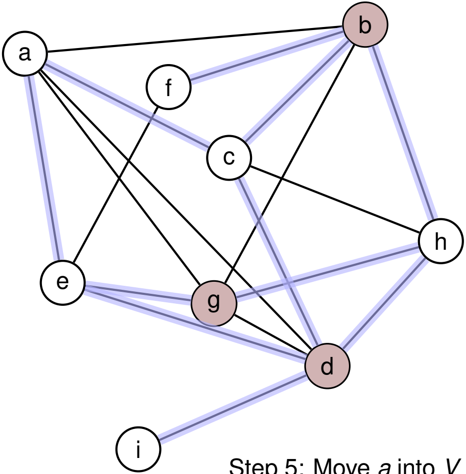
Cut $= \cancel{10}$

Illustration of Local Search



Cut $= 11$

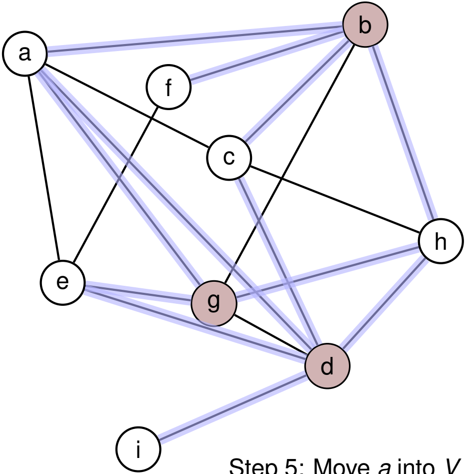
Illustration of Local Search



Step 5: Move a into $V \setminus S$

Cut $= 11$

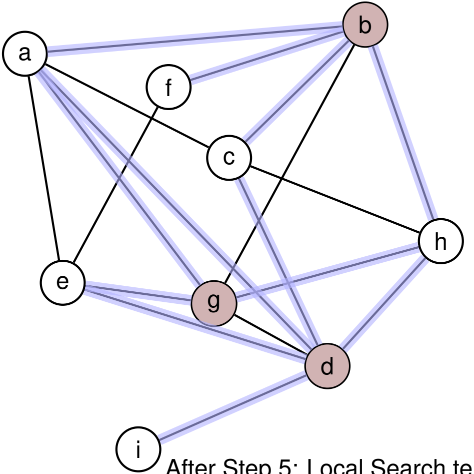
Illustration of Local Search



Step 5: Move a into $V \setminus S$

Cut $= \cancel{11}$

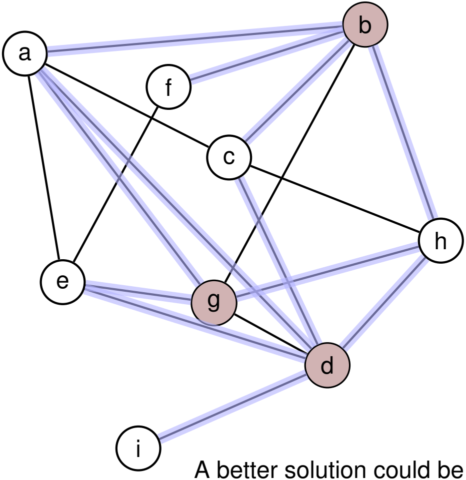
Illustration of Local Search



After Step 5: Local Search terminates

$\text{Cut} = 12$

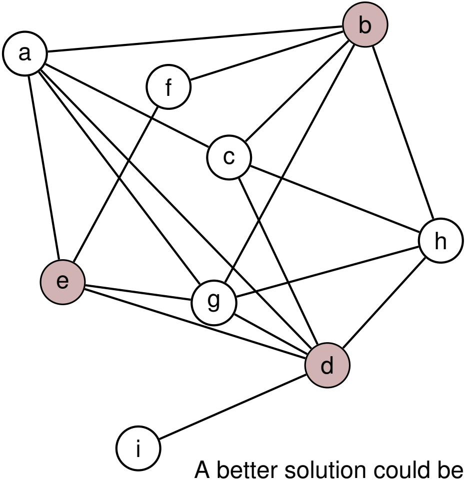
Illustration of Local Search



A better solution could be found:

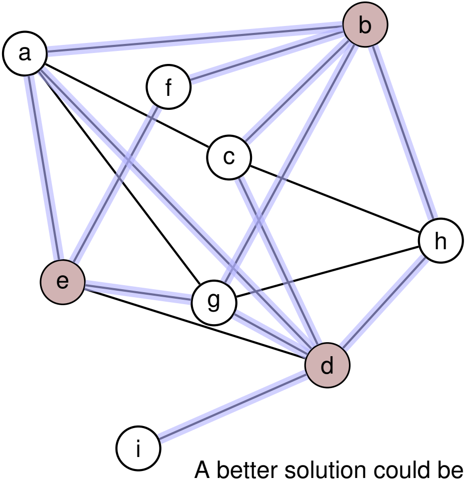
$\text{Cut} = 12$

Illustration of Local Search



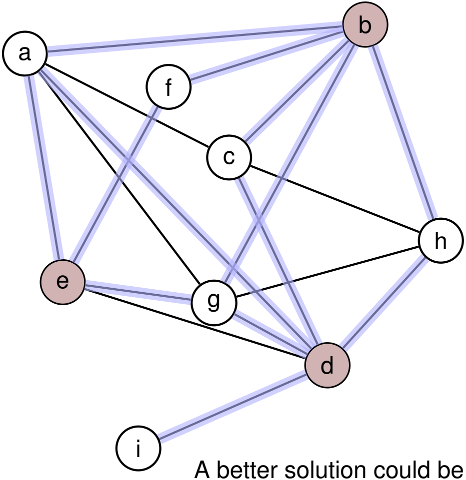
A better solution could be found:

Illustration of Local Search



A better solution could be found:

Illustration of Local Search



A better solution could be found:

$\text{Cut} = 13$

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2) W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u, v\}) \geq \sum_{v \in S, v \sim u} w(\{u, v\}), \tag{1}$$

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u, v\}) \geq \sum_{v \in S, v \sim u} w(\{u, v\}), \tag{1}$$

- Similarly, for any vertex $u \in V \setminus S$:

