Approximation Algorithms

Lecture 1: Introduction and Vertex Cover

Part I: Organizational

Alexander Wolff

Winter 2023/24

Organizational

Lectures: on site (English/German, depending on audience)

Organizational

Lectures: on site (English/German, depending on audience)

Fri, 10:15–11:45 (ÜR I)

Organizational

Lectures: on site (English/German, depending on audience)

Fri, 10:15–11:45 (ÜR I)

Tutorials: roughly one exercise sheet per lecture

Organizational

Lectures:	on site (English/German, depending on audience)
	Fri, 10:15–11:45 (ÜR I)
Tutorials:	roughly one exercise sheet per lecture
	discussing old solutions and solving new tasks

Organizational

Lectures: on site (English/German, depending on audience)
Fri, 10:15–11:45 (ÜR I)

Tutorials: roughly one exercise sheet per lecture
discussing old solutions and solving new tasks
Tue, 10:15–11:45 (SE I), starting Oct. 24.

Organizational

Lectures: on site (English/German, depending on audience)

Fri, 10:15–11:45 (ÜR I)

Tutorials: roughly one exercise sheet per lecture

discussing old solutions and solving new tasks

Tue, 10:15–11:45 (SE I), starting Oct. 24.

Bonus (+0.3 on final grade) for $\geq$ 50% points

Organizational

Lectures: on site (English/German, depending on audience)

Fri, 10:15–11:45 (ÜR I)

Tutorials: roughly one exercise sheet per lecture

discussing old solutions and solving new tasks

Tue, 10:15–11:45 (SE I), starting Oct. 24.

Bonus (+0.3 on final grade) for $\geq$ 50% points

Questions/Tasks during the lecture

Organizational

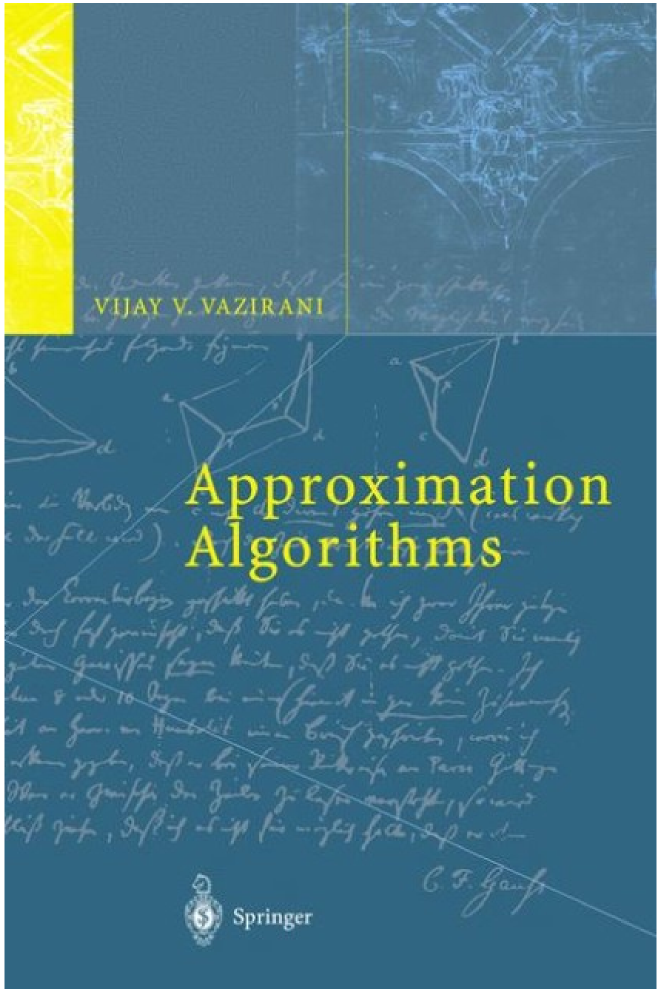
Lectures: on site (English/German, depending on audience)
Fri, 10:15–11:45 (ÜR I)

Tutorials: roughly one exercise sheet per lecture
discussing old solutions and solving new tasks
Tue, 10:15–11:45 (SE I), starting Oct. 24.
Bonus (+0.3 on final grade) for $\geq$ 50% points

Questions/Tasks during the lecture

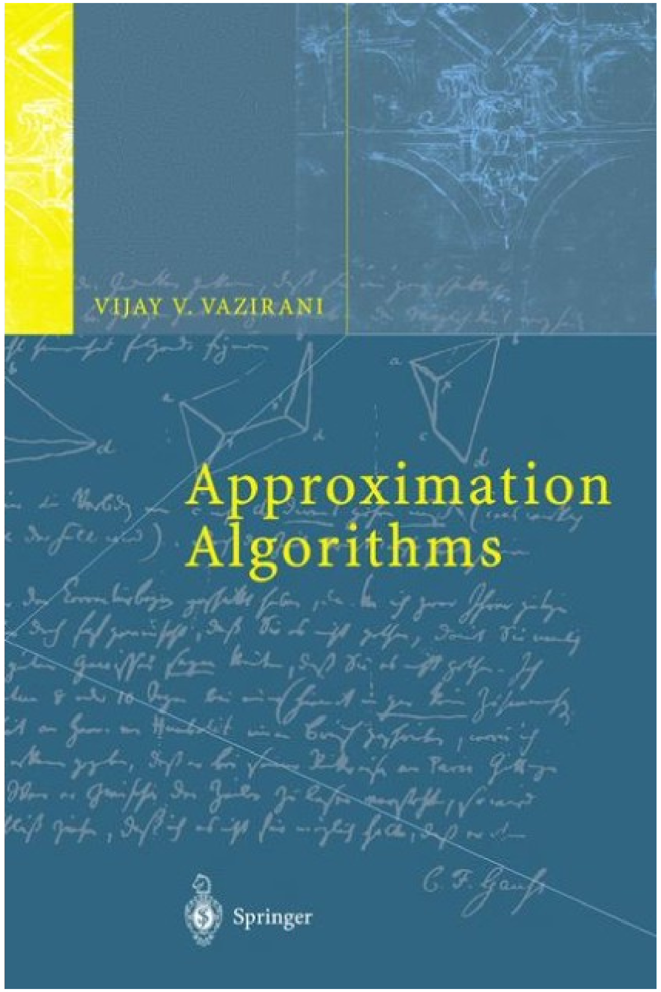
Most slides are due to Joachim Spoerhase,
polishing & colors are due to Philipp Kindermann – thanks!

Textbooks

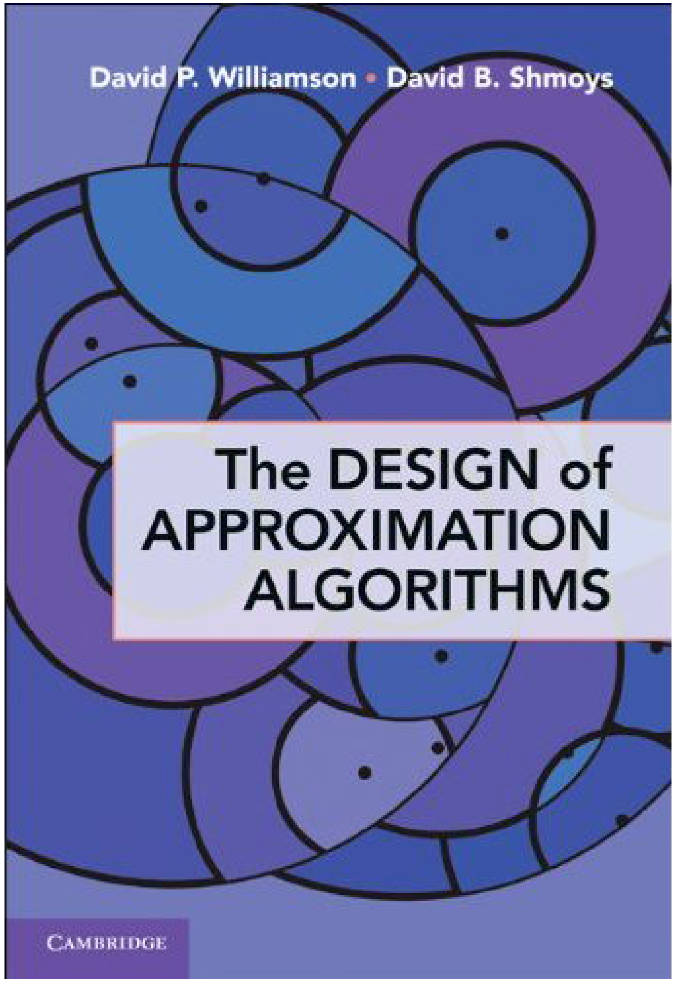


Vijay V. Vazirani:
Approximation Algorithms
Springer-Verlag, 2003.

Textbooks



Vijay V. Vazirani:
Approximation Algorithms
Springer-Verlag, 2003.



