Lecture 13: Linear Programming Relaxation and Rounding

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

June 4, 2025

Overview

- Part I
  - Why Relax & Round?
- Vertex Cover
- Set Cover
- Conclusion
- Acknowledgements

Motivation - NP-hard problems

- Many important optimization problems are NP-hard to solve.

Motivation - NP-hard problems

- Many important optimization problems are NP-hard to solve.
- What do we do when we see one?

Motivation - NP-hard problems

- Many important optimization problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!

Motivation - NP-hard problems

- Many important optimization problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!
  2. Sometimes we even do that for problems in P (but we want much much faster solutions)

Motivation - NP-hard problems

- Many important optimization problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!
  2. Sometimes we even do that for problems in P (but we want much much faster solutions)
- **Integer Linear Program (ILP):**

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \in \mathbb{N}^n \end{aligned}$$

Motivation - NP-hard problems

- Many important optimization problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!
  2. Sometimes we even do that for problems in P (but we want much much faster solutions)
- **Integer Linear Program (ILP):**

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \in \mathbb{N}^n \end{aligned}$$

- Advantage of ILPs: very expressive language to formulate optimization problems (capture many combinatorial optimization problems)

Motivation - NP-hard problems

- Many important optimization problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!
  2. Sometimes we even do that for problems in P (but we want much much faster solutions)
- **Integer Linear Program (ILP):**

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \in \mathbb{N}^n \end{aligned}$$

- Advantage of ILPs: very expressive language to formulate optimization problems (capture many combinatorial optimization problems)
- Disadvantage of ILPs: capture even NP-hard problems (thus NP-hard)

Motivation - NP-hard problems

- Many important optimization problems are NP-hard to solve.
- What do we do when we see one?
  1. Find approximate solutions in polynomial time!
  2. Sometimes we even do that for problems in P (but we want much much faster solutions)
- **Integer Linear Program (ILP):**

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \in \mathbb{N}^n \end{aligned}$$

- Advantage of ILPs: very expressive language to formulate optimization problems (capture many combinatorial optimization problems)
- Disadvantage of ILPs: capture even NP-hard problems (thus NP-hard)
- But we know how to solve LPs. Can we get partial credit in life?

Example

Maximum Independent Set:

$$G(V, E) \text{ graph.}$$

$$\text{Independent set } S \subseteq V \text{ such that } u, v \in S \Rightarrow \{u, v\} \notin E.$$

Example

Maximum Independent Set:

$$G(V, E) \text{ graph.}$$

$$\text{Independent set } S \subseteq V \text{ such that } u, v \in S \Rightarrow \{u, v\} \notin E.$$

Integer Linear Program:

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & x_u + x_v \leq 1 \quad \text{for } \{u, v\} \in E \\ & x_v \in \{0, 1\} \quad \text{for } v \in V \end{aligned}$$

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate combinatorial optimization problem as ILP

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate combinatorial optimization problem as ILP
2. Derive LP from the ILP by removing the integral constraints

This is called an *LP relaxation*.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate combinatorial optimization problem as ILP
2. Derive LP from the ILP by removing the integral constraints

   This is called an *LP relaxation*.

3. We are still minimizing the same objective function, but over a (potentially) larger set of solutions.

$$opt(LP) \leq opt(ILP)$$

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate combinatorial optimization problem as ILP
2. Derive LP from the ILP by removing the integral constraints

   This is called an *LP relaxation*.
3. We are still minimizing the same objective function, but over a (potentially) larger set of solutions.

   $$opt(LP) \leq opt(ILP)$$
4. Solve LP optimally using efficient algorithm.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate combinatorial optimization problem as ILP
2. Derive LP from the ILP by removing the integral constraints

   This is called an *LP relaxation*.
3. We are still minimizing the same objective function, but over a (potentially) larger set of solutions.

$$opt(LP) \leq opt(ILP)$$

4. Solve LP optimally using efficient algorithm.
   1. If solution to LP has *integral values*, then it is a solution to ILP and we are done

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate combinatorial optimization problem as ILP
2. Derive LP from the ILP by removing the integral constraints

   This is called an *LP relaxation*.
3. We are still minimizing the same objective function, but over a (potentially) larger set of solutions.

$$opt(LP) \leq opt(ILP)$$

4. Solve LP optimally using efficient algorithm.
   1. If solution to LP has *integral values*, then it is a solution to ILP and we are done
   2. If solution has *fractional values*, then we have to devise *rounding procedure* that transforms

      fractional solutions $\rightarrow$ integral solutions

$$opt(LP) \leq \text{ rounded solution } \leq c \cdot opt(ILP)$$

Not all LPs created equal

When solving LP

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

it is important to understand *geometry of feasible set* & how nice the *corner points* are, as they are the candidates to *optimum* solution.

Not all LPs created equal

When solving LP

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

it is important to understand *geometry of feasible set* & how nice the *corner points* are, as they are the candidates to *optimum* solution.