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u,v\}) \geq \sum_{v \in S, v \sim u} w(\{u,v\}), \tag{1}$$

- Similarly, for any vertex $u \in V \setminus S$:

$$\sum_{v \in S, v \sim u} w(\{u,v\}) \geq \sum_{v \in V \setminus S, v \sim u} w(\{u,v\}). \tag{2}$$

- Adding up equation 1 for all vertices in S

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u,v\}) \geq \sum_{v \in S, v \sim u} w(\{u,v\}), \tag{1}$$

- Similarly, for any vertex $u \in V \setminus S$:

$$\sum_{v \in S, v \sim u} w(\{u,v\}) \geq \sum_{v \in V \setminus S, v \sim u} w(\{u,v\}). \tag{2}$$

- Adding up equation 1 for all vertices in S

$$w(S) \geq 2 \cdot \sum_{v \in S, u \in S, u \sim v} w(\{u,v\})$$

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u, v\}) \geq \sum_{v \in S, v \sim u} w(\{u, v\}), \tag{1}$$

- Similarly, for any vertex $u \in V \setminus S$:

$$\sum_{v \in S, v \sim u} w(\{u, v\}) \geq \sum_{v \in V \setminus S, v \sim u} w(\{u, v\}). \tag{2}$$

- Adding up equation 1 for all vertices in S and equation 2 for all vertices in $V \setminus S$,

$$w(S) \geq 2 \cdot \sum_{v \in S, u \in S, u \sim v} w(\{u, v\})$$

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u, v\}) \geq \sum_{v \in S, v \sim u} w(\{u, v\}), \tag{1}$$

- Similarly, for any vertex $u \in V \setminus S$:

$$\sum_{v \in S, v \sim u} w(\{u, v\}) \geq \sum_{v \in V \setminus S, v \sim u} w(\{u, v\}). \tag{2}$$

- Adding up equation 1 for all vertices in S and equation 2 for all vertices in $V \setminus S$,

$$w(S) \geq 2 \cdot \sum_{v \in S, u \in S, u \sim v} w(\{u, v\}) \quad \text{and} \quad w(S) \geq 2 \cdot \sum_{v \in V \setminus S, u \in V \setminus S, u \sim v} w(\{u, v\}).$$

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u,v\}) \geq \sum_{v \in S, v \sim u} w(\{u,v\}), \tag{1}$$

- Similarly, for any vertex $u \in V \setminus S$:

$$\sum_{v \in S, v \sim u} w(\{u,v\}) \geq \sum_{v \in V \setminus S, v \sim u} w(\{u,v\}). \tag{2}$$

- Adding up equation 1 for all vertices in S and equation 2 for all vertices in $V \setminus S$,

$$w(S) \geq 2 \cdot \sum_{v \in S, u \in S, u \sim v} w(\{u,v\}) \quad \text{and} \quad w(S) \geq 2 \cdot \sum_{v \in V \setminus S, u \in V \setminus S, u \sim v} w(\{u,v\}).$$

- Adding up these two inequalities, and diving by 2 yields

$$w(S) \geq \sum_{v \in S, u \in S, u \sim v} w(\{u,v\}) + \sum_{v \in V \setminus S, u \in V \setminus S, u \sim v} w(\{u,v\}).$$

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u,v\}) \geq \sum_{v \in S, v \sim u} w(\{u,v\}), \qquad (1)$$

- Similarly, for any vertex $u \in V \setminus S$:

$$\sum_{v \in S, v \sim u} w(\{u,v\}) \geq \sum_{v \in V \setminus S, v \sim u} w(\{u,v\}). \qquad (2)$$

- Adding up equation 1 for all vertices in S and equation 2 for all vertices in $V \setminus S$,

$$w(S) \geq 2 \cdot \sum_{v \in S, u \in S, u \sim v} w(\{u,v\}) \quad \text{and} \quad w(S) \geq 2 \cdot \sum_{v \in V \setminus S, u \in V \setminus S, u \sim v} w(\{u,v\}).$$

- Adding up these two inequalities, and diving by 2 yields

$$w(S) \geq \sum_{v \in S, u \in S, u \sim v} w(\{u,v\}) + \sum_{v \in V \setminus S, u \in V \setminus S, u \sim v} w(\{u,v\}).$$

Every edge appears on one of the two sides.

Analysis of Local Search (1/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Proof:

- At the time of termination, for every vertex $u \in S$:

$$\sum_{v \in V \setminus S, v \sim u} w(\{u,v\}) \geq \sum_{v \in S, v \sim u} w(\{u,v\}), \qquad (1)$$

- Similarly, for any vertex $u \in V \setminus S$:

$$\sum_{v \in S, v \sim u} w(\{u,v\}) \geq \sum_{v \in V \setminus S, v \sim u} w(\{u,v\}). \qquad (2)$$

- Adding up equation 1 for all vertices in S and equation 2 for all vertices in $V \setminus S$,

$$w(S) \geq 2 \cdot \sum_{v \in S, u \in S, u \sim v} w(\{u,v\}) \quad \text{and} \quad w(S) \geq 2 \cdot \sum_{v \in V \setminus S, u \in V \setminus S, u \sim v} w(\{u,v\}).$$

- Adding up these two inequalities, and diving by 2 yields

$$w(S) \geq \sum_{v \in S, u \in S, u \sim v} w(\{u,v\}) + \sum_{v \in V \setminus S, u \in V \setminus S, u \sim v} w(\{u,v\}). \qquad \square$$

Every edge appears on one of the two sides.

Analysis of Local Search (2/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

Analysis of Local Search (2/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

What is the running time of LOCAL-SEARCH?

Analysis of Local Search (2/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

What is the running time of LOCAL-SEARCH?

- **Unweighted Graphs:** Cut increases by at least one in each iteration $\Rightarrow$ at most n^2 iterations

Analysis of Local Search (2/2)

Theorem

The cut returned by LOCAL-SEARCH satisfies $W \geq (1/2)W^*$.

What is the running time of LOCAL-SEARCH?

- **Unweighted Graphs:** Cut increases by at least one in each iteration $\Rightarrow$ at most n^2 iterations
- **Weighted Graphs:** could take exponential time in n (not obvious...)