D. P. Williamson & D. B. Shmoys:
The Design of Approximation Algorithms
Cambridge-Verlag, 2011.

http://www.designofapproxalgs.com/

Approximation Algorithms

„All exact science is dominated by the idea of approximation.“

– Bertrand Russell (1872 – 1970)

Approximation Algorithms

- Many optimization problems are NP-hard!
  (For example, the traveling salesperson problem.)

Approximation Algorithms

- Many optimization problems are NP-hard!
  (For example, the traveling salesperson problem.)
- $\rightsquigarrow$ an optimal solution cannot be efficiently computed unless P=NP.

Approximation Algorithms

- Many optimization problems are NP-hard!
  (For example, the traveling salesperson problem.)
- $\rightsquigarrow$ an optimal solution cannot be efficiently computed unless P=NP.
- However, good approximate solutions can often be found efficiently!

Approximation Algorithms

- Many optimization problems are NP-hard!
  (For example, the traveling salesperson problem.)
- $\leadsto$ an optimal solution cannot be efficiently computed unless P=NP.
- However, good approximate solutions can often be found efficiently!
- **Techniques** for the design and analysis of approximation algorithms arise from studying specific optimization problems.

Overview

Combinatorial algorithms

- Introduction (Vertex Cover)
- Set Cover via Greedy
- Shortest Superstring
  via reduction to SC
- Steiner Tree via MST
- Multiway Cut via Greedy
- k-Center via Parametrized Pruning
- Min-Degree Spanning Tree
  and local search
- Knapsack via DP and Scaling
- Euclidean TSP via Quadtrees

Overview

Combinatorial algorithms

- Introduction (Vertex Cover)
- Set Cover via Greedy
- Shortest Superstring via reduction to SC
- Steiner Tree via MST
- Multiway Cut via Greedy
- k-Center via Parametrized Pruning
- Min-Degree Spanning Tree and local search
- Knapsack via DP and Scaling
- Euclidean TSP via Quadtrees

LP-based algorithms

- introduction to LP-Duality
- Set Cover via LP Rounding
- Set Cover via Primal–Dual Schema
- Maximum Satisfiability
- Scheduling und Extreme Point Solutions
- Steiner Forest via Primal–Dual

Approximation Algorithms

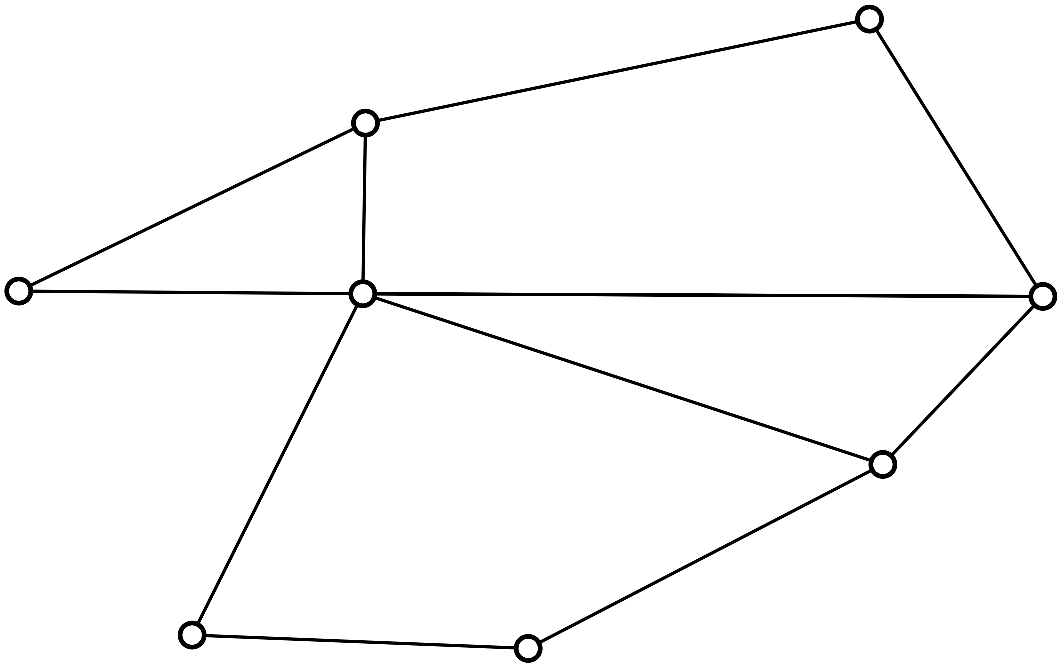
Lecture 1:
Introduction and Vertex Cover

Part II:
(Cardinality) Vertex Cover

VertexCover (card.)

Input: Graph $G = (V, E)$

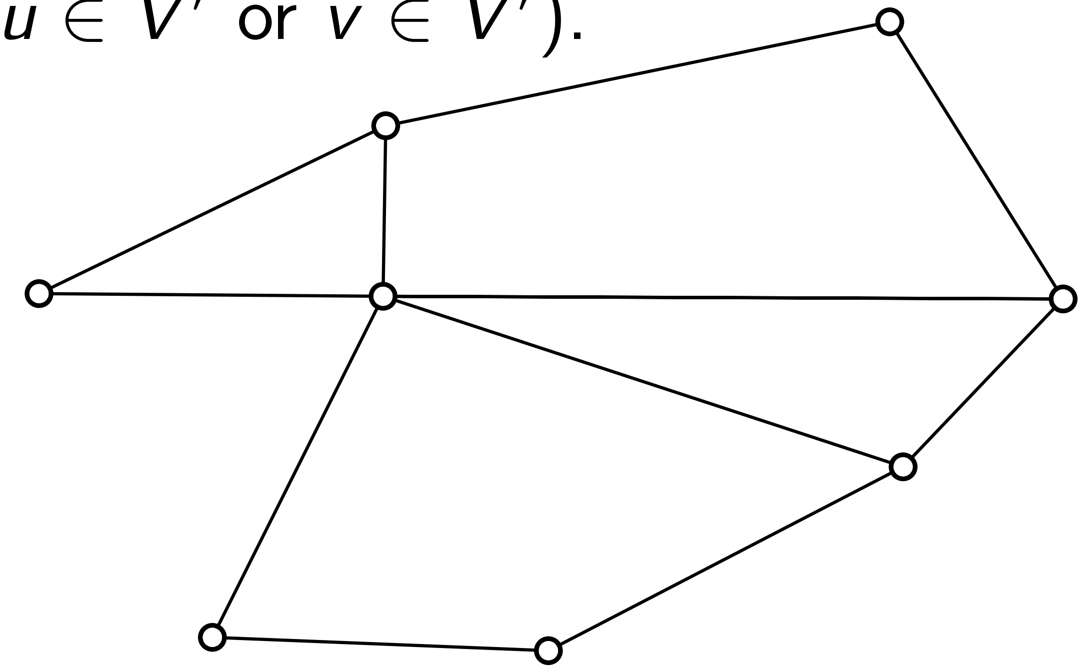
Output:



VertexCover (card.)

Input: Graph $G = (V, E)$

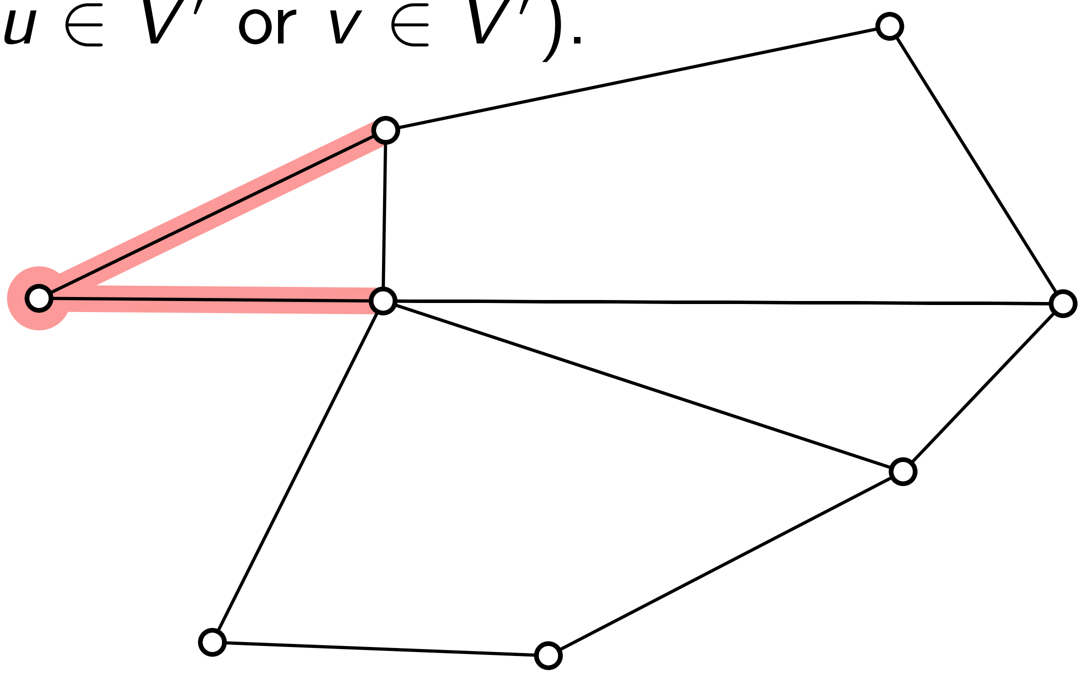
Output: a minimum **vertex cover**, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover (card.)

