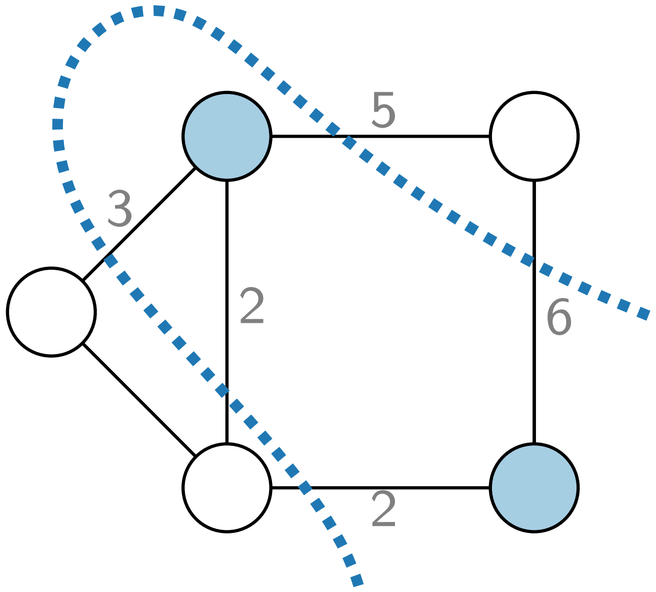
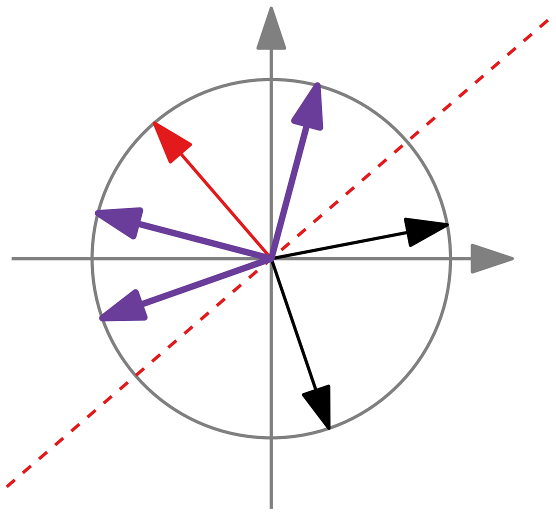
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Advanced Algorithms

QP-Relaxation
for Max Cut

Jonathan Klawitter · WS20

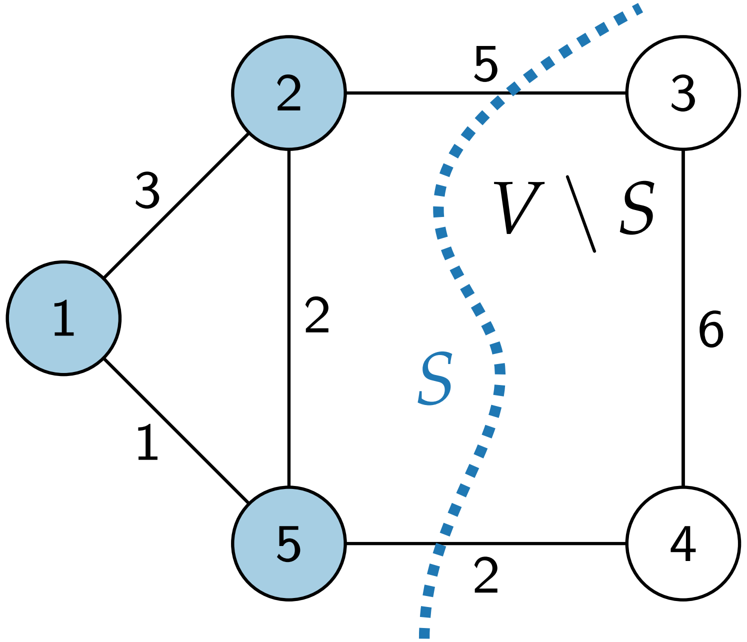




Cut

- Let $G = (V, E)$ be a graph with edge weights $c\colon E \to \mathbb{N}$.
- A **cut** of G is a partition $(S, V \setminus S)$ of V.
- The **weight** of a cut $(S, V \setminus S)$ is

$$c(S, V \setminus S) = \sum_{\substack{uv \in E, \\ u \in S, v \in V \setminus S}} c(uv)$$

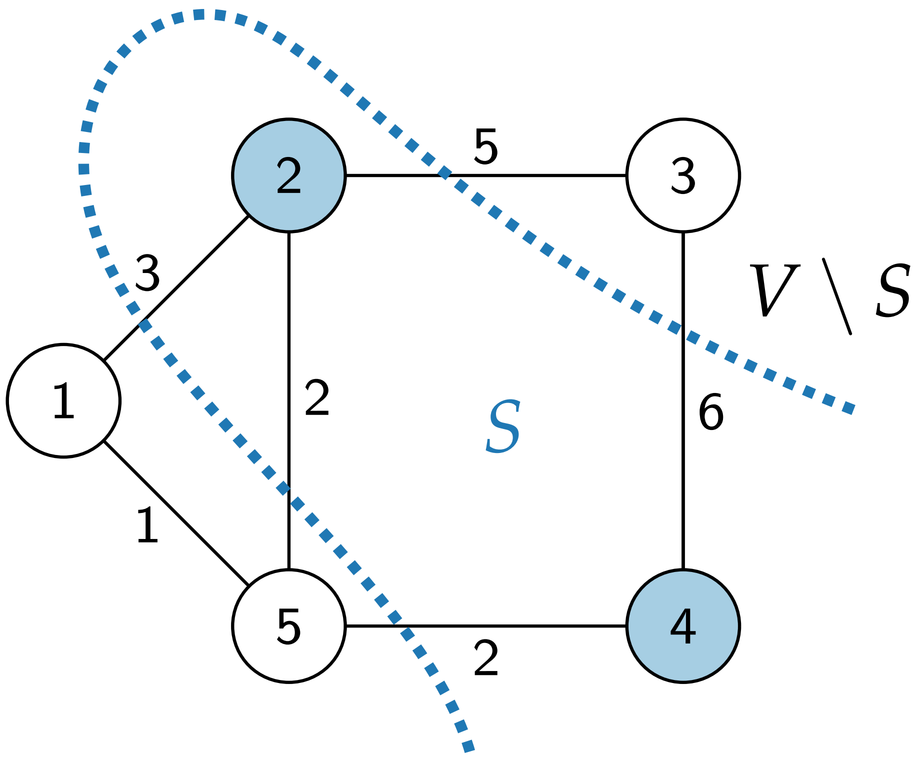


$$c(\{1, 2, 5\}, \{3, 4\}) = 7$$

The **MaxCut** Problem

Input. Graph $G = (V, E)$, edge weights $c\colon E \to \mathbb{N}$.

Output. Cut $(S, V \setminus S)$ of G with **maximum** weight.



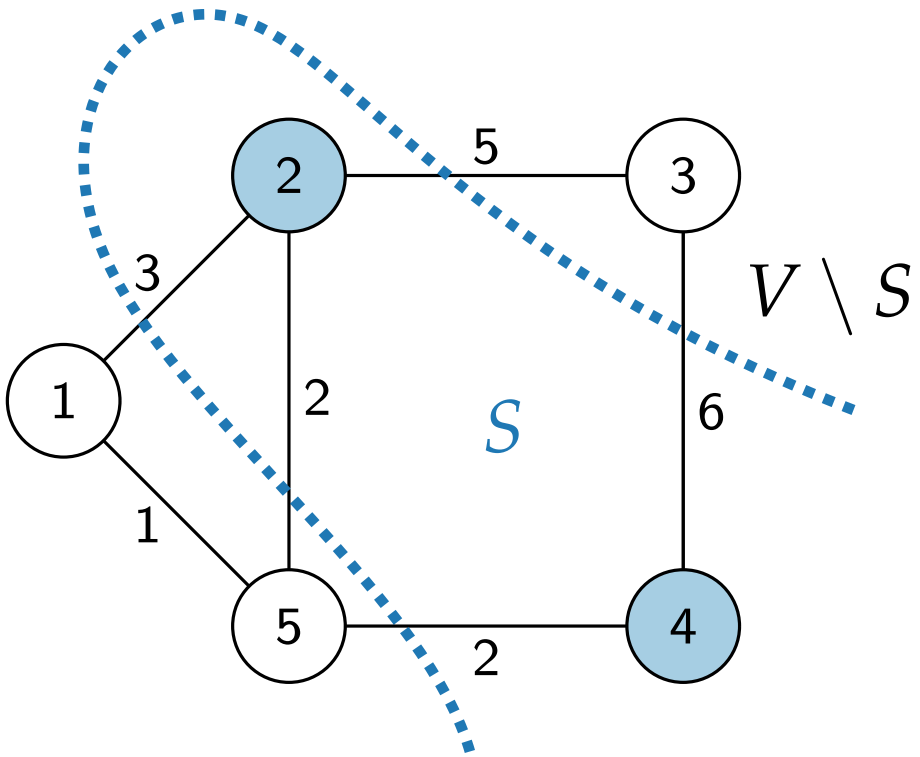
$c(S, V \setminus V) = 18$

The **MaxCut** Problem

Input. Graph $G = (V, E)$, edge weights $c\colon E \to \mathbb{N}$.

Output. Cut $(S, V \setminus S)$ of G with **maximum** weight.

- MaxCut is NP-hard.



$c(S, V \setminus V) = 18$

Randomized 0.5-approximation for (unweighted) MaxCut

$\textsc{CoinFlipMaxCut}(G, c\colon E \to 1)$

$S \leftarrow \emptyset$
foreach $v \in V$ **do**
  if coin flip shows $\textsc{Heads}$ **then**
    $S \leftarrow S \cup \{v\}$
return $c(S, V \setminus S), S$

Randomized 0.5-approximation for (unweighted) MaxCut

Theorem 1.
CoinFlipMaxCut is a randomized 0.5-approximation algorithm for MaxCut.

CoinFlipMaxCut$(G, c\colon E \to 1)$

- $S \leftarrow \emptyset$
- **foreach** $v \in V$ **do**
  - **if** coin flip shows Heads **then**
    - $S \leftarrow S \cup \{v\}$
- **return** $c(S, V \setminus S), S$

Randomized 0.5-approximation for (unweighted) MaxCut

Theorem 1.
CoinFlipMaxCut is a randomized 0.5-approximation algorithm for MaxCut.

Proof.

- Runs in $O(n+m)$.

CoinFlipMaxCut$(G, c\colon E \to 1)$

```
S ← ∅
foreach v ∈ V do
    if coin flip shows Heads then
        S ← S ∪ {v}
return c(S, V \ S), S
```

Randomized 0.5-approximation for (unweighted) MaxCut

Theorem 1.
$\textsc{CoinFlipMaxCut}$ is a randomized 0.5-approximation algorithm for MaxCut.

$\textsc{CoinFlipMaxCut}(G, c\colon E \to 1)$

```
S ← ∅
foreach v ∈ V do
    if coin flip shows HEADS then
        S ← S ∪ {v}
return c(S, V \ S), S
```

Proof.

- Runs in $O(n+m)$.
- Compute expected weight of cut:

$$\mathrm{E}[c(\textsc{CoinFlipMaxCut}(G))]$$

$$\geq \frac{1}{2}\mathrm{OPT}(G)$$

Randomized 0.5-approximation for (unweighted) MaxCut

Theorem 1.
$\textsc{CoinFlipMaxCut}$ is a randomized 0.5-approximation algorithm for MaxCut.

$\textsc{CoinFlipMaxCut}(G, c\colon E \to 1)$

$S \leftarrow \emptyset$
foreach $v \in V$ **do**
  if coin flip shows $\textsc{Heads}$ **then**
    $S \leftarrow S \cup \{v\}$
return $c(S, V \setminus S), S$

Proof.

- Runs in $O(n+m)$.
- Compute expected weight of cut:

$$\mathsf{E}[c(\textsc{CoinFlipMaxCut}(G))] = \mathsf{E}\big[|E(S, V \setminus S)|\big]$$
$$=$$
$$\geq \frac{1}{2}\mathsf{OPT}(G)$$

Randomized 0.5-approximation for (unweighted) MaxCut

Theorem 1.
CoinFlipMaxCut is a randomized 0.5-approximation algorithm for MaxCut.

$\textsc{CoinFlipMaxCut}(G, c\colon E \to 1)$

$S \leftarrow \emptyset$
foreach $v \in V$ **do**
  if coin flip shows Heads **then**
    $S \leftarrow S \cup \{v\}$
return $c(S, V \setminus S), S$

Proof.

- Runs in $O(n+m)$.
- Compute expected weight of cut:

$$\begin{aligned} \mathrm{E}[c(\textsc{CoinFlipMaxCut}(G))] &= \mathrm{E}\big[|E(S, V \setminus S)|\big] \\ &= \sum_{e \in E} \mathrm{P}[e \in E(S, V \setminus S)] \\ &= \qquad\qquad \geq \frac{1}{2}\mathrm{OPT}(G) \end{aligned}$$

Randomized 0.5-approximation for (unweighted) MaxCut

Theorem 1.
CoinFlipMaxCut is a randomized 0.5-approximation algorithm for MaxCut.

$\textsc{CoinFlipMaxCut}(G, c\colon E \to 1)$

- $S \leftarrow \emptyset$
- **foreach** $v \in V$ **do**
  - **if** coin flip shows Heads **then**
    - $S \leftarrow S \cup \{v\}$
- **return** $c(S, V \setminus S), S$

Proof.

- Runs in $O(n+m)$.
- Compute expected weight of cut:

$$\begin{aligned} \mathsf{E}[c(\textsc{CoinFlipMaxCut}(G))] &= \mathsf{E}\big[|E(S, V \setminus S)|\big] \\ &= \sum_{e \in E} \mathsf{P}[e \in E(S, V \setminus S)] \\ &= \sum_{e \in E} \frac{1}{2} = \frac{1}{2}|E| \geq \frac{1}{2}\mathrm{OPT}(G) \end{aligned}$$

Randomized 0.5-approximation for (unweighted) MaxCut

Theorem 1.
CoinFlipMaxCut is a randomized 0.5-approximation algorithm for MaxCut.

$\textsc{CoinFlipMaxCut}(G, c\colon E \to 1)$

```
S ← ∅
foreach v ∈ V do
    if coin flip shows HEADS then
        S ← S ∪ {v}
return c(S, V \ S), S
```

Proof.

- Runs in $O(n+m)$.
- Compute expected weight of cut:

$$\begin{aligned}\mathsf{E}[c(\textsc{CoinFlipMaxCut}(G))] &= \mathsf{E}\big[|E(S, V\setminus S)|\big] \\ &= \sum_{e\in E} \mathsf{P}[e \in E(S, V\setminus S)] \\ &= \sum_{e\in E} \frac{1}{2} = \frac{1}{2}|E| \geq \frac{1}{2}\mathsf{OPT}(G)\end{aligned}$$

- Can be "derandomized". Exercise.