Outline

Simple Algorithms for MAX-CUT

A Solution based on Semidefinite Programming

Summary

Max-Cut Problem

High-Level-Approach:

Max-Cut Problem

High-Level-Approach:

1. Describe the Max-Cut Problem as a quadratic optimisation problem

Max-Cut Problem

High-Level-Approach:

1. Describe the Max-Cut Problem as a quadratic optimisation problem
2. Solve a corresponding semidefinite program that is a relaxation of the original problem

Max-Cut Problem

High-Level-Approach:

1. Describe the Max-Cut Problem as a quadratic optimisation problem
2. Solve a corresponding semidefinite program that is a relaxation of the original problem
3. Recover an approximation for the original problem from the approximation for the semidefinite program

Max-Cut Problem

High-Level-Approach:

1. Describe the Max-Cut Problem as a quadratic optimisation problem
2. Solve a corresponding semidefinite program that is a relaxation of the original problem
3. Recover an approximation for the original problem from the approximation for the semidefinite program

Quadratic program

$$\begin{array}{lll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1-y_iy_j) & \\ \text{subject to} & y_i\in\{-1,+1\}, & i=1,\ldots,n. \end{array}$$

High-Level-Approach:

1. Describe the Max-Cut Problem as a quadratic optimisation problem
2. Solve a corresponding semidefinite program that is a relaxation of the original problem
3. Recover an approximation for the original problem from the approximation for the semidefinite program

Label vertices by $1, 2, \ldots, n$ and express weight function etc. as a $n \times n$-matrix.

Quadratic program

maximize $$\frac{1}{2} \sum_{(i,j) \in E} w_{i,j} \cdot (1 - y_i y_j)$$

subject to $$y_i \in \{-1, +1\}, \qquad i = 1, \ldots, n.$$

Max-Cut Problem

High-Level-Approach:

1. Describe the Max-Cut Problem as a quadratic optimisation problem
2. Solve a corresponding semidefinite program that is a relaxation of the original problem
3. Recover an approximation for the original problem from the approximation for the semidefinite program

Label vertices by $1, 2, \ldots, n$ and express weight function etc. as a $n \times n$-matrix.

Quadratic program

maximize	$\frac{1}{2} \sum_{(i,j) \in E} w_{i,j} \cdot (1 - y_i y_j)$
subject to	$y_i \in \{-1, +1\}, \qquad i = 1, \ldots, n.$

This models the MAX-CUT problem

Max-Cut Problem

High-Level-Approach:

1. Describe the Max-Cut Problem as a quadratic optimisation problem
2. Solve a corresponding semidefinite program that is a relaxation of the original problem
3. Recover an approximation for the original problem from the approximation for the semidefinite program

Label vertices by $1, 2, \ldots, n$ and express weight function etc. as a $n \times n$-matrix.

Quadratic program

maximize $$\frac{1}{2} \sum_{(i,j) \in E} w_{i,j} \cdot (1 - y_i y_j)$$

subject to $$y_i \in \{-1, +1\}, \qquad i = 1, \ldots, n.$$

This models the MAX-CUT problem

$$S = \{i \in V : y_i = +1\},$$
$$V \setminus S = \{i \in V : y_i = -1\}$$

Relaxation

Quadratic program

$$\begin{array}{lll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1-y_iy_j) & \\ \text{subject to} & y_i \in \{-1,+1\}, & i = 1,\ldots,n. \end{array}$$

Relaxation

Quadratic program

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1-y_iy_j) & \\
\text{subject to} & y_i \in \{-1,+1\}, & i=1,\ldots,n.
\end{array}
$$

Vector Programming Relaxation

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1-v_iv_j) & \\
\text{subject to} & v_i\cdot v_i = 1 & i=1,\ldots,n. \\
 & v_i \in \mathbb{R}^n &
\end{array}
$$

Relaxation

Quadratic program

$$\begin{array}{lll} \text{maximize} & \frac{1}{2} \sum_{(i,j) \in E} w_{i,j} \cdot (1 - y_i y_j) & \\ \text{subject to} & y_i \in \{-1, +1\}, & i = 1, \ldots, n. \end{array}$$

Any solution of the original program can be recovered by setting $v_i = (y_i, 0, 0, \ldots, 0)$!

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \frac{1}{2} \sum_{(i,j) \in E} w_{i,j} \cdot (1 - v_i v_j) & \\ \text{subject to} & v_i \cdot v_i = 1 & i = 1, \ldots, n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Examples:

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Examples:

$$A = \begin{pmatrix} 18 & 2 \\ 2 & 6 \end{pmatrix}$$

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Examples:

$$A = \begin{pmatrix} 18 & 2 \\ 2 & 6 \end{pmatrix} = \begin{pmatrix} 4 & -1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 1 \\ -1 & 2 \end{pmatrix},$$

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Examples:

$$A = \begin{pmatrix} 18 & 2 \\ 2 & 6 \end{pmatrix} = \begin{pmatrix} 4 & -1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 1 \\ -1 & 2 \end{pmatrix}, \quad \text{so } A \text{ is SPD.}$$

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Examples:

$$A = \begin{pmatrix} 18 & 2 \\ 2 & 6 \end{pmatrix} = \begin{pmatrix} 4 & -1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 1 \\ -1 & 2 \end{pmatrix}, \qquad \text{so } A \text{ is SPD.}$$

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Examples:

$$A = \begin{pmatrix} 18 & 2 \\ 2 & 6 \end{pmatrix} = \begin{pmatrix} 4 & -1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 1 \\ -1 & 2 \end{pmatrix}, \quad \text{so } A \text{ is SPD.}$$

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \quad \text{since } \begin{pmatrix} 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Examples:

$$A = \begin{pmatrix} 18 & 2 \\ 2 & 6 \end{pmatrix} = \begin{pmatrix} 4 & -1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 1 \\ -1 & 2 \end{pmatrix}, \quad \text{so } A \text{ is SPD.}$$

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \quad \text{since } \begin{pmatrix} 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix} = -2,$$

Positive Definite Matrices

Definition

A matrix $A \in \mathbb{R}^{n\times n}$ is positive semidefinite iff for all $y \in \mathbb{R}^n$,

$$y^T \cdot A \cdot y \geq 0.$$

Remark

1. A is symmetric and positive definite iff there exists a $n \times n$ matrix B with $B^T \cdot B = A$.
2. If A is symmetric and positive definite, then the matrix B above can be computed in polynomial time.

