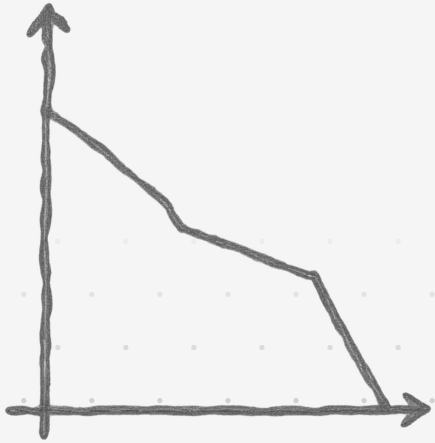
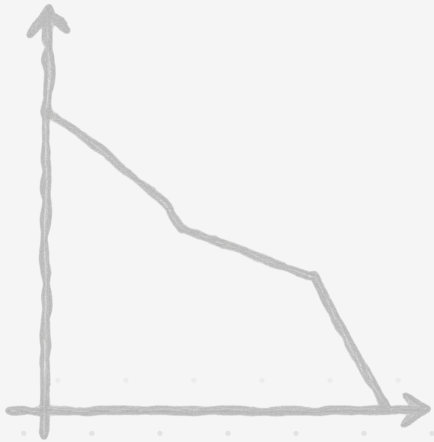


Optimization ▸ Linear Programming

Lecture 12

- The Approximation Framework
- Linear Programs
- Solving Linear Programs
- Relaxations and Roundings





Optimization ▸ Linear Programming

Lecture 12

- The Approximation Framework
- Linear Programs
- Solving Linear Programs
- Relaxations and Roundings

Recap: Approximation Algorithms

Our Goal: Find approximate solutions to hard optimization problems

Optimization problem: given input I, find the best feasible solution

- Minimize a cost, or maximize a quality
- $\mathrm{OPT}(I)$ is the value of the best solution

Approximation: an algorithm returning value $A(I)$ is an α-approximation if, for every input I,

- **Minimization:** $A(I) \leq \alpha\,\mathrm{OPT}(I)$
- **Maximization:** $A(I) \geq \mathrm{OPT}(I)/\alpha$

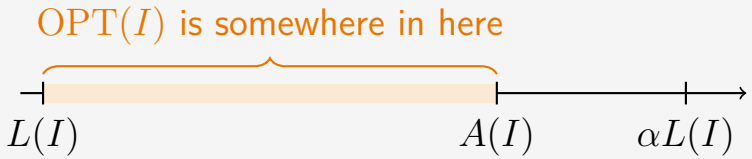
The challenge: to prove a guarantee we must compare with $\mathrm{OPT}(I)$, but computing it is the hard problem we are avoiding!

Certificates: The General Recipe

For a minimization problem, a lower-bound certificate is a value $L(I) \leq \mathrm{OPT}(I)$ that we can actually compute.

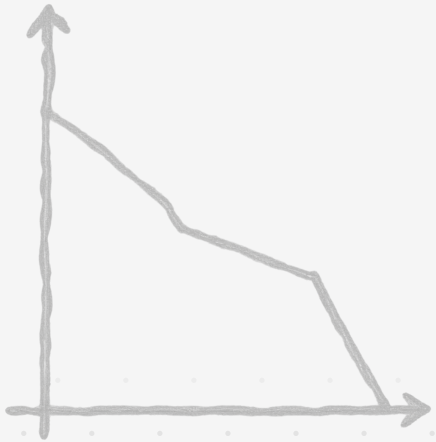
Recipe:

1. Find a certificate $L(I) \leq \mathrm{OPT}(I)$
2. Design an algorithm with $A(I) \leq \alpha L(I)$
3. Conclude $A(I) \leq \alpha L(I) \leq \alpha\,\mathrm{OPT}(I)$



Last lecture: $L = |M|$ for a maximal matching M, and $A = 2|M|$.
Today: L is the optimal value of a linear program.

For maximization, flip it: an upper bound $U(I) \geq \mathrm{OPT}(I)$ and $A(I) \geq U(I)/\alpha$.





