CS498: Algorithmic Engineering

Lecture 4: LP Relaxations, Vertex Cover, & When They Fail

Elfarouk Harb

University of Illinois Urbana-Champaign

01/29/2026

Outline

Theme of Today

Last time:

- Built linear programs (LPs) and solved them with Gurobi.

Theme of Today

Last time:

- Built linear programs (LPs) and solved them with Gurobi.
- Interpreted duals as **shadow prices** and did sensitivity/network examples.

Theme of Today

Last time:

- Built linear programs (LPs) and solved them with Gurobi.
- Interpreted duals as **shadow prices** and did sensitivity/network examples.

Today:

- Use LPs to attack **discrete** problems on graphs and real-world settings.

Theme of Today

Last time:

- Built linear programs (LPs) and solved them with Gurobi.
- Interpreted duals as **shadow prices** and did sensitivity/network examples.

Today:

- Use LPs to attack **discrete** problems on graphs and real-world settings.
- See how LPs give **approximate** solutions to NP-hard problems.

Theme of Today

Last time:

- Built linear programs (LPs) and solved them with Gurobi.
- Interpreted duals as **shadow prices** and did sensitivity/network examples.

Today:

- Use LPs to attack **discrete** problems on graphs and real-world settings.
- See how LPs give **approximate** solutions to NP-hard problems.
- See one case where LPs are **great** (Vertex Cover), and one case where they **fail badly** (Independent Set).

Course Roadmap Around Today

Week 02 (this week): Linear Programming in depth

- Modeling, duality, sensitivity, network flow.
- LPs with continuous variables.

Course Roadmap Around Today

Week 02 (this week): Linear Programming in depth

- Modeling, duality, sensitivity, network flow.
- LPs with continuous variables.

Week 03: Integer Programming

- Some variables must be **integers** (0/1 decisions).
- Branch-and-bound, cutting planes, modeling tricks.

Course Roadmap Around Today

Week 02 (this week): Linear Programming in depth

- Modeling, duality, sensitivity, network flow.
- LPs with continuous variables.

Week 03: Integer Programming

- Some variables must be **integers** (0/1 decisions).
- Branch-and-bound, cutting planes, modeling tricks.

Today: "What happens if we *pretend* the integer decisions are continuous?"

What Is an Integer Program? (Preview)

So far: LP variables like $x_1, x_2, \ldots$ can be any **real number** in a range.

What Is an Integer Program? (Preview)

So far: LP variables like $x_1, x_2, \ldots$ can be any **real number** in a range.

Integer variable:

- x must be an **integer**, e.g., $x \in \{0, 1, 2, 3, \ldots\}$ or just $x \in \{0, 1\}$.

What Is an Integer Program? (Preview)

So far: LP variables like $x_1, x_2, \ldots$ can be any **real number** in a range.

Integer variable:

- x must be an **integer**, e.g., $x \in \{0, 1, 2, 3, \ldots\}$ or just $x \in \{0, 1\}$.
- For modelling, this helps with a typical meaning: "turn something on/off".

What Is an Integer Program? (Preview)

So far: LP variables like $x_1, x_2, \ldots$ can be any **real number** in a range.

Integer variable:

- x must be an **integer**, e.g., $x \in \{0, 1, 2, 3, \ldots\}$ or just $x \in \{0, 1\}$.
- For modelling, this helps with a typical meaning: "turn something on/off".

Integer Program (IP):

- Linear objective and linear constraints (like LP).
- **But some variables are restricted to integer values.**

What Is an Integer Program? (Preview)

So far: LP variables like $x_1, x_2, \ldots$ can be any **real number** in a range.

Integer variable:

- x must be an **integer**, e.g., $x \in \{0, 1, 2, 3, \ldots\}$ or just $x \in \{0, 1\}$.
- For modelling, this helps with a typical meaning: “turn something on/off”.

Integer Program (IP):

- Linear objective and linear constraints (like LP).
- **But some variables are restricted to integer values.**

  Today our discrete decisions will be $x \in \{0, 1\}$: turn on or off this variable.

Vertex Cover: Problem Statement

Input: an undirected graph $G = (V, E)$.

Vertex Cover: Problem Statement

Input: an undirected graph $G = (V, E)$.
A vertex cover is a set $C \subseteq V$ such that

$$\forall (u, v) \in E : \quad u \in C \text{ or } v \in C.$$

Vertex Cover: Problem Statement

Input: an undirected graph $G = (V, E)$.
A vertex cover is a set $C \subseteq V$ such that

$$\forall (u, v) \in E : \quad u \in C \text{ or } v \in C.$$

i.e., every edge has **at least one endpoint** in C.

Vertex Cover: Problem Statement

Input: an undirected graph $G = (V, E)$.
A vertex cover is a set $C \subseteq V$ such that

$$\forall (u, v) \in E : \quad u \in C \text{ or } v \in C.$$

i.e., every edge has **at least one endpoint** in C.
Goal: Find a vertex cover C with minimum size $|C|$.

Vertex Cover: Problem Statement

Input: an undirected graph $G = (V, E)$.
A vertex cover is a set $C \subseteq V$ such that

$$\forall (u, v) \in E : \quad u \in C \text{ or } v \in C.$$

i.e., every edge has **at least one endpoint** in C.
Goal: Find a vertex cover C with minimum size $|C|$. Trivial fact: The entire vertex set V is always a vertex cover.

Vertex Cover: Problem Statement

Input: an undirected graph $G = (V, E)$.
A vertex cover is a set $C \subseteq V$ such that

$$\forall (u, v) \in E : \quad u \in C \text{ or } v \in C.$$

i.e., every edge has **at least one endpoint** in C.
Goal: Find a vertex cover C with minimum size $|C|$. Trivial fact: The entire vertex set V is always a vertex cover.

But we want a **smaller** cover.

Tiny Examples to Build Intuition

Path of length 3: $a - b - c$

- Is $\{b\}$ a vertex cover?

Tiny Examples to Build Intuition

Path of length 3: $a - b - c$

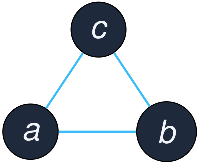
- Is $\{b\}$ a vertex cover?
- Is $\{a\}$ or $\{c\}$ alone a vertex cover?

Tiny Examples to Build Intuition

Path of length 3: $a - b - c$

- Is $\{b\}$ a vertex cover?
- Is $\{a\}$ or $\{c\}$ alone a vertex cover?

Triangle:

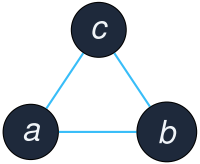


Tiny Examples to Build Intuition

Path of length 3: $a - b - c$

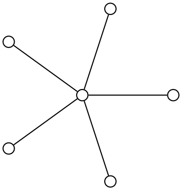
- Is $\{b\}$ a vertex cover?
- Is $\{a\}$ or $\{c\}$ alone a vertex cover?