using Cholesky-decomposition

Examples:

$$A = \begin{pmatrix} 18 & 2 \\ 2 & 6 \end{pmatrix} = \begin{pmatrix} 4 & -1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 1 \\ -1 & 2 \end{pmatrix}, \quad \text{so } A \text{ is SPD.}$$

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \quad \text{since } \begin{pmatrix} 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix} = -2, \quad A \text{ is not SPD.}$$

Reformulating the Quadratic Program as a Semidefinite Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \displaystyle \frac{1}{2} \sum_{(i,j) \in E} w_{i,j} \cdot (1 - v_i v_j) & \\ \text{subject to} & v_i \cdot v_i = 1 & i = 1, \ldots, n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Reformulating the Quadratic Program as a Semidefinite Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \displaystyle \frac{1}{2} \sum_{(i,j) \in E} w_{i,j} \cdot (1 - v_i v_j) & \\ \text{subject to} & v_i \cdot v_i = 1 & i = 1, \ldots, n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Reformulation:

Reformulating the Quadratic Program as a Semidefinite Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1-v_i v_j) & \\ \text{subject to} & v_i\cdot v_i = 1 & i=1,\ldots,n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Reformulation:

- Introduce n^2 variables $a_{i,j} = v_i \cdot v_j$, which give rise to a matrix A

Reformulating the Quadratic Program as a Semidefinite Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1 - v_i v_j) & \\ \text{subject to} & v_i \cdot v_i = 1 & i = 1,\ldots,n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Reformulation:

- Introduce n^2 variables $a_{i,j} = v_i \cdot v_j$, which give rise to a matrix A
- If V is the matrix given by the vectors $(v_1, v_2, \ldots, v_n)$, then $A = V^T \cdot V$ is symmetric and positive definite

Reformulating the Quadratic Program as a Semidefinite Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1 - v_i v_j) & \\ \text{subject to} & v_i \cdot v_i = 1 & i = 1,\ldots,n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Reformulation:

- Introduce n^2 variables $a_{i,j} = v_i \cdot v_j$, which give rise to a matrix A
- If V is the matrix given by the vectors $(v_1, v_2, \ldots, v_n)$, then $A = V^T \cdot V$ is symmetric and positive definite

Semidefinite Program

$$\begin{array}{ll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1 - a_{i,j}) \\ \text{subject to} & A = (a_{i,j}) \text{ is symmetric and positive definite,} \\ & \text{and } a_{i,i} = 1 \text{ for all } i = 1,\ldots,n \end{array}$$

Reformulating the Quadratic Program as a Semidefinite Program

Vector Programming Relaxation

$$\begin{array}{lll}
\text{maximize} & \displaystyle \frac{1}{2} \sum_{(i,j)\in E} w_{i,j} \cdot (1 - v_i v_j) & \\
\text{subject to} & v_i \cdot v_i = 1 & i = 1, \ldots, n. \\
 & v_i \in \mathbb{R}^n &
\end{array}$$

Reformulation:

- Introduce n^2 variables $a_{i,j} = v_i \cdot v_j$, which give rise to a matrix A
- If V is the matrix given by the vectors $(v_1, v_2, \ldots, v_n)$, then $A = V^T \cdot V$ is symmetric and positive definite

Solve this (which can be done in polynomial time), and recover V using Cholesky Decomposition.

Semidefinite Program

$$\begin{array}{ll}
\text{maximize} & \displaystyle \frac{1}{2} \sum_{(i,j)\in E} w_{i,j} \cdot (1 - a_{i,j}) \\
\text{subject to} & A = (a_{i,j}) \text{ is symmetric and positive definite,} \\
 & \text{and } a_{i,i} = 1 \text{ for all } i = 1, \ldots, n
\end{array}$$

Rounding the Vector Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1 - v_i v_j) & \\ \text{subject to} & v_i \cdot v_i = 1 & i = 1,\ldots,n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Rounding the Vector Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1-v_i v_j) & \\ \text{subject to} & v_i\cdot v_i = 1 & i=1,\ldots,n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Rounding by a random hyperplane :

1. Pick a random vector $r=(r_1,r_2,\ldots,r_n)$ by drawing each component from $\mathcal{N}(0,1)$

Rounding the Vector Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \displaystyle\frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1 - v_i v_j) & \\ \text{subject to} & v_i \cdot v_i = 1 & i = 1,\ldots,n. \\ & v_i \in \mathbb{R}^n & \end{array}$$

Rounding by a random hyperplane :

1. Pick a random vector $r = (r_1, r_2, \ldots, r_n)$ by drawing each component from $\mathcal{N}(0,1)$
2. Put $i \in V$ if $v_i \cdot r \geq 0$ and $i \in V \setminus S$ otherwise

Rounding the Vector Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1-v_iv_j) & \\ \text{subject to} & v_i\cdot v_i = 1 & i=1,\dots,n. \\ & v_i\in\mathbb{R}^n & \end{array}$$

Rounding by a random hyperplane :

1. Pick a random vector $r=(r_1,r_2,\dots,r_n)$ by drawing each component from $\mathcal{N}(0,1)$
2. Put $i\in V$ if $v_i\cdot r\geq 0$ and $i\in V\setminus S$ otherwise

Lemma 1

The probability that two vectors $v_i, v_j\in\mathbb{R}^n$ are separated by the (random) hyperplane given by r equals $\frac{\arccos(v_i\cdot v_j)}{\pi}$.

Rounding the Vector Program

Vector Programming Relaxation

$$\begin{array}{lll} \text{maximize} & \frac{1}{2}\sum_{(i,j)\in E} w_{i,j}\cdot(1-v_iv_j) & \\ \text{subject to} & v_i\cdot v_i = 1 & i=1,\ldots,n. \\ & v_i\in\mathbb{R}^n & \end{array}$$

Rounding by a random hyperplane :

1. Pick a random vector $r = (r_1, r_2, \ldots, r_n)$ by drawing each component from $\mathcal{N}(0,1)$
2. Put $i \in V$ if $v_i \cdot r \geq 0$ and $i \in V \setminus S$ otherwise

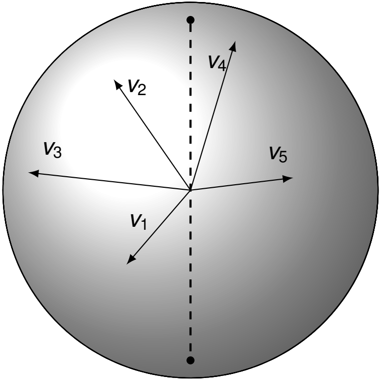
Lemma 1

The probability that two vectors $v_i, v_j \in \mathbb{R}^n$ are separated by the (random) hyperplane given by r equals $\frac{\arccos(v_i\cdot v_j)}{\pi}$.