Optimization ▸ Linear Programming

Lecture 12

- The Approximation Framework
- Linear Programs
- Solving Linear Programs
- Relaxations and Roundings

Recall/Review: Maximum Flow

Input: Directed graph $G = (V, E)$, source s, sink t, edge capacities $c : E \to \mathbb{R}_{\geq 0}$

Goal: Find maximum flow from s to t respecting capacities

Recall: A flow $f : E \to \mathbb{R}_{\geq 0}$ is valid if:

- **Capacity:** For all edges $e \in E$: $0 \leq f(e) \leq c(e)$
- **Conservation:** For all vertices $v \in V \setminus \{s, t\}$: Flow in $=$ Flow out

Value: $\text{val}(f) = \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ into } s} f(e)$

Maximum Flow as an Optimization Problem

Variables: Associate a variable f_e to each edge $e \in E$

Objective: Maximize $\sum_{e=(s,v)\in E} f_e - \sum_{e=(u,s)\in E} f_e$

Constraints:

- For all $e \in E$: $f_e \leq c_e$ (capacity)
- For all $e \in E$: $f_e \geq 0$ (non-negativity)
- For all $v \in V \setminus \{s,t\}$: $\sum_{e=(v,u)\in E} f_e = \sum_{e=(u,v)\in E} f_e$ (conservation)

All constraints are linear (in)equalities! This is a linear program.

A General Linear Program

Input:

- n real-valued variables $x_1, \ldots, x_n$
- A linear objective function to maximize or minimize
- m linear constraints (inequalities or equalities)

Output: Values for $x_1, \ldots, x_n$ that:

- Satisfy all constraints
- Maximize/minimize the objective function

Important: Only $\leq$, $\geq$, and $=$ allowed (no strict inequalities $<$ or $>$)

General LP: Formal Definition

Maximize (or Minimize): $c_1x_1 + c_2x_2 + \cdots + c_nx_n$

Subject to:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &\leq b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &\leq b_2 \\ &\vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n &\leq b_m \end{aligned}$$

- All $a_{i,j}$, b_i, c_j are real constants
- Can also use $\geq$ or $=$ constraints
- Can add non-negativity: $x_i \geq 0$ for some/all variables

Example 1: Nutrition Planning

Problem: Design a meal plan to minimize cost while meeting nutritional goals

Available foods:

Food	Cost (per serving)	Protein (g/serving)	Sugar (g/serving)
Tofu	\$2.50	20g	1g
Rice	\$0.50	4g	1g
Beans	\$1.00	15g	3g

Requirements:

- At least 50g of protein
- At most 7g of sugar

Classic application: diet planning, resource allocation, production planning

Example 1: LP Formulation and Solution

Variables: $x_1 =$ servings of tofu, $x_2 =$ servings of rice, $x_3 =$ servings of beans

Minimize: $2.5x_1 + 0.5x_2 + x_3$ (total cost in dollars)

Subject to:

$$\begin{aligned} 20x_1 + 4x_2 + 15x_3 &\geq 50 \quad \text{(protein requirement)} \\ x_1 + x_2 + 3x_3 &\leq 7 \quad \text{(sugar limit)} \\ x_1, x_2, x_3 &\geq 0 \quad \text{(non-negativity)} \end{aligned}$$

Optimal solution: $x_1^* = 1$, $x_2^* = 0$, $x_3^* = 2$ with cost $\$4.50$
(1 serving of tofu, 2 servings of beans, and no rice)

Example 2: Linear Regression

Input: Points $(x_1, y_1), \ldots, (x_n, y_n)$ in the plane

Goal: Find a line $y = ax + b$ that minimizes the L1 distance to the points:

$$\sum_{i=1}^{n} |y_i - (a \cdot x_i + b)|$$

Remark

In a Machine Learning class you might have seen a similar problem with L2 distance (i.e., least squares):

$$\sum_i (y_i - (ax_i + b))^2$$

Example 2: A First Attempt

Variables: a, b (line parameters)
(The data x_i, y_i are fixed constants, not variables.)

Minimize: $\sum_{i=1}^{n} |y_i - (a \cdot x_i + b)|$

Subject to: No constraints!

What is wrong with this formulation?