Input: Graph $G = (V, E)$

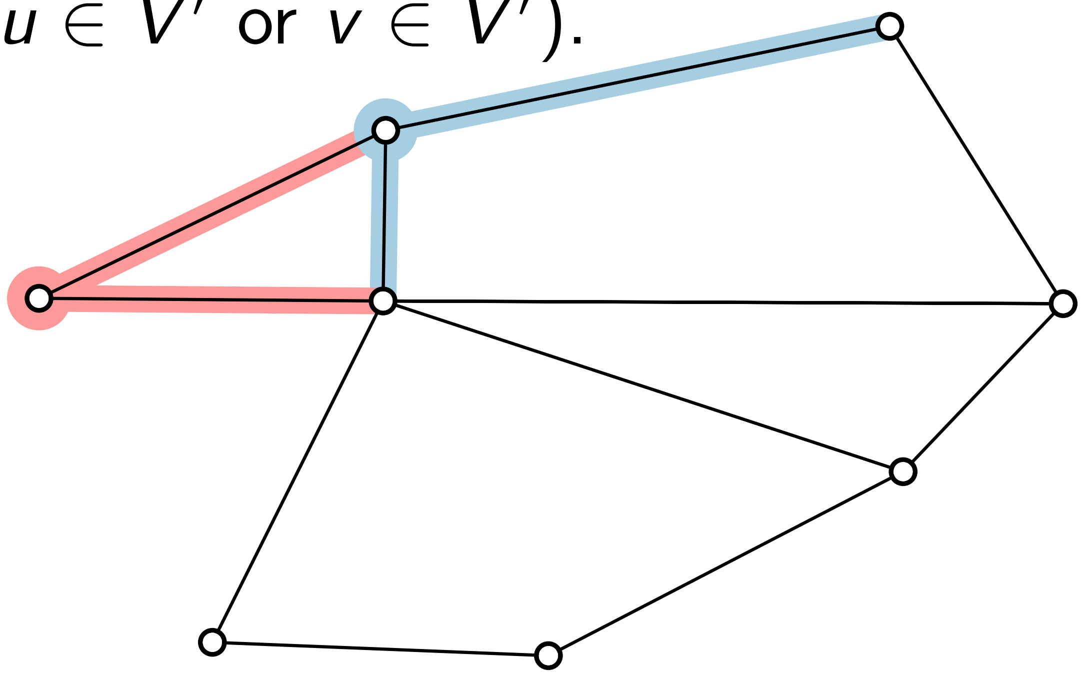
Output: a minimum **vertex cover**, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover (card.)

Input: Graph $G = (V, E)$

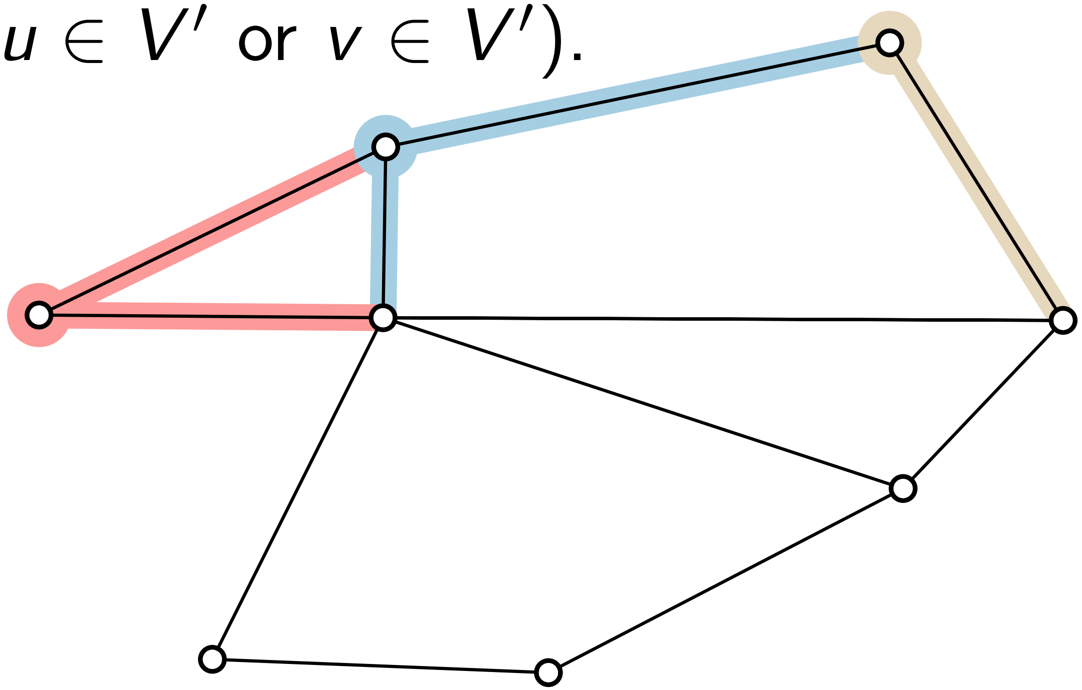
Output: a minimum **vertex cover**, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover (card.)

Input: Graph $G = (V, E)$

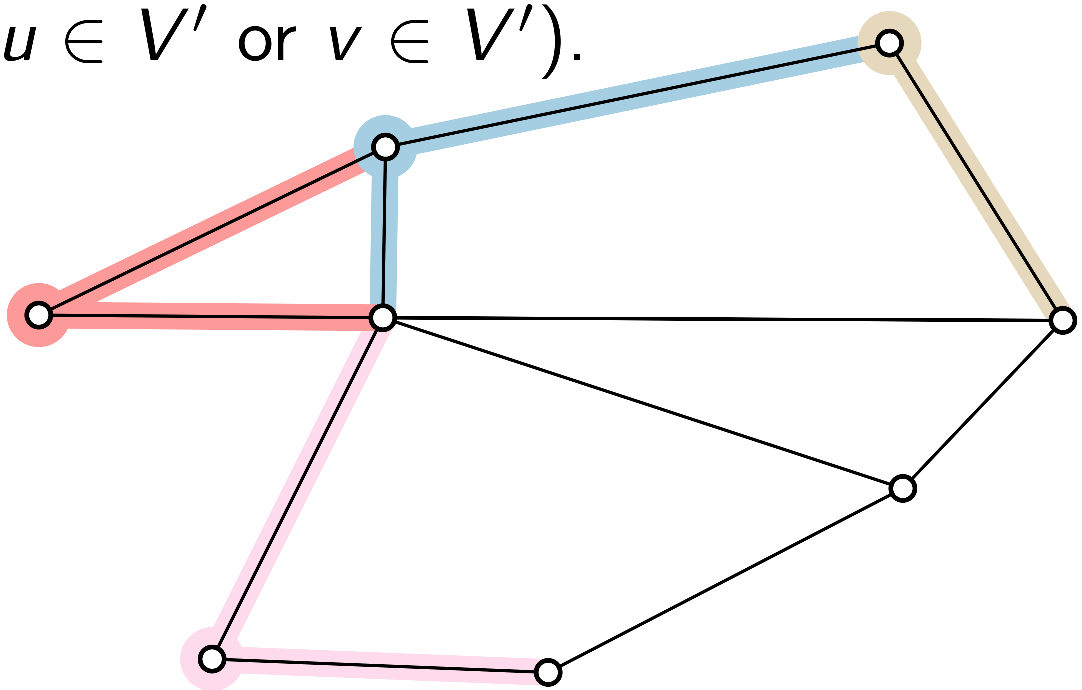
Output: a minimum **vertex cover**, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover (card.)