LP-Relaxation

Integer Linear Program

$$
\begin{array}{lrcr}
\textbf{maximize} & c^\intercal x & & \\
\textbf{subject to} & Ax & \leq & b \\
 & x & \geq & 0 \\
 & x & \in & \mathbb{Z}
\end{array}
$$

LP-Relaxation

Integer Linear Program

$$\begin{array}{llcl} \textbf{maximize} & c^\intercal x & & \\ \textbf{subject to} & Ax & \leq & b \\ & x & \geq & 0 \\ & x & \in & \mathbb{Z} \end{array}$$

LP-Relaxation →

Linear Program

$$\begin{array}{llcl} \textbf{maximize} & c^\intercal x & & \\ \textbf{subject to} & Ax & \leq & b \\ & x & \geq & 0 \end{array}$$

LP-Relaxation

Integer Linear Program

$$\begin{array}{llcl} \textbf{maximize} & c^\intercal x & & \\ \textbf{subject to} & Ax & \leq & b \\ & x & \geq & 0 \\ & x & \in & \mathbb{Z} \end{array}$$

LP-Relaxation

Linear Program

$$\begin{array}{llcl} \textbf{maximize} & c^\intercal x & & \\ \textbf{subject to} & Ax & \leq & b \\ & x & \geq & 0 \end{array}$$

Solve in polynomial time

Solution for LP

$x^\star$

LP-Relaxation

Integer Linear Program

$$\begin{array}{llcl} \textbf{maximize} & c^\intercal x & & \\ \textbf{subject to} & Ax & \leq & b \\ & x & \geq & 0 \\ & x & \in & \mathbb{Z} \end{array}$$

LP-Relaxation →

Linear Program

$$\begin{array}{llcl} \textbf{maximize} & c^\intercal x & & \\ \textbf{subject to} & Ax & \leq & b \\ & x & \geq & 0 \end{array}$$

Solve in polynomial time ↓

Solution for LP

$x^\star$

← e.g. rounding

Assignment for ILP

$x^\star$

LP-Relaxation

Integer Linear Program

$$\begin{array}{llll} \textbf{maximize} & c^\intercal x & & \\ \textbf{subject to} & Ax & \leq & b \\ & x & \geq & 0 \\ & x & \in & \mathbb{Z} \end{array}$$

LP-Relaxation

Linear Program

$$\begin{array}{llll} \textbf{maximize} & c^\intercal x & & \\ \textbf{subject to} & Ax & \leq & b \\ & x & \geq & 0 \end{array}$$

Solve in polynomial time

Solution for LP

$x^\star$

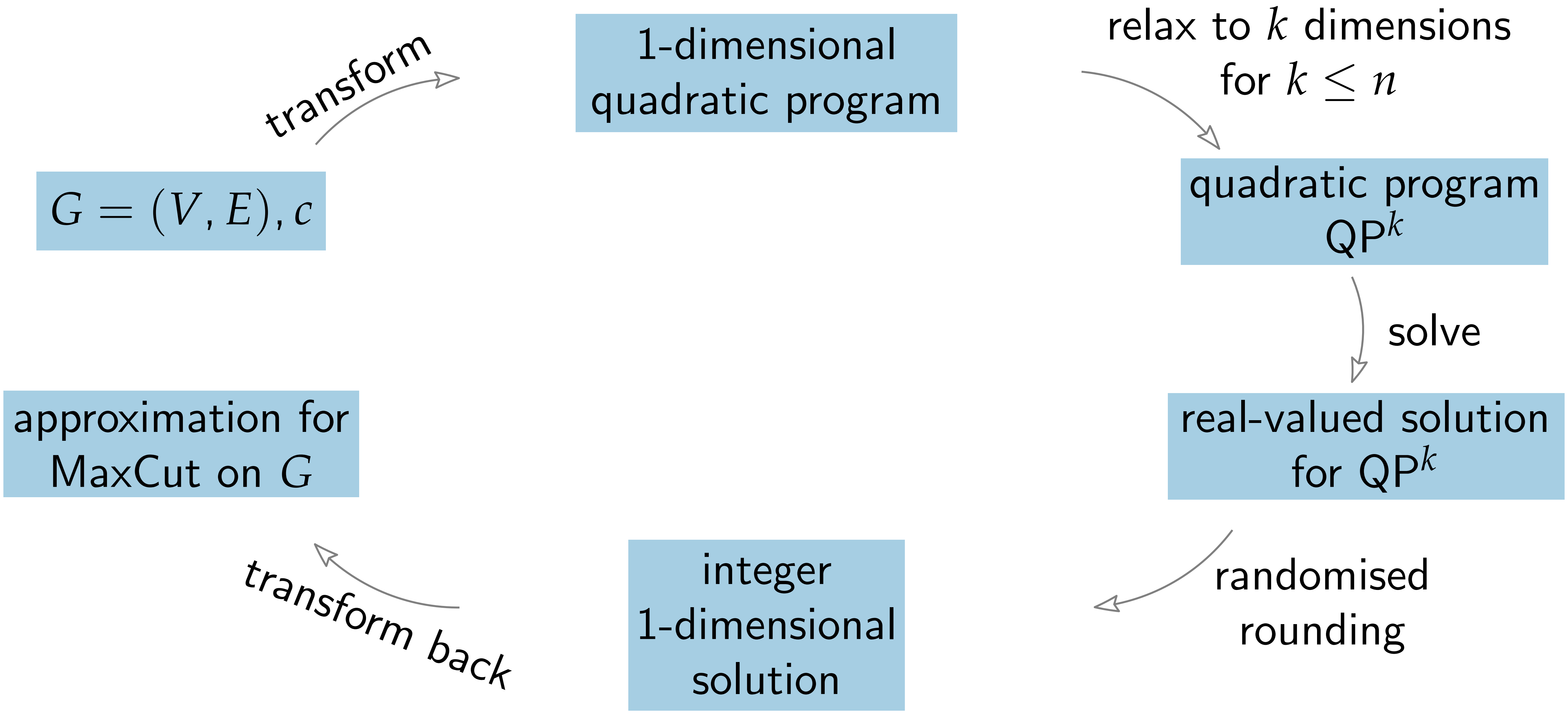
e.g. rounding

Assignment for ILP

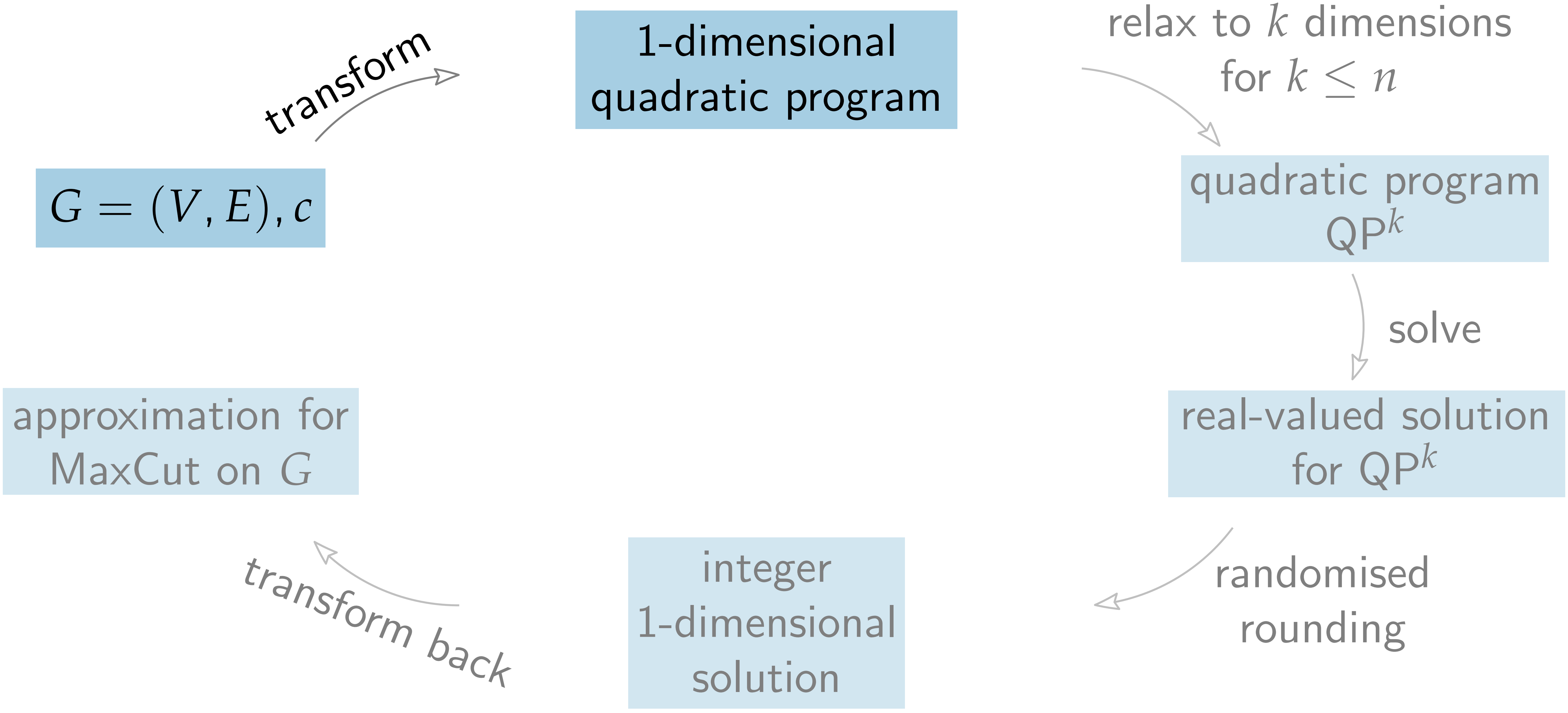
$x^\star$

Solution, approximation, or bound

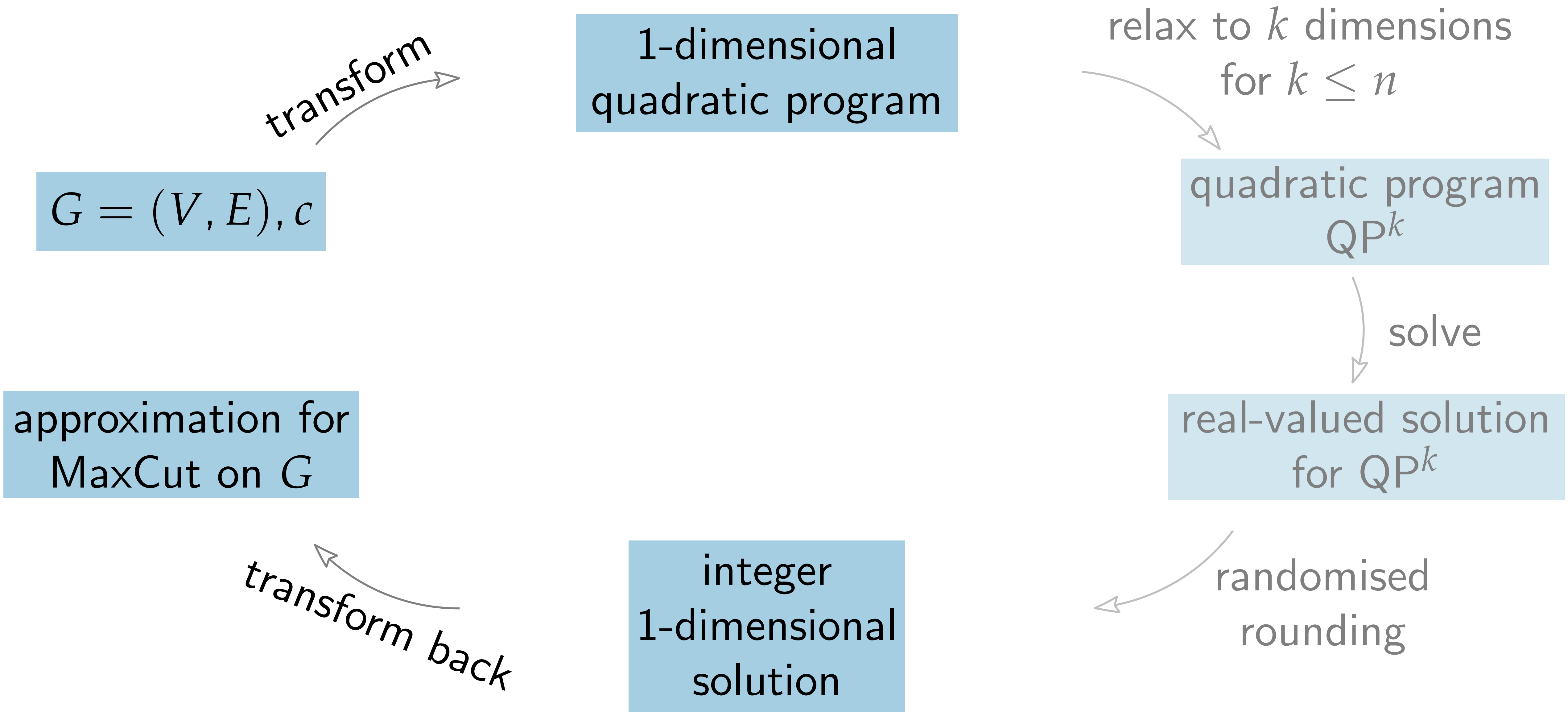
Goemans-Williamson algorithm for MaxCut



Goemans-Williamson algorithm for MaxCut

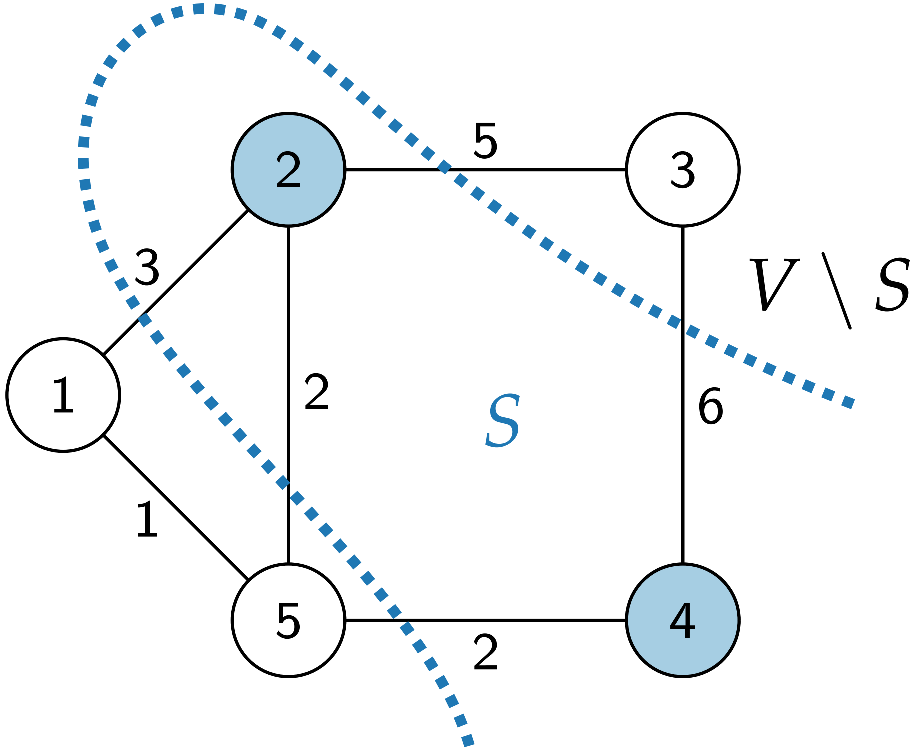


Goemans-Williamson algorithm for MaxCut



QP(G, c)

Idea.