Triangle:



- Minimum vertex cover size?

Star: one center, many leaves

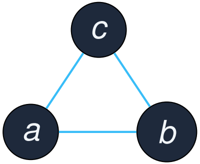


Tiny Examples to Build Intuition

Path of length 3: $a - b - c$

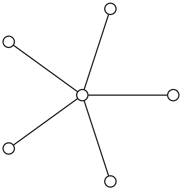
- Is $\{b\}$ a vertex cover?
- Is $\{a\}$ or $\{c\}$ alone a vertex cover?

Triangle:



- Minimum vertex cover size?

Star: one center, many leaves



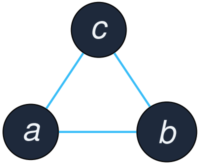
- Minimum cover is just the center.

Tiny Examples to Build Intuition

Path of length 3: $a - b - c$

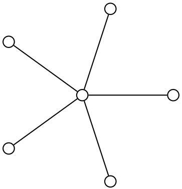
- Is $\{b\}$ a vertex cover?
- Is $\{a\}$ or $\{c\}$ alone a vertex cover?

Triangle:



- Minimum vertex cover size?

Star: one center, many leaves



- Minimum cover is just the center.

Next step: turn this into an optimization model with variables.

Binary Decision Variables for Vertex Cover

We want to encode “take vertex v” as a variable.

Binary Decision Variables for Vertex Cover

We want to encode "take vertex v" as a variable.

$$x_v = \begin{cases} 1 & \text{if we include vertex } v \text{ in the cover,} \\ 0 & \text{if we do not.} \end{cases}$$

Binary Decision Variables for Vertex Cover

We want to encode "take vertex v" as a variable.

$$x_v = \begin{cases} 1 & \text{if we include vertex } v \text{ in the cover,} \\ 0 & \text{if we do not.} \end{cases}$$

Objective: minimize the total number of chosen vertices:

$$\min \sum_{v \in V} x_v .$$

Binary Decision Variables for Vertex Cover

We want to encode "take vertex v" as a variable.

$$x_v = \begin{cases} 1 & \text{if we include vertex } v \text{ in the cover,} \\ 0 & \text{if we do not.} \end{cases}$$

Objective: minimize the total number of chosen vertices:

$$\min \sum_{v \in V} x_v .$$

Still need constraints to enforce: *every edge is covered by at least one chosen endpoint.*

Edge Constraints: Cover Every Edge

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

Edge Constraints: Cover Every Edge

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

- If $x_u = 0$ and $x_v = 0$ then edge (u, v) is **uncovered**.

Edge Constraints: Cover Every Edge

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

- If $x_u = 0$ and $x_v = 0$ then edge (u, v) is **uncovered**.
- We want to forbid this situation.

Edge Constraints: Cover Every Edge

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

- If $x_u = 0$ and $x_v = 0$ then edge (u, v) is **uncovered**.
- We want to forbid this situation.

Natural linear constraint:

$$x_u + x_v \geq 1 \quad \forall (u, v) \in E.$$

Edge Constraints: Cover Every Edge

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

- If $x_u = 0$ and $x_v = 0$ then edge (u, v) is **uncovered**.
- We want to forbid this situation.

Natural linear constraint:

$$x_u + x_v \geq 1 \quad \forall (u, v) \in E.$$

- If both are 0, constraint violated.
- If at least one is 1, constraint satisfied.

Edge Constraints: Cover Every Edge

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

- If $x_u = 0$ and $x_v = 0$ then edge (u, v) is **uncovered**.
- We want to forbid this situation.

Natural linear constraint:

$$x_u + x_v \;\geq\; 1 \quad \forall (u, v) \in E.$$

- If both are 0, constraint violated.
- If at least one is 1, constraint satisfied.

Put everything together . . .

Integer Program for Minimum Vertex Cover

IP formulation:

$$
\begin{aligned}
\min \quad & \sum_{v \in V} x_v \\
\text{s.t.} \quad & x_u + x_v \geq 1 \quad \forall (u, v) \in E \\
& x_v \in \{0, 1\} \quad \forall v \in V.
\end{aligned}
$$

Integer Program for Minimum Vertex Cover

IP formulation:

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ \text{s.t.} \quad & x_u + x_v \geq 1 \quad \forall (u,v) \in E \\ & x_v \in \{0,1\} \quad \forall v \in V. \end{aligned}$$

This is our first full **integer program** in this course.

Why Every Vertex Cover Gives a Feasible IP Solution

Let $C \subseteq V$ be any vertex cover.

Why Every Vertex Cover Gives a Feasible IP Solution

Let $C \subseteq V$ be any vertex cover.
Define:

$$x_v = \begin{cases} 1 & \text{if } v \in C, \\ 0 & \text{if } v \notin C. \end{cases}$$

Why Every Vertex Cover Gives a Feasible IP Solution

Let $C \subseteq V$ be any vertex cover.
Define:

$$x_v = \begin{cases} 1 & \text{if } v \in C, \\ 0 & \text{if } v \notin C. \end{cases}$$

Check feasibility:

- $x_v \in \{0, 1\}$ by construction.

Why Every Vertex Cover Gives a Feasible IP Solution

Let $C \subseteq V$ be any vertex cover.
Define:

$$x_v = \begin{cases} 1 & \text{if } v \in C, \\ 0 & \text{if } v \notin C. \end{cases}$$

Check feasibility:

- $x_v \in \{0, 1\}$ by construction.
- Take any edge $(u, v) \in E$.
  - Because C is a vertex cover, $u \in C$ or $v \in C$.

Why Every Vertex Cover Gives a Feasible IP Solution

Let $C \subseteq V$ be any vertex cover.
Define:

$$x_v = \begin{cases} 1 & \text{if } v \in C, \\ 0 & \text{if } v \notin C. \end{cases}$$

Check feasibility:

- $x_v \in \{0, 1\}$ by construction.
- Take any edge $(u, v) \in E$.
  - Because C is a vertex cover, $u \in C$ or $v \in C$.
  - So $x_u = 1$ or $x_v = 1$, hence $x_u + x_v \geq 1$.

Why Every Vertex Cover Gives a Feasible IP Solution

Let $C \subseteq V$ be any vertex cover.
Define:

$$x_v = \begin{cases} 1 & \text{if } v \in C, \\ 0 & \text{if } v \notin C. \end{cases}$$

Check feasibility:

- $x_v \in \{0, 1\}$ by construction.
- Take any edge $(u, v) \in E$.
  - Because C is a vertex cover, $u \in C$ or $v \in C$.
  - So $x_u = 1$ or $x_v = 1$, hence $x_u + x_v \geq 1$.