- Let $P := \{x \in \mathbb{R}^n_{\geq 0} \mid Ax = b\}$

Not all LPs created equal

When solving LP

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

it is important to understand *geometry of feasible set* & how nice the *corner points* are, as they are the candidates to *optimum* solution.

- Let $P := \{x \in \mathbb{R}^n_{\geq 0} \mid Ax = b\}$
- **Vertex Solutions:** a solution $x \in P$ is a vertex solution if $\nexists y \neq 0$ such that $x + y \in P$ *and* $x - y \in P$

Not all LPs created equal

When solving LP

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

it is important to understand *geometry of feasible set* & how nice the *corner points* are, as they are the candidates to *optimum* solution.

- Let $P := \{x \in \mathbb{R}^n_{\geq 0} \mid Ax = b\}$
- **Vertex Solutions:** a solution $x \in P$ is a vertex solution if $\nexists y \neq 0$ such that $x + y \in P$ *and* $x - y \in P$
- **Extreme Point Solutions:** $x \in P$ is an extreme point solution if $\exists u \in \mathbb{R}^n$ such that x is the unique optimum solution to the LP with constraint P and objective $u^T x$.

Not all LPs created equal

When solving LP

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

it is important to understand *geometry of feasible set* & how nice the *corner points* are, as they are the candidates to *optimum* solution.

- Let $P := \{x \in \mathbb{R}^n_{\geq 0} \mid Ax = b\}$
- **Vertex Solutions:** a solution $x \in P$ is a vertex solution if $\nexists y \neq 0$ such that $x + y \in P$ *and* $x - y \in P$
- **Extreme Point Solutions:** $x \in P$ is an extreme point solution if $\exists u \in \mathbb{R}^n$ such that x is the unique optimum solution to the LP with constraint P and objective $u^T x$.
- **Basic Solutions:** let $supp(x) := \{i \in [n] \mid x_i > 0\}$ be the set of nonzero coordinates of x. Then $x \in P$ is a basic solution $\Leftrightarrow$ the columns of A indexed by $supp(x)$ are linearly independent.

Not all LPs created equal

- Let $P := \{x \in \mathbb{R}^n_{\geq 0} \mid Ax = b\}$
- **Vertex Solutions:** a solution $x \in P$ is a vertex solution if $\nexists y \neq 0$ such that $x + y \in P$ *and* $x - y \in P$
- **Extreme Point Solutions:** $x \in P$ is an extreme point solution if $\exists u \in \mathbb{R}^n$ such that x is the unique optimum solution to the LP with constraint P and objective $u^T x$.
- **Basic Solutions:** let $supp(x) := \{i \in [n] \mid x_i > 0\}$ be the set of nonzero coordinates of x. Then $x \in P$ is a basic solution $\Leftrightarrow$ the columns of A indexed by $supp(x)$ are linearly independent.

Proposition

The three definitions above are equivalent!

See `https://cs.uwaterloo.ca/~lapchi/cs466-2020/notes/L17.pdf` for a proof.

- Part I
  - Why Relax & Round?
- Vertex Cover
- Set Cover
- Conclusion
- Acknowledgements

Vertex Cover

Setup:

- **Input:** a graph $G(V, E)$.
- **Output:** Minimum number of vertices that "touches" all edges of graph. That is, minimum set S such that for each edge $\{u, v\} \in E$ we have
$$|S \cap \{u, v\}| \geq 1.$$

Vertex Cover

Setup:

- **Input:** a graph $G(V, E)$.
- **Output:** Minimum number of vertices that "touches" all edges of graph. That is, minimum set S such that for each edge $\{u, v\} \in E$ we have
$$|S \cap \{u, v\}| \geq 1.$$
- **Weighted version:** associate to each vertex $v \in V$ a cost $c_v \in \mathbb{R}_{\geq 0}$.

Vertex Cover

Setup:

- **Input:** a graph $G(V, E)$.
- **Output:** Minimum number of vertices that "touches" all edges of graph. That is, minimum set S such that for each edge $\{u, v\} \in E$ we have

$$|S \cap \{u, v\}| \geq 1.$$

- **Weighted version:** associate to each vertex $v \in V$ a cost $c_v \in \mathbb{R}_{\geq 0}$.