QP(G, c)

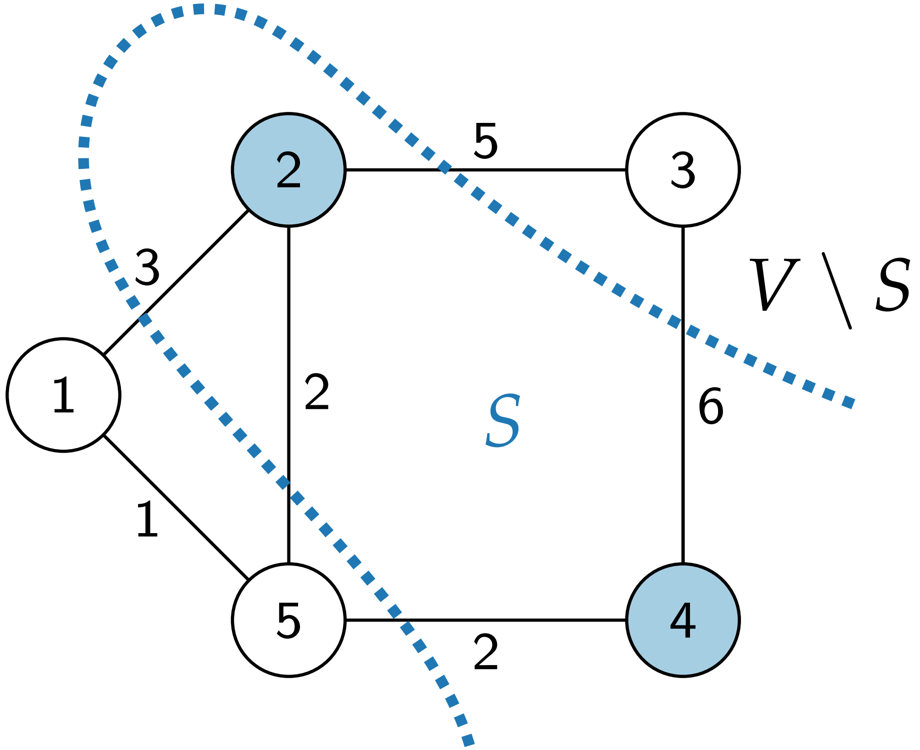
maximize

subject to

QP(G, c)

Idea.

- Indicator variables $x_i \in \{1, -1\}$



QP(G, c)

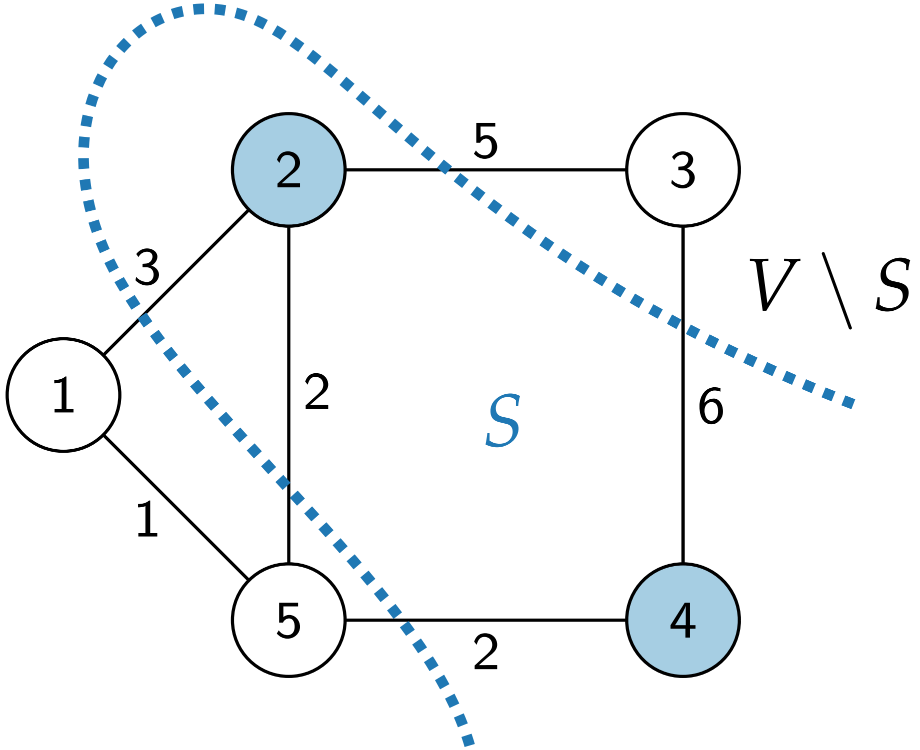
maximize

subject to

QP(G, c)

Idea.

- Indicator variables $x_i \in \{1, -1\}$



QP(G, c)

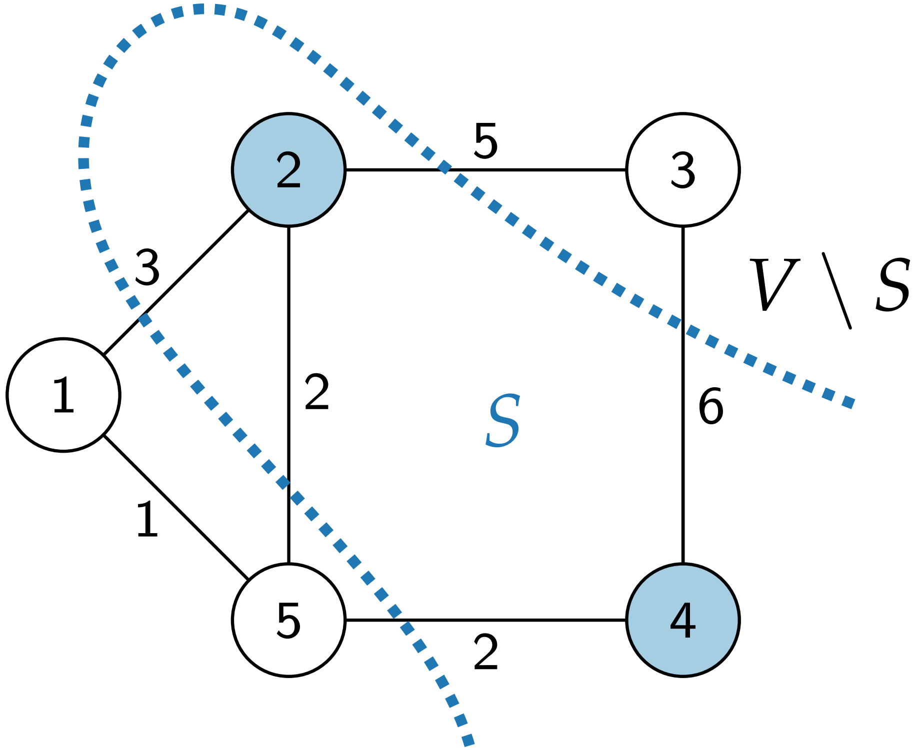
maximize

subject to $x_i^2 = 1$

QP(G, c)

Idea.

- Indicator variables $x_i \in \{1, -1\}$
- $x_i x_j = \begin{cases} 1 & \text{if } i, j \text{ in same partition} \\ -1 & \text{otherwise} \end{cases}$



QP(G, c)

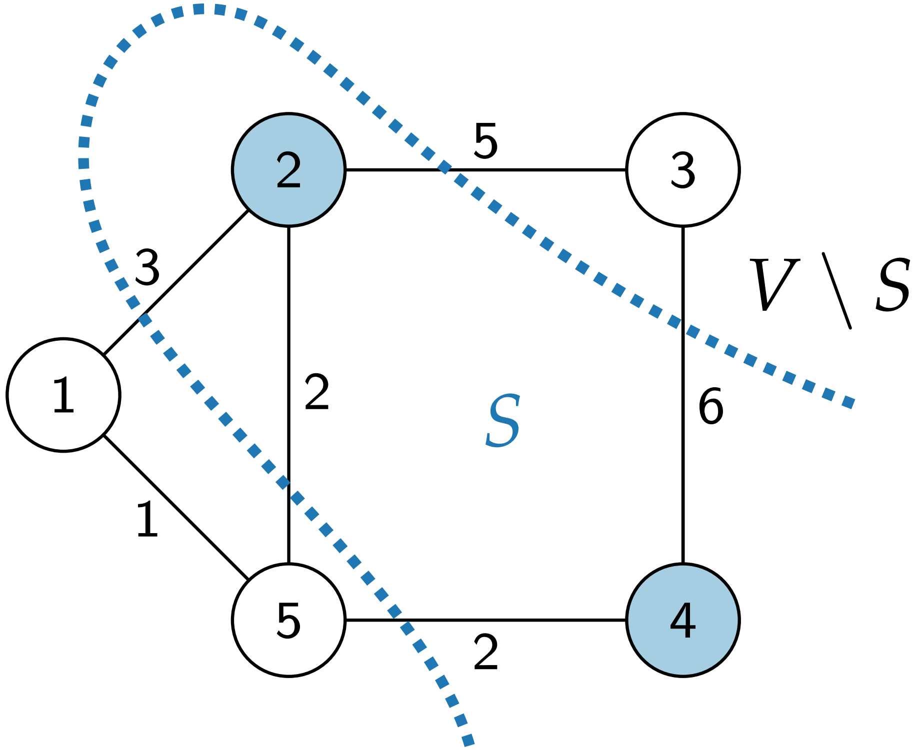
maximize

subject to $\quad x_i^2 = 1$

QP(G, c)

Idea.

- Indicator variables $x_i \in \{1, -1\}$
- $x_i x_j = \begin{cases} 1 & \text{if } i, j \text{ in same partition} \\ -1 & \text{otherwise} \end{cases}$



QP(G, c)

maximize $(1 - x_i x_j)$

subject to $x_i^2 = 1$

QP(G, c)

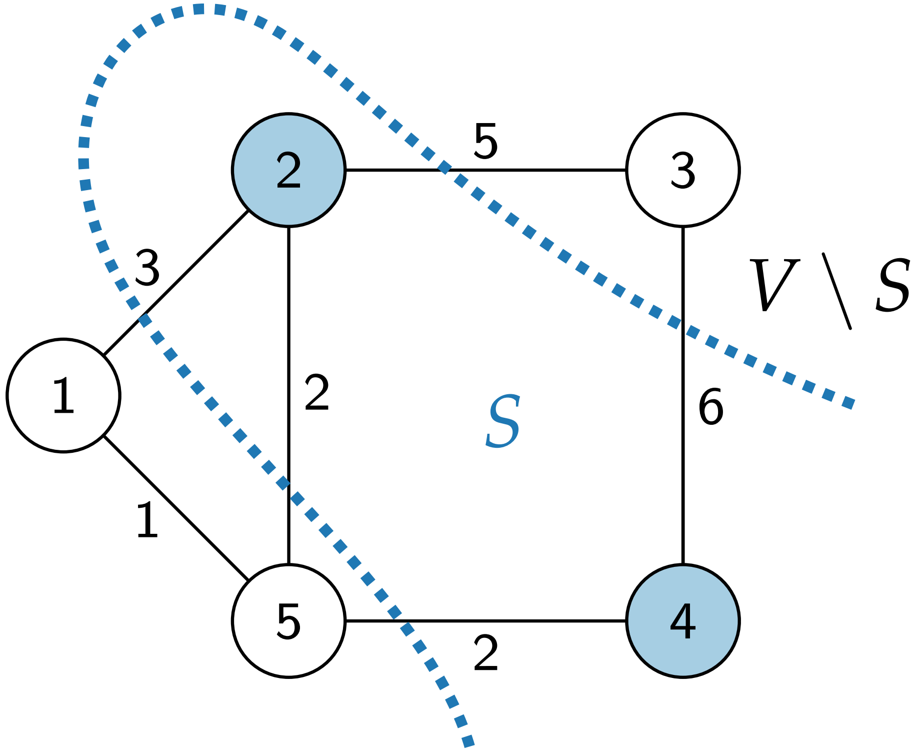
Idea.

- Indicator variables $x_i \in \{1, -1\}$
- $x_i x_j = \begin{cases} 1 & \text{if } i, j \text{ in same partition} \\ -1 & \text{otherwise} \end{cases}$

QP(G, c)

maximize $\quad c_{ij}(1 - x_i x_j)$

subject to $\quad x_i^2 = 1$



- Weight matrix c_{ij}

	1	2	3	4	5
1		3			1
2	3		5		2
3		5		6	
4			6		2
5	1	2		2	

QP(G, c)

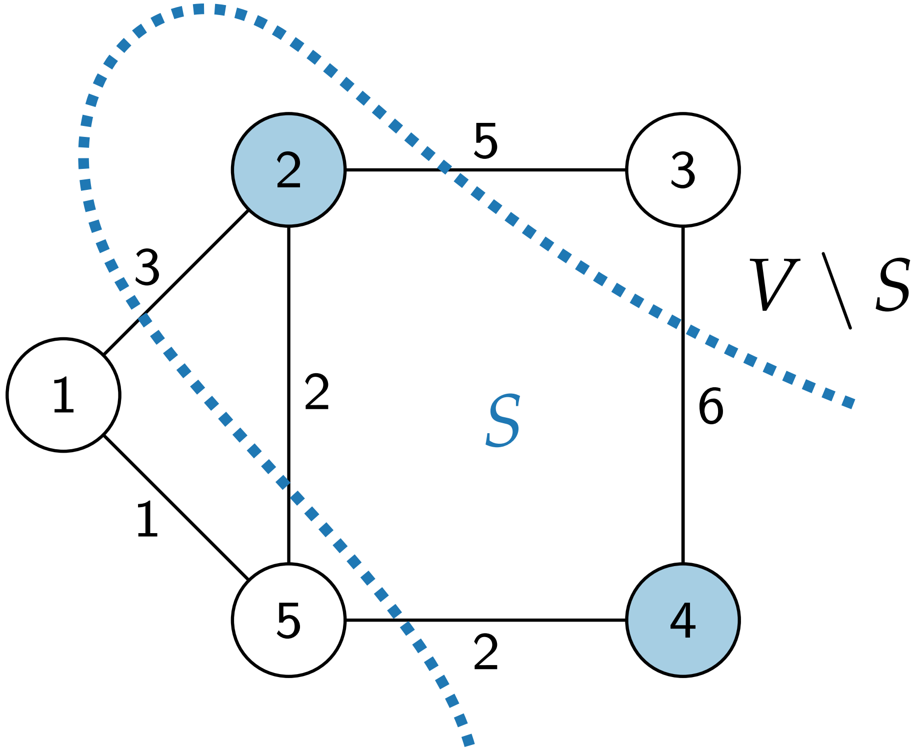
Idea.

- Indicator variables $x_i \in \{1, -1\}$
- $x_i x_j = \begin{cases} 1 & \text{if } i, j \text{ in same partition} \\ -1 & \text{otherwise} \end{cases}$

QP(G, c)

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j)$

subject to $x_i^2 = 1$



- Weight matrix c_{ij}

	1	2	3	4	5
1		3			1
2	3		5		2
3		5		6	
4			6		2
5	1	2		2	

$\mathrm{QP}(G, c)$

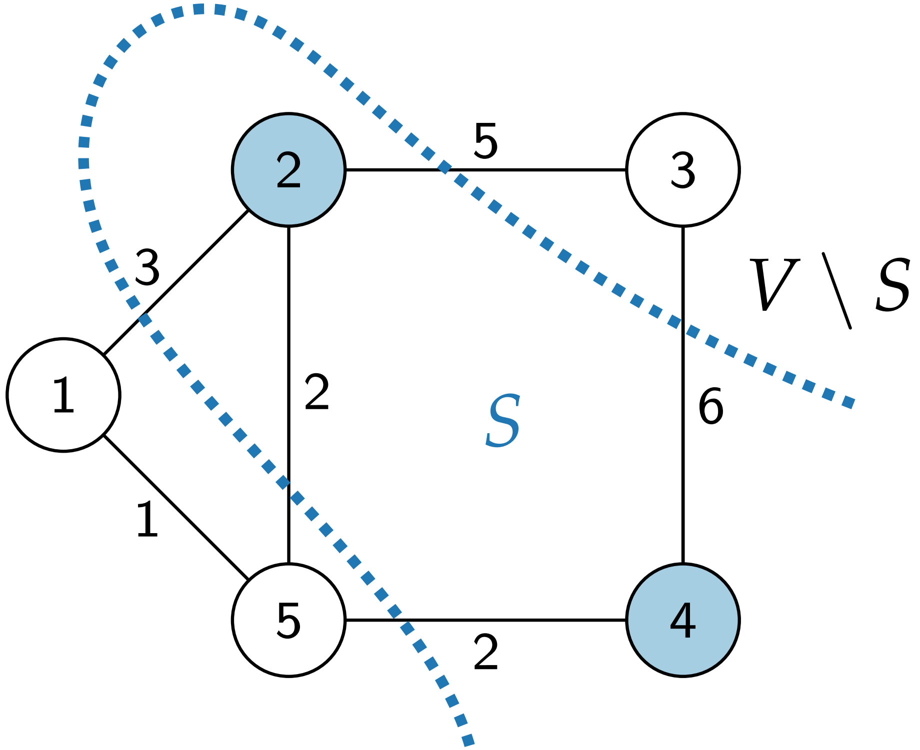
Idea.

