Lectures: Mondays, 12:15-13:45

Exercise sessions: Tuesdays (bi-weekly) 16:00-18:00
(given by Marcus Kaiser)

Book: The Design of Approximation Algorithms
by D. Williamson and D. Shmoys

More info on the website.
Introduction to Approximation Algorithms
Algorithmic wishlist

1. **fast** (run in polynomial time)
2. **universal** (work for any instance)
3. **optimal** (find best solution)
Algorithmic wishlist

1. fast (run in polynomial time)
2. universal (work for any instance)
3. optimal (find best solution)

Choose two.
(unless P = NP)
Algorithmic wishlist

1. fast (run in polynomial time)
2. universal (work for any instance)
3. approximately optimal (find provably good solution)

⇓

Approximation Algorithms
Definition An $\alpha$-approximation algorithm for an optimization problem is an algorithm that

- runs in polynomial time and
- computes for any instance of the problem a solution,
- whose value is within a factor of $\alpha$ of the optimal solution.
Definition An $\alpha$-approximation algorithm for an optimization problem is an algorithm that

- runs in polynomial time and
- computes for any instance of the problem a solution,
- whose value is within a factor of $\alpha$ of the optimal solution.

ALG: value of solution computed by algorithm

OPT: value of optimal solution

for maximization problems: $\text{ALG} \geq \alpha \cdot \text{OPT}$

for minimization problems: $\text{ALG} \leq \alpha \cdot \text{OPT}$

($\alpha \leq 1$) 

($\alpha \geq 1$)
**Definition** An $\alpha$-approximation algorithm for an optimization problem is an algorithm that

- runs in polynomial time and
- computes for any instance of the problem a solution,
- whose value is within a factor of $\alpha$ of the optimal solution.

ALG: value of solution computed by algorithm
OPT: value of optimal solution

for maximization problems: $\text{ALG} \geq \alpha \cdot \text{OPT}$  
($\alpha \leq 1$)  

for minimization problems: $\text{ALG} \leq \alpha \cdot \text{OPT}$  
($\alpha \geq 1$)

We call $\alpha$ approximation factor or performance guarantee.
Example: Set Cover
The **Set Cover** problem

**Input:** elements $E$, sets $S \subseteq 2^E$, weights $w : S \to \mathbb{R}_+$

**Task:** find $S' \subseteq S$ with $\bigcup_{S \in S'} S = E$
minimizing $\sum_{S \in S'} w(S)$
The **Set Cover** problem

**Input:** elements $E$, sets $S \subseteq 2^E$, weights $w : S \rightarrow \mathbb{R}_+$

**Task:** find $S' \subseteq S$ with $\bigcup_{S \in S'} S = E$
minimizing $\sum_{S \in S'} w(S)$

**Special case:** **Vertex Cover**
The **Set Cover** problem

**Input:** elements $E$, sets $S \subseteq 2^E$, weights $w : S \rightarrow \mathbb{R}_+$

**Task:** find $S' \subseteq S$ with $\bigcup_{S \in S'} S = E$
minimizing $\sum_{S \in S'} w(S)$

---

**Special case:** **Vertex Cover**
How to design an approximation algorithm?
How to design an approximation algorithm?

We don’t know OPT, but we can get lower bounds.
\[
\begin{align*}
\text{min} & \quad \sum_{S \in S} w(S)x(S) \\
\text{s.t.} & \quad \sum_{S : e \in S} x(S) \geq 1 \quad \forall e \in E \\
& \quad x(S) \in \{0, 1\} \quad \forall S \in S
\end{align*}
\]
LP relaxation

\[
\begin{align*}
\min & \quad \sum_{S \in S} w(S)x(S) \\
\text{s.t.} & \quad \sum_{S : e \in S} x(S) \geq 1 \quad \forall e \in E \\
& \quad x(S) \geq 0 \quad \forall S \in S
\end{align*}
\]
LP relaxation

\[
\begin{align*}
\min & \quad \sum_{S \in \mathcal{S}} w(S)x(S) \\
\text{s.t.} & \quad \sum_{S : e \in S} x(S) \geq 1 \quad \forall \ e \in \mathcal{E} \\
& \quad x(S) \geq 0 \quad \forall \ S \in \mathcal{S}
\end{align*}
\]
LP relaxation

\[ Z^* := \min \sum_{S \in \mathcal{S}} w(S) x(S) \]

s.t. \[ \sum_{S : e \in S} x(S) \geq 1 \quad \forall e \in \mathcal{E} \]

\[ x(S) \geq 0 \quad \forall S \in \mathcal{S} \]

LP value is lower bound:

\[ Z^* \leq \text{OPT} \]
(Deterministic) LP Rounding
LP rounding

**Idea:** Select $S$ if $x(S) \geq \frac{1}{f}$.

$$f := \max_{e \in E} |\{S : e \in S\}|$$
**Idea:** Select $S$ if $x(S) \geq \frac{1}{f}$.

$$f := \max_{e \in E} |\{S : e \in S\}|$$

**Theorem**

LP rounding is an $f$-approximation algorithm for **Set Cover**.
**LP rounding**

**Idea:** Select $S$ if $x(S) \geq \frac{1}{f}$. 

\[ f := \max_{e \in E} |\{S : e \in S\}| \]

**Theorem**

LP rounding is an $f$-approximation algorithm for SET COVER.