Input: Graph $G = (V, E)$

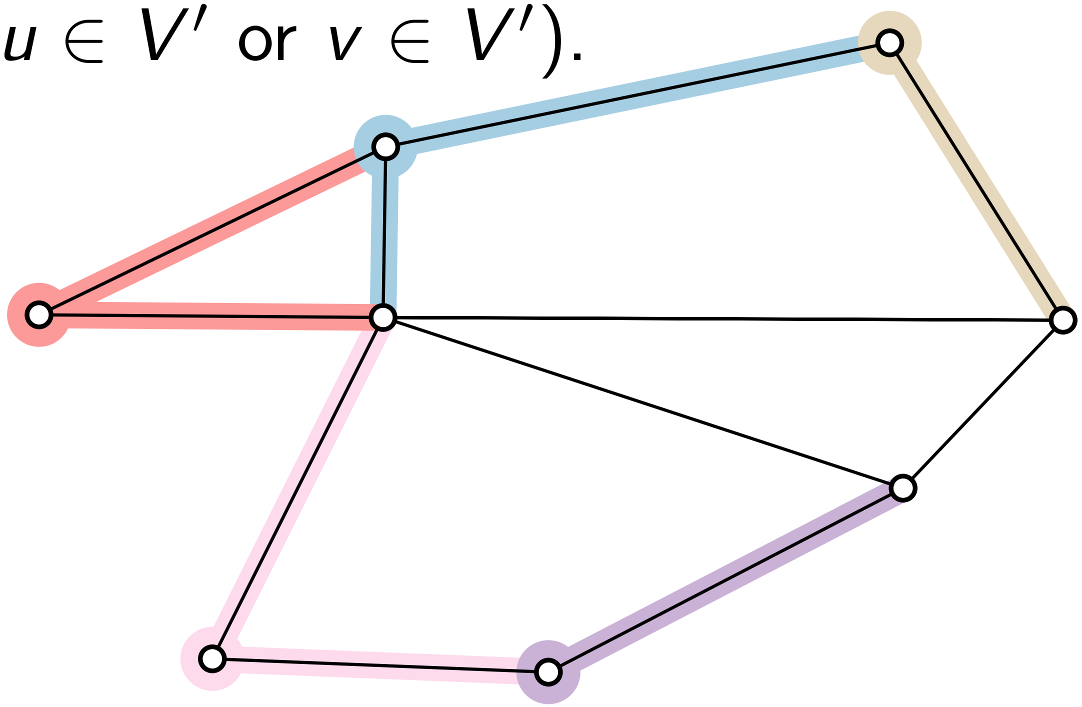
Output: a minimum **vertex cover**, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover (card.)

Input: Graph $G = (V, E)$

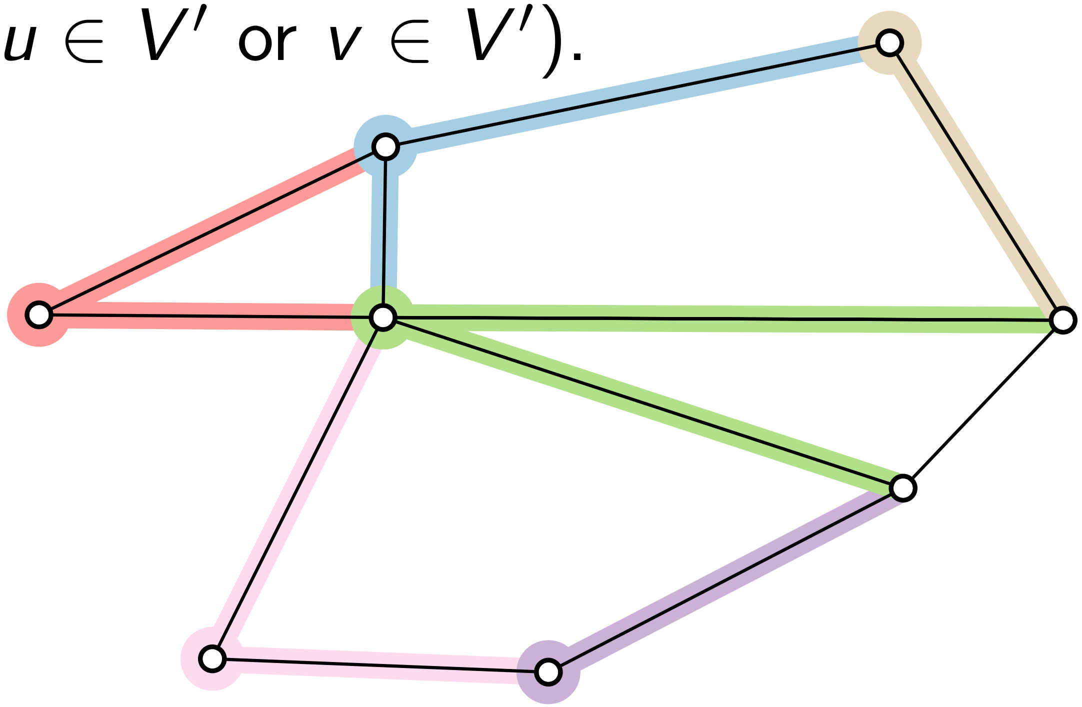
Output: a minimum **vertex cover**, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



Vertex Cover (card.)

Input: Graph $G = (V, E)$

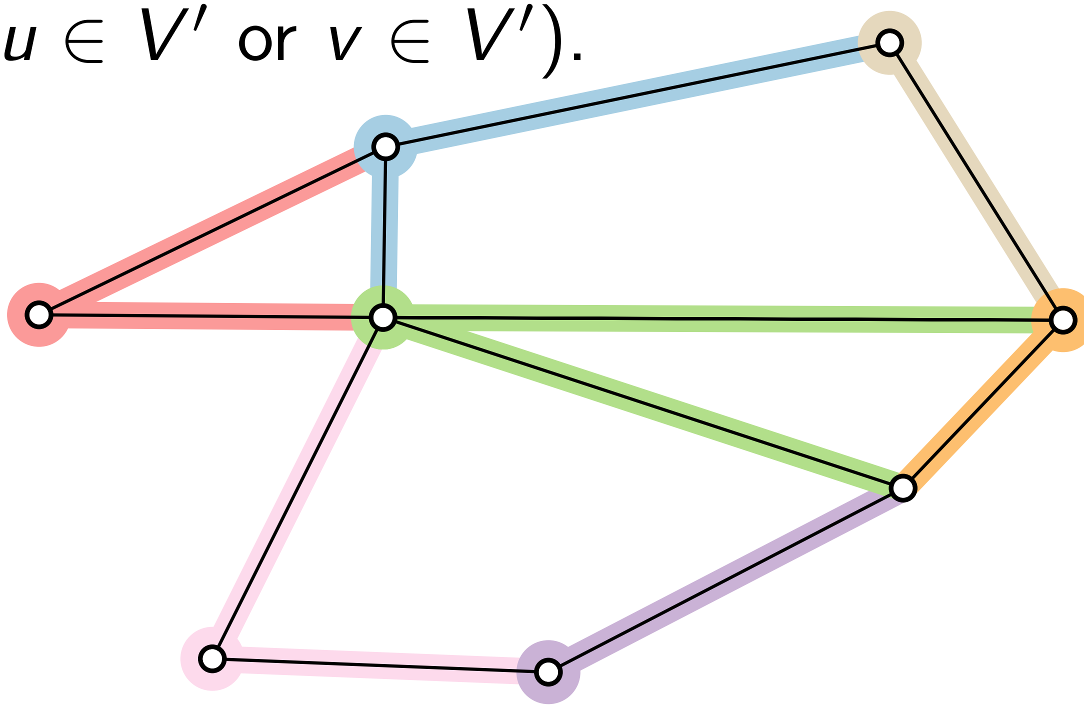
Output: a minimum **vertex cover**, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover (card.)

Input: Graph $G = (V, E)$

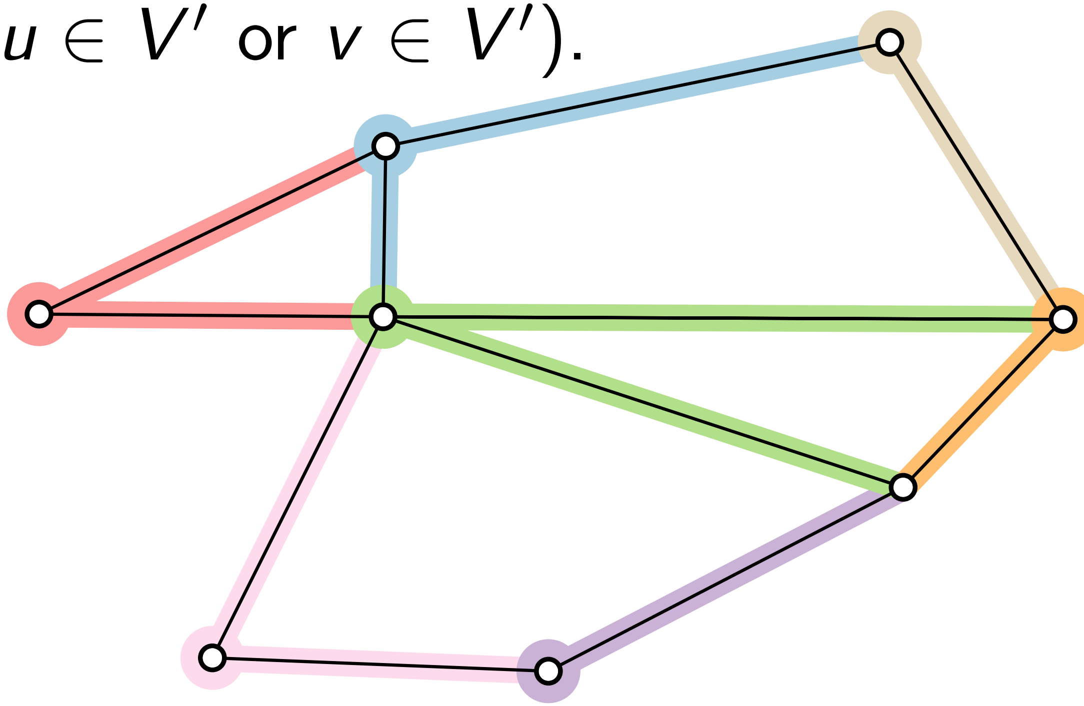
Output: a minimum **vertex cover**, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



VertexCover (card.)