Example 2: Modeling Absolute Value

Key Idea: For each point i, introduce variable z_i to represent $|y_i - (ax_i + b)|$

We need: $z_i \geq |y_i - (ax_i + b)|$

This is equivalent to:

$$z_i \geq y_i - (ax_i + b)$$
$$z_i \geq -(y_i - (ax_i + b)) = ax_i + b - y_i$$

When we minimize $\sum_i z_i$, at an optimal solution each z_i equals $|y_i - (ax_i + b)|$ (not larger)

Example 2: LP Formulation

Variables: a, b (line parameters) and $z_1, \ldots, z_n$ (helper variables)
(The data x_i, y_i are fixed constants, not variables.)

Minimize: $z_1 + z_2 + \cdots + z_n$

Subject to: For all $i = 1, \ldots, n$:

$$z_i \geq y_i - ax_i - b$$
$$z_i \geq -y_i + ax_i + b$$

This shows how to model certain non-linear functions using LPs by introducing helper variables!

Example 3: Satisfiability

Input: A CNF formula over Boolean variables $x_1, \ldots, x_n$, e.g.

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_3)$$

Goal: Find a satisfying assignment

Variables: x_i for each Boolean variable ($1 =$ true, $0 =$ false), with $0 \leq x_i \leq 1$

Constraints: Each clause needs at least one true literal, where $\neg x_i$ becomes $1 - x_i$:

$$\begin{aligned} x_1 + (1 - x_2) + x_3 &\geq 1 \\ (1 - x_1) + x_2 &\geq 1 \\ (1 - x_1) + (1 - x_3) &\geq 1 \end{aligned}$$

Objective: None; any feasible solution will do

What is wrong with this formulation?

Example 3: An Integer Program

Problem: Setting every $x_i = 1/2$ satisfies every clause with at least two literals, e.g. $\frac{1}{2} + (1 - \frac{1}{2}) + \frac{1}{2} \geq 1$. So the LP says nothing about the formula!

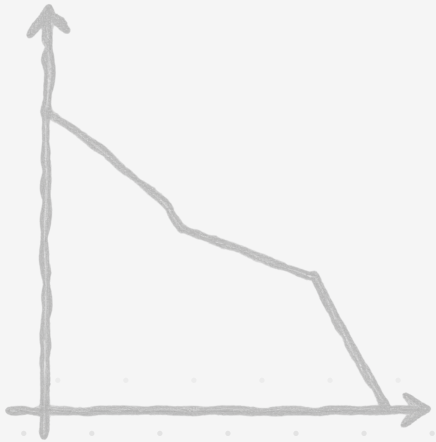
Fix: Require $x_i \in \{0, 1\}$. Then a feasible solution is exactly a satisfying assignment.

Definition: Integer Program (IP)

An LP with the extra constraint that the variables take integer values (often $x_i \in \{0, 1\}$).

SAT can be written as an IP, so solving general IPs is **NP-hard**!

Dropping $x_i \in \{0, 1\}$ back to $0 \leq x_i \leq 1$ is called a relaxation: the result is an LP, which is easy to solve, but its solution may be fractional.





Optimization ▸ Linear Programming

Lecture 12

- The Approximation Framework
- Linear Programs
- Solving Linear Programs
- Relaxations and Roundings

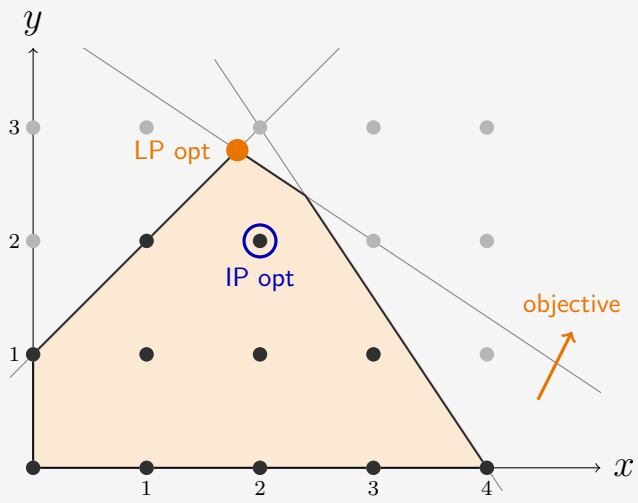
What an LP Looks Like

Maximize: $x + 2y$

Subject to:

$$\begin{aligned} -x + y &\leq 1 \\ 3x + 2y &\leq 12 \\ 2x + 3y &\leq 12 \\ x, y &\geq 0 \end{aligned}$$

- Each constraint is a half-plane
- Feasible region: a polygon
- The optimum is at a corner



If we require integers (an IP), the best point is $(2, 2)$, not a rounding of the LP optimum: the nearest point $(2, 3)$ is not even feasible!