1. Setup ILP:

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & x_u \in \{0, 1\} \quad \text{for } u \in V \end{aligned}$$

Simple 2-approximation (unweighted)

1. List edges of E in any order. Set $S = \emptyset$

Simple 2-approximation (unweighted)

1. List edges of E in any order. Set $S = \emptyset$
2. For each $\{u, v\} \in E$:
    1. If $S \cap \{u, v\} = \emptyset$, then $S \leftarrow S \cup \{u, v\}$

Simple 2-approximation (unweighted)

1. List edges of E in any order. Set $S = \emptyset$
2. For each $\{u, v\} \in E$:
    1. If $S \cap \{u, v\} = \emptyset$, then $S \leftarrow S \cup \{u, v\}$
3. return S

Simple 2-approximation (unweighted)

1. List edges of E in any order. Set $S = \emptyset$
2. For each $\{u, v\} \in E$:
   1. If $S \cap \{u, v\} = \emptyset$, then $S \leftarrow S \cup \{u, v\}$
3. return S

Proof of correctness:

Simple 2-approximation (unweighted)

1. List edges of E in any order. Set $S = \emptyset$
2. For each $\{u, v\} \in E$:
    1. If $S \cap \{u, v\} = \emptyset$, then $S \leftarrow S \cup \{u, v\}$
3. return S

Proof of correctness:

- By construction, S is a vertex cover.

Simple 2-approximation (unweighted)

1. List edges of E in any order. Set $S = \emptyset$
2. For each $\{u, v\} \in E$:
   1. If $S \cap \{u, v\} = \emptyset$, then $S \leftarrow S \cup \{u, v\}$
3. return S

Proof of correctness:

- By construction, S is a vertex cover.
- If added elements to S k times, then $|S| = 2k$ and G has a matching of size k, which means that optimum vertex cover is at least k.

Simple 2-approximation (unweighted)

1. List edges of E in any order. Set $S = \emptyset$
2. For each $\{u, v\} \in E$:
    1. If $S \cap \{u, v\} = \emptyset$, then $S \leftarrow S \cup \{u, v\}$
3. return S

Proof of correctness:

- By construction, S is a vertex cover.
- If added elements to S k times, then $|S| = 2k$ and G has a matching of size k, which means that optimum vertex cover is at least k.
- Thus, we get a 2-approximation.

What can go wrong in the weighted case?

Original Algo

Heuristic: pick lowest weight only

Vertex Cover - LP relaxation

1. Setup ILP:

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & x_u \in \{0, 1\} \quad \text{for } u \in V \end{aligned}$$

Vertex Cover - LP relaxation

1. Setup ILP:

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & x_u \in \{0, 1\} \quad \text{for } u \in V \end{aligned}$$

2. Drop integrality constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & 0 \leq x_u \leq 1 \quad \text{for } u \in V \end{aligned}$$

Vertex Cover - LP relaxation

1. Setup ILP:

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & x_u \in \{0, 1\} \quad \text{for } u \in V \end{aligned}$$

2. Drop integrality constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & 0 \leq x_u \leq 1 \quad \text{for } u \in V \end{aligned}$$

3. Solve LP. Get optimal solution z for LP, where $z = (z_u)_{u \in V}$.

Vertex Cover - LP relaxation

1. Setup ILP:

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & x_u \in \{0, 1\} \quad \text{for } u \in V \end{aligned}$$

2. Drop integrality constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & 0 \leq x_u \leq 1 \quad \text{for } u \in V \end{aligned}$$

3. Solve LP. Get optimal solution z for LP, where $z = (z_u)_{u \in V}$.
4. Round LP as follows: round z_v to nearest integer.

Vertex Cover - Analysis

1. Drop integrality constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & 0 \leq x_u \leq 1 \quad \text{for } u \in V \end{aligned}$$

2. Solve LP. Get optimal solution z for LP.

Vertex Cover - Analysis

1. Drop integrality constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & 0 \leq x_u \leq 1 \quad \text{for } u \in V \end{aligned}$$

2. Solve LP. Get optimal solution z for LP.
3. Round z_v to nearest integer. That is $y_v = \begin{cases} 1, & \text{if } z_v \geq 1/2 \\ 0, & \text{if } 0 \leq z_v < 1/2 \end{cases}$

Note that $y_v \leq 2z_v$

Vertex Cover - Analysis

1. Drop integrality constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & 0 \leq x_u \leq 1 \quad \text{for } u \in V \end{aligned}$$

2. Solve LP. Get optimal solution z for LP.
3. Round z_v to nearest integer. That is $y_v = \begin{cases} 1, & \text{if } z_v \geq 1/2 \\ 0, & \text{if } 0 \leq z_v < 1/2 \end{cases}$
4. y is an integral cover by construction

Vertex Cover - Analysis

1. Drop integrality constraints

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} c_u \cdot x_u \\ \text{subject to} \quad & x_u + x_v \geq 1 \quad \text{for } \{u, v\} \in E \\ & 0 \leq x_u \leq 1 \quad \text{for } u \in V \end{aligned}$$

2. Solve LP. Get optimal solution z for LP.
3. Round z_v to nearest integer. That is $y_v = \begin{cases} 1, & \text{if } z_v \geq 1/2 \\ 0, & \text{if } 0 \leq z_v < 1/2 \end{cases}$
4. y is an integral cover by construction
5. each edge is covered, since given $\{u, v\} \in E$, at least one of z_u, z_v is $\geq 1/2$ (by feasibility of LP)