$\Rightarrow$ Every vertex cover C corresponds to a feasible IP vector x.

Why Every Feasible IP Solution is a Vertex Cover

Now suppose we have a **feasible IP solution** x:

- $x_v \in \{0, 1\}$ for all v,
- $x_u + x_v \geq 1$ for each edge $(u, v) \in E$.

Why Every Feasible IP Solution is a Vertex Cover

Now suppose we have a **feasible IP solution** x:

- $x_v \in \{0, 1\}$ for all v,
- $x_u + x_v \geq 1$ for each edge $(u, v) \in E$.

Define:

$$C := \{v \in V : x_v = 1\}.$$

Why Every Feasible IP Solution is a Vertex Cover

Now suppose we have a **feasible IP solution** x:

- $x_v \in \{0, 1\}$ for all v,
- $x_u + x_v \geq 1$ for each edge $(u, v) \in E$.

Define:

$$C := \{v \in V : x_v = 1\}.$$

Take any edge (u, v):

- Constraint says $x_u + x_v \geq 1$.

Why Every Feasible IP Solution is a Vertex Cover

Now suppose we have a **feasible IP solution** x:

- $x_v \in \{0, 1\}$ for all v,
- $x_u + x_v \geq 1$ for each edge $(u, v) \in E$.

Define:

$$C := \{v \in V : x_v = 1\}.$$

Take any edge (u, v):

- Constraint says $x_u + x_v \geq 1$.
- Since $x_u, x_v \in \{0, 1\}$, the only way this can hold is: at least one is 1.

Why Every Feasible IP Solution is a Vertex Cover

Now suppose we have a **feasible IP solution** x:

- $x_v \in \{0, 1\}$ for all v,
- $x_u + x_v \geq 1$ for each edge $(u, v) \in E$.

Define:

$$C := \{v \in V : x_v = 1\}.$$

Take any edge (u, v):

- Constraint says $x_u + x_v \geq 1$.
- Since $x_u, x_v \in \{0, 1\}$, the only way this can hold is: at least one is 1.
- So either $u \in C$ or $v \in C$.

Why Every Feasible IP Solution is a Vertex Cover

Now suppose we have a **feasible IP solution** x:

- $x_v \in \{0, 1\}$ for all v,
- $x_u + x_v \geq 1$ for each edge $(u, v) \in E$.

Define:

$$C := \{v \in V : x_v = 1\}.$$

Take any edge (u, v):

- Constraint says $x_u + x_v \geq 1$.
- Since $x_u, x_v \in \{0, 1\}$, the only way this can hold is: at least one is 1.
- So either $u \in C$ or $v \in C$.

$$\Rightarrow C \text{ is a vertex cover.}$$

Feasible IP solutions $\iff$ **vertex covers.**

Objective Values: IP vs Vertex Cover Size

Under the mapping:

$$C \leftrightarrow x,$$

we have:

Objective Values: IP vs Vertex Cover Size

Under the mapping:

$$C \leftrightarrow x,$$

we have:

- If $x_v = 1$ exactly when $v \in C$, then

$$\sum_{v \in V} x_v = |C|.$$

Objective Values: IP vs Vertex Cover Size

Under the mapping:

$$C \leftrightarrow x,$$

we have:

- If $x_v = 1$ exactly when $v \in C$, then

$$\sum_{v \in V} x_v = |C|.$$

- Minimizing $\sum_v x_v$ is exactly minimizing $|C|$.

Objective Values: IP vs Vertex Cover Size

Under the mapping:

$$C \leftrightarrow x,$$

we have:

- If $x_v = 1$ exactly when $v \in C$, then

$$\sum_{v \in V} x_v = |C|.$$

- Minimizing $\sum_v x_v$ is exactly minimizing $|C|$.

So **Integer Program** models **minimum vertex cover** exactly.

Relaxing the Integrality

IP constraint:

$$x_v \in \{0, 1\}.$$

Relaxing the Integrality

IP constraint:

$$x_v \in \{0, 1\}.$$

Relaxation idea: allow x_v to be fractional:

$$0 \leq x_v \leq 1.$$

Relaxing the Integrality

IP constraint:

$$x_v \in \{0, 1\}.$$

Relaxation idea: allow x_v to be fractional:

$$0 \leq x_v \leq 1.$$

Vertex Cover LP Relaxation:

$$\begin{aligned}
\min \quad & \sum_{v \in V} x_v \\
\text{s.t.} \quad & x_u + x_v \ \geq \ 1 \quad \forall (u, v) \in E \\
& 0 \leq x_v \leq 1 \quad \forall v \in V.
\end{aligned}$$

Relaxing the Integrality

IP constraint:

$$x_v \in \{0, 1\}.$$

Relaxation idea: allow x_v to be fractional:

$$0 \le x_v \le 1.$$

Vertex Cover LP Relaxation:

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ \text{s.t.} \quad & x_u + x_v \ \ge \ 1 \quad \forall (u, v) \in E \\ & 0 \le x_v \le 1 \quad \forall v \in V. \end{aligned}$$

Now this is a standard LP. Gurobi can solve it quickly.

Why LP$^\star \le$ OPT for Minimization

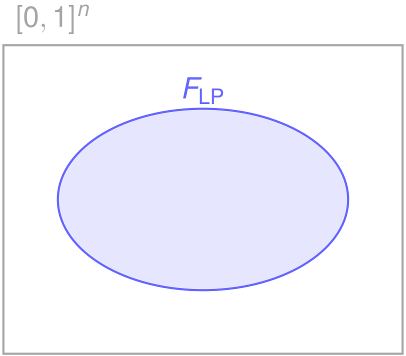
$[0,1]^n$

$$F_{\mathrm{IP}} \subseteq F_{\mathrm{LP}} \subseteq [0,1]^n$$

$$\mathrm{LP}^\star = \min_{x \in F_{\mathrm{LP}}} \sum_v x_v \;\le\; \min_{x \in F_{\mathrm{IP}}} \sum_v x_v = \mathrm{OPT}.$$

LP minimizes over a **bigger set**, so its value can only be **smaller or equal**.