- Indicator variables $x_i \in \{1, -1\}$
- $x_i x_j = \begin{cases} 1 & \text{if } i, j \text{ in same partition} \\ -1 & \text{otherwise} \end{cases}$

QP(G, c)

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j)$

subject to $x_i^2 = 1$



- Weight matrix c_{ij}

	1	2	3	4	5
1		3			1
2	3		5		2
3		5		6	
4			6		2
5	1	2		2	

- Solution

$x_2 = x_4 = 1$

$x_1 = x_3 = x_5 = -1$

QP(G, c)

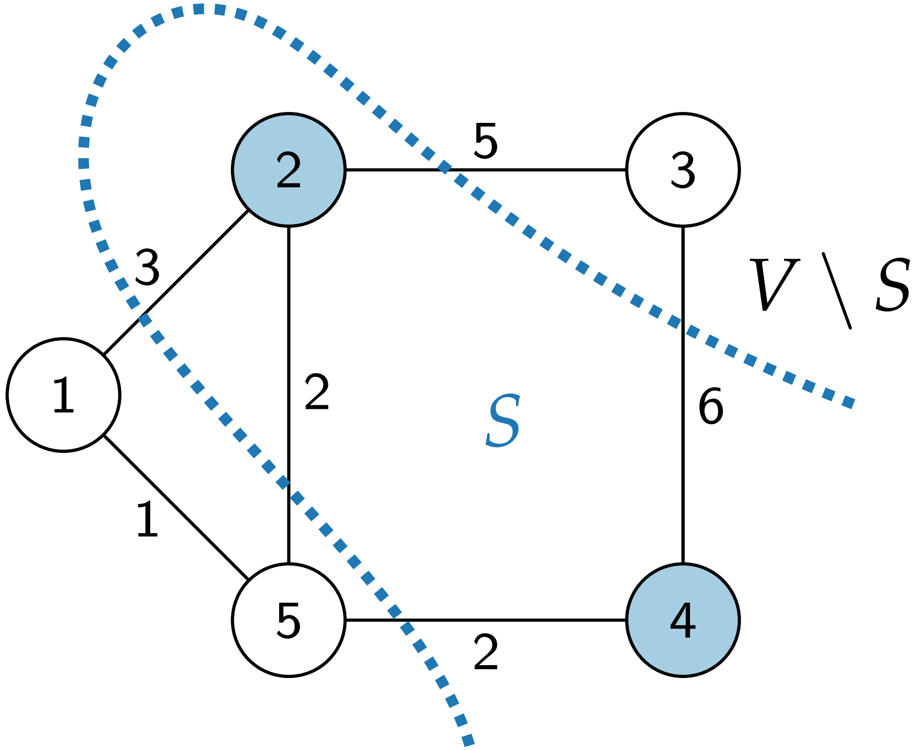
Idea.

- Indicator variables $x_i \in \{1, -1\}$
- $x_i x_j = \begin{cases} 1 & \text{if } i, j \text{ in same partition} \\ -1 & \text{otherwise} \end{cases}$

QP(G, c)

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j)$

subject to $x_i^2 = 1$



- Weight matrix c_{ij}

	1	2	3	4	5
1		3			1
2	3		5		2
3		5		6	
4			6		2
5	1	2		2	

- Solution

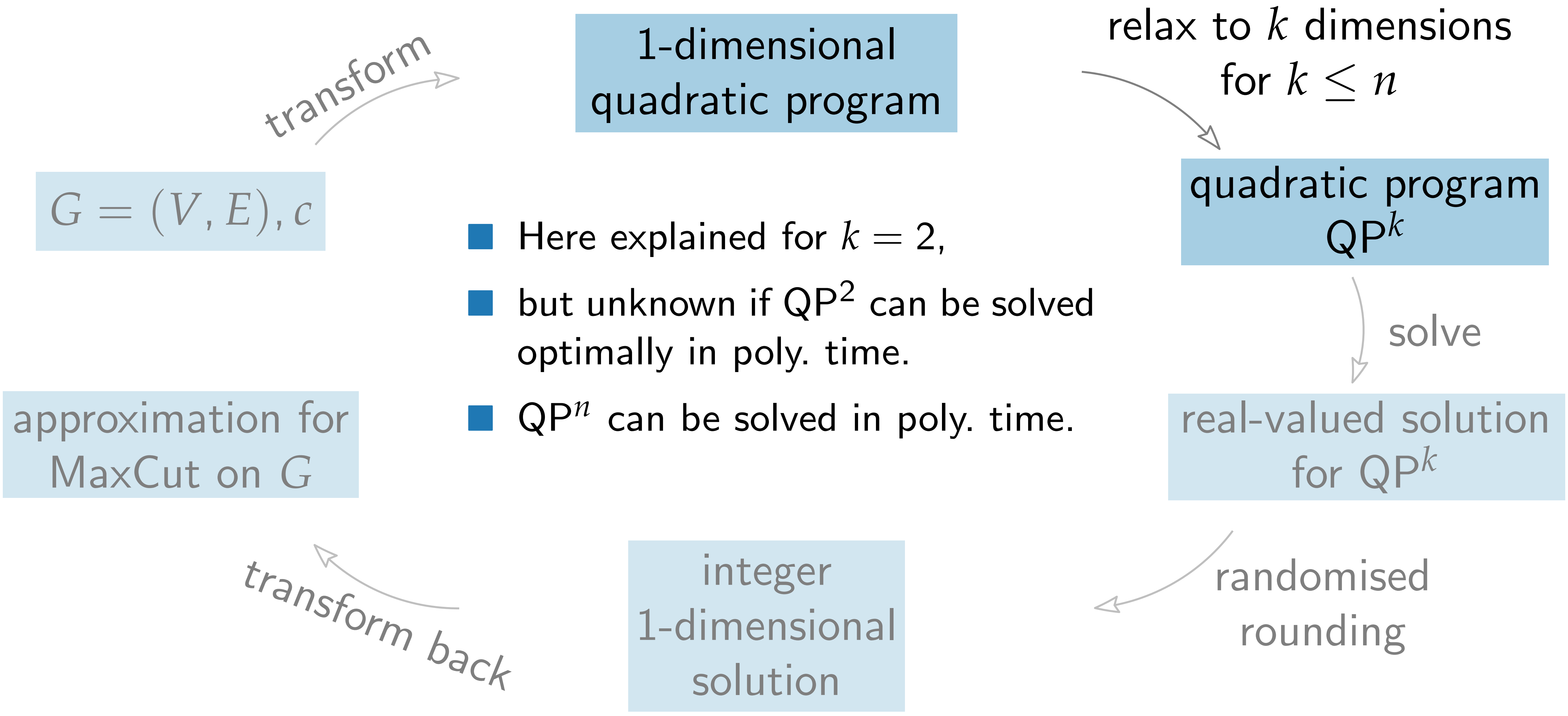
  $x_2 = x_4 = 1$

  $x_1 = x_3 = x_5 = -1$

Note.

- Solving QP(G) is NP-hard.
- Otherwise MaxCut wouldn't be NP-hard.

Goemans-Williamson algorithm for MaxCut



- Here explained for $k = 2$,
- but unknown if QP^2 can be solved optimally in poly. time.
- QP^n can be solved in poly. time.

Relaxation of QP(G, c)

QP$^2(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to $x^i \cdot x^i = 1$

$x^i = (x^i_1, x^i_2) \in \mathbb{R}^2$

Relaxation of QP(G, c)

QP$^2(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to $x^i \cdot x^i = 1$

$x^i = (x^i_1, x^i_2) \in \mathbb{R}^2$

- "$\cdot$" is scalar product.

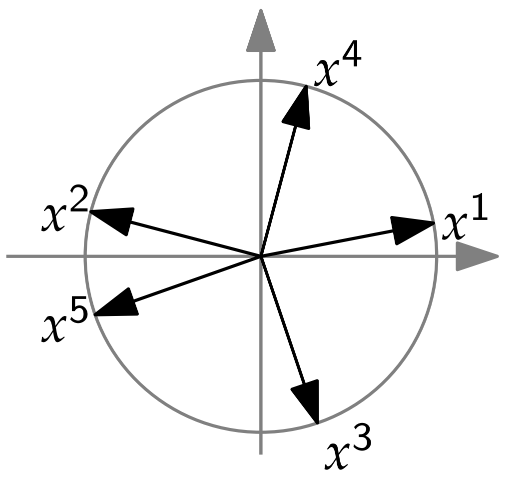
Relaxation of QP(G, c)

QP$^2(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to
$$x^i \cdot x^i = 1$$
$$x^i = (x^i_1, x^i_2) \in \mathbb{R}^2$$

- " $\cdot$ " is scalar product.
- x^i lies on unit circle.



Relaxation of QP(G, c)

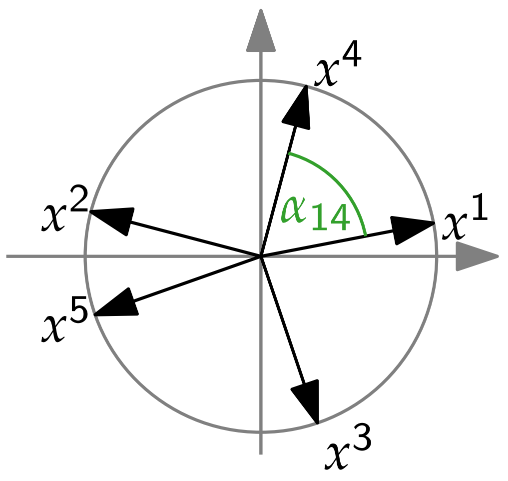
QP$^2(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to $x^i \cdot x^i = 1$

$x^i = (x_1^i, x_2^i) \in \mathbb{R}^2$

- " $\cdot$ " is scalar product.
- x^i lies on unit circle.
- $x^i x^j = x_1^i x_1^j + x_2^i x_2^j = \cos(\alpha_{ij})$ with $0 \leq \alpha_{ij} \leq \pi$.



Relaxation of QP(G, c)

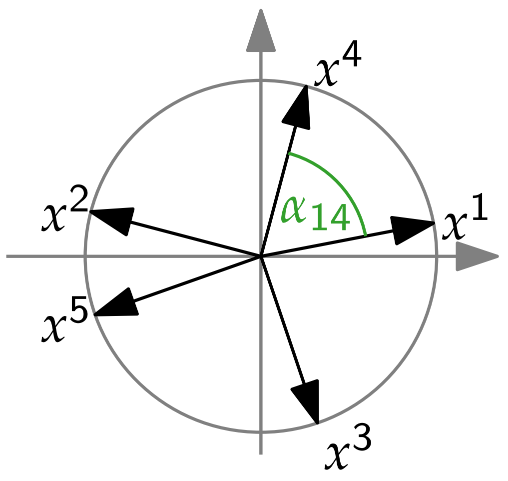
QP$^2(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to $x^i \cdot x^i = 1$

$x^i = (x_1^i, x_2^i) \in \mathbb{R}^2$

- " $\cdot$ " is scalar product.
- x^i lies on unit circle.
- $x^i x^j = x_1^i x_1^j + x_2^i x_2^j = \cos(\alpha_{ij})$ with $0 \leq \alpha_{ij} \leq \pi$.



- We maximize angles α_{ij}: $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - \cos(\alpha_{ij}))$

Relaxation of QP(G, c)

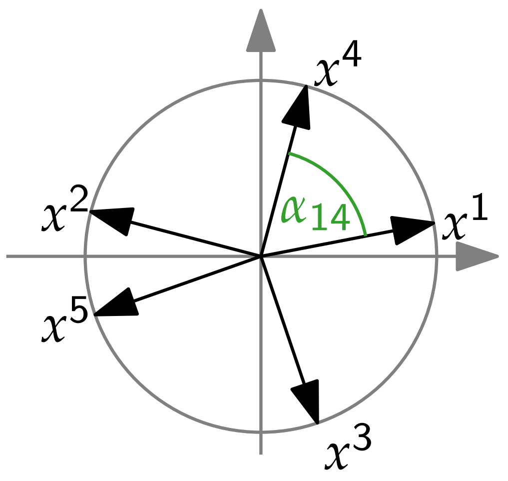
QP$^2(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

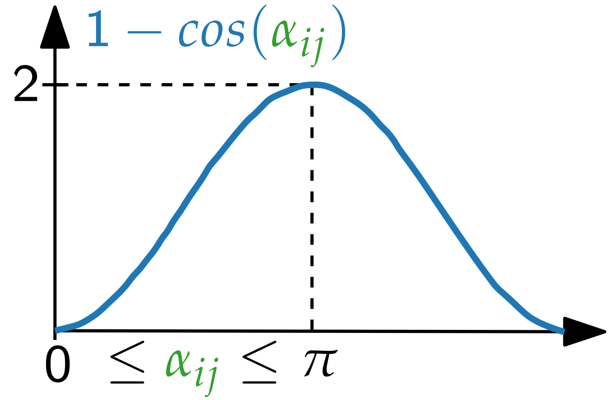
subject to $x^i \cdot x^i = 1$

$x^i = (x^i_1, x^i_2) \in \mathbb{R}^2$

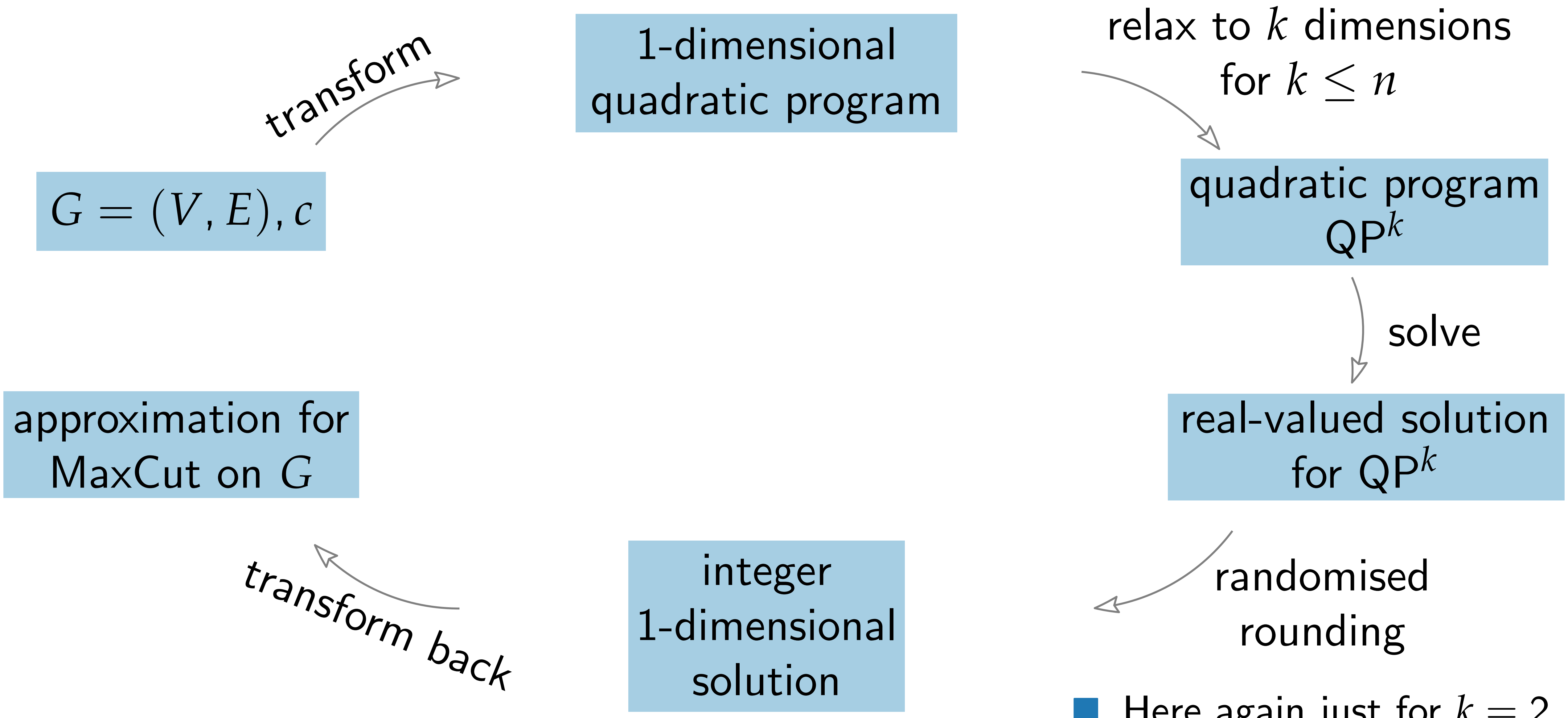
- " $\cdot$ " is scalar product.
- x^i lies on unit circle.
- $x^i x^j = x^i_1 x^j_1 + x^i_2 x^j_2 = \cos(\alpha_{ij})$ with $0 \leq \alpha_{ij} \leq \pi$.