Vertex Cover - Analysis

2. Solve LP. Get optimal solution z for LP.
3. Round z_v to nearest integer. That is $y_v = \begin{cases} 1, & \text{if } z_v \geq 1/2 \\ 0, & \text{if } 0 \leq z_v < 1/2 \end{cases}$
4. y is an integral cover by construction
5. each edge is covered, since given $\{u, v\} \in E$, at least one of z_u, z_v is $\geq 1/2$ (by feasibility of LP)
6. Cost of y is:

$$\sum_{u \in V} c_u \cdot y_u \leq \sum_{u \in V} c_u \cdot (2 \cdot z_u) \leq 2 \cdot OPT(ILP)$$

Set Cover

Setup:

- **Input:** a finite set U and a collection $S_1, S_2, \ldots, S_n$ of subsets of U.
- **Output:** The fewest collection of sets $I \subseteq [n]$ such that

$$\bigcup_{i \in I} S_i = U.$$

Set Cover

Setup:

- **Input:** a finite set U and a collection $S_1, S_2, \ldots, S_n$ of subsets of U.
- **Output:** The fewest collection of sets $I \subseteq [n]$ such that

$$\bigcup_{i \in I} S_i = U.$$

- **Weighted version:** associate to each set S_i a weight $w_i \in \mathbb{R}_{\geq 0}$.

Set Cover

Setup:

- **Input:** a finite set U and a collection $S_1, S_2, \ldots, S_n$ of subsets of U.
- **Output:** The fewest collection of sets $I \subseteq [n]$ such that

$$\bigcup_{i \in I} S_i = U.$$

- **Weighted version:** associate to each set S_i a weight $w_i \in \mathbb{R}_{\geq 0}$.

1. Setup ILP:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in [n]} w_i \cdot x_i \\ \text{subject to} \quad & \sum_{i \text{ s.t. } v \in S_i} x_i \geq 1 \quad \text{for } v \in U \\ & x_i \in \{0, 1\} \quad \text{for } i \in [n] \end{aligned}$$

Set Cover - Relax...

1. Obtain LP relaxation:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i \in [n]} w_i \cdot x_i \\
\text{subject to} \quad & \sum_{i \text{ s.t. } v \in S_i} x_i \geq 1 \quad \text{for } v \in U \\
& 0 \leq x_i \leq 1 \quad \text{for } i \in [n]
\end{aligned}
$$

Set Cover - Relax...

1. Obtain LP relaxation:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in [n]} w_i \cdot x_i \\ \text{subject to} \quad & \sum_{i \text{ s.t. } v \in S_i} x_i \geq 1 \quad \text{for } v \in U \\ & 0 \leq x_i \leq 1 \quad \text{for } i \in [n] \end{aligned}$$

2. Suppose we end up with fractional solution $z \in [0,1]^n$ when we solve the LP above. Now need to come up with a rounding scheme.

Set Cover - Relax...

1. Obtain LP relaxation:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in [n]} w_i \cdot x_i \\ \text{subject to} \quad & \sum_{i \text{ s.t. } v \in S_i} x_i \geq 1 \quad \text{for } v \in U \\ & 0 \leq x_i \leq 1 \quad \text{for } i \in [n] \end{aligned}$$

2. Suppose we end up with fractional solution $z \in [0,1]^n$ when we solve the LP above. Now need to come up with a rounding scheme.
3. Can we just round each coordinate z_i to the nearest integer (like in vertex cover)?

Set Cover - Relax...

1. Obtain LP relaxation:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in [n]} w_i \cdot x_i \\ \text{subject to} \quad & \sum_{i \text{ s.t. } v \in S_i} x_i \geq 1 \quad \text{for } v \in U \\ & 0 \leq x_i \leq 1 \quad \text{for } i \in [n] \end{aligned}$$

2. Suppose we end up with fractional solution $z \in [0,1]^n$ when we solve the LP above. Now need to come up with a rounding scheme.
3. Can we just round each coordinate z_i to the nearest integer (like in vertex cover)?
4. Not really. Say $v \in U$ is in 20 sets, and we got $z_i = 1/20$ for each of the sets $v \in S_i$. Then rounding procedure above would not select any such set!

Set Cover - Rounding

1. Think of z_i as the "probability" that we would pick set S_i.

Set Cover - Rounding

1. Think of z_i as the “probability” that we would pick set S_i.
2. Solution z describes an “optimal probability distribution” over ways to chose the sets S_i.

Set Cover - Rounding

1. Think of z_i as the "probability" that we would pick set S_i.
2. Solution z describes an "optimal probability distribution" over ways to chose the sets S_i.
3. Okay, but how do we cover?

Set Cover - Rounding

1. Think of z_i as the "probability" that we would pick set S_i.
2. Solution z describes an "optimal probability distribution" over ways to chose the sets S_i.
3. Okay, but how do we cover?