Why LP$^\star \leq$ OPT for Minimization

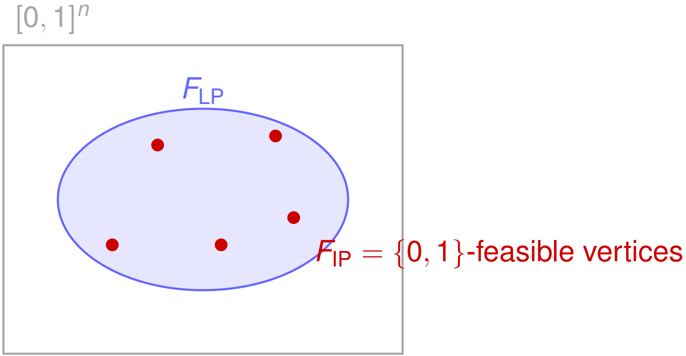


$$F_{\mathrm{IP}} \subseteq F_{\mathrm{LP}} \subseteq [0,1]^n$$

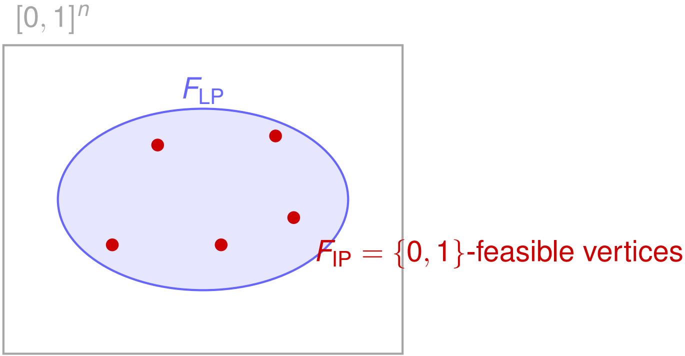
$$\mathrm{LP}^\star = \min_{x \in F_{\mathrm{LP}}} \sum_v x_v \;\leq\; \min_{x \in F_{\mathrm{IP}}} \sum_v x_v = \mathrm{OPT}.$$

LP minimizes over a **bigger set**, so its value can only be **smaller or equal**.

Why $\mathrm{LP}^\star \leq \mathrm{OPT}$ for Minimization

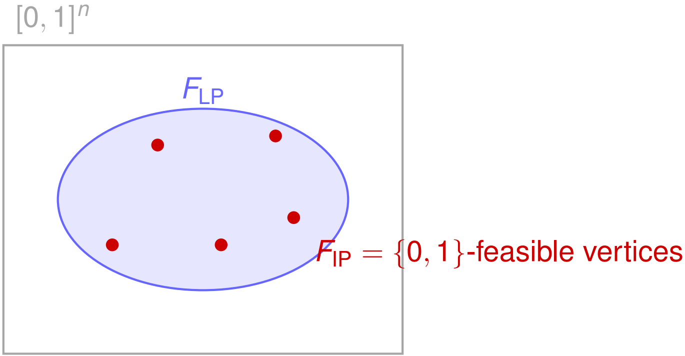


Why LP* $\leq$ OPT for Minimization



$$F_{\text{IP}} \subseteq F_{\text{LP}} \subseteq [0,1]^n$$

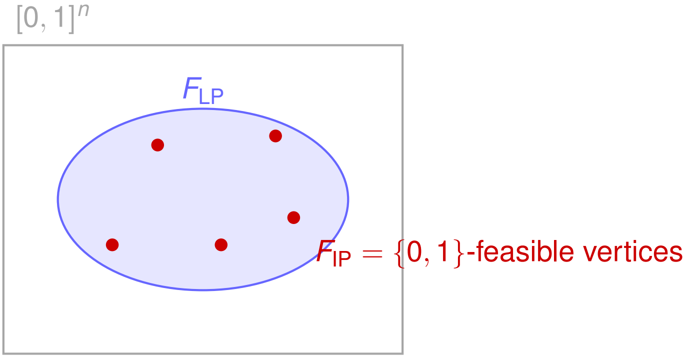
Why $\text{LP}^\star \le \text{OPT}$ for Minimization



$$F_{\text{IP}} \subseteq F_{\text{LP}} \subseteq [0,1]^n$$

$$\text{LP}^\star = \min_{x \in F_{\text{LP}}} \sum_v x_v \le \min_{x \in F_{\text{IP}}} \sum_v x_v = \text{OPT}.$$

Why LP$^\star \le$ OPT for Minimization

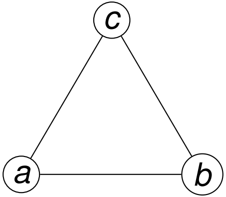


$$F_{\text{IP}} \subseteq F_{\text{LP}} \subseteq [0,1]^n$$

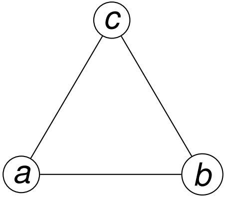
$$\text{LP}^\star = \min_{x \in F_{\text{LP}}} \sum_v x_v \le \min_{x \in F_{\text{IP}}} \sum_v x_v = \text{OPT}.$$

LP minimizes over a **bigger set**, so its value can only be **smaller or equal**.

Small LP Example with a Fractional Optimum



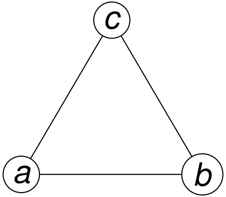
Small LP Example with a Fractional Optimum



The LP solver returns:

$$x_a^\star = x_b^\star = x_c^\star = \frac{1}{2}.$$

Small LP Example with a Fractional Optimum

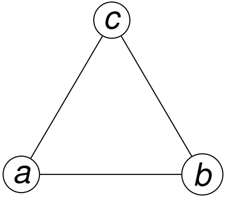


The LP solver returns:

$$x_a^\star = x_b^\star = x_c^\star = \frac{1}{2}.$$

- This is **fractional**: not a true vertex cover.

Small LP Example with a Fractional Optimum

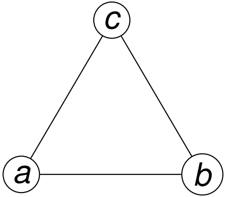


The LP solver returns:

$$x_a^\star = x_b^\star = x_c^\star = \tfrac{1}{2}.$$

- This is **fractional**: not a true vertex cover.
- The integral optimum has size 2, but $\mathrm{LP}^\star = 1.5$.

Small LP Example with a Fractional Optimum

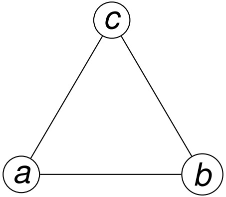


The LP solver returns:

$$x_a^\star = x_b^\star = x_c^\star = \tfrac{1}{2}.$$

- This is **fractional**: not a true vertex cover.
- The integral optimum has size 2, but $\text{LP}^\star = 1.5$.
- Fractional values suggest which vertices matter most.

Small LP Example with a Fractional Optimum



The LP solver returns:

$$x_a^\star = x_b^\star = x_c^\star = \frac{1}{2}.$$

- This is **fractional**: not a true vertex cover.
- The integral optimum has size 2, but $\mathrm{LP}^\star = 1.5$.
- Fractional values suggest which vertices matter most.