- We maximize angles α_{ij}: $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - \cos(\alpha_{ij}))$
- since larger α_{ij}, increases contribution of c_{ij}.

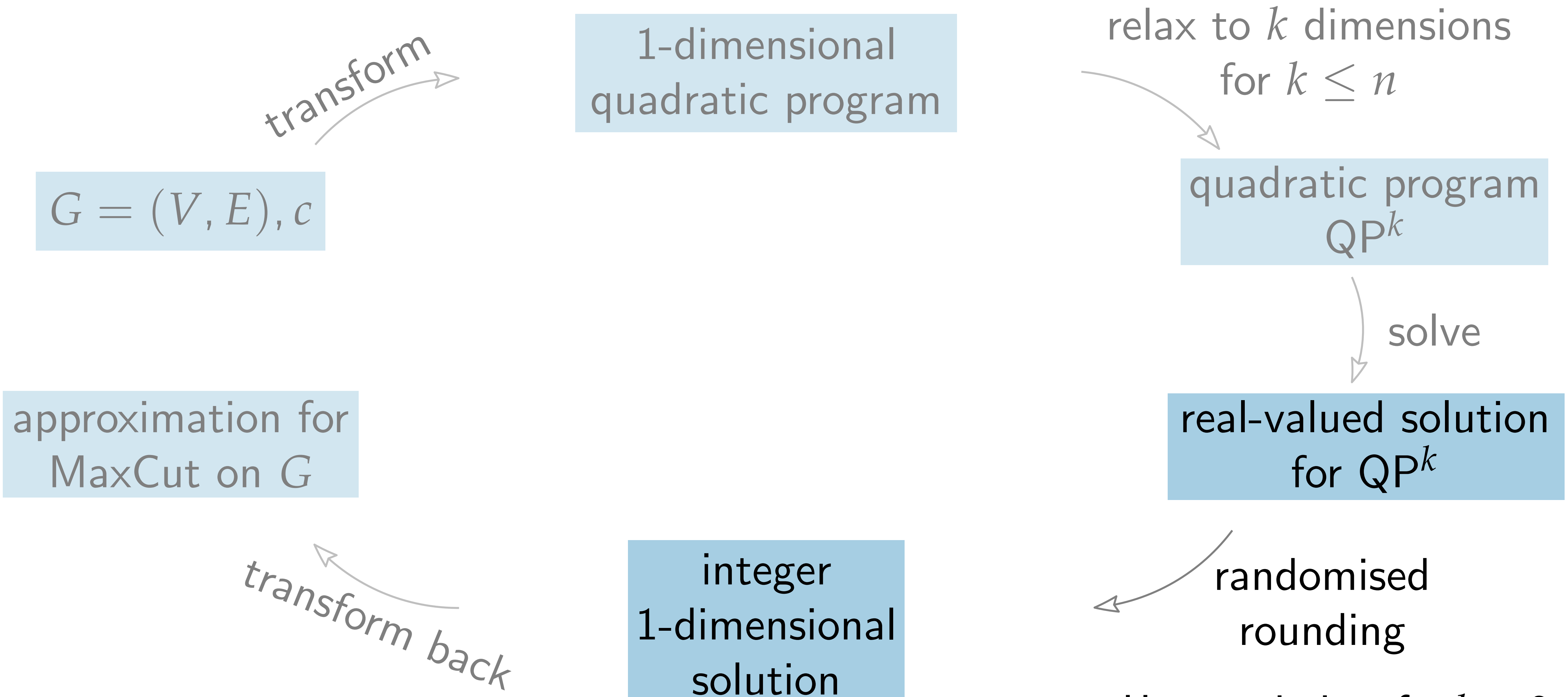


Goemans-Williamson algorithm for MaxCut



- Here again just for $k = 2$.

Goemans-Williamson algorithm for MaxCut



- Here again just for $k = 2$.

Algorithm $\textsc{RandomizedMaxCut}$

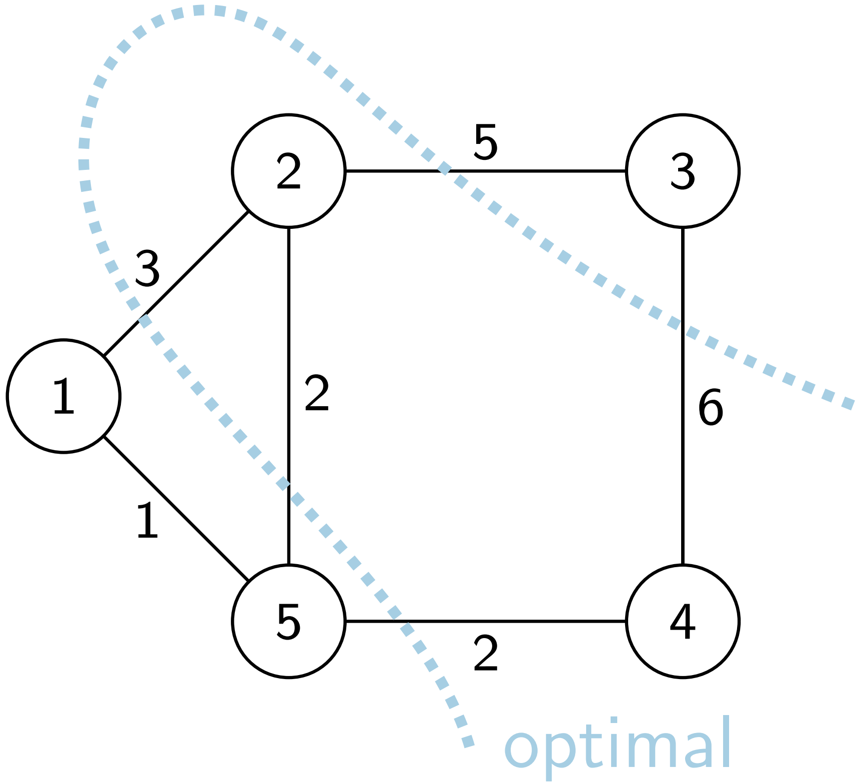
$\textsc{RandomizedMaxCut}(G, c)$

Compute optimal solution $(\tilde{x}^1, \ldots, \tilde{x}^n)$ for $\mathrm{QP}^2(G, c)$
Pick random vector $r \in \mathbb{R}^2$
$S \leftarrow \{i \in V : \tilde{x}^i \cdot r \geq 0\}$
return $c(S, V \setminus S)$

Algorithm RandomizedMaxCut

RandomizedMaxCut(G, c)

Compute optimal solution $(\tilde{x}^1, \dots, \tilde{x}^n)$ for $\text{QP}^2(G, c)$
Pick random vector $r \in \mathbb{R}^2$
$S \leftarrow \{i \in V : \tilde{x}^i \cdot r \geq 0\}$
return $c(S, V \setminus S)$



Algorithm RandomizedMaxCut

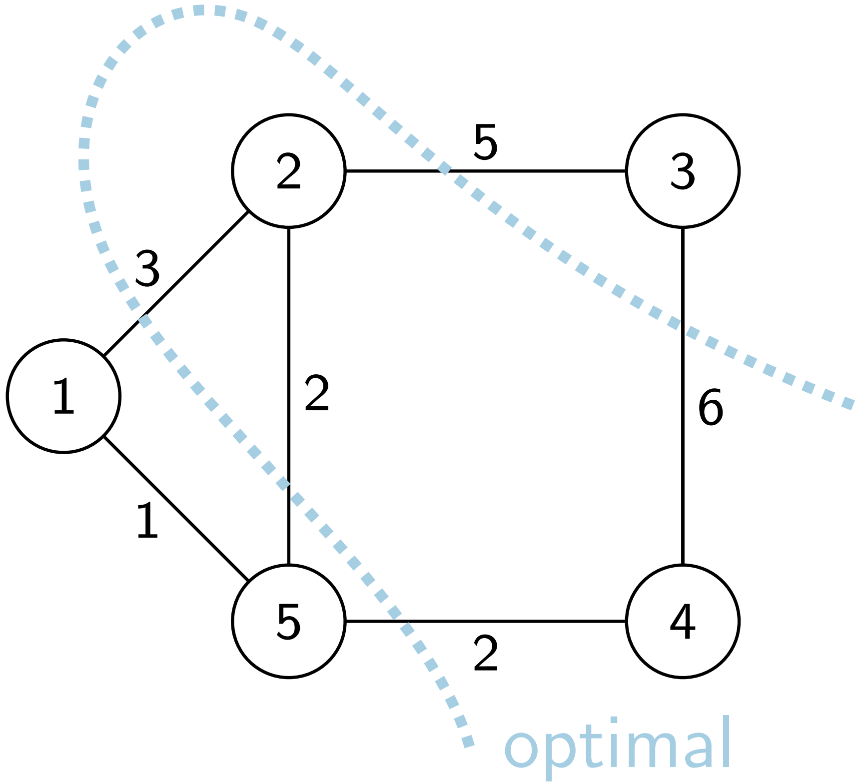
RandomizedMaxCut(G, c)

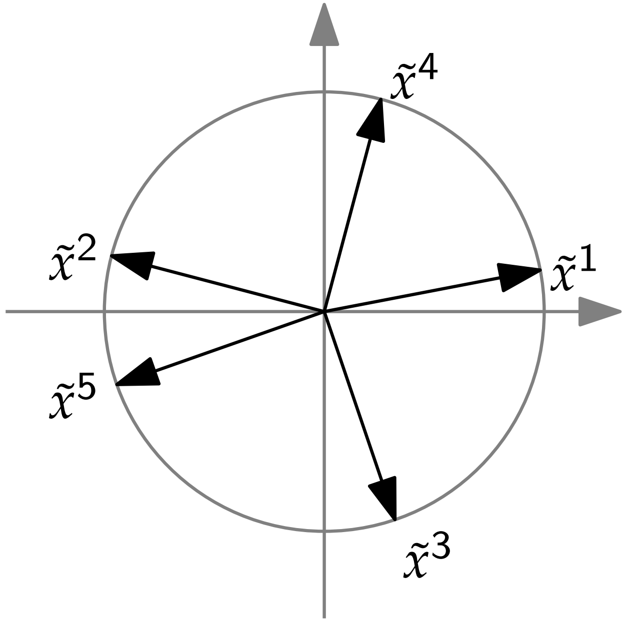
Compute optimal solution $(\tilde{x}^1, \ldots, \tilde{x}^n)$ for QP$^2(G, c)$

Pick random vector $r \in \mathbb{R}^2$

$S \leftarrow \{i \in V : \tilde{x}^i \cdot r \geq 0\}$

return $c(S, V \setminus S)$





(Sketch)

Algorithm RandomizedMaxCut

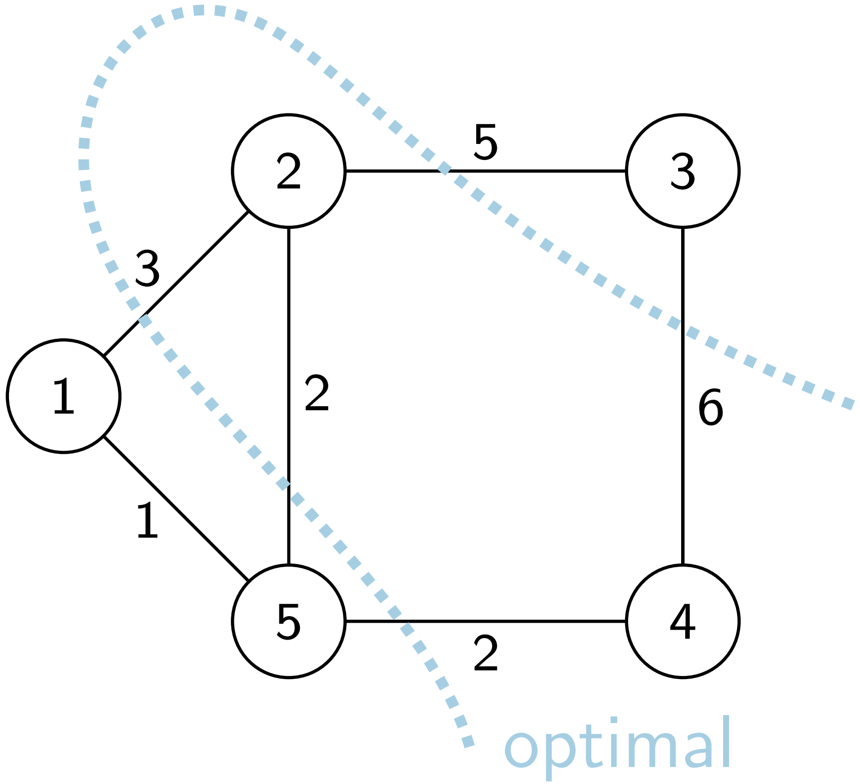
RandomizedMaxCut(G, c)

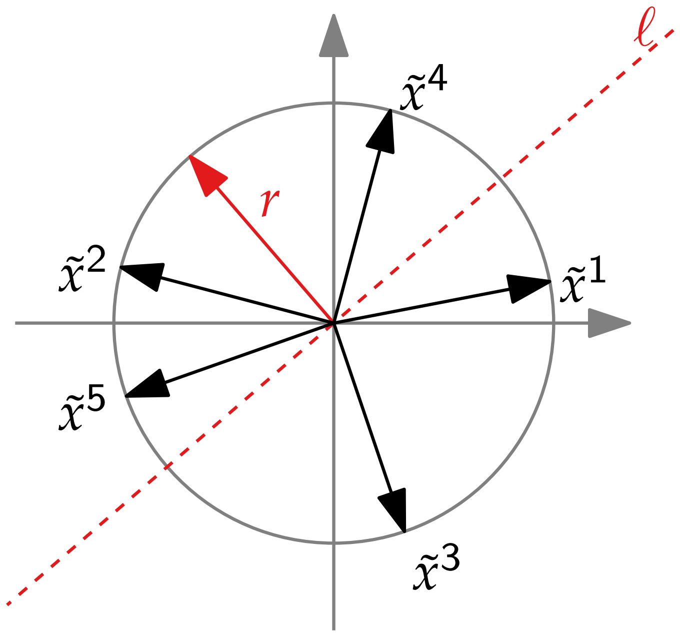
Compute optimal solution $(\tilde{x}^1, \ldots, \tilde{x}^n)$ for $QP^2(G, c)$