Input: Graph $G = (V, E)$

Output: a minimum **vertex cover**, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

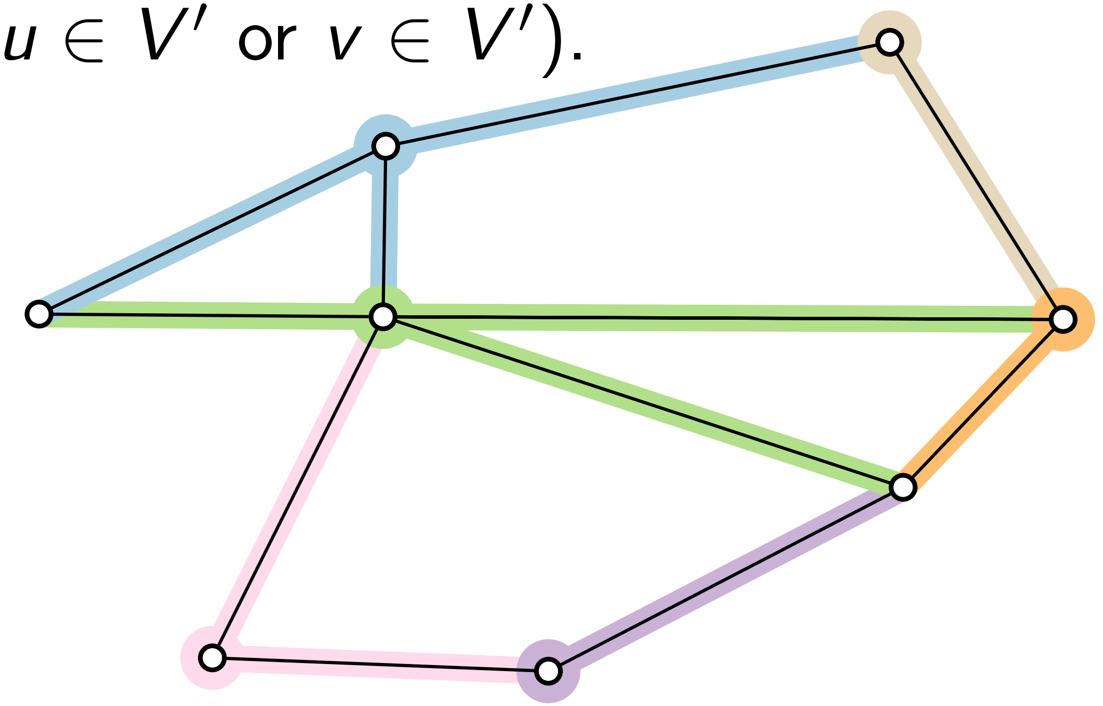


any vertex cover

VertexCover (card.)

Input: Graph $G = (V, E)$

Output: a minimum **vertex cover**, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

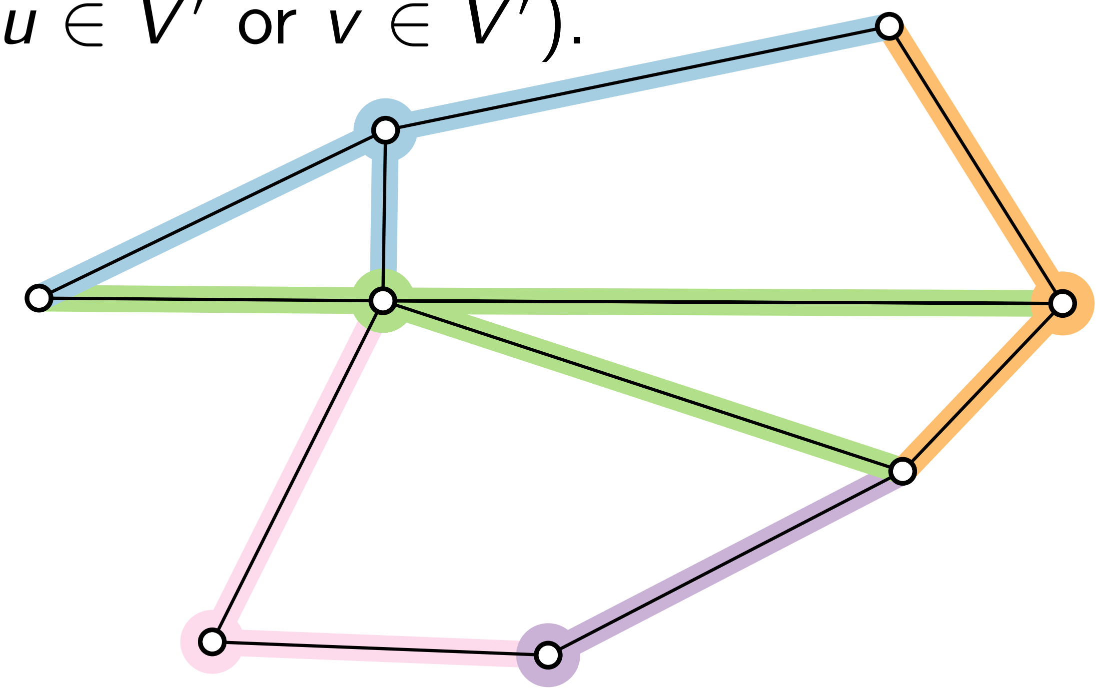


any vertex cover

VertexCover (card.)

Input: Graph $G = (V, E)$

Output: a minimum **vertex cover**, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

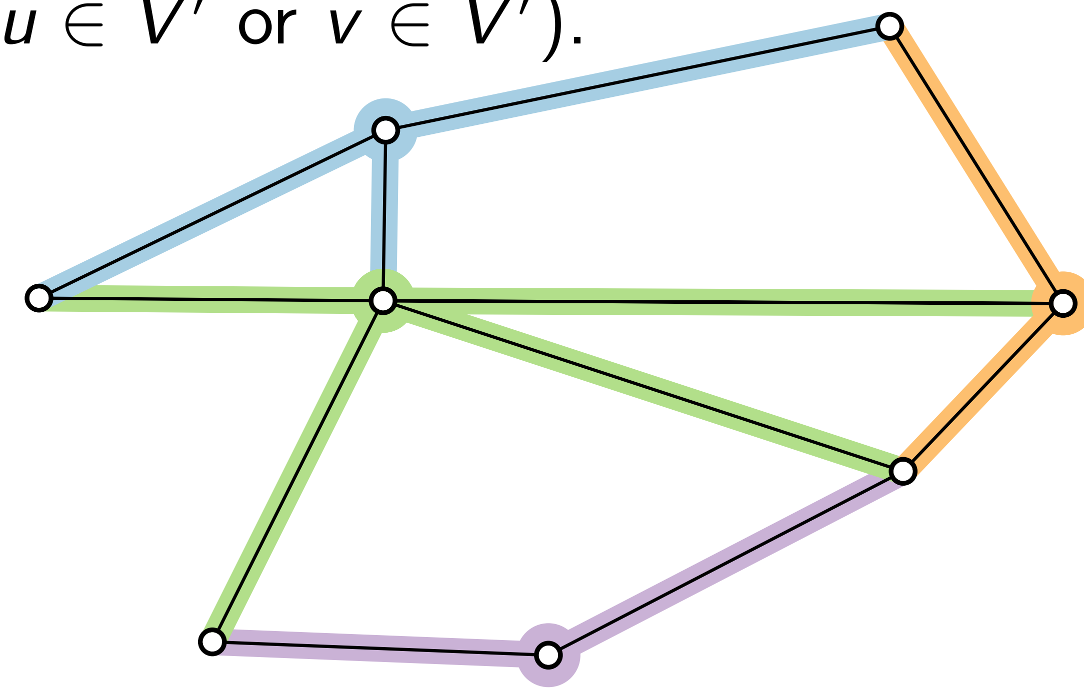


any vertex cover

VertexCover (card.)