Next: how to round $x^\star$ into an actual vertex cover C?

Rounding Scheme: The Half-Threshold Trick

Algorithm:

- Solve the LP and get optimal solution $x^\star$.

Rounding Scheme: The Half-Threshold Trick

Algorithm:

- Solve the LP and get optimal solution $x^\star$.
- Define
$$C := \{v \in V : x_v^\star \geq 1/2\}.$$

Rounding Scheme: The Half-Threshold Trick

Algorithm:

- Solve the LP and get optimal solution $x^\star$.
- Define
$$C := \{v \in V : x_v^\star \geq 1/2\}.$$
- Output C as our vertex cover.

Rounding Scheme: The Half-Threshold Trick

Algorithm:

- Solve the LP and get optimal solution $x^\star$.
- Define
$$C := \{v \in V : x_v^\star \geq 1/2\}.$$
- Output C as our vertex cover.

Two questions:

1. Is C always a valid vertex cover?

Rounding Scheme: The Half-Threshold Trick

Algorithm:

- Solve the LP and get optimal solution $x^\star$.
- Define
$$C := \{v \in V : x_v^\star \geq 1/2\}.$$
- Output C as our vertex cover.

Two questions:

1. Is C always a valid vertex cover?
2. How big can $|C|$ be compared to the true optimum vertex cover?

Feasibility: C Covers All Edges

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

Feasibility: C Covers All Edges

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

- LP constraint: $x_u^\star + x_v^\star \geq 1$.

Feasibility: C Covers All Edges

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

- LP constraint: $x_u^\star + x_v^\star \geq 1$.
- Suppose (for contradiction) that $u \notin C$ and $v \notin C$.
  - Then $x_u^\star < 1/2$ and $x_v^\star < 1/2$.

Feasibility: C Covers All Edges

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

- LP constraint: $x_u^\star + x_v^\star \geq 1$.
- Suppose (for contradiction) that $u \notin C$ and $v \notin C$.
  - Then $x_u^\star < 1/2$ and $x_v^\star < 1/2$.
  - So $x_u^\star + x_v^\star < 1$.

Feasibility: C Covers All Edges

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

- LP constraint: $x_u^\star + x_v^\star \geq 1$.
- Suppose (for contradiction) that $u \notin C$ and $v \notin C$.
  - Then $x_u^\star < 1/2$ and $x_v^\star < 1/2$.
  - So $x_u^\star + x_v^\star < 1$.
  - Contradiction to the LP constraint.

Feasibility: C Covers All Edges

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

- LP constraint: $x_u^\star + x_v^\star \geq 1$.
- Suppose (for contradiction) that $u \notin C$ and $v \notin C$.
  - Then $x_u^\star < 1/2$ and $x_v^\star < 1/2$.
  - So $x_u^\star + x_v^\star < 1$.
  - Contradiction to the LP constraint.

So for every edge (u, v), at least one endpoint lies in C.

Feasibility: C Covers All Edges

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

- LP constraint: $x_u^\star + x_v^\star \geq 1$.
- Suppose (for contradiction) that $u \notin C$ and $v \notin C$.
  - Then $x_u^\star < 1/2$ and $x_v^\star < 1/2$.
  - So $x_u^\star + x_v^\star < 1$.
  - Contradiction to the LP constraint.

So for every edge (u, v), at least one endpoint lies in C.

$\Rightarrow$ C is always a **valid vertex cover**.

Size Guarantee: 2-Approximation

We want to compare $|C|$ to OPT.

Size Guarantee: 2-Approximation

We want to compare $|C|$ to OPT.
For each $v \in C$, we know $x_v^\star \geq 1/2$. Hence:

$$1 \leq 2x_v^\star \quad \forall v \in C.$$

Size Guarantee: 2-Approximation

We want to compare $|C|$ to OPT.
For each $v \in C$, we know $x_v^\star \geq 1/2$. Hence:

$$1 \leq 2x_v^\star \quad \forall v \in C.$$

Summing over $v \in C$:

$$|C| = \sum_{v \in C} 1 \leq \sum_{v \in C} 2x_v^\star \leq 2 \sum_{v \in V} x_v^\star = 2\,\mathrm{LP}^\star.$$

Size Guarantee: 2-Approximation

We want to compare $|C|$ to OPT.
For each $v \in C$, we know $x_v^\star \geq 1/2$. Hence:

$$1 \leq 2x_v^\star \quad \forall v \in C.$$

Summing over $v \in C$:

$$|C| = \sum_{v \in C} 1 \leq \sum_{v \in C} 2x_v^\star \leq 2 \sum_{v \in V} x_v^\star = 2\,\mathrm{LP}^\star.$$

And we know $\mathrm{LP}^\star \leq \mathrm{OPT}$, so:

$$|C| \leq 2\,\mathrm{OPT}.$$

Size Guarantee: 2-Approximation

We want to compare $|C|$ to OPT.
For each $v \in C$, we know $x_v^\star \geq 1/2$. Hence:

$$1 \leq 2x_v^\star \quad \forall v \in C.$$

Summing over $v \in C$:

$$|C| = \sum_{v \in C} 1 \leq \sum_{v \in C} 2x_v^\star \leq 2 \sum_{v \in V} x_v^\star = 2\,\mathrm{LP}^\star.$$

And we know $\mathrm{LP}^\star \leq \mathrm{OPT}$, so:

$$|C| \leq 2\,\mathrm{OPT}.$$

Our rounded cover is at most a factor 2 worse than the best possible cover!

Gurobi Code Sketch for Vertex Cover LP

```python
import gurobipy as gp
from gurobipy import GRB

def solve_vertex_cover_lp(V, E):
    m = gp.Model("vertex_cover_lp")
    m.Params.OutputFlag = 0
```

Gurobi Code Sketch for Vertex Cover LP

```
import gurobipy as gp
from gurobipy import GRB

def solve_vertex_cover_lp(V, E):
    m = gp.Model("vertex_cover_lp")
    m.Params.OutputFlag = 0

    # x[v] in [0,1]
    x = {v: m.addVar(lb=0.0, ub=1.0, name=f"x_{v}") for v in V}
```

Gurobi Code Sketch for Vertex Cover LP

```python
import gurobipy as gp
from gurobipy import GRB

def solve_vertex_cover_lp(V, E):
    m = gp.Model("vertex_cover_lp")
    m.Params.OutputFlag = 0

    # x[v] in [0,1]
    x = {v: m.addVar(lb=0.0, ub=1.0, name=f"x_{v}") for v in V}

    # Edge constraints: x_u + x_v >= 1
    for (u, v) in E:
        m.addConstr(x[u] + x[v] >= 1, name=f"edge_{u}_{v}")
```