**Proof.**

\[ S' := \left\{ S : x(S) \geq \frac{1}{f} \right\} \]
**LP rounding**

**Idea:** Select $S$ if $x(S) \geq \frac{1}{f}$. 

$f := \max_{e \in E} |\{S : e \in S\}|$

---

**Theorem**

LP rounding is an $f$-approximation algorithm for \textsc{Set Cover}.

**Proof.**

$S' := \left\{ S : x(S) \geq \frac{1}{f} \right\}$

- Is every element covered?
LP rounding

**Idea:** Select $S$ if $x(S) \geq \frac{1}{f}$.

$$f := \max_{e \in E} |\{ S : e \in S \}|$$

**Theorem**

LP rounding is an $f$-approximation algorithm for **Set Cover**.

**Proof.**

$$S' := \left\{ S : x(S) \geq \frac{1}{f} \right\}$$

- Is every element covered?

$$\sum_{S \in S : e \in S} x(S) \geq 1 \quad \Rightarrow \quad \exists S \in S : x(S) \geq \frac{1}{f}$$
LP rounding

**Idea:** Select $S$ if $x(S) \geq \frac{1}{f}$.

$$f := \max_{e \in E} |\{S : e \in S\}|$$

**Theorem**

LP rounding is an $f$-approximation algorithm for **Set Cover**.

**Proof.**

$$S' := \left\{ S : x(S) \geq \frac{1}{f} \right\}$$

- Is every element covered?

$$\sum_{S \in S : e \in S} x(S) \geq 1 \implies \exists S \in S : x(S) \geq \frac{1}{f}$$

- Is the approximation factor fulfilled?
LP rounding

Idea: Select $S$ if $x(S) \geq \frac{1}{f}$.

\[ f := \max_{e \in E} |\{S : e \in S\}| \]

Theorem

LP rounding is an $f$-approximation algorithm for SET COVER.

Proof.

\[ S' := \left\{ S : x(S) \geq \frac{1}{f} \right\} \]

- Is every element covered?

\[ \sum_{S \in S : e \in S} x(S) \geq 1 \quad \Rightarrow \quad \exists S \in S : x(S) \geq \frac{1}{f} \]

- Is the approximation factor fulfilled?

\[ \sum_{S \in S'} w(S) \leq \sum_{S \in S'} w(S) \cdot f \cdot x(S) \]
Idea: Select $S$ if $x(S) \geq \frac{1}{f}$.

$$f := \max_{e \in E} \left| \{S : e \in S\} \right|$$

Theorem

LP rounding is an $f$-approximation algorithm for Set Cover.

Proof.

$$S' := \left\{ S : x(S) \geq \frac{1}{f} \right\}$$

- Is every element covered?

$$\sum_{S \in S : e \in S} x(S) \geq 1 \quad \Rightarrow \quad \exists S \in S : x(S) \geq \frac{1}{f}$$

- Is the approximation factor fulfilled?

$$\sum_{S \in S'} w(S) \leq \sum_{S \in S'} w(S) \cdot f \cdot x(S) \leq f \cdot \sum_{S \in S} w(S) x(S) = f \cdot Z^*$$
Consequences

A priori vs. a fortiori

- The LP rounding analysis gives us an a priori guarantee: $\text{ALG} \leq f \cdot \text{OPT}$ for any instance of Set Cover.

- For a concrete run of the algorithm, we get an a fortiori guarantee: If $\text{ALG} \big/ Z^* \leq \alpha$, we know that $\text{ALG} \leq \alpha \cdot \text{OPT}$. 
A priori vs. a fortiori

- The LP rounding analysis gives us an a priori guarantee: \( \text{ALG} \leq f \cdot \text{OPT} \) for any instance of Set Cover.
- For a concrete run of the algorithm, we get an a fortiori guarantee: If \( \text{ALG}/Z^* \leq \alpha \), we know that \( \text{ALG} \leq \alpha \cdot \text{OPT} \).

Integrality gap

The integrality gap of an LP is the ratio \( \frac{Z^*}{\text{OPT}} \).