Input: Graph $G = (V, E)$

Output: a minimum **vertex cover**, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

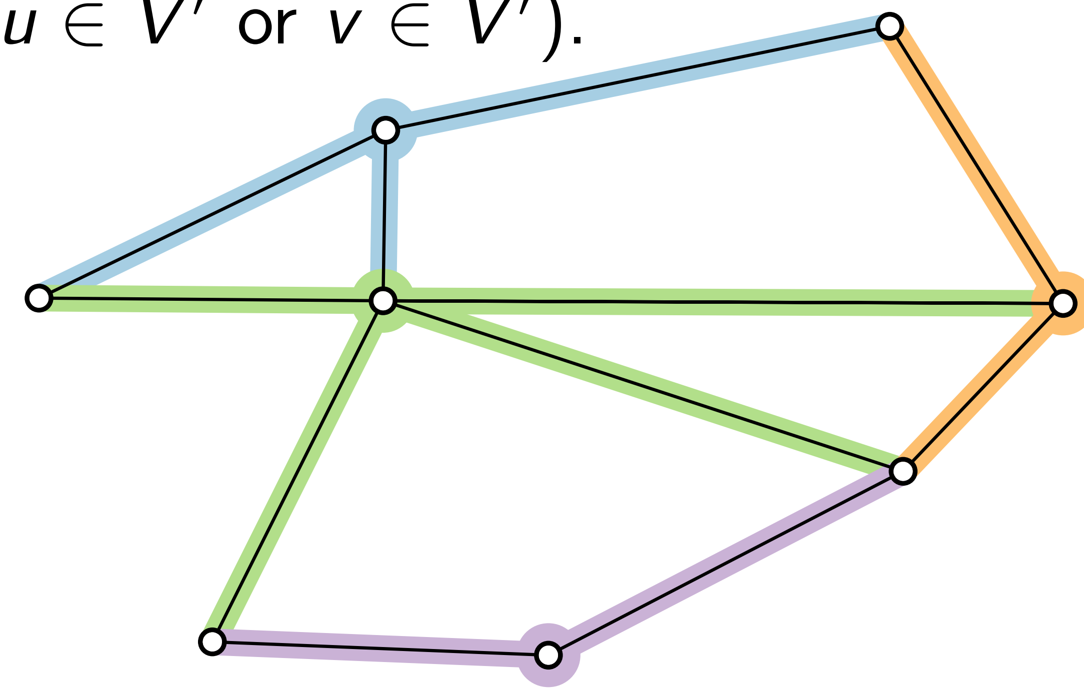


any vertex cover

VertexCover (card.)

Input: Graph $G = (V, E)$

Output: a minimum **vertex cover**, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

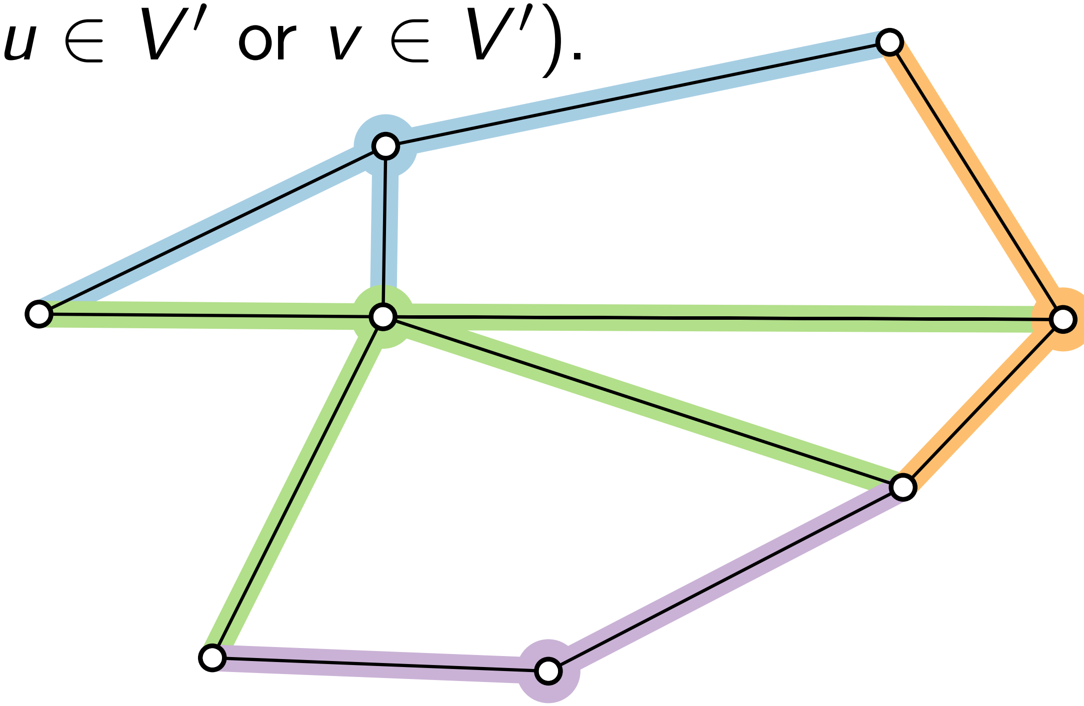


Optimum ($\mathrm{OPT} = 4$)

VertexCover (card.)

Input: Graph $G = (V, E)$

Output: a minimum **vertex cover**, that is,
a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).

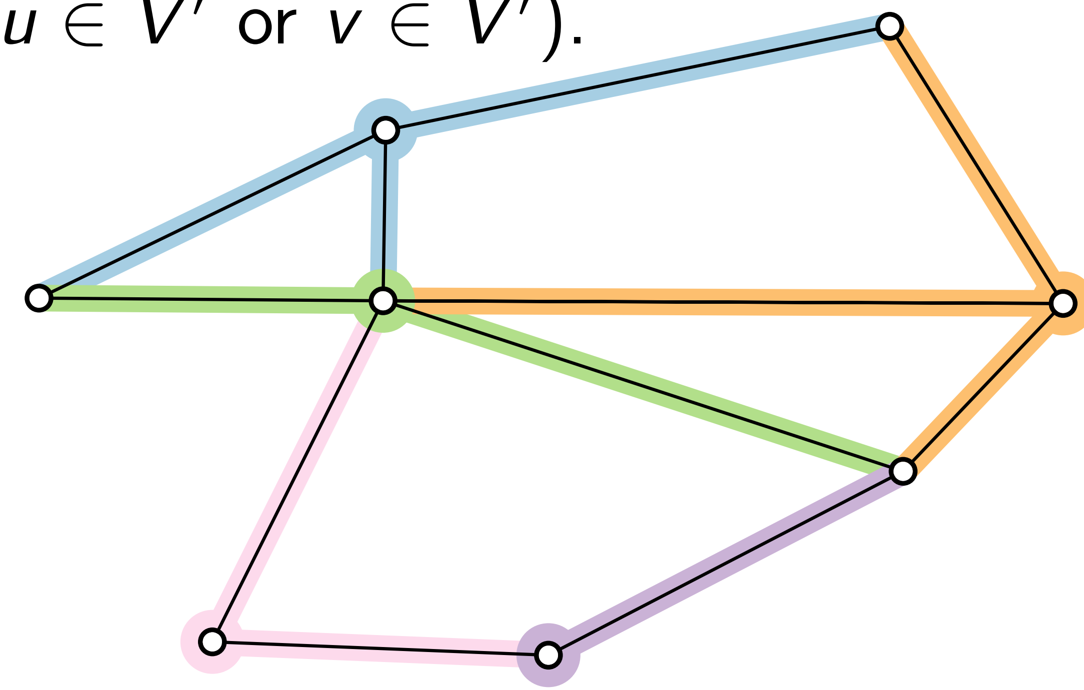


Optimum (OPT $= 4$) – but in general NP-hard to find :-(

VertexCover (card.)

Input: Graph $G = (V, E)$

Output: a minimum **vertex cover**, that is, a minimum-cardinality vertex set $V' \subseteq V$ such that every edge is **covered** (i.e., for every $uv \in E$, it holds that $u \in V'$ or $v \in V'$).



"good" (5/4-) approximate solution

Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part III:
NP-Optimization Problem

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$,
  a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$,
  a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:
  - for each solution $s \in S_\Pi(I)$,
    its size $|s|$ is polynomially bounded in $|I|$, and

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$,
  a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:
  - for each solution $s \in S_\Pi(I)$,
    its size $|s|$ is polynomially bounded in $|I|$, and
  - there is a polynomial-time algorithm that decides,
    for each pair (s, I), whether $s \in S_\Pi(I)$.

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$,
  a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:
  - for each solution $s \in S_\Pi(I)$,
    its size $|s|$ is polynomially bounded in $|I|$, and
  - there is a polynomial-time algorithm that decides,
    for each pair (s, I), whether $s \in S_\Pi(I)$.
- A polynomial time computable **objective function** obj_Π
  which assigns a positive objective value $\mathrm{obj}_\Pi(I, s) \geq 0$ to
  any given pair (s, I) with $s \in S_\Pi(I)$.

NP-Optimization Problem

An **NP-optimization problem** Π is given by:

- A set D_Π of **instances**.
  We denote the size of an instance $I \in D_\Pi$ by $|I|$.
- For each instance $I \in D_\Pi$,
  a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I such that:
  - for each solution $s \in S_\Pi(I)$,
    its size $|s|$ is polynomially bounded in $|I|$, and
  - there is a polynomial-time algorithm that decides,
    for each pair (s, I), whether $s \in S_\Pi(I)$.
- A polynomial time computable **objective function** obj_Π
  which assigns a positive objective value $\text{obj}_\Pi(I, s) \geq 0$ to
  any given pair (s, I) with $s \in S_\Pi(I)$.
- Π is either a minimization or maximization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$

For $I \in D_\Pi$: $|I| =$

$S_\Pi(I) =$

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$

Π is a m...imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$

$S_\Pi(I) =$

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$

Π is a m...imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$

$G = (V, E)$ $S_\Pi(I) =$

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$

Π is a m...imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for Π = Vertex Cover.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

$G = (V, E)$ $S_\Pi(I) =$

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$

Π is a m...imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

$G=(V,E)$ $S_\Pi(I) =$ Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$

Π is a m...imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

$G=(V,E)$ $S_\Pi(I) =$ Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

  $s \subseteq V \Rightarrow |s| \leq |V| = |I|$

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$?