Pick random vector $r \in \mathbb{R}^2$

$S \leftarrow \{i \in V : \tilde{x}^i \cdot r \geq 0\}$

return $c(S, V \setminus S)$





(Sketch)

Algorithm RandomizedMaxCut

RandomizedMaxCut(G, c)

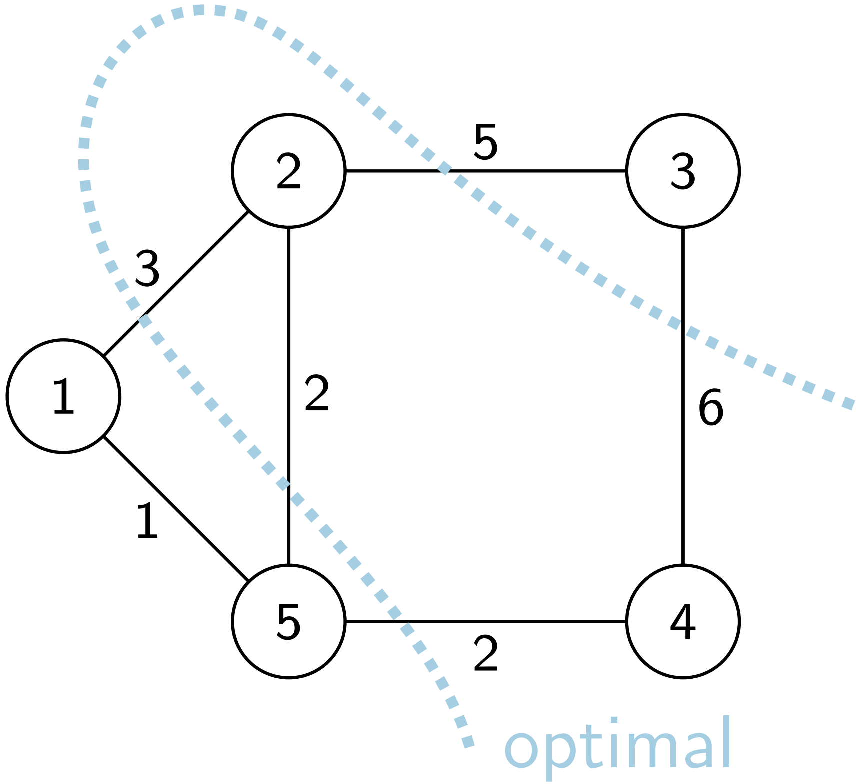
Compute optimal solution $(\tilde{x}^1, \ldots, \tilde{x}^n)$ for $QP^2(G, c)$

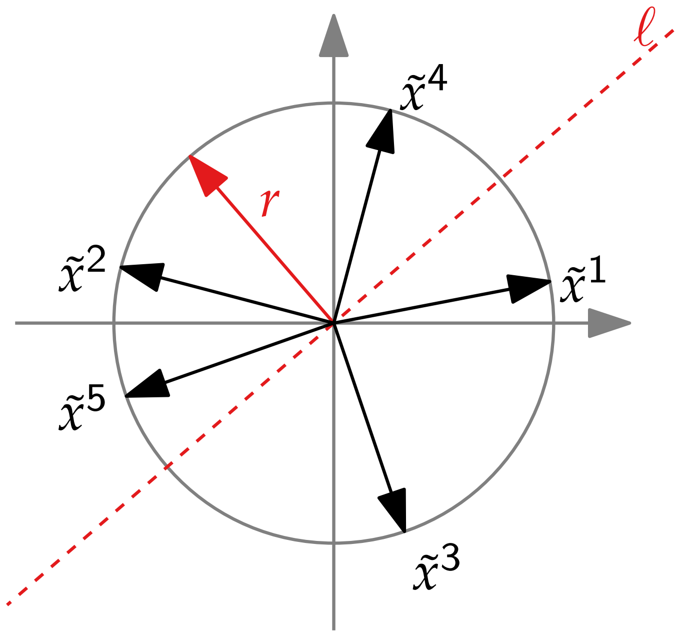
Pick random vector $r \in \mathbb{R}^2$

$S \leftarrow \{i \in V : \tilde{x}^i \cdot r \geq 0\}$

return $c(S, V \setminus S)$

- $\tilde{x}^i$ lies above line ℓ orthogonal to r





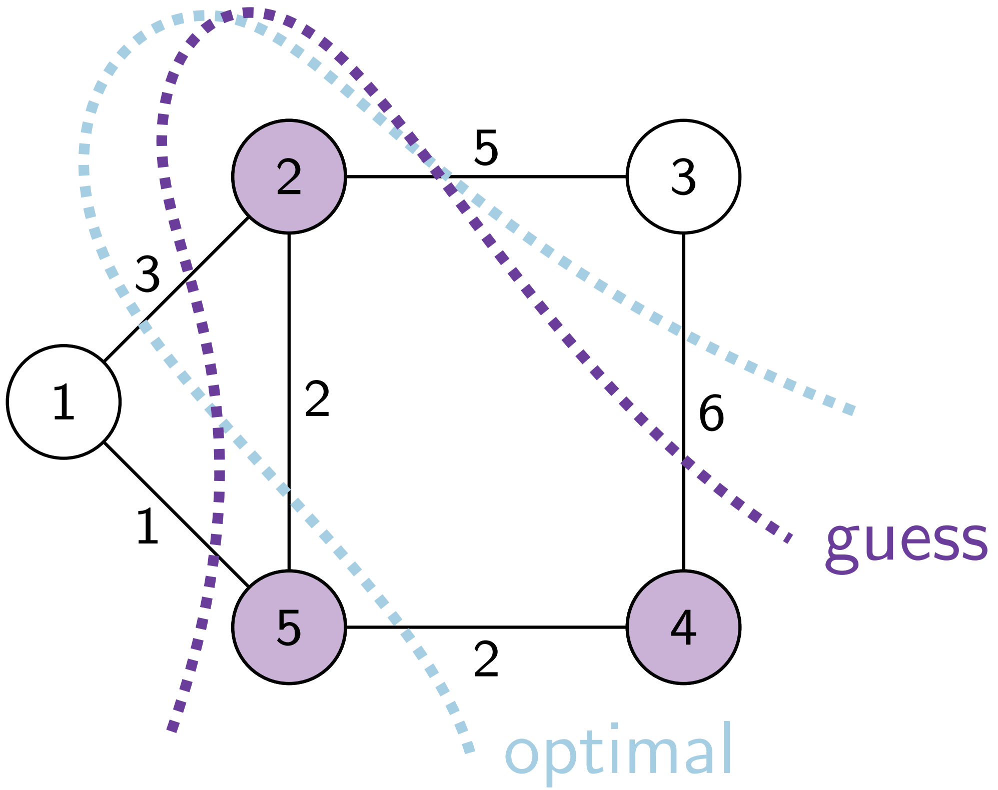
(Sketch)

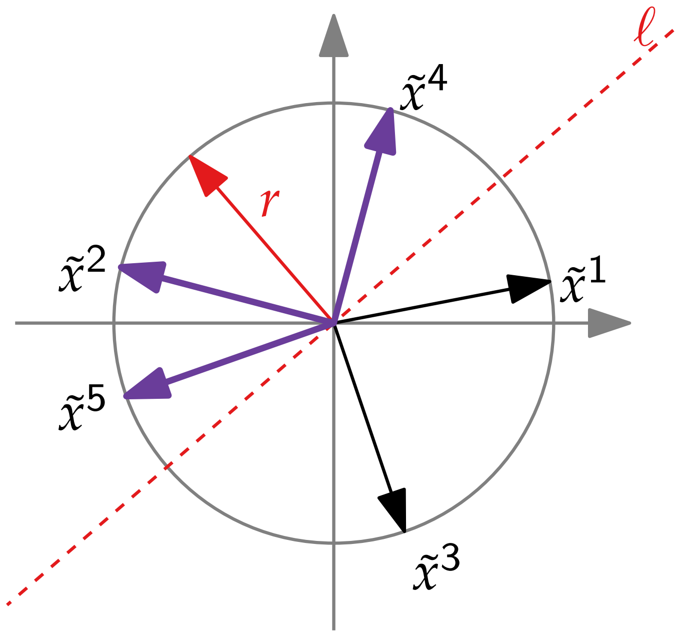
Algorithm RandomizedMaxCut

RandomizedMaxCut(G, c)

Compute optimal solution $(\tilde{x}^1, \dots, \tilde{x}^n)$ for $\text{QP}^2(G, c)$
Pick random vector $r \in \mathbb{R}^2$
$S \leftarrow \{i \in V : \tilde{x}^i \cdot r \geq 0\}$
return $c(S, V \setminus S)$

- $\tilde{x}^i$ lies above line ℓ orthogonal to r





(Sketch)

RandomMaxCut – expected value

Lemma 2.
Let X be the solution of RandomizedMaxCut(G, c).
If r is picked uniformally at random, then

$$\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}.$$

RandomMaxCut – expected value

Lemma 2.
Let X be the solution of RandomizedMaxCut(G, c).
If r is picked uniformally at random, then

$$\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}.$$

Proof.

- $E[X] =$

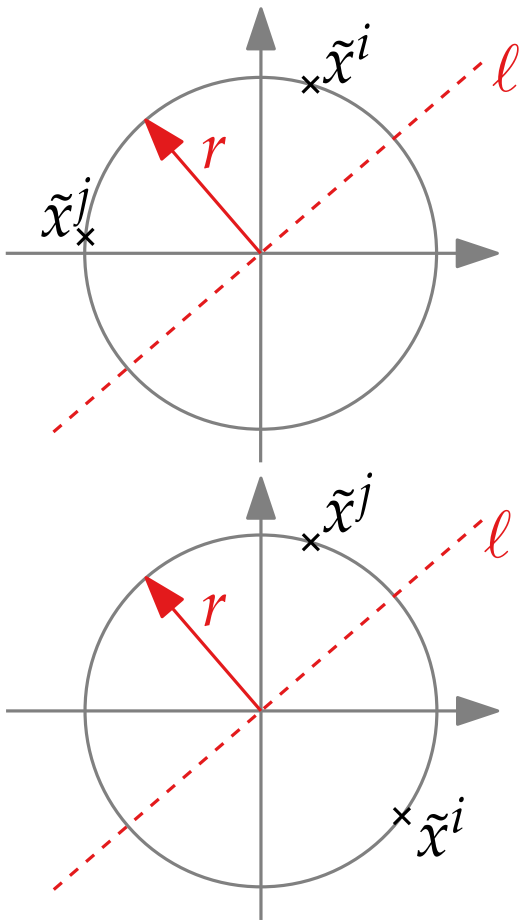
RandomMaxCut – expected value

Lemma 2.
Let X be the solution of RandomizedMaxCut(G, c).
If r is picked uniformally at random, then

$$\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}.$$

Proof.

- $E[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \, \mathrm{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j]$



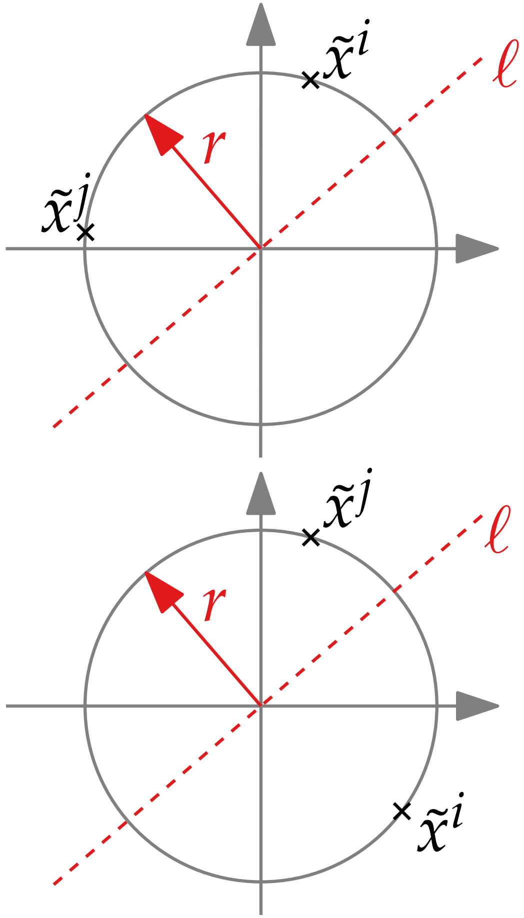
RandomMaxCut – expected value

Lemma 2.
Let X be the solution of RandomizedMaxCut(G, c).
If r is picked uniformally at random, then

$$\mathsf{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}.$$

Proof.

- $E[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}$ P[ℓ separates $\tilde{x}^i, \tilde{x}^j$]
- P[ℓ separates $\tilde{x}^i, \tilde{x}^j$] $=$



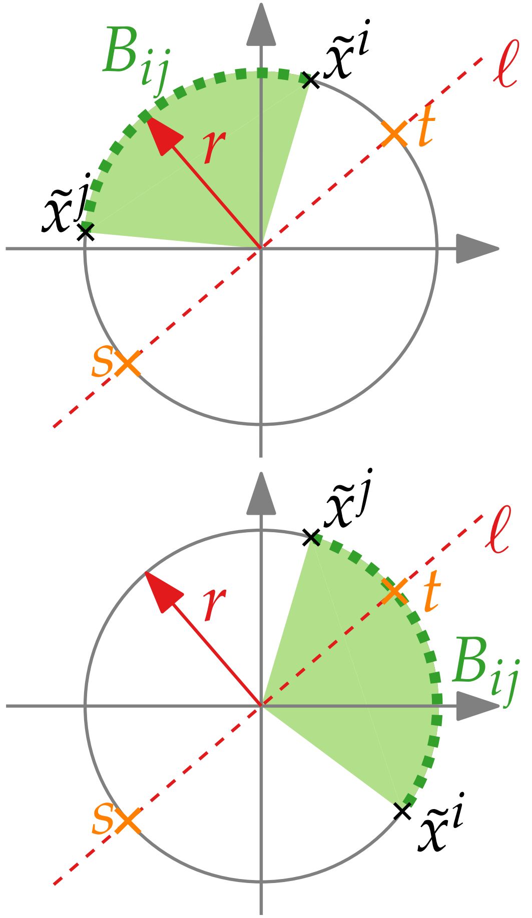
RandomMaxCut – expected value

Lemma 2.
Let X be the solution of RandomizedMaxCut(G, c).
If r is picked uniformly at random, then

$$\mathsf{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}.$$

Proof.

- $E[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \, \mathsf{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j]$
- $\mathsf{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j] = \mathsf{P}[s \text{ or } t \text{ lies on } B_{ij}]$
  $=$



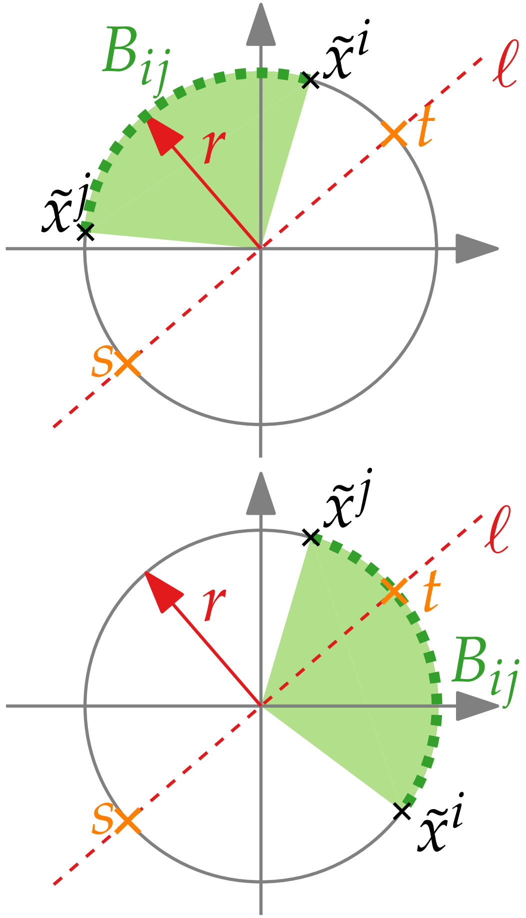
RandomMaxCut – expected value

Lemma 2.
Let X be the solution of RandomizedMaxCut(G, c).
If r is picked uniformally at random, then

$$\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}.$$

Proof.