Follows by projecting on the plane given by v_i and v_j.

Illustration of the Hyperplane

N



S

Illustration of the Hyperplane

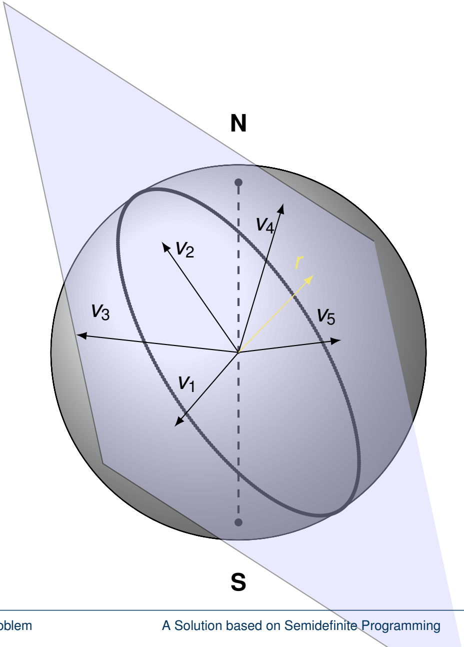
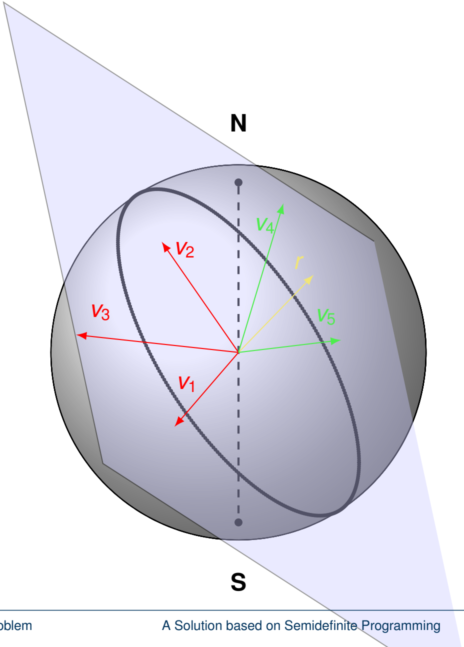


Illustration of the Hyperplane



A second (technical) Lemma

Lemma 2

For any $x \in [-1, 1]$,

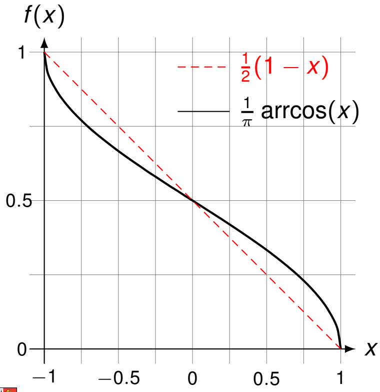
$$\frac{1}{\pi} \arccos(x) \geq 0.878 \cdot \frac{1}{2}(1 - x).$$

A second (technical) Lemma

Lemma 2

For any $x \in [-1, 1]$,

$$\frac{1}{\pi} \arccos(x) \geq 0.878 \cdot \frac{1}{2}(1 - x).$$

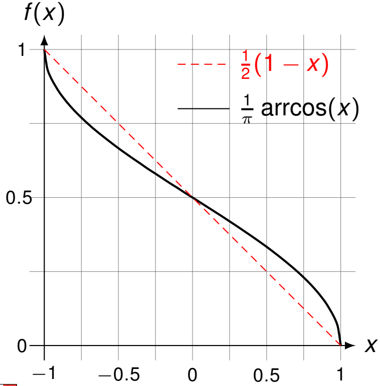


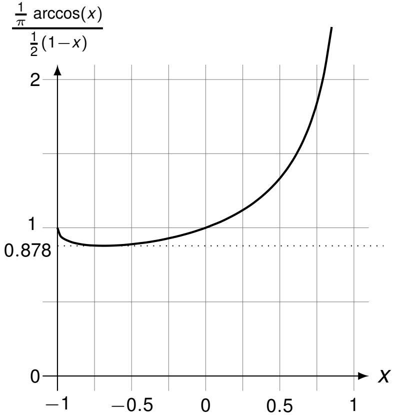
A second (technical) Lemma

Lemma 2

For any $x \in [-1, 1]$,

$$\frac{1}{\pi} \arccos(x) \geq 0.878 \cdot \frac{1}{2}(1 - x).$$





Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof:

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\mathbf{E}\left[\, w(S) \,\right]$$

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\mathbf{E}[\,w(S)\,] = \mathbf{E}\left[\sum_{\{i,j\}\in E} X_{i,j}\right]$$

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned} \mathbf{E}\left[\, w(S) \,\right] &= \mathbf{E}\left[\sum_{\{i,j\} \in E} X_{i,j} \right] \\ &= \sum_{\{i,j\} \in E} \mathbf{E}\left[\, X_{i,j} \,\right] \end{aligned}$$

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned} \mathbf{E}\left[\, w(S) \,\right] &= \mathbf{E}\left[\sum_{\{i,j\} \in E} X_{i,j} \right] \\ &= \sum_{\{i,j\} \in E} \mathbf{E}\left[\, X_{i,j} \,\right] \\ &= \sum_{\{i,j\} \in E} w_{i,j} \cdot \mathbf{Pr}\left[\, \{i,j\} \in E \text{ is in the cut} \,\right] \end{aligned}$$