Algorithm (Random Pick)

1. **Input:** $z = (z_1, \ldots, z_n) \in [0, 1]^n$ *such that* z *is OPT solution to our LP*
2. **Output:** *a set cover for* U

Set Cover - Rounding

1. Think of z_i as the "probability" that we would pick set S_i.
2. Solution z describes an "optimal probability distribution" over ways to chose the sets S_i.
3. Okay, but how do we cover?

Algorithm (Random Pick)

1. **Input:** $z = (z_1, \ldots, z_n) \in [0, 1]^n$ *such that* z *is OPT solution to our LP*
2. **Output:** *a set cover for* U
3. *Set* $I = \emptyset$

Set Cover - Rounding

1. Think of z_i as the "probability" that we would pick set S_i.
2. Solution z describes an "optimal probability distribution" over ways to chose the sets S_i.
3. Okay, but how do we cover?

Algorithm (Random Pick)

1. **Input:** $z = (z_1, \ldots, z_n) \in [0,1]^n$ *such that z is OPT solution to our LP*
2. **Output:** *a set cover for U*
3. *Set* $I = \emptyset$
4. *for* $i = 1, \ldots n$
   - *with probability* z_i, *set* $I = I \cup \{i\}$

Set Cover - Rounding

1. Think of z_i as the "probability" that we would pick set S_i.
2. Solution z describes an "optimal probability distribution" over ways to chose the sets S_i.
3. Okay, but how do we cover?

Algorithm (Random Pick)

1. **Input:** $z = (z_1, \ldots, z_n) \in [0,1]^n$ *such that* z *is OPT solution to our LP*
2. **Output:** *a set cover for* U
3. *Set* $I = \emptyset$
4. *for* $i = 1, \ldots n$
    - *with probability* z_i*, set* $I = I \cup \{i\}$
5. *return* I

Set Cover - Rounding

1. Think of z_i as the "probability" that we would pick set S_i.
2. Solution z describes an "optimal probability distribution" over ways to chose the sets S_i.
3. Okay, but how do we cover?

Algorithm (Random Pick)

1. **Input:** $z = (z_1, \ldots, z_n) \in [0,1]^n$ *such that z is OPT solution to our LP*
2. **Output:** *a set cover for U*
3. *Set* $I = \emptyset$
4. *for* $i = 1, \ldots n$
   - *with probability* z_i, *set* $I = I \cup \{i\}$
5. *return* I

4. Expected cost of the sets is $\sum_{i=1}^{n} w_i \cdot z_i$, which is the optimum for the LP. But will this process cover U?

Analyzing Random Pick

Let's consider the Random Pick process from point of view of $v \in U$.

Analyzing Random Pick

Let's consider the Random Pick process from point of view of $v \in U$.

- $v \in S_1, \ldots, S_k$ (for simplicity)

Analyzing Random Pick

Let's consider the Random Pick process from point of view of $v \in U$.

- $v \in S_1, \ldots, S_k$ (for simplicity)
- As long as we select one of S_i's above we are good (w.r.t. v)

Analyzing Random Pick

Let's consider the Random Pick process from point of view of $v \in U$.

- $v \in S_1, \ldots, S_k$ (for simplicity)
- As long as we select one of S_i's above we are good (w.r.t. v)
- We select S_i with probability z_i such that

$$\sum_{i=1}^{k} z_i \geq 1$$

Because z is a solution to our LP

Analyzing Random Pick

Let's consider the Random Pick process from point of view of $v \in U$.

- $v \in S_1, \ldots, S_k$ (for simplicity)
- As long as we select one of S_i's above we are good (w.r.t. v)
- We select S_i with probability z_i such that

$$\sum_{i=1}^{k} z_i \geq 1$$

  Because z is a solution to our LP
- What is probability that v is covered in Random Pick?

Analyzing Random Pick

Let's consider the Random Pick process from point of view of $v \in U$.

- $v \in S_1, \ldots, S_k$ (for simplicity)
- As long as we select one of S_i's above we are good (w.r.t. v)
- We select S_i with probability z_i such that

$$\sum_{i=1}^{k} z_i \geq 1$$

  Because z is a solution to our LP
- What is probability that v is covered in Random Pick?
- Definitely not 1. Think about case $k = 2$ and $z_1 = z_2 = 1/2$.

Analyzing Random Pick

Let's consider the Random Pick process from point of view of $v \in U$.

- $v \in S_1, \ldots, S_k$ (for simplicity)
- As long as we select one of S_i's above we are good (w.r.t. v)
- We select S_i with probability z_i such that

$$\sum_{i=1}^{k} z_i \geq 1$$

  Because z is a solution to our LP
- What is probability that v is covered in Random Pick?
- Definitely not 1. Think about case $k = 2$ and $z_1 = z_2 = 1/2$.
- If had many elements like that, would expect many elements uncovered. How to deal with this?

Analyzing Random Pick

Let's consider the Random Pick process from point of view of $v \in U$.