The LP rounding algorithm implies that the integrality gap of the Set Cover LP is bounded by \( f \).
The Primal-Dual Method
Primal-dual method

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} y(e) \\
\text{s.t.} & \quad \sum_{e \in S} y(e) \leq w(S) \quad \forall S \in S \\
\quad & \quad y(e) \geq 0 \quad \forall e \in E
\end{align*}
\]

Algorithm:

1. Initialize \( y(e) = 0 \) for all \( e \in E \).
2. While \( \exists \) uncovered element \( e' \):
   - Increase \( y(e') \) until a set \( S \) with \( e' \in S \) becomes tight.
3. Add \( S \) to \( S' \). \( \sum_{e \in S} y(e) = w(S) \)
4. Return \( S' \).

Theorem: Primal-dual is an \( f \)-approximation algorithm for Set Cover.
Primal-dual method

\[
\begin{align*}
\max & \quad \sum_{e \in E} y(e) \\
\text{s.t.} & \quad \sum_{e \in S} y(e) \leq w(S) \quad \forall S \in S \\
& \quad y(e) \geq 0 \quad \forall e \in E
\end{align*}
\]

Algorithm:

1. Initialize \( y(e) = 0 \) for all \( e \in E \).
2. while (\( \exists \) uncovered element \( e' \))
   
   Increase \( y(e) \) until a set \( S \) with \( e \in S \) becomes tight.
   
   Add \( S \) to \( S' \).

3. Return \( S' \).

Theorem

Primal-dual is an \( f \)-approximation algorithm for Set Cover.
Primal-dual method

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} y(e) \\
\text{s.t.} & \quad \sum_{e \in S} y(e) \leq w(S) \quad \forall S \in \mathcal{S} \\
& \quad y(e) \geq 0 \quad \forall e \in E
\end{align*}
\]

Algorithm:

1. Initialize \( y(e) = 0 \) for all \( e \in E \).
2. while (\( \exists \) uncovered element \( e' \))
   Increase \( y(e) \) until a set \( S \) with \( e \in S \) becomes tight.
   Add \( S \) to \( S' \).
   \( (\sum_{e \in S} y(e) = w(S)) \)
3. Return \( S' \).

Theorem

Primal-dual is an \( f \)-approximation algorithm for Set Cover.
Greedy algorithm
Greedy algorithm

Algorithm:

1. while (∃ uncovered element)
   - Choose $S'$ minimizing $\frac{w(S')}{|S' \setminus \bigcup_{S \in S'} S|}$
   - Add $S'$ to $S'$.
2. Return $S'$.

Theorem
The Greedy Algorithm is an $H_n$-approximation for Set Cover.

Lemma
For every iteration $i$:
$$w(S_i) \leq n_i - n_i + 1$$
$$n_i$$OPT.
Greedy algorithm

Algorithm:

1. while (exists uncovered element)
   
   Choose $S'$ minimizing $\frac{w(S')}{|S' \setminus \bigcup_{s \in S'} S|}$

   Add $S'$ to $S'$.

2. Return $S'$.

$$n := |E|, \quad H_n := \sum_{i=1}^{n} \frac{1}{i}$$

Theorem

The Greedy Algorithm is an $H_n$-approximation for Set Cover.
Greedy algorithm

Algorithm:

1. while (∃ uncovered element)
   
   Choose $S'$ minimizing $\frac{w(S')}{|S'\setminus \bigcup_{S \in S'} S|}$

   Add $S'$ to $S'$.

2. Return $S'$.

$n := |E|, \ H_n := \sum_{i=1}^{n} \frac{1}{i}$

Theorem

The Greedy Algorithm is an $H_n$-approximation for Set Cover.

$S_i$: set selected in iteration $i$

$n_i$: uncovered elements at start of iteration $i$
Greedy algorithm

Algorithm:

1. while (exists uncovered element)
   
   Choose $S'$ minimizing $\frac{w(S')}{|S' \setminus \bigcup_{S \in S'} S|}$
   
   Add $S'$ to $S'$.

2. Return $S'$.

$n := |E|, \quad H_n := \sum_{i=1}^{n} \frac{1}{i}$

Theorem

The Greedy Algorithm is an $H_n$-approximation for Set Cover.

$S_i$: set selected in iteration $i$

$n_i$: uncovered elements at start of iteration $i$

Lemma

For every iteration $i$: $w(S_i) \leq \frac{n_i - n_{i+1}}{n_i} \cdot \text{OPT}$.
Greedy algorithm

Algorithm:

1. while (∃ uncovered element)
   Choose $S'$ minimizing $\frac{w(S')}{|S'\setminus \bigcup_{S \in S'} S|}$
   Add $S'$ to $S'$.
2. Return $S'$.

Theorem

$\sum_{S \in S'} w(S) \leq H_g \cdot Z^*$, where $g := \max_{S \in S} |S|$. 