Solving LPs: Theory and Practice

Simplex method (Dantzig, 1947)

- Walks from corner to better corner until none is better
- Exponential time in the worst case (Klee–Minty examples)
- Very fast in practice (smoothed analysis, Spielman–Teng, explains why)

Polynomial-time algorithms

- Ellipsoid method (Khachiyan, 1979): first polynomial-time algorithm
- Interior-point methods (Karmarkar, 1984): polynomial and practical

Software: Gurobi, CPLEX, HiGHS, GLPK

LPs can be solved in polynomial time, while general IPs are NP-hard.

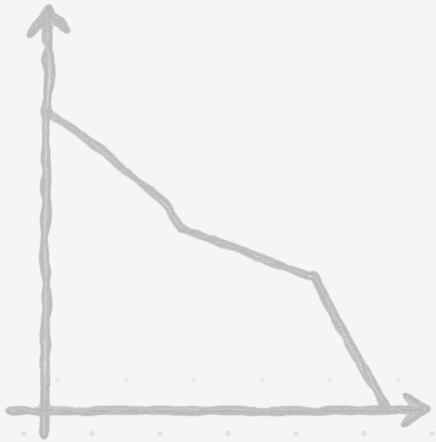
Why Should We Care About LPs?

Three key reasons:

1. **Efficiently solvable**, both in theory (polynomial time) and in practice (fast solvers)
2. **Very expressive**
   - Flows, regression, planning, scheduling, ...
   - Powerful tool for algorithm design (as we'll see next!)
3. **Rich and fundamental theory**
   - Deep connections to optimization, geometry, game theory, economics

Remark

We won't focus on LP theory or algorithms in this course. For the rich theory and practice of LPs, see courses like ORF 307 (Optimization).





Optimization ▸ Linear Programming

Lecture 12

- The Approximation Framework
- Linear Programs
- Solving Linear Programs
- Relaxations and Roundings

The Relaxation and Rounding Framework

Recall Our Goal: Find approximate solutions to hard optimization problems

Key Idea: Certificates for lower/upper bounds on OPT

- Help us prove approximation guarantees
- We use LP relaxations as certificates

General Approach:

1. Model problem as an Integer Program (IP)
2. Relax to a Linear Program (LP) by allowing fractional values
3. Solve LP to get OPT_{LP} (polynomial time!)
4. OPT_{LP} provides a certificate: $\text{OPT}_{\text{LP}} \leq \text{OPT}$
5. Round LP solution to get integral solution
6. Compare rounded solution quality to OPT_{LP}
7. By proxy, this tells us about quality w.r.t. OPT

Recall/Review: Vertex Cover

Input: Undirected graph $G = (V, E)$

Goal: Find a minimum-size subset $S \subseteq V$ such that every edge has at least one endpoint in S

Formally: For every edge $(u, v) \in E$: $u \in S$ or $v \in S$ (or both)

What we know:

- Vertex Cover is NP-hard
- Last lecture: 2-approximation algorithm (greedy maximal matching approach)
- Today: Another 2-approximation using LP relaxation

Vertex Cover: Integer Program vs. LP Relaxation

Integer Program (IP)

Variables: x_v for each $v \in V$
($x_v = 1$ means $v \in S$)

Minimize: $\sum_{v \in V} x_v$

Subject to:

- For every edge $(u, v) \in E$:

$$x_u + x_v \geq 1$$

- $x_v \in \{0, 1\}$ for all v

LP Relaxation

Variables: x_v for each $v \in V$

Minimize: $\sum_{v \in V} x_v$

Subject to:

- For every edge $(u, v) \in E$:

$$x_u + x_v \geq 1$$

- $0 \leq x_v \leq 1$ for all v

Key Idea: $\text{OPT}_{\text{LP}} \leq \text{OPT}$ (LP has larger feasible set)
We will round the LP solution to get an integral solution and use OPT_{LP} as our certificate/lower bound.