- $v \in S_1, \ldots, S_k$ (for simplicity)
- As long as we select one of S_i's above we are good (w.r.t. v)
- We select S_i with probability z_i such that

$$\sum_{i=1}^{k} z_i \geq 1$$

  Because z is a solution to our LP
- What is probability that v is covered in Random Pick?
- Definitely not 1. Think about case $k = 2$ and $z_1 = z_2 = 1/2$.
- If had many elements like that, would expect many elements uncovered. How to deal with this?
- By perseverance! :)

Probability that Element is Covered

Lemma (Probability of Covering an Element)

In a sequence of k independent experiments, in which the i^{th} experiment has success probability p_i, and

$$\sum_{i=1}^{k} p_i \geq 1$$

then there is a probability $\geq 1 - 1/e$ that at least one experiment is successful.

Probability that Element is Covered

Lemma (Probability of Covering an Element)

In a sequence of k independent experiments, in which the i^{th} experiment has success probability p_i, and

$$\sum_{i=1}^{k} p_i \geq 1$$

then there is a probability $\geq 1 - 1/e$ that at least one experiment is successful.

- Probability that no experiment is successful:

$$(1 - p_1) \cdot (1 - p_2) \cdots (1 - p_k)$$

Probability that Element is Covered

Lemma (Probability of Covering an Element)

In a sequence of k independent experiments, in which the i^{th} experiment has success probability p_i, and

$$\sum_{i=1}^{k} p_i \geq 1$$

then there is a probability $\geq 1 - 1/e$ that at least one experiment is successful.

- Probability that no experiment is successful:

$$(1 - p_1) \cdot (1 - p_2) \cdots (1 - p_k)$$

- $1 - x \leq e^{-x}$ for $x \in [0, 1]$

Probability that Element is Covered

Lemma (Probability of Covering an Element)

In a sequence of k independent experiments, in which the i^{th} experiment has success probability p_i, and

$$\sum_{i=1}^{k} p_i \geq 1$$

then there is a probability $\geq 1 - 1/e$ that at least one experiment is successful.

- Probability that no experiment is successful:

$$(1 - p_1) \cdot (1 - p_2) \cdots (1 - p_k)$$

- $1 - x \leq e^{-x}$ for $x \in [0, 1]$
- Thus probability of failure is

$$\prod_{i=1}^{k} (1 - p_i) \leq \prod_{i=1}^{k} e^{-p_i} = e^{-p_1 - \cdots - p_k} \leq 1/e$$

Randomized Rounding

Algorithm (Randomized Rounding)

1. **Input:** *values* $z = (z_1, \ldots, z_n) \in [0,1]^n$ *s.t.* z *is a solution to our LP*
2. **Output:** *a set cover for* U

Randomized Rounding

Algorithm (Randomized Rounding)

1. **Input:** *values* $z = (z_1, \ldots, z_n) \in [0,1]^n$ *s.t.* z *is a solution to our LP*
2. **Output:** *a set cover for* U
3. *Set* $I = \emptyset$
4. *While there is element* $v \in U$ *uncovered:*
   For $i = 1, \ldots, n$*:*
   - *with probability* z_i, *set* $I = I \cup \{i\}$
5. *return* I

Randomized Rounding

Algorithm (Randomized Rounding)

1. **Input:** *values* $z = (z_1, \ldots, z_n) \in [0,1]^n$ *s.t.* z *is a solution to our LP*
2. **Output:** *a set cover for* U
3. *Set* $I = \emptyset$
4. *While there is element* $v \in U$ *uncovered:*
   For $i = 1, \ldots, n$*:*
   - *with probability* z_i*, set* $I = I \cup \{i\}$
5. *return* I

To analyze this, need to show that we don't execute the for loop too many times.

Randomized Rounding

Algorithm (Randomized Rounding)

1. **Input:** *values* $z = (z_1, \ldots, z_n) \in [0,1]^n$ *s.t.* z *is a solution to our LP*
2. **Output:** *a set cover for* U
3. *Set* $I = \emptyset$
4. *While there is element* $v \in U$ *uncovered:*
   For $i = 1, \ldots, n$*:*
   - *with probability* z_i, *set* $I = I \cup \{i\}$
5. *return* I

To analyze this, need to show that we don't execute the for loop too many times.

Lemma (Probability Decay)

Let $t \in \mathbb{N}$. *The probability that the for loop will be executed more than* $\ln(|U|) + t$ *times is at most* e^{-t}.

Proof of Lemma

Lemma (Probability Decay)

Let $t \in \mathbb{N}$. The probability that the for loop will be executed more than $\ln(|U|) + t$ times is at most e^{-t}.

Proof of Lemma

Lemma (Probability Decay)

Let $t \in \mathbb{N}$. The probability that the for loop will be executed more than $\ln(|U|) + t$ times is at most e^{-t}.

- Probability that for loop is executed more than $\ln(|U|) + t$ times is the probability that there is an *uncovered* element after the $\ln(|U|) + t$ iteration.

Proof of Lemma

Lemma (Probability Decay)

Let $t \in \mathbb{N}$. The probability that the for loop will be executed more than $\ln(|U|) + t$ times is at most e^{-t}.