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned} \mathbf{E}[\, w(S) \,] &= \mathbf{E}\left[\sum_{\{i,j\} \in E} X_{i,j} \right] \\ &= \sum_{\{i,j\} \in E} \mathbf{E}[\, X_{i,j} \,] \\ &= \sum_{\{i,j\} \in E} w_{i,j} \cdot \mathbf{Pr}[\, \{i,j\} \in E \text{ is in the cut} \,] \end{aligned}$$

By Lemma 1

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned} \mathbf{E}\left[\, w(S) \,\right] &= \mathbf{E}\left[\sum_{\{i,j\} \in E} X_{i,j} \right] \\ &= \sum_{\{i,j\} \in E} \mathbf{E}\left[\, X_{i,j} \,\right] \\ &= \sum_{\{i,j\} \in E} w_{i,j} \cdot \mathbf{Pr}\left[\, \{i,j\} \in E \text{ is in the cut} \,\right] \\ &= \sum_{\{i,j\} \in E} w_{i,j} \cdot \frac{1}{\pi} \arccos(v_i \cdot v_j) \end{aligned}$$

By Lemma 1

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned} \mathbf{E}[\,w(S)\,] &= \mathbf{E}\left[\sum_{\{i,j\}\in E} X_{i,j}\right] \\ &= \sum_{\{i,j\}\in E} \mathbf{E}[\,X_{i,j}\,] \\ &= \sum_{\{i,j\}\in E} w_{i,j} \cdot \mathbf{Pr}[\,\{i,j\} \in E \text{ is in the cut}\,] \\ &= \sum_{\{i,j\}\in E} w_{i,j} \cdot \frac{1}{\pi} \arccos(v_i \cdot v_j) \end{aligned}$$

By Lemma 1

By Lemma 2

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned}
\mathbf{E}\left[\, w(S) \,\right] &= \mathbf{E}\left[\sum_{\{i,j\}\in E} X_{i,j} \right] \\
&= \sum_{\{i,j\}\in E} \mathbf{E}\left[\, X_{i,j} \,\right] \\
&= \sum_{\{i,j\}\in E} w_{i,j} \cdot \mathbf{Pr}\left[\, \{i,j\} \in E \text{ is in the cut} \,\right] \\
\text{By Lemma 1} \quad &= \sum_{\{i,j\}\in E} w_{i,j} \cdot \frac{1}{\pi} \arccos(v_i \cdot v_j) \\
\text{By Lemma 2} \quad &\geq 0.878 \cdot \frac{1}{2} \sum_{\{i,j\}\in E} w_{i,j} \cdot (1 - v_i \cdot v_j)
\end{aligned}$$

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned}
\mathbf{E}[\, w(S) \,] &= \mathbf{E}\left[\sum_{\{i,j\} \in E} X_{i,j} \right] \\
&= \sum_{\{i,j\} \in E} \mathbf{E}[\, X_{i,j} \,] \\
&= \sum_{\{i,j\} \in E} w_{i,j} \cdot \mathbf{Pr}[\, \{i,j\} \in E \text{ is in the cut} \,] \\
\text{By Lemma 1} \quad &= \sum_{\{i,j\} \in E} w_{i,j} \cdot \frac{1}{\pi} \arccos(v_i \cdot v_j) \\
\text{By Lemma 2} \quad &\geq 0.878 \cdot \frac{1}{2} \sum_{\{i,j\} \in E} w_{i,j} \cdot (1 - v_i \cdot v_j) = 0.878 \cdot z^*
\end{aligned}$$

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned}
\mathbf{E}[\, w(S) \,] &= \mathbf{E}\left[\sum_{\{i,j\} \in E} X_{i,j} \right] \\
&= \sum_{\{i,j\} \in E} \mathbf{E}[\, X_{i,j} \,] \\
&= \sum_{\{i,j\} \in E} w_{i,j} \cdot \mathbf{Pr}[\, \{i,j\} \in E \text{ is in the cut} \,] \\
&= \sum_{\{i,j\} \in E} w_{i,j} \cdot \frac{1}{\pi} \arccos(v_i \cdot v_j) \\
&\geq 0.878 \cdot \frac{1}{2} \sum_{\{i,j\} \in E} w_{i,j} \cdot (1 - v_i \cdot v_j) = 0.878 \cdot z^* \geq 0.878 \cdot W^*.
\end{aligned}$$

By Lemma 1

By Lemma 2

Putting Everything Together

Theorem (Goemans, Willamson'96)

The algorithm has an approximation ratio of $\frac{1}{0.878} \approx 1.139$.

Proof: Define an indicator variable

$$X_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ are on different sides of the hyperplane} \\ 0 & \text{otherwise.} \end{cases}$$

Hence for the (random) weight of the computed cut,

$$\begin{aligned}
\mathbf{E}\left[\, w(S) \,\right] &= \mathbf{E}\left[\sum_{\{i,j\}\in E} X_{i,j} \right] \\
&= \sum_{\{i,j\}\in E} \mathbf{E}\left[\, X_{i,j} \,\right] \\
&= \sum_{\{i,j\}\in E} w_{i,j} \cdot \mathbf{Pr}\left[\, \{i,j\} \in E \text{ is in the cut} \,\right] \\
\text{By Lemma 1} \quad &= \sum_{\{i,j\}\in E} w_{i,j} \cdot \frac{1}{\pi} \arccos(v_i \cdot v_j) \\
\text{By Lemma 2} \quad &\geq 0.878 \cdot \frac{1}{2} \sum_{\{i,j\}\in E} w_{i,j} \cdot (1 - v_i \cdot v_j) = 0.878 \cdot z^* \geq 0.878 \cdot W^*. \qquad \square
\end{aligned}$$

MAX-CUT: Concluding Remarks

Theorem (Goemans, Willamson'96)

There is a randomised polynomial-time 1.139-approximation algorithm for MAX-CUT.

MAX-CUT: Concluding Remarks

Theorem (Goemans, Willamson'96)

There is a randomised polynomial-time 1.139-approximation algorithm for MAX-CUT.

can be derandomized (with some effort)

MAX-CUT: Concluding Remarks

Theorem (Goemans, Willamson'96)

There is a randomised polynomial-time 1.139-approximation algorithm for MAX-CUT.

can be derandomized (with some effort)

Similar approach can be applied to MAX-3-CNF and yields an approximation ratio of 1.345

MAX-CUT: Concluding Remarks

Theorem (Goemans, Willamson'96)

There is a randomised polynomial-time 1.139-approximation algorithm for MAX-CUT.

can be derandomized (with some effort)

Similar approach can be applied to MAX-3-CNF and yields an approximation ratio of 1.345

Theorem (Håstad'97)

Unless P$=$NP, there is no ρ-approximation algorithm for MAX-CUT with $\rho \leq \frac{17}{16} = 1.0625$.