From LP Solution to Vertex Cover

Question: The LP gives us fractional values x_v^*. How do we get an actual vertex cover?

Algorithm: LP Rounding for Vertex Cover

1. Solve the LP relaxation to get x_v^* for all $v \in V$
2. Let $S = \{v \in V : x_v^* \geq 1/2\}$
3. Return S as the vertex cover

Key Questions:

- Is S a valid vertex cover?
- How large is S compared to the optimal vertex cover?

Goal: Compare $|S|$ to OPT_{LP}, which by proxy gives us a bound on $|S|$ vs. OPT

Analysis: Validity

Lemma:

$S = \{v : x_v^* \geq 1/2\}$ is a valid vertex cover.

Proof

Consider any edge $(u, v) \in E$.

By the LP constraints, $x_u^* + x_v^* \geq 1$, so at least one of x_u^* or x_v^* must be $\geq 1/2$.

Therefore, at least one of u or v is in S.

Since this holds for every edge, S is a valid vertex cover. □

Analysis: Approximation Ratio

Lemma:

$|S| \leq 2 \cdot \text{OPT}$, where OPT is the size of the optimal vertex cover.

Proof

Let x_v^* be an optimal LP solution, and let $\text{OPT}_{\text{LP}} = \sum_v x_v^*$ denote its value.

Now, we bound the size of S in terms of OPT_{LP}. Recall that $S = \{v : x_v^* \geq 1/2\}$. For each vertex $v \in S$, we have $x_v^* \geq 1/2$, which means $2x_v^* \geq 1$. Therefore:

$$|S| = \sum_{v \in S} 1 \leq \sum_{v \in S} 2x_v^* \leq 2 \sum_{v \in V} x_v^* = 2 \cdot \text{OPT}_{\text{LP}}$$

Recall that $\text{OPT}_{\text{LP}} \leq \text{OPT}$, since the LP relaxation has a larger feasible set than the IP.

Combining these two inequalities, we obtain $|S| \leq 2 \cdot \text{OPT}_{\text{LP}} \leq 2 \cdot \text{OPT}$. □

Conclusion: This algorithm is a **2-approximation** for Vertex Cover.

Why This Approach?

We already knew a 2-approximation for Vertex Cover. Why care about this one?

- **Simplicity:** Once you understand LP relaxations, this approach is conceptually clean and systematic
- **Generality:** The LP relaxation framework extends to many other problems where ad-hoc techniques may not work

Example: Weighted Vertex Cover
Given a graph $G = (V, E)$ where each vertex v has a weight $w_v > 0$, find a vertex cover S that minimizes $\sum_{v \in S} w_v$.

Exercise: Weighted Vertex Cover

Exercise: Show that the same LP relaxation approach (with appropriately weighted objective) gives a 2-approximation for weighted Vertex Cover.

Hint: The LP relaxation is:

$$\textbf{Minimize:} \quad \sum_{v \in V} w_v \cdot x_v$$

$$\textbf{Subject to:} \quad x_u + x_v \geq 1 \quad \forall (u,v) \in E$$

$$0 \leq x_v \leq 1 \quad \forall v \in V$$

Round the same way: $S = \{v : x_v^* \geq 1/2\}$.

Summary: LP Relaxation Technique

General Strategy:

1. Model NP-hard problem as an Integer Program
2. Relax integrality constraints to get an LP
3. Solve the LP in polynomial time
4. LP optimum provides a certificate for lower/upper bound on OPT
5. Round the fractional solution to an integral one
6. Analyze approximation ratio by comparing to LP certificate

For Vertex Cover:

- LP relaxation: $\text{OPT}_{\text{LP}} \leq \text{OPT}$ (certificate!)
- Simple rounding ($x_v^* \geq 1/2$) gives 2-approximation
- Proof: $|S| \leq 2 \cdot \text{OPT}_{\text{LP}} \leq 2 \cdot \text{OPT}$

LPs provide certificates for proving approximation guarantees!