- Probability that for loop is executed more than $\ln(|U|) + t$ times is the probability that there is an *uncovered* element after the $\ln(|U|) + t$ iteration.
- Let $v \in U$. For each iteration of the loop, there is a probability of $1/e$ that v is not covered. (by our previous lemma)

Proof of Lemma

Lemma (Probability Decay)

Let $t \in \mathbb{N}$. The probability that the for loop will be executed more than $\ln(|U|) + t$ times is at most e^{-t}.

- Probability that for loop is executed more than $\ln(|U|) + t$ times is the probability that there is an *uncovered* element after the $\ln(|U|) + t$ iteration.
- Let $v \in U$. For each iteration of the loop, there is a probability of $1/e$ that v is not covered. (by our previous lemma)
- Probability that v not covered after $\ln(|U|) + t$ iterations is

$$\left(\frac{1}{e}\right)^{\ln(|U|)+t} = \frac{1}{|U|} \cdot e^{-t}$$

Proof of Lemma

Lemma (Probability Decay)

Let $t \in \mathbb{N}$. The probability that the for loop will be executed more than $\ln(|U|) + t$ times is at most e^{-t}.

- Probability that for loop is executed more than $\ln(|U|) + t$ times is the probability that there is an *uncovered* element after the $\ln(|U|) + t$ iteration.
- Let $v \in U$. For each iteration of the loop, there is a probability of $1/e$ that v is not covered. (by our previous lemma)
- Probability that v not covered after $\ln(|U|) + t$ iterations is

$$\left(\frac{1}{e}\right)^{\ln(|U|)+t} = \frac{1}{|U|} \cdot e^{-t}$$

- Union bound.

Cost of Rounded Solution

Now that we know we will cover with high probability, we need to bound the cost of the solution we came up with.

Cost of Rounded Solution

Now that we know we will cover with high probability, we need to bound the cost of the solution we came up with.

- At each implementation of for loop, our expected cover weight is

$$\sum_{i=1}^{n} w_i \cdot z_i$$

Cost of Rounded Solution

Now that we know we will cover with high probability, we need to bound the cost of the solution we came up with.

- At each implementation of for loop, our expected cover weight is

$$\sum_{i=1}^{n} w_i \cdot z_i$$

- After t iterations of for loop, expected weight is

$$\omega := t \cdot \sum_{i=1}^{n} w_i \cdot z_i$$

Cost of Rounded Solution

Now that we know we will cover with high probability, we need to bound the cost of the solution we came up with.

- At each implementation of for loop, our expected cover weight is

$$\sum_{i=1}^{n} w_i \cdot z_i$$

- After t iterations of for loop, expected weight is

$$\omega := t \cdot \sum_{i=1}^{n} w_i \cdot z_i$$

- By Markov:

$$\Pr[X \geq 2 \cdot \mathbb{E}[X]] \leq 1/2.$$

Cost of Rounded Solution

Now that we know we will cover with high probability, we need to bound the cost of the solution we came up with.

- At each implementation of for loop, our expected cover weight is

$$\sum_{i=1}^{n} w_i \cdot z_i$$

- After t iterations of for loop, expected weight is

$$\omega := t \cdot \sum_{i=1}^{n} w_i \cdot z_i$$

- By Markov:

$$\Pr[X \geq 2 \cdot \mathbb{E}[X]] \leq 1/2.$$

Lemma (Cost of Rounding)

Given z optimal for the LP, our randomized rounding outputs, with probability ≥ 0.45 a feasible solution to set cover with $\leq 2 \cdot (\ln(|U|) + 3) \cdot OPT(ILP)$ sets

Cost of Rounding

Lemma (Cost of Rounding)

Given z optimal for the LP, our randomized rounding outputs, with probability ≥ 0.45 a feasible solution to set cover with $\leq 2 \cdot (\ln(|U|) + 3) \cdot OPT(ILP)$ sets

Cost of Rounding

Lemma (Cost of Rounding)

Given z optimal for the LP, our randomized rounding outputs, with probability ≥ 0.45 a feasible solution to set cover with $\leq 2 \cdot (\ln(|U|) + 3) \cdot OPT(ILP)$ sets

1. Let $t = \ln(|U|) + 3$. There is a probability at most $e^{-3} < 0.05$ that while loop runs for more than t steps.

Cost of Rounding

Lemma (Cost of Rounding)

Given z optimal for the LP, our randomized rounding outputs, with probability ≥ 0.45 a feasible solution to set cover with $\leq 2 \cdot (\ln(|U|) + 3) \cdot OPT(ILP)$ sets

1. Let $t = \ln(|U|) + 3$. There is a probability at most $e^{-3} < 0.05$ that while loop runs for more than t steps.
2. After t steps, expected weight is

$$\omega := t \cdot \sum w_i \cdot z_i \leq t \cdot OPT(ILP)$$

Cost of Rounding

Lemma (Cost of Rounding)