Gurobi Code Sketch for Vertex Cover LP

```python
import gurobipy as gp
from gurobipy import GRB

def solve_vertex_cover_lp(V, E):
    m = gp.Model("vertex_cover_lp")
    m.Params.OutputFlag = 0

    # x[v] in [0,1]
    x = {v: m.addVar(lb=0.0, ub=1.0, name=f"x_{v}") for v in V}

    # Edge constraints: x_u + x_v >= 1
    for (u, v) in E:
        m.addConstr(x[u] + x[v] >= 1, name=f"edge_{u}_{v}")

    # Minimize number of chosen vertices
    m.setObjective(gp.quicksum(x[v] for v in V), GRB.MINIMIZE)
    m.optimize()

    return m, x
```

Rounding the LP Solution in Code

```
def round_vertex_cover_lp(V, E):
    m, x = solve_vertex_cover_lp(V, E)
    C = {v for v in V if x[v].X >= 0.5}
    return C
```

Assignment Problem

Scenario: Assign drivers to delivery routes.

Assignment Problem

Scenario: Assign drivers to delivery routes.

- Set of drivers D.
- Set of routes R.
- Cost c_{ij} = fuel/time/risk if driver i does route j.

Assignment Problem

Scenario: Assign drivers to delivery routes.

- Set of drivers D.
- Set of routes R.
- Cost c_{ij} = fuel/time/risk if driver i does route j.

Goal: each driver does exactly one route; each route done by at most one driver; total cost is minimized.

Assignment Problem

Scenario: Assign drivers to delivery routes.

- Set of drivers D.
- Set of routes R.
- Cost c_{ij} = fuel/time/risk if driver i does route j.

Goal: each driver does exactly one route; each route done by at most one driver; total cost is minimized.
This is a classic **assignment problem**. It can be written as an IP.

Assignment as IP (and LP)

Binary variables:

$$x_{ij} = \begin{cases} 1 & \text{if driver } i \text{ is assigned route } j, \\ 0 & \text{otherwise.} \end{cases}$$

Assignment as IP (and LP)

Binary variables:

$$x_{ij} = \begin{cases} 1 & \text{if driver } i \text{ is assigned route } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$\min \quad \sum_{i \in D} \sum_{j \in R} c_{ij} x_{ij}$$

s.t.

Assignment as IP (and LP)

Binary variables:

$$x_{ij} = \begin{cases} 1 & \text{if driver } i \text{ is assigned route } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \text{min} \quad & \sum_{i \in D} \sum_{j \in R} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in R} x_{ij} = 1 \quad \forall i \in D \quad \text{(each driver gets 1 route)} \end{aligned}$$

Assignment as IP (and LP)

Binary variables:

$$x_{ij} = \begin{cases} 1 & \text{if driver } i \text{ is assigned route } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \min \quad & \sum_{i \in D} \sum_{j \in R} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in R} x_{ij} = 1 \quad \forall i \in D \quad \text{(each driver gets 1 route)} \\ & \sum_{i \in D} x_{ij} \leq 1 \quad \forall j \in R \quad \text{(each route used at most once)} \end{aligned}$$

Assignment as IP (and LP)

Binary variables:

$$x_{ij} = \begin{cases} 1 & \text{if driver } i \text{ is assigned route } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$\begin{aligned}
\min \quad & \sum_{i \in D} \sum_{j \in R} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j \in R} x_{ij} = 1 \quad \forall i \in D \quad \text{(each driver gets 1 route)} \\
& \sum_{i \in D} x_{ij} \leq 1 \quad \forall j \in R \quad \text{(each route used at most once)} \\
& x_{ij} \in \{0, 1\}.
\end{aligned}$$

Assignment as IP (and LP)

Binary variables:

$$x_{ij} = \begin{cases} 1 & \text{if driver } i \text{ is assigned route } j, \\ 0 & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \min \quad & \sum_{i \in D} \sum_{j \in R} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in R} x_{ij} = 1 \quad \forall i \in D \quad \text{(each driver gets 1 route)} \\ & \sum_{i \in D} x_{ij} \le 1 \quad \forall j \in R \quad \text{(each route used at most once)} \\ & x_{ij} \in \{0, 1\}. \end{aligned}$$

LP relaxation: replace $x_{ij} \in \{0, 1\}$ by $0 \le x_{ij} \le 1$.

Magic: LP Relaxation is Exact for Assignment

For this particular constraint structure:

Magic: LP Relaxation is Exact for Assignment

For this particular constraint structure:

- Every extreme point (vertex) of the LP feasible region has $x_{ij} \in \{0, 1\}$.

Magic: LP Relaxation is Exact for Assignment

For this particular constraint structure:

- Every extreme point (vertex) of the LP feasible region has $x_{ij} \in \{0, 1\}$.
- So the LP solver will **already return integer solutions**!!

Magic: LP Relaxation is Exact for Assignment

For this particular constraint structure:

- Every extreme point (vertex) of the LP feasible region has $x_{ij} \in \{0, 1\}$.
- So the LP solver will **already return integer solutions**!!
- That means **LP optimum = IP optimum**.

Magic: LP Relaxation is Exact for Assignment

For this particular constraint structure:

- Every extreme point (vertex) of the LP feasible region has $x_{ij} \in \{0, 1\}$.
- So the LP solver will **already return integer solutions**!!
- That means **LP optimum = IP optimum**.

High-level reason (no proof today):

- The constraint matrix is **totally unimodular**.

Magic: LP Relaxation is Exact for Assignment

For this particular constraint structure:

- Every extreme point (vertex) of the LP feasible region has $x_{ij} \in \{0, 1\}$.
- So the LP solver will **already return integer solutions**!!
- That means **LP optimum = IP optimum**.

High-level reason (no proof today):

- The constraint matrix is **totally unimodular**.
- With integer right-hand sides, this implies integrality of vertices.

Magic: LP Relaxation is Exact for Assignment

For this particular constraint structure:

- Every extreme point (vertex) of the LP feasible region has $x_{ij} \in \{0, 1\}$.
- So the LP solver will **already return integer solutions**!!
- That means **LP optimum = IP optimum**.

High-level reason (no proof today):

- The constraint matrix is **totally unimodular**.
- With integer right-hand sides, this implies integrality of vertices.

Assignment problem is a case where LP relaxation is **perfect**. You can solve it exactly and integrally using an LP.