MAX-CUT: Concluding Remarks

Theorem (Goemans, Willamson'96)

There is a randomised polynomial-time 1.139-approximation algorithm for MAX-CUT.

can be derandomized (with some effort)

Similar approach can be applied to MAX-3-CNF and yields an approximation ratio of 1.345

Theorem (Håstad'97)

Unless P=NP, there is no ρ-approximation algorithm for MAX-CUT with $\rho \leq \frac{17}{16} = 1.0625$.

Theorem (Khot, Kindler, Mossel, O'Donnell'04)

Assuming the so-called Unique Games Conjecture holds, unless P=NP there is no ρ-approximation algorithm for MAX-CUT with

$$\rho \leq \max_{-1 \leq x \leq 1} \frac{\frac{1}{2}(1-x)}{\frac{1}{\pi}\arccos(x)} \leq 1.139$$

Other Approximation Algorithms for MAX-CUT

Theorem (Mathieu, Schudy'08)

For any $\epsilon > 0$, there is a randomised algorithm with running time $O(n^2)2^{O(1/\epsilon^2)}$ so that the expected value of the output deviates from the maximum cut value by at most $O(\epsilon \cdot n^2)$.

Other Approximation Algorithms for MAX-CUT

Theorem (Mathieu, Schudy'08)

For any $\epsilon > 0$, there is a randomised algorithm with running time $O(n^2)2^{O(1/\epsilon^2)}$ so that the expected value of the output deviates from the maximum cut value by at most $O(\epsilon \cdot n^2)$.

This is an additive approximation!

Other Approximation Algorithms for MAX-CUT

Theorem (Mathieu, Schudy'08)

For any $\epsilon > 0$, there is a randomised algorithm with running time $O(n^2)2^{O(1/\epsilon^2)}$ so that the expected value of the output deviates from the maximum cut value by at most $O(\epsilon \cdot n^2)$.

This is an additive approximation!

Algorithm (1):

1. Take a sample S of $x = O(1/\epsilon^2)$ vertices chosen uniformly at random
2. For each of the 2^x possible cuts, go through vertices in $V \setminus S$ in random order and place them on the side of the cut which maximizes the crossing edges
3. Output the best cut found

Other Approximation Algorithms for MAX-CUT

Theorem (Mathieu, Schudy'08)

For any $\epsilon > 0$, there is a randomised algorithm with running time $O(n^2)2^{O(1/\epsilon^2)}$ so that the expected value of the output deviates from the maximum cut value by at most $O(\epsilon \cdot n^2)$.

This is an additive approximation!

Algorithm (1):

1. Take a sample S of $x = O(1/\epsilon^2)$ vertices chosen uniformly at random
2. For each of the 2^x possible cuts, go through vertices in $V \setminus S$ in random order and place them on the side of the cut which maximizes the crossing edges
3. Output the best cut found

Theorem (Trevisan'08)

There is a randomised 1.833-approximation algorithm for MAX-CUT which runs in $O(n^2 \cdot \text{polylog}(n))$ time.

Other Approximation Algorithms for MAX-CUT

Theorem (Mathieu, Schudy'08)

For any $\epsilon > 0$, there is a randomised algorithm with running time $O(n^2)2^{O(1/\epsilon^2)}$ so that the expected value of the output deviates from the maximum cut value by at most $O(\epsilon \cdot n^2)$.

This is an additive approximation!

Algorithm (1):

1. Take a sample S of $x = O(1/\epsilon^2)$ vertices chosen uniformly at random
2. For each of the 2^x possible cuts, go through vertices in $V \setminus S$ in random order and place them on the side of the cut which maximizes the crossing edges
3. Output the best cut found

Theorem (Trevisan'08)

There is a randomised 1.833-approximation algorithm for MAX-CUT which runs in $O(n^2 \cdot \text{polylog}(n))$ time.

Exploits relation between the smallest eigenvalue and the structure of the graph.

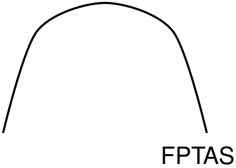
Outline

Simple Algorithms for MAX-CUT

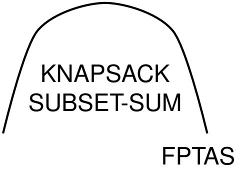
A Solution based on Semidefinite Programming