- $E[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \, \mathrm{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j]$
- $\mathrm{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j] = \mathrm{P}[s \text{ or } t \text{ lies on } B_{ij}]$
  $=$
- B_{ij} has length $\alpha_{ij} = \arccos(\tilde{x}^i \cdot \tilde{x}^j)$.



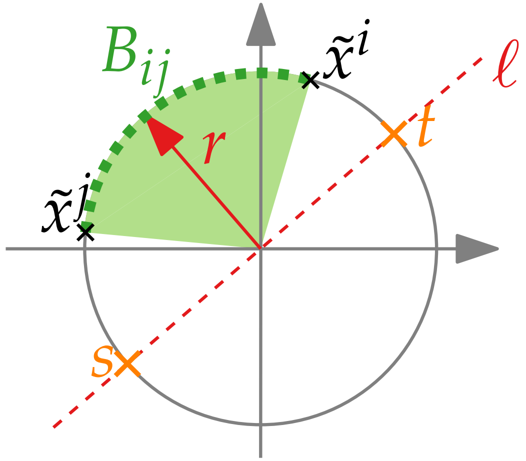
RandomMaxCut – expected value

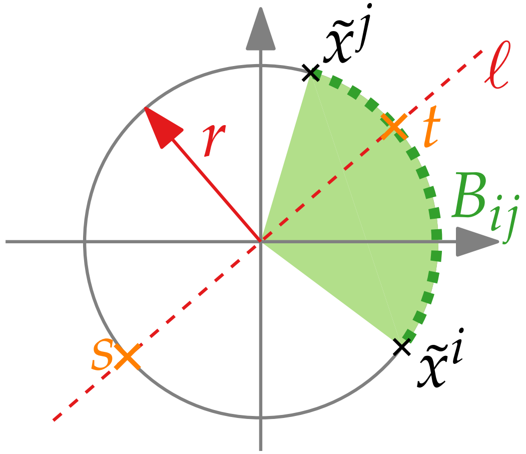
Lemma 2.
Let X be the solution of RandomizedMaxCut(G, c).
If r is picked uniformly at random, then

$$\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}.$$

Proof.

- $E[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}\, \mathrm{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j]$
- $\mathrm{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j] = \mathrm{P}[s \text{ or } t \text{ lies on } B_{ij}]$
  $= \frac{\alpha_{ij}}{2\pi} + \frac{\alpha_{ij}}{2\pi} =$
- B_{ij} has length $\alpha_{ij} = \arccos(\tilde{x}^i \cdot \tilde{x}^j)$.





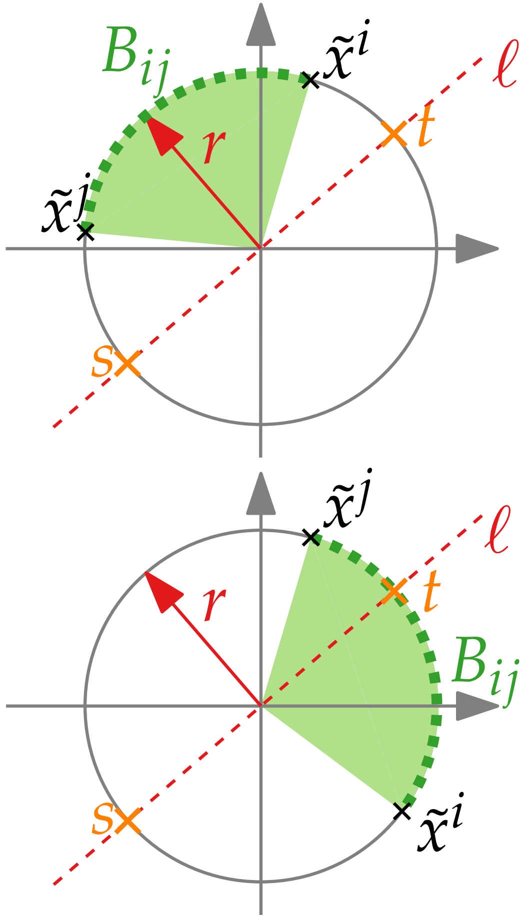
RandomMaxCut – expected value

Lemma 2.
Let X be the solution of RandomizedMaxCut(G, c).
If r is picked uniformly at random, then

$$\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}.$$

Proof.

- $E[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \, \mathrm{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j]$
- $\mathrm{P}[\ell \text{ separates } \tilde{x}^i, \tilde{x}^j] = \mathrm{P}[s \text{ or } t \text{ lies on } B_{ij}]$
  $= \frac{\alpha_{ij}}{2\pi} + \frac{\alpha_{ij}}{2\pi} = \frac{\alpha_{ij}}{\pi}$
- B_{ij} has length $\alpha_{ij} = \arccos(\tilde{x}^i \cdot \tilde{x}^j)$.



RandomMaxCut – quality

Theorem 3.
Let X be the solution of RandomizedMaxCut(G, c).
Then

$$\frac{\mathrm{E}[X]}{\mathrm{OPT}(G,c)} \geq 0.8785.$$

RandomMaxCut – quality

Theorem 3.
Let X be the solution of RandomizedMaxCut(G, c).
Then

$$\frac{\mathsf{E}[X]}{\mathsf{OPT}(G,c)} \geq 0.8785.$$

Proof.

- Lemma 2: $\mathsf{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}$

RandomMaxCut – quality

Theorem 3.
Let X be the solution of RandomizedMaxCut(G, c).
Then
$$\frac{\mathrm{E}[X]}{\mathrm{OPT}(G,c)} \geq 0.8785.$$

Proof.

- Lemma 2: $\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}$
- Optimal solution for QP^2:

RandomMaxCut – quality

Theorem 3.
Let X be the solution of RandomizedMaxCut(G, c).
Then

$$\frac{\mathsf{E}[X]}{\mathsf{OPT}(G,c)} \geq 0.8785.$$

Proof.

- Lemma 2: $\mathsf{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}$
- Optimal solution for QP^2:

$$\mathsf{QP}^2(G,c) = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{1 - \cos(\alpha_{ij})}{2}$$

RandomMaxCut – quality

Theorem 3.
Let X be the solution of RandomizedMaxCut(G, c). Then

$$\frac{\mathsf{E}[X]}{\mathsf{OPT}(G,c)} \geq 0.8785.$$

Proof.

- Lemma 2: $\mathsf{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}$
- Optimal solution for QP^2:

$$QP^2(G,c) = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{1-\cos(\alpha_{ij})}{2}$$

- $QP^2(G,c)$ is relaxation of $QP(G,c)$:

$$QP^2(G,c) \geq \mathsf{OPT}(G,c)$$

RandomMaxCut – quality

Theorem 3.
Let X be the solution of RandomizedMaxCut(G, c).
Then
$$\frac{\mathrm{E}[X]}{\mathrm{OPT}(G,c)} \geq 0.8785.$$

Proof.

- Lemma 2: $\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}$

- Optimal solution for QP^2:
$$\mathrm{QP}^2(G,c) = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{1-\cos(\alpha_{ij})}{2}$$

- $\mathrm{QP}^2(G,c)$ is relaxation of $\mathrm{QP}(G,c)$:
$$\mathrm{QP}^2(G,c) \geq \mathrm{OPT}(G,c)$$

- $\frac{\mathrm{E}[X]}{\mathrm{OPT}(G,c)} \geq \frac{\mathrm{E}[X]}{\mathrm{QP}^2(G,c)}$

RandomMaxCut – quality

Theorem 3.
Let X be the solution of RandomizedMaxCut(G, c). Then

$$\frac{\mathrm{E}[X]}{\mathrm{OPT}(G,c)} \geq 0.8785.$$

Proof.

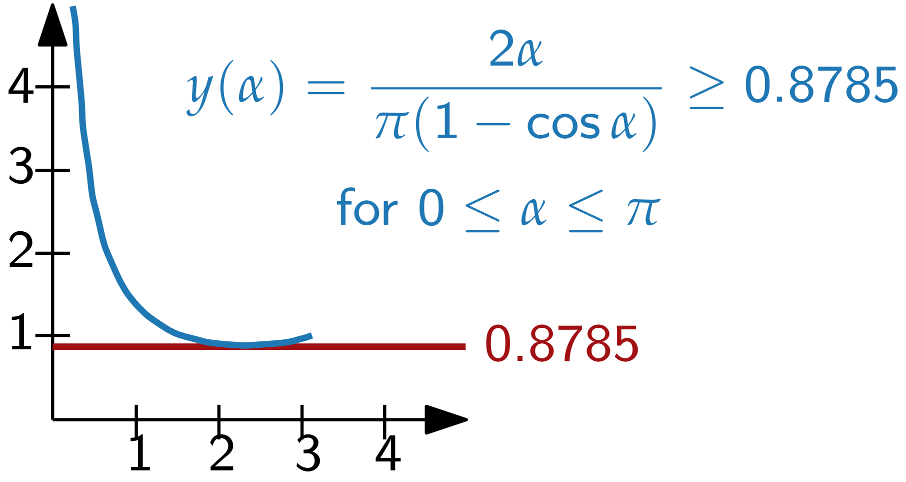
- Lemma 2: $\mathrm{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{\alpha_{ij}}{\pi}$
- Optimal solution for QP^2:

$$\mathrm{QP}^2(G,c) = \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij} \frac{1-\cos(\alpha_{ij})}{2}$$

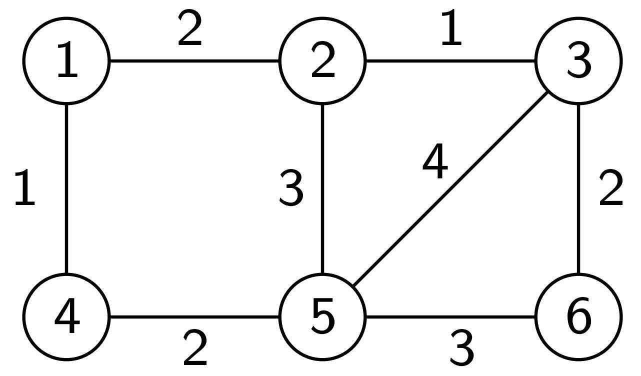
- $\mathrm{QP}^2(G,c)$ is relaxation of $\mathrm{QP}(G,c)$:

$$\mathrm{QP}^2(G,c) \geq \mathrm{OPT}(G,c)$$

- $\frac{\mathrm{E}[X]}{\mathrm{OPT}(G,c)} \geq \frac{\mathrm{E}[X]}{\mathrm{QP}^2(G,c)}$
- $\frac{\alpha_{ij}}{\pi} \geq \frac{1-\cos(\alpha_{ij})}{2} \cdot 0.8785$



Example



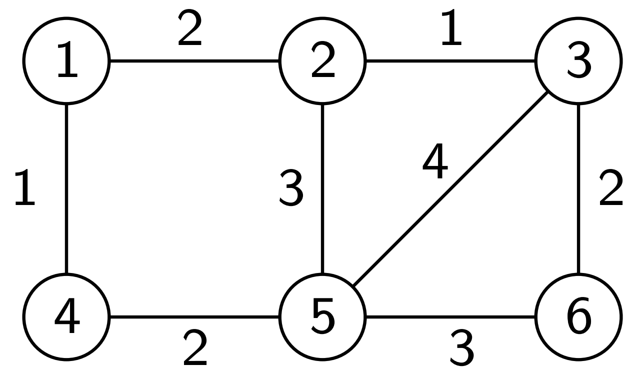
Example

1. Step: Build QP

$$\begin{aligned} &\textbf{maximize} && \tfrac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j) \\ &\textbf{subject to} && x_i^2 = 1 \end{aligned}$$

Weight matrix c_{ij}

	1	2	3	4	5	6
1		2		1		
2	2		1		3	
3		1			4	2
4	1				2	
5		3	4	2		3
6			2		3	



Example

1. Step: Build QP

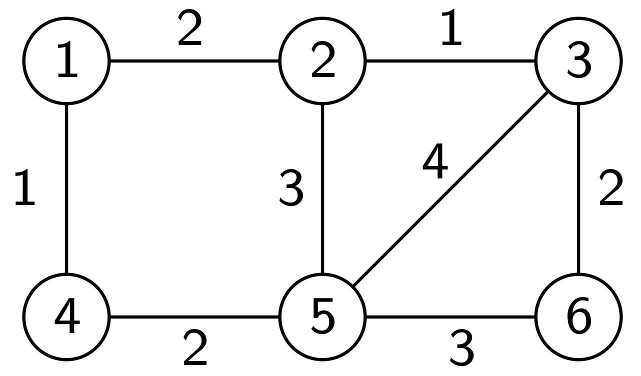
$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j) \\ \textbf{subject to} \quad & x_i^2 = 1 \end{aligned}$$

2. Step: Relax QP to QP^2

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j) \\ \textbf{subject to} \quad & x^i \cdot x^i = 1 \\ & x^i = (x_1^i, x_2^i) \in \mathbb{R}^2 \end{aligned}$$

Weight matrix c_{ij}

	1	2	3	4	5	6
1		2		1		
2	2		1		3	
3		1			4	2
4	1				2	
5		3	4	2		3
6			2		3	



Example

1. Step: Build QP

maximize $\frac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j)$

subject to $x_i^2 = 1$

Weight matrix c_{ij}

	1	2	3	4	5	6
1		2		1		
2	2		1		3	
3		1			4	2
4	1				2	
5		3	4	2		3
6			2		3	

2. Step: Relax QP to QP^2

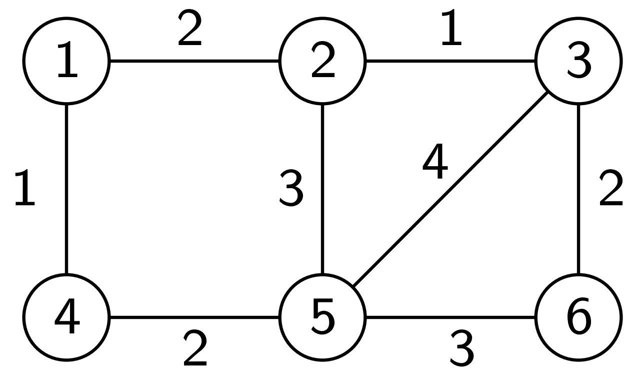
maximize $\frac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

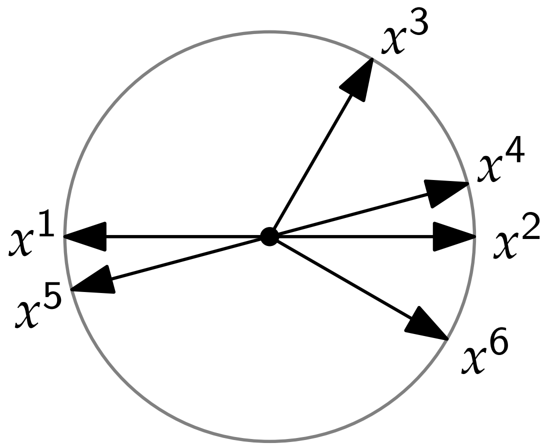
subject to $x^i \cdot x^i = 1$

$x^i = (x_1^i, x_2^i) \in \mathbb{R}^2$

3. Step: Solve QP^2

Variable	x^1	x^2	x^3	x^4	x^5	x^6
Angle	0	180	120	165	345	210





Example

1. Step: Build QP

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j) \\ \textbf{subject to} \quad & x_i^2 = 1 \end{aligned}$$