Gurobi Code Sketch for Assignment

```python
import gurobipy as gp
from gurobipy import GRB

def solve_assignment(drivers, routes, cost):
    m = gp.Model("assignment")
    m.Params.OutputFlag = 0

    # LP variables (we don't enforce integrality here!)
    x = m.addVars(drivers, routes, lb=0.0, ub=1.0, name="x")

    # each driver takes exactly one route
    for i in drivers:
        m.addConstr(gp.quicksum(x[i,j] for j in routes) == 1,  name=f"driver_{i}")

    # each route used at most once
    for j in routes:
        m.addConstr(gp.quicksum(x[i,j] for i in drivers) <= 1, name=f"route_{j}")

    m.setObjective(
        gp.quicksum(cost[i,j] * x[i,j] for i in drivers for j in routes), GRB.MINIMIZE
    )
    m.optimize()

    # inspect solution: you should see x[i,j] in {0,1}
    return m, x
```

Summary: Vertex Cover vs Assignment

Vertex Cover:

- IP is NP-hard.
- LP relaxation can be solved but gives fractional solutions.
- Rounding gives a 2-approximation.

Summary: Vertex Cover vs Assignment

Vertex Cover:

- IP is NP-hard.
- LP relaxation can be solved but gives fractional solutions.
- Rounding gives a 2-approximation.

Assignment:

- LP relaxation is **already integral**, which means IP is polynomial-time solvable.
- No rounding needed; LP = exact integral solution.

Summary: Vertex Cover vs Assignment

Vertex Cover:

- IP is NP-hard.
- LP relaxation can be solved but gives fractional solutions.
- Rounding gives a 2-approximation.

Assignment:

- LP relaxation is **already integral**, which means IP is polynomial-time solvable.
- No rounding needed; LP = exact integral solution.

Next: an example where the LP relaxation is **terrible**.

Maximum Independent Set: Definition

Independent set $S \subseteq V$:

- No edge has both endpoints in S.
- Formally:

$$\forall (u, v) \in E : \text{ not } (u \in S \text{ and } v \in S).$$

Maximum Independent Set: Definition

Independent set $S \subseteq V$:

- No edge has both endpoints in S.
- Formally:

$$\forall (u, v) \in E : \text{ not } (u \in S \text{ and } v \in S).$$

Maximum Independent Set (MIS):

$$\text{Find } S \text{ with maximum size } |S| \text{ that is independent.}$$

Maximum Independent Set: Definition

Independent set $S \subseteq V$:

- No edge has both endpoints in S.
- Formally:

$$\forall (u, v) \in E : \text{ not } (u \in S \text{ and } v \in S).$$

Maximum Independent Set (MIS):

Find S with maximum size $|S|$ that is independent.

This is also NP-hard, and looks similar to Vertex Cover:

IP for Maximum Independent Set

Binary variables:

$$y_v = \begin{cases} 1 & \text{if we include } v \text{ in the independent set,} \\ 0 & \text{otherwise.} \end{cases}$$

IP for Maximum Independent Set

Binary variables:

$$y_v = \begin{cases} 1 & \text{if we include } v \text{ in the independent set,} \\ 0 & \text{otherwise.} \end{cases}$$

Idea: for each edge (u, v), we cannot pick both u and v:

$$y_u + y_v \ \leq \ 1 \quad \forall (u, v) \in E.$$

IP for Maximum Independent Set

Binary variables:

$$y_v = \begin{cases} 1 & \text{if we include } v \text{ in the independent set,} \\ 0 & \text{otherwise.} \end{cases}$$

Idea: for each edge (u, v), we cannot pick both u and v:

$$y_u + y_v \;\le\; 1 \quad \forall (u, v) \in E.$$

IP for MIS:

$$\begin{aligned} \max \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & y_u + y_v \le 1 \quad \forall (u, v) \in E \\ & y_v \in \{0, 1\} \quad \forall v \in V. \end{aligned}$$

LP Relaxation for MIS

Relax integrality:

$$y_v \in \{0, 1\} \quad \rightsquigarrow \quad 0 \leq y_v \leq 1.$$

LP Relaxation for MIS

Relax integrality:

$$y_v \in \{0,1\} \quad \rightsquigarrow \quad 0 \le y_v \le 1.$$

MIS LP Relaxation:

$$\begin{aligned} \max \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & y_u + y_v \le 1 \quad \forall (u,v) \in E \\ & 0 \le y_v \le 1 \quad \forall v \in V. \end{aligned}$$

LP Relaxation for MIS

Relax integrality:

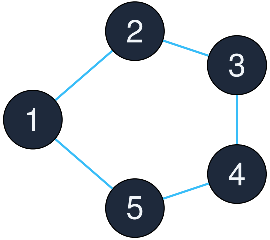
$$y_v \in \{0,1\} \quad \rightsquigarrow \quad 0 \le y_v \le 1.$$

MIS LP Relaxation:

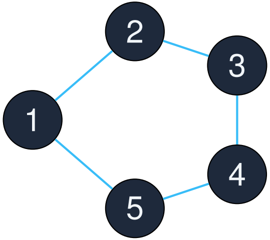
$$\begin{aligned} \max \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & y_u + y_v \le 1 \quad \forall (u,v) \in E \\ & 0 \le y_v \le 1 \quad \forall v \in V. \end{aligned}$$

Now we ask the same question: **How good is this LP as an approximation?**

Example: 5-Cycle



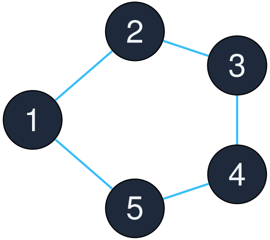
Example: 5-Cycle



True MIS:

- Maximum independent set size is 2: we can pick at most 2 non-adjacent vertices.

Example: 5-Cycle



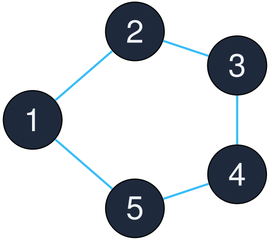
True MIS:

- Maximum independent set size is 2: we can pick at most 2 non-adjacent vertices.

LP solution:

- Set $y_v = 0.5$ for all 5 vertices.

Example: 5-Cycle



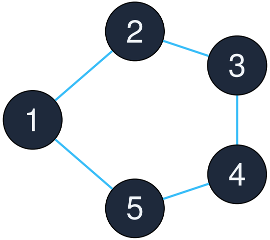
True MIS:

- Maximum independent set size is 2: we can pick at most 2 non-adjacent vertices.

LP solution:

- Set $y_v = 0.5$ for all 5 vertices.
- Then for any edge (u, v), $y_u + y_v = 1 \leq 1$ (constraint satisfied).

Example: 5-Cycle



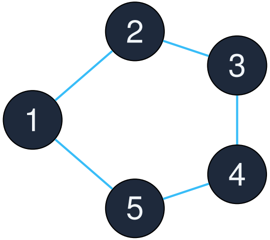
True MIS:

- Maximum independent set size is 2: we can pick at most 2 non-adjacent vertices.