Given z optimal for the LP, our randomized rounding outputs, with probability ≥ 0.45 a feasible solution to set cover with $\leq 2 \cdot (\ln(|U|) + 3) \cdot OPT(ILP)$ sets

1. Let $t = \ln(|U|) + 3$. There is a probability at most $e^{-3} < 0.05$ that while loop runs for more than t steps.
2. After t steps, expected weight is

$$\omega := t \cdot \sum w_i \cdot z_i \leq t \cdot OPT(ILP)$$

3. Markov $\Rightarrow$ probability that our solution has weight $\geq 2 \cdot \omega$ is $\leq 1/2$

Cost of Rounding

Lemma (Cost of Rounding)

Given z optimal for the LP, our randomized rounding outputs, with probability ≥ 0.45 a feasible solution to set cover with $\leq 2 \cdot (\ln(|U|) + 3) \cdot OPT(ILP)$ sets

1. Let $t = \ln(|U|) + 3$. There is a probability at most $e^{-3} < 0.05$ that while loop runs for more than t steps.
2. After t steps, expected weight is

$$\omega := t \cdot \sum w_i \cdot z_i \leq t \cdot OPT(ILP)$$

3. Markov $\Rightarrow$ probability that our solution has weight $\geq 2 \cdot \omega$ is $\leq 1/2$
4. Union bound, with probability ≤ 0.55 either run for more than t times, or our solution has weight $\geq 2\omega$

Cost of Rounding

Lemma (Cost of Rounding)

Given z optimal for the LP, our randomized rounding outputs, with probability ≥ 0.45 a feasible solution to set cover with $\leq 2 \cdot (\ln(|U|) + 3) \cdot OPT(ILP)$ sets

1. Let $t = \ln(|U|) + 3$. There is a probability at most $e^{-3} < 0.05$ that while loop runs for more than t steps.
2. After t steps, expected weight is

$$\omega := t \cdot \sum w_i \cdot z_i \leq t \cdot OPT(ILP)$$

3. Markov $\Rightarrow$ probability that our solution has weight $\geq 2 \cdot \omega$ is $\leq 1/2$
4. Union bound, with probability ≤ 0.55 either run for more than t times, or our solution has weight $\geq 2\omega$
5. Thus, with probability ≥ 0.45 we stop at t iterations **and** construct solution to set cover with cost $\leq 2t \cdot OPT(ILP)$

Putting Everything Together

1. Formulate set cover problem as ILP

Putting Everything Together

1. Formulate set cover problem as ILP
2. Derive LP from the ILP *LP relaxation*

Putting Everything Together

1. Formulate set cover problem as ILP
2. Derive LP from the ILP *LP relaxation*
3. We are still minimizing the same objective function (weight of cover), but over a (potentially) larger (*fractional*) set of solutions.

$$OPT(LP) \leq OPT(ILP)$$

Putting Everything Together

1. Formulate set cover problem as ILP
2. Derive LP from the ILP *LP relaxation*
3. We are still minimizing the same objective function (weight of cover), but over a (potentially) larger (*fractional*) set of solutions.

$$OPT(LP) \leq OPT(ILP)$$

4. Solve LP optimally using efficient algorithm.

Putting Everything Together

1. Formulate set cover problem as ILP
2. Derive LP from the ILP *LP relaxation*
3. We are still minimizing the same objective function (weight of cover), but over a (potentially) larger (*fractional*) set of solutions.

$$OPT(LP) \leq OPT(ILP)$$

4. Solve LP optimally using efficient algorithm.
    1. If solution to LP has *integral values*, then it is a solution to ILP and we are done

Putting Everything Together

1. Formulate set cover problem as ILP
2. Derive LP from the ILP *LP relaxation*
3. We are still minimizing the same objective function (weight of cover), but over a (potentially) larger (*fractional*) set of solutions.

$$OPT(LP) \leq OPT(ILP)$$

4. Solve LP optimally using efficient algorithm.
    1. If solution to LP has *integral values*, then it is a solution to ILP and we are done
    2. If have *fractional values*, *rounding procedure*

       Randomized Rounding algorithm, with probability ≥ 0.45 we get

$$cost(\text{rounded solution}) \leq 2 \cdot (\ln(|U|) + 3) \cdot OPT(ILP)$$

Conclusion

- Integer Linear programming - very general, and pervasive in (combinatorial) algorithmic life
- ILP NP-hard
- Rounding for the rescue!
- Solve LP and round the solution
  - Deterministic rounding when solutions are nice
  - Randomized rounding when things a bit more complicated

Acknowledgement

- Lecture based largely on:
  - Lectures 7-8 of Luca's Optimization class
- See Luca's vertex cover notes at `https://lucatrevisan.github.io/teaching/cs261-11/lecture07.pdf`
- See Luca's set cover notes at `https://lucatrevisan.github.io/teaching/cs261-11/lecture08.pdf`