Weight matrix c_{ij}

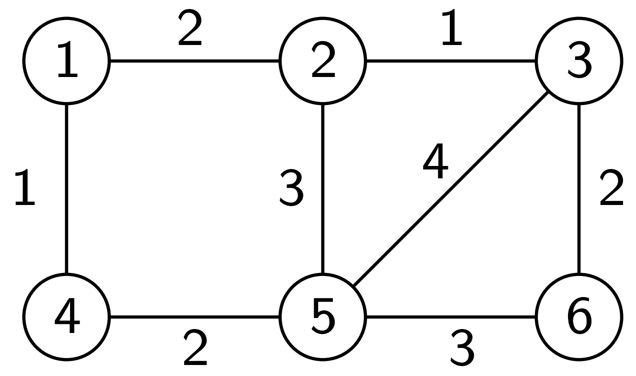
	1	2	3	4	5	6
1		2		1		
2	2		1		3	
3		1			4	2
4	1				2	
5		3	4	2		3
6			2		3	

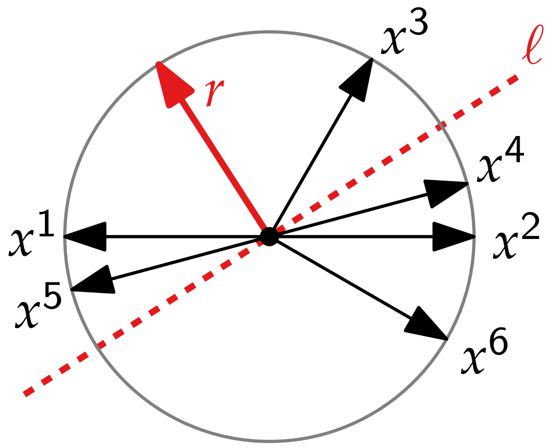
2. Step: Relax QP to QP^2

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j) \\ \textbf{subject to} \quad & x^i \cdot x^i = 1 \\ & x^i = (x_1^i, x_2^i) \in \mathbb{R}^2 \end{aligned}$$

3. Step: Solve QP^2

Variable	x^1	x^2	x^3	x^4	x^5	x^6
Angle	0	180	120	165	345	210





4. Step: Guess r

Example

1. Step: Build QP

maximize $\frac{1}{2} \sum_{j=1}^{6} \sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j)$

subject to $x_i^2 = 1$

Weight matrix c_{ij}

	1	2	3	4	5	6
1		2		1		
2	2		1		3	
3		1			4	2
4	1				2	
5		3	4	2		3
6			2		3	

2. Step: Relax QP to QP^2

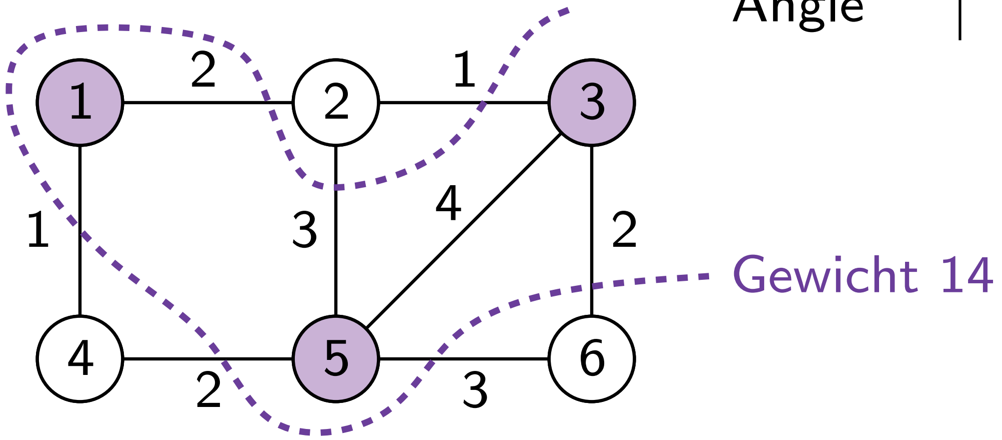
maximize $\frac{1}{2} \sum_{j=1}^{6} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

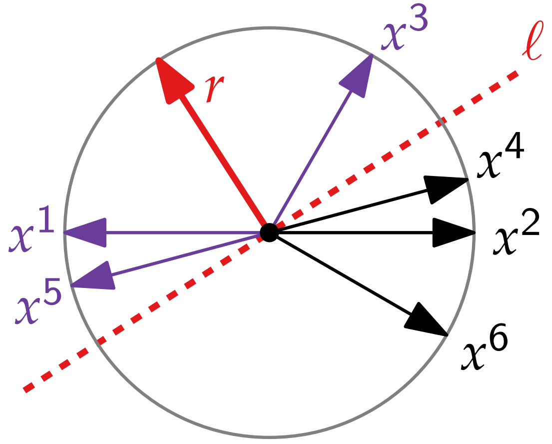
subject to $x^i \cdot x^i = 1$

$x^i = (x_1^i, x_2^i) \in \mathbb{R}^2$

3. Step: Solve QP^2

Variable	x^1	x^2	x^3	x^4	x^5	x^6
Angle	0	180	120	165	345	210





4. Step: Guess r

5. Step: Derive S

Example

1. Step: Build QP

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x_i x_j) \\ \textbf{subject to} \quad & x_i^2 = 1 \end{aligned}$$

Weight matrix c_{ij}

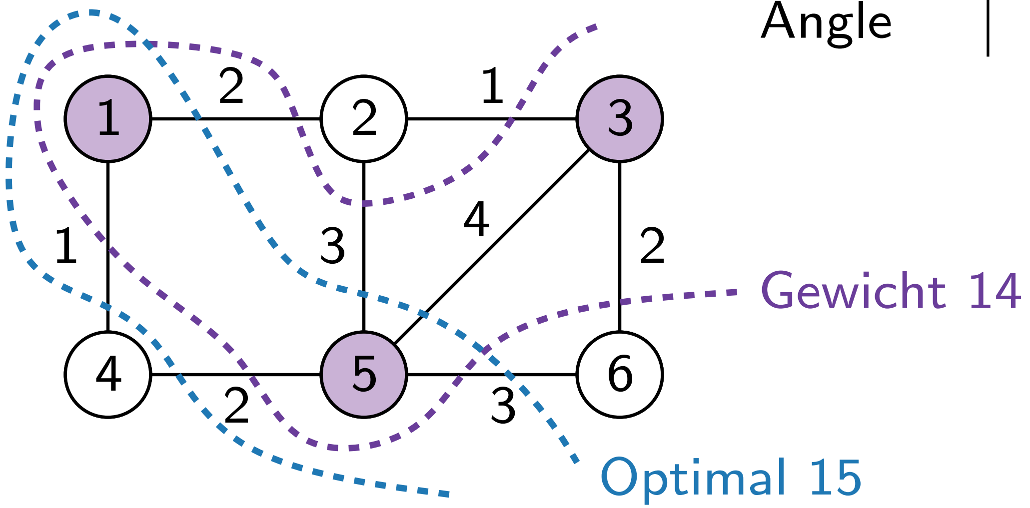
	1	2	3	4	5	6
1		2		1		
2	2		1		3	
3		1			4	2
4	1				2	
5		3	4	2		3
6			2		3	

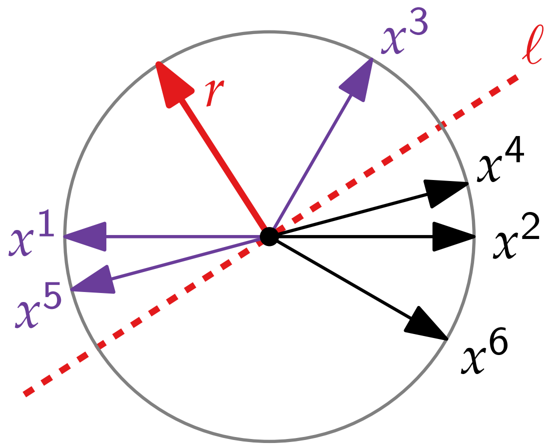
2. Step: Relax QP to QP^2

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2}\sum_{j=1}^{6}\sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j) \\ \textbf{subject to} \quad & x^i \cdot x^i = 1 \\ & x^i = (x^i_1, x^i_2) \in \mathbb{R}^2 \end{aligned}$$

3. Step: Solve QP^2

Variable	x^1	x^2	x^3	x^4	x^5	x^6
Angle	0	180	120	165	345	210

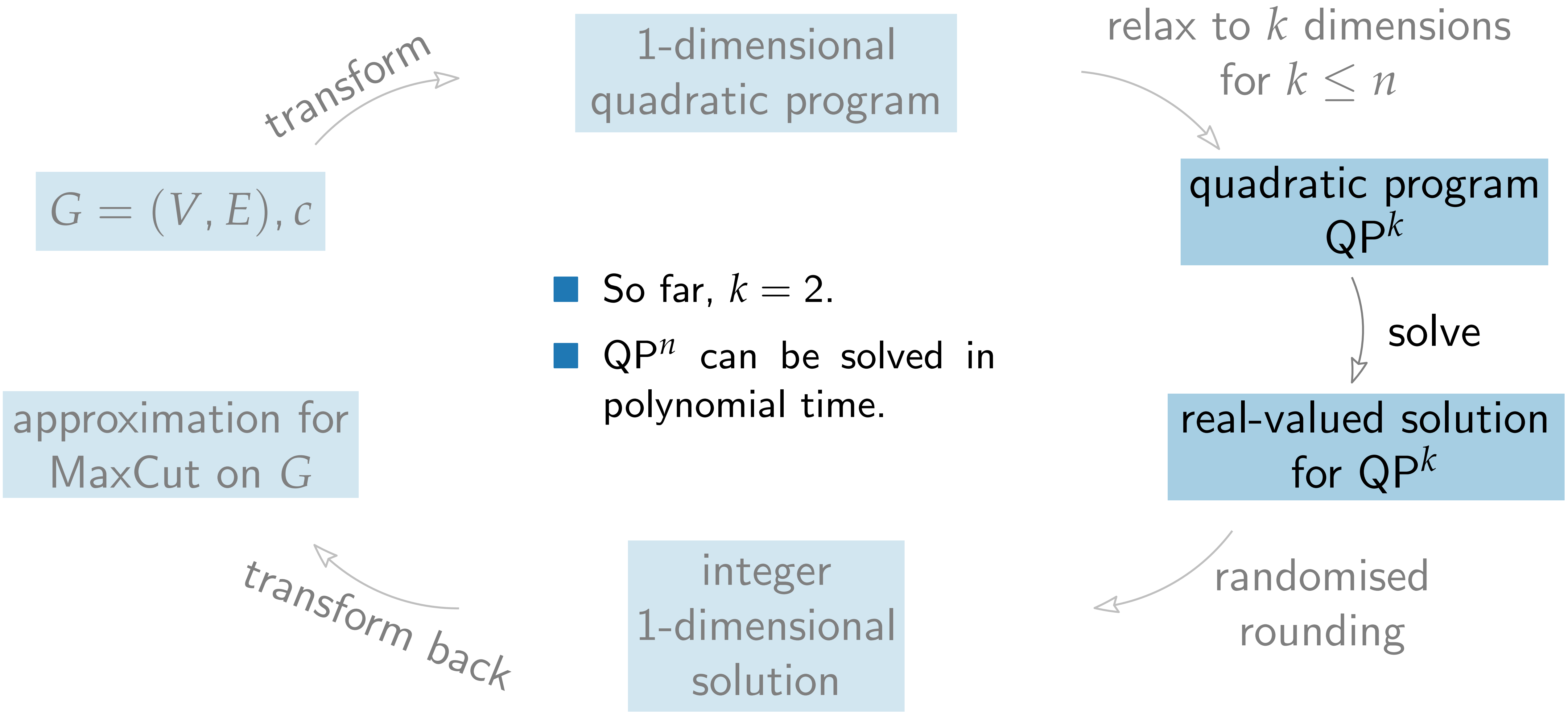




4. Step: Guess r

5. Step: Derive S

Goemans-Williamson algorithm for MaxCut



- So far, $k = 2$.
- QP^n can be solved in polynomial time.

$\mathrm{QP}^n(G, c)$

$\mathbf{QP}^2(G, c)$

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j) \\ \textbf{subject to} \quad & x^i \cdot x^i = 1 \\ & x^i = (x^i_1, x^i_2) \in \mathbb{R}^2 \end{aligned}$$

$\mathbf{QP}^n(G, c)$

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j) \\ \textbf{subject to} \quad & x^i \cdot x^i = 1 \\ & x^i \in \mathbb{R}^n \end{aligned}$$

$\mathrm{QP}^n(G, c)$

$\mathrm{QP}^2(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to $x^i \cdot x^i = 1$

$x^i = (x^i_1, x^i_2) \in \mathbb{R}^2$

$\mathrm{QP}^n(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to $x^i \cdot x^i = 1$

$x^i \in \mathbb{R}^n$

- A matrix M is called **positive semidefinite** if, for any vector $v \in \mathbb{R}^n$:

$$v^\mathsf{T} \cdot M \cdot v \geq 0$$

$\mathrm{QP}^n(G, c)$

$\mathrm{QP}^2(G, c)$

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j) \\ \textbf{subject to} \quad & x^i \cdot x^i = 1 \\ & x^i = (x^i_1, x^i_2) \in \mathbb{R}^2 \end{aligned}$$

$\mathrm{QP}^n(G, c)$

$$\begin{aligned} \textbf{maximize} \quad & \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j) \\ \textbf{subject to} \quad & x^i \cdot x^i = 1 \\ & x^i \in \mathbb{R}^n \end{aligned}$$

- A matrix M is called **positive semidefinite** if, for any vector $v \in \mathbb{R}^n$:

$$v^\mathsf{T} \cdot M \cdot v \geq 0$$

- $M = (m_{ij}) = (x^i \cdot x^j)$ is positive semidefinite.

$\mathrm{QP}^n(G, c)$

$\mathbf{QP}^2(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to $x^i \cdot x^i = 1$

$x^i = (x^i_1, x^i_2) \in \mathbb{R}^2$

$\mathbf{QP}^n(G, c)$

maximize $\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{ij}(1 - x^i \cdot x^j)$

subject to $x^i \cdot x^i = 1$

$x^i \in \mathbb{R}^n$

- A matrix M is called **positive semidefinite** if, for any vector $v \in \mathbb{R}^n$:
  $$v^\top \cdot M \cdot v \geq 0$$
- $M = (m_{ij}) = (x^i \cdot x^j)$ is positive semidefinite.
- $\mathrm{QP}^n(G, c)$ becomes problem $\textsc{SemiDefiniteCut}(G, c)$.
  - Can be approximated in time poly. in (G, c) and $1/\varepsilon$ with additive guarantee ε.
  - For $\varepsilon = 10^{-5}$, approximation guarantee for $\textsc{Random-MaxCut}$ is achieved.

Discussion

- Semidefinite programming is a powerful tool to develop approximation algorithms
- Whole book on this topic:
  - [Gärtner, Matoušek] “Approximation Algorithms and Semidefinite Progamming”

Discussion

- Semidefinite programming is a powerful tool to develop approximation algorithms
- Whole book on this topic:
  - [Gärtner, Matoušek] “Approximation Algorithms and Semidefinite Progamming”
- Using randomness is another tool to design approximation algorithms.
- See future lectures.

Literature

Original paper:

- [GW ’95] “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming”

Source:

- [Vazirani Ch26] “Approximation Algorithms”

Whole book on this topic:

- [Gärtner, Matoušek] “Approximation Algorithms and Semidefinite Progamming”