LP solution:

- Set $y_v = 0.5$ for all 5 vertices.
- Then for any edge (u, v), $y_u + y_v = 1 \leq 1$ (constraint satisfied).
- Objective value = $\sum_v y_v = 5 \times 0.5 = 2.5$.

Example: 5-Cycle



True MIS:

- Maximum independent set size is 2: we can pick at most 2 non-adjacent vertices.

LP solution:

- Set $y_v = 0.5$ for all 5 vertices.
- Then for any edge (u, v), $y_u + y_v = 1 \leq 1$ (constraint satisfied).
- Objective value = $\sum_v y_v = 5 \times 0.5 = 2.5$.

$\text{LP}^\star \geq 2.5$ but $\text{OPT} = 2$. Not catastrophic yet.

Worse Example: Clique K_n

Let G be the complete graph (clique) on n vertices.

Worse Example: Clique K_n

Let G be the complete graph (clique) on n vertices.

True MIS:

- Any two vertices have an edge between them.
- So we can pick at most one vertex.
- $\mathrm{OPT} = 1$.

Worse Example: Clique K_n

Let G be the complete graph (clique) on n vertices.

True MIS:

- Any two vertices have an edge between them.
- So we can pick at most one vertex.
- $\text{OPT} = 1$.

LP solution idea:

- Try a uniform solution $y_v = 1/2$ for all v.

Worse Example: Clique K_n

Let G be the complete graph (clique) on n vertices.

True MIS:

- Any two vertices have an edge between them.
- So we can pick at most one vertex.
- $\text{OPT} = 1$.

LP solution idea:

- Try a uniform solution $y_v = 1/2$ for all v.
- Objective value: $\sum_v y_v = n \cdot \frac{1}{2} = \frac{n}{2}$.

Worse Example: Clique K_n

Let G be the complete graph (clique) on n vertices.
True MIS:

- Any two vertices have an edge between them.
- So we can pick at most one vertex.
- $\text{OPT} = 1$.

LP solution idea:

- Try a uniform solution $y_v = 1/2$ for all v.
- Objective value: $\sum_v y_v = n \cdot \frac{1}{2} = \frac{n}{2}$.

$\text{LP}^\star \geq n/2$ vs $\text{OPT} = 1$. **Gap factor** $\approx n/2$ **— terrible!**

Takeaways from MIS Example

- MIS IP and VC IP look very similar:
  - VC: $x_u + x_v \geq 1$ (cover edges).
  - MIS: $y_u + y_v \leq 1$ (avoid picking both endpoints).

Takeaways from MIS Example

- MIS IP and VC IP look very similar:
  - VC: $x_u + x_v \geq 1$ (cover edges).
  - MIS: $y_u + y_v \leq 1$ (avoid picking both endpoints).
- But their LP relaxations behave very differently:
  - VC LP: rounding gives a 2-approximation.
  - MIS LP: can overestimate the optimum by a factor $\Theta(n)$.

Takeaways from MIS Example

- MIS IP and VC IP look very similar:
  - VC: $x_u + x_v \geq 1$ (cover edges).
  - MIS: $y_u + y_v \leq 1$ (avoid picking both endpoints).
- But their LP relaxations behave very differently:
  - VC LP: rounding gives a 2-approximation.
  - MIS LP: can overestimate the optimum by a factor $\Theta(n)$.
- Moral:
  - **Choice of formulation** and **relaxation** matters a lot.
  - Not every natural IP $\to$ LP relaxation is useful algorithmically.

Takeaways from MIS Example

- MIS IP and VC IP look very similar:
  - VC: $x_u + x_v \geq 1$ (cover edges).
  - MIS: $y_u + y_v \leq 1$ (avoid picking both endpoints).
- But their LP relaxations behave very differently:
  - VC LP: rounding gives a 2-approximation.
  - MIS LP: can overestimate the optimum by a factor $\Theta(n)$.
- Moral:
  - **Choice of formulation** and **relaxation** matters a lot.
  - Not every natural IP $\rightarrow$ LP relaxation is useful algorithmically.

LP relaxations are a tool, not magic, they can shine or fail.

Summary: LP Relaxations, Successes, and Failures

What we did today:

- Built an **integer program** for Minimum Vertex Cover from scratch.

Summary: LP Relaxations, Successes, and Failures

What we did today:

- Built an **integer program** for Minimum Vertex Cover from scratch.
- Carefully showed the equivalence: vertex covers $\leftrightarrow$ feasible IP solutions.

Summary: LP Relaxations, Successes, and Failures

What we did today:

- Built an **integer program** for Minimum Vertex Cover from scratch.
- Carefully showed the equivalence: vertex covers $\leftrightarrow$ feasible IP solutions.
- Relaxed integrality to get a Vertex Cover LP and proved:
  - $\mathrm{LP}^\star \leq \mathrm{OPT}$ (for minimization),
  - Simple threshold rounding gives a 2-approximation.

Summary: LP Relaxations, Successes, and Failures

What we did today:

- Built an **integer program** for Minimum Vertex Cover from scratch.
- Carefully showed the equivalence: vertex covers $\leftrightarrow$ feasible IP solutions.
- Relaxed integrality to get a Vertex Cover LP and proved:
  - $\text{LP}^\star \leq \text{OPT}$ (for minimization),
  - Simple threshold rounding gives a 2-approximation.
- Saw the **assignment problem** where LP relaxation is exact (integral polytope).

Summary: LP Relaxations, Successes, and Failures

What we did today:

- Built an **integer program** for Minimum Vertex Cover from scratch.
- Carefully showed the equivalence: vertex covers $\leftrightarrow$ feasible IP solutions.
- Relaxed integrality to get a Vertex Cover LP and proved:
  - $\text{LP}^{\star} \leq \text{OPT}$ (for minimization),
  - Simple threshold rounding gives a 2-approximation.
- Saw the **assignment problem** where LP relaxation is exact (integral polytope).
- Built the MIS IP and its LP relaxation, and demonstrated **huge integrality gaps** (clique example).

Summary: LP Relaxations, Successes, and Failures

What we did today:

- Built an **integer program** for Minimum Vertex Cover from scratch.
- Carefully showed the equivalence: vertex covers $\leftrightarrow$ feasible IP solutions.
- Relaxed integrality to get a Vertex Cover LP and proved:
  - $\mathrm{LP}^\star \leq \mathrm{OPT}$ (for minimization),
  - Simple threshold rounding gives a 2-approximation.
- Saw the **assignment problem** where LP relaxation is exact (integral polytope).
- Built the MIS IP and its LP relaxation, and demonstrated **huge integrality gaps** (clique example).

Next time: start doing Integer Programming *for real* (branch-and-bound, cutting planes, and more modeling tricks).