Summary

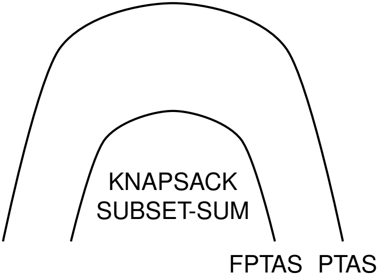
Spectrum of Approximations



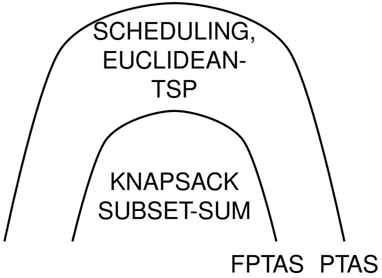
Spectrum of Approximations



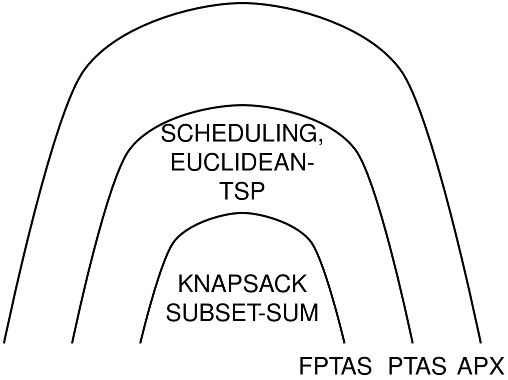
Spectrum of Approximations



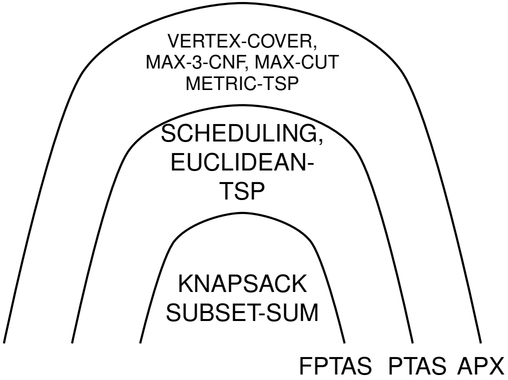
Spectrum of Approximations



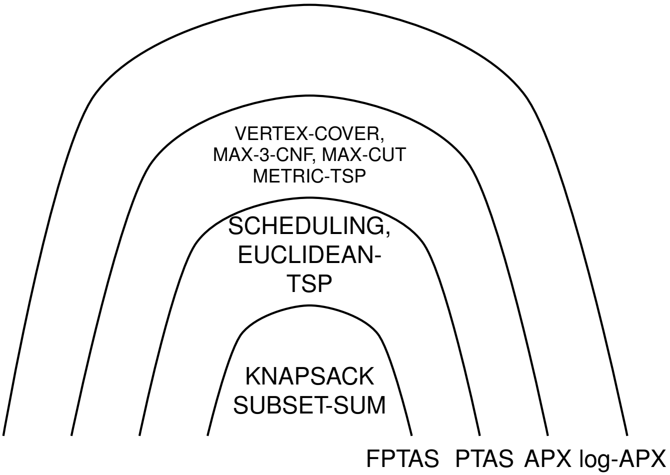
Spectrum of Approximations



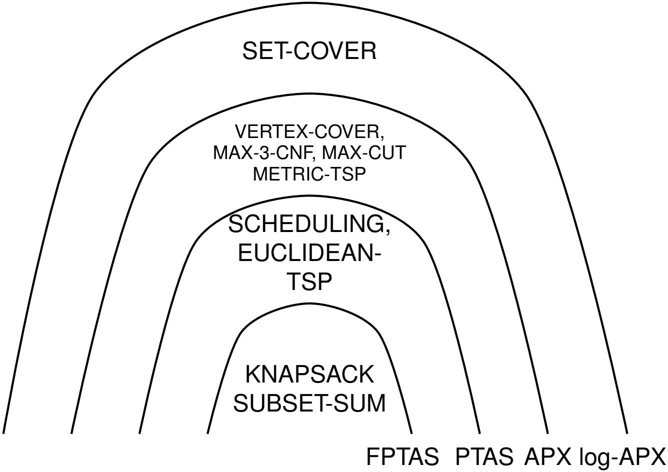
Spectrum of Approximations



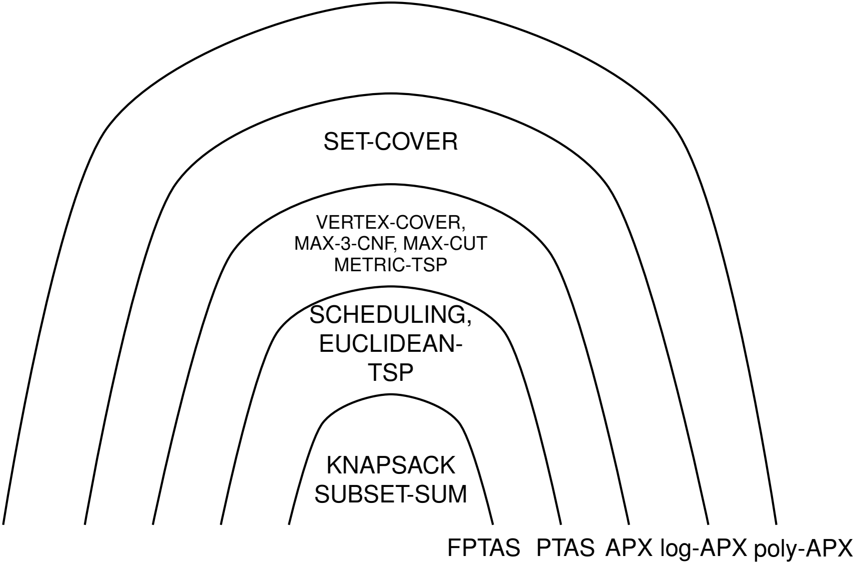
Spectrum of Approximations



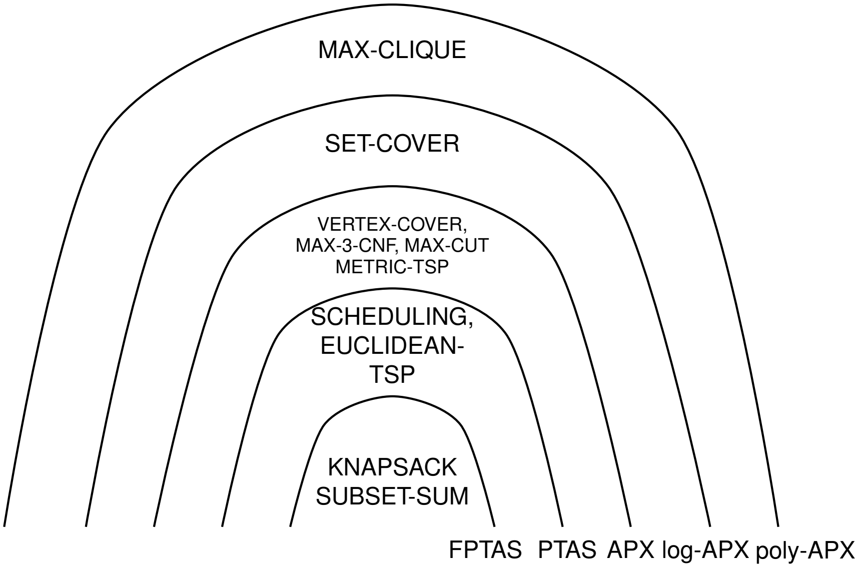
Spectrum of Approximations



Spectrum of Approximations



Spectrum of Approximations



The End

Thank you very much and
Best Wishes for the Exam!