$\text{obj}_\Pi(I, s) =$

Π is a m...imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

$G=(V,E)$ $S_\Pi(I) =$ Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

  $s \subseteq V \Rightarrow |s| \leq |V| = |I|$

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$? Test whether all edges are covered.

$\text{obj}_\Pi(I, s) =$

Π is a m...imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi = \text{Vertex Cover}$.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $|I| =$ Number of vertices $|V|$

$G=(V,E)$ $S_\Pi(I) =$ Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?
  $s \subseteq V \Rightarrow |s| \leq |V| = |I|$
- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$? Test whether all edges are covered.

$\text{obj}_\Pi(I, s) = |s|$

Π is a m...imization problem.

VertexCover: NP-Optimization Problem

Task: Fill in the gaps for $\Pi = \textsc{Vertex Cover}$.

$D_\Pi =$ Set of all graphs

For $I \in D_\Pi$: $\quad |I| =$ Number of vertices $|V|$

$G = (V, E)$ $\quad S_\Pi(I) =$ Set of all vertex covers of G

- Why is $|s| \in \text{poly}(|I|)$ for every $s \in S_\Pi(I)$?

  $s \subseteq V \Rightarrow |s| \leq |V| = |I|$

- For a given pair (s, I), how can we efficiently decide whether $s \in S_\Pi(I)$? Test whether all edges are covered.

$\text{obj}_\Pi(I, s) = |s|$

Π is a minimization problem.

Optimum and Optimal Objective Value

Let Π be a minimization problem and $I \in D_\Pi$ an instance of Π.

Optimum and Optimal Objective Value

Let Π be a minimization problem and $I \in D_\Pi$ an instance of Π.

A feasible solution $s^* \in S_\Pi(I)$ is **optimal** if $\mathrm{obj}_\Pi(I, s^*)$ is minimal among the objective values attained by the feasible solutions of I.

Optimum and Optimal Objective Value

maximization problem

Let Π be a minimization problem and $I \in D_\Pi$ an instance of Π.

A feasible solution $s^* \in S_\Pi(I)$ is **optimal** if

maximal

$\text{obj}_\Pi(I, s^*)$ is minimal among the objective values attained by the feasible solutions of I.

Optimum and Optimal Objective Value

maximization problem
Let Π be a minimization problem and $I \in D_\Pi$ an instance of Π.

A feasible solution $s^* \in S_\Pi(I)$ is **optimal** if

maximal
$\mathrm{obj}_\Pi(I, s^*)$ is minimal among the objective values

attained by the feasible solutions of I.

The optimal value $\mathrm{obj}_\Pi(I, s^*)$ of the objective function is denoted by $\mathrm{OPT}_\Pi(I)$ or simply by OPT in context.

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^+$.

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^+$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^+$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)}$$

Approximation Algorithms

Let Π be a minimization problem and $\alpha \in \mathbb{Q}^+$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)} \leq \alpha.$$

Approximation Algorithms

Let Π be a minimization problem and ~~$\alpha \in \mathbb{Q}^+$~~ $\alpha \colon \mathbb{N} \to \mathbb{Q}$.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)} \leq \cancel{\alpha}\ \alpha(|I|).$$

Approximation Algorithms

maximization problem $\alpha\colon \mathbb{N} \to \mathbb{Q}$

Let Π be a minimization problem and ~~$\alpha \in \mathbb{Q}^+$~~.

A factor-α approximation algorithm for Π is an efficient algorithm that provides, for **any** instance $I \in D_\Pi$, a feasible solution $s \in S_\Pi(I)$ such that

$$\frac{\mathrm{obj}_\Pi(I, s)}{\mathrm{OPT}_\Pi(I)} \;\overset{\geq}{\leq}\; \cancel{\alpha}\,\alpha(|I|).$$

Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part IV:
Approximation Algorithm for VertexCover

Approximation Alg. for VertexCover

Ideas?

Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy

Approximation Alg. for VertexCover

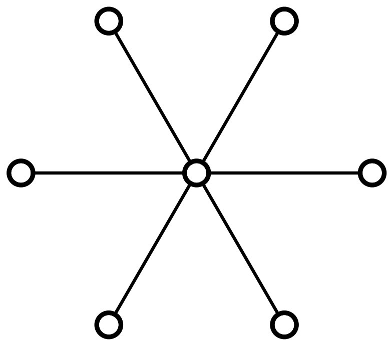
Ideas?

- Edge-Greedy
- Vertex-Greedy

Approximation Alg. for VertexCover

Ideas?

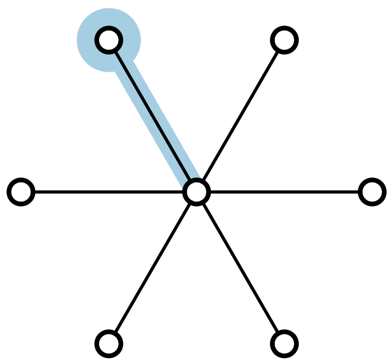
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

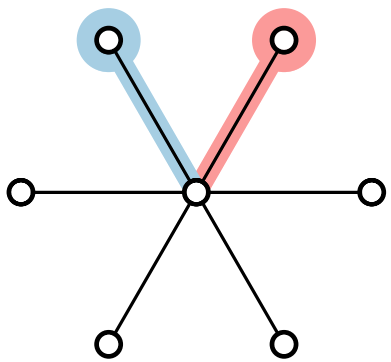
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

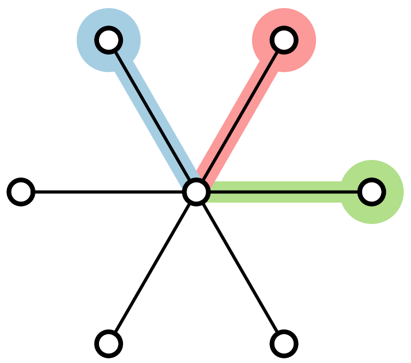
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

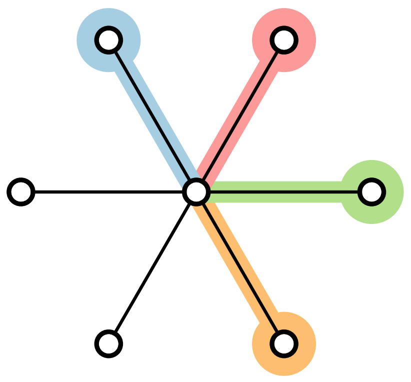
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

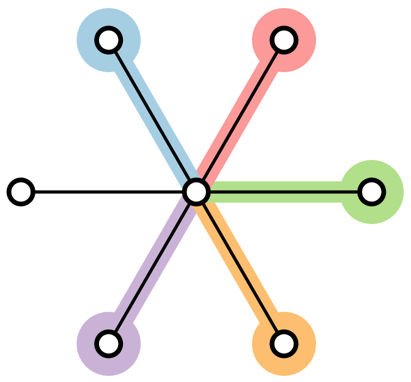
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

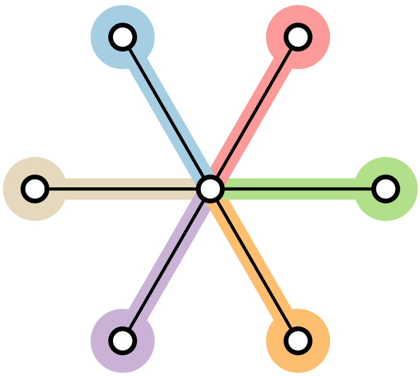
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

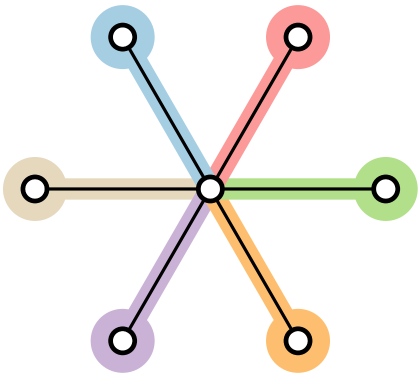
- Edge-Greedy
- Vertex-Greedy



Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy

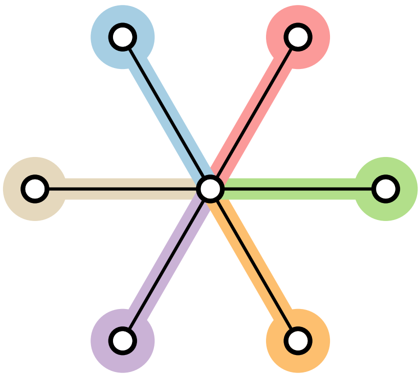


Quality?

Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



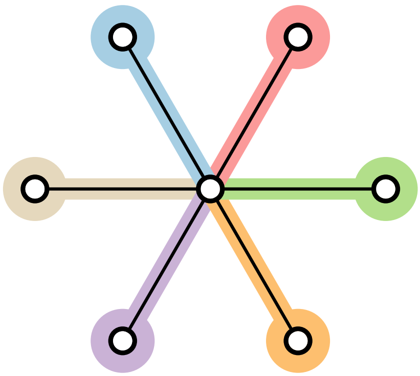
Quality?

Problem: How can we estimate $\mathrm{obj}_{\Pi}(I, s)/\mathrm{OPT}$,
when it is hard to compute OPT?

Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



Quality?

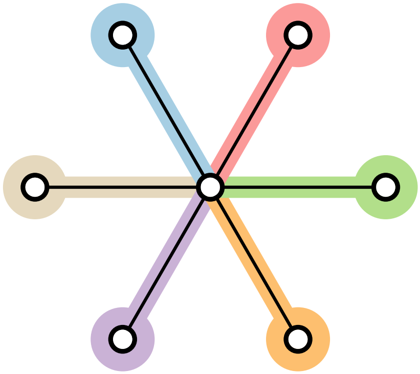
Problem: How can we estimate $\text{obj}_\Pi(I, s)/\text{OPT}$, when it is hard to compute OPT?

Idea: Find a "good" lower bound $L \leq \text{OPT}$ for OPT and compare it to our approximate solution.

Approximation Alg. for VertexCover

Ideas?

- Edge-Greedy
- Vertex-Greedy



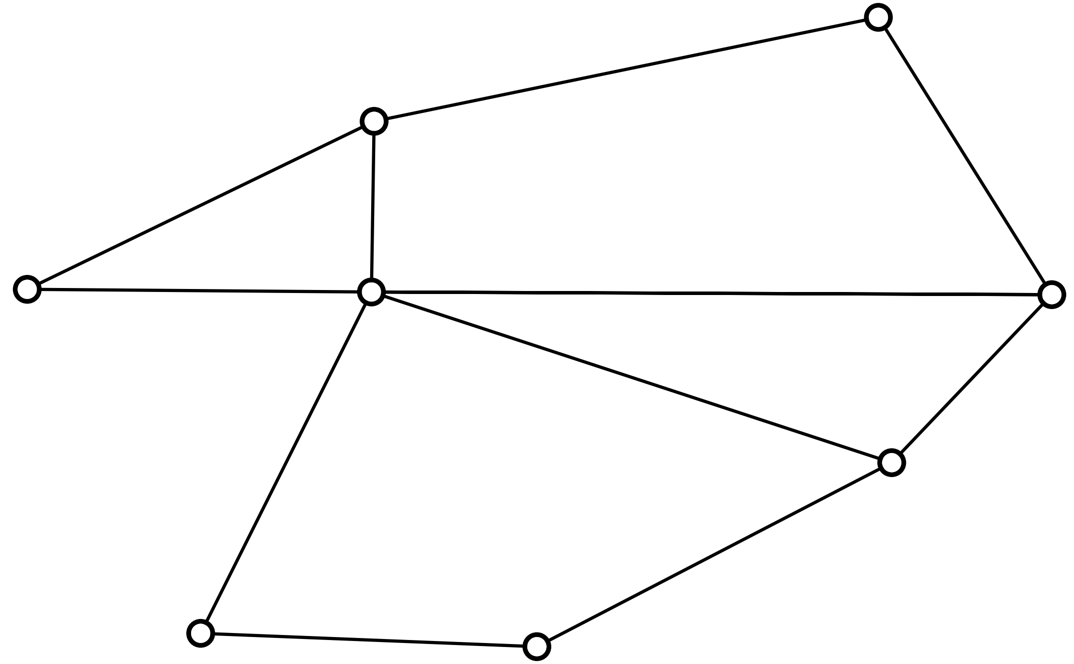
Quality?

Problem: How can we estimate $\text{obj}_\Pi(I, s)/\text{OPT}$, when it is hard to compute OPT?

Idea: Find a "good" lower bound $L \leq \text{OPT}$ for OPT and compare it to our approximate solution.

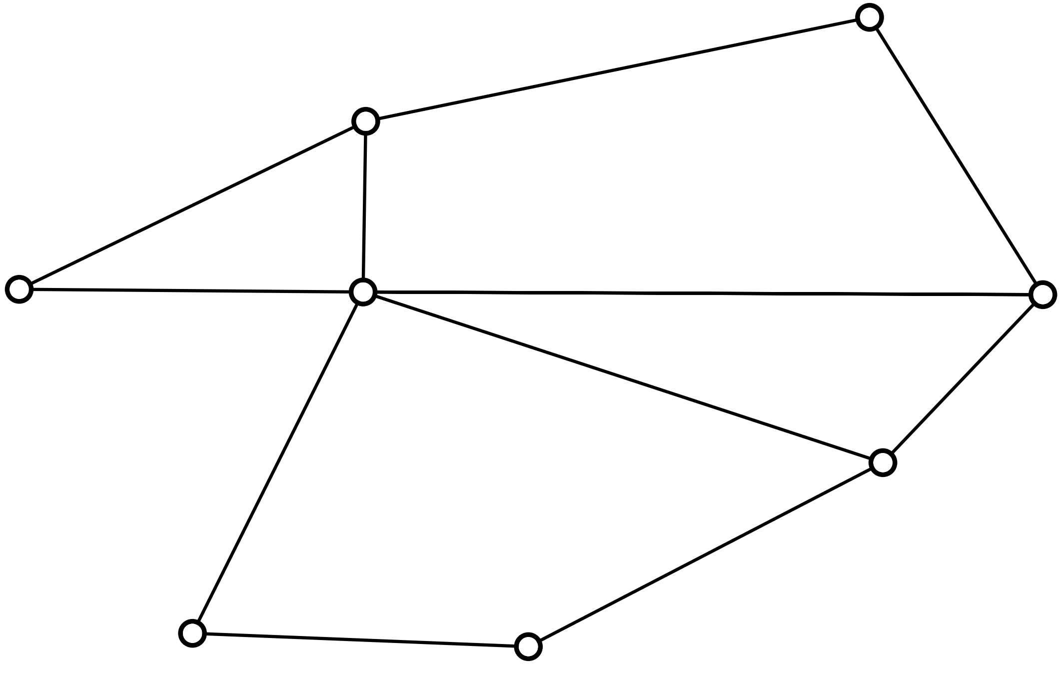
$$\frac{\text{obj}_\Pi(I, s)}{\text{OPT}} \leq \frac{\text{obj}_\Pi(I, s)}{L}$$

Lower Bound by Matchings



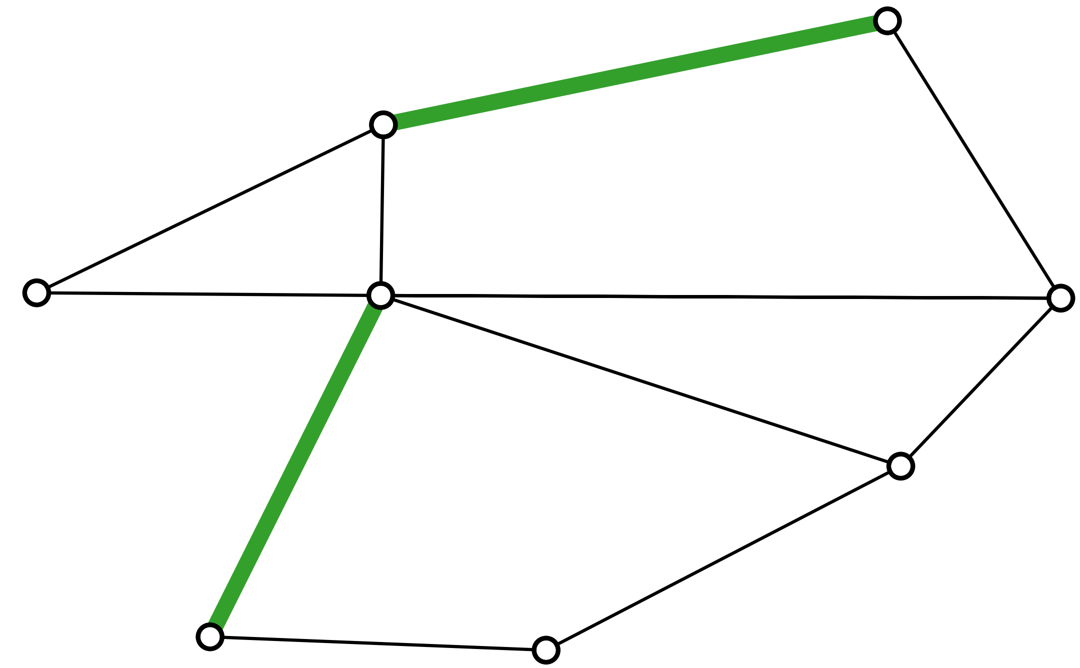
Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).



Lower Bound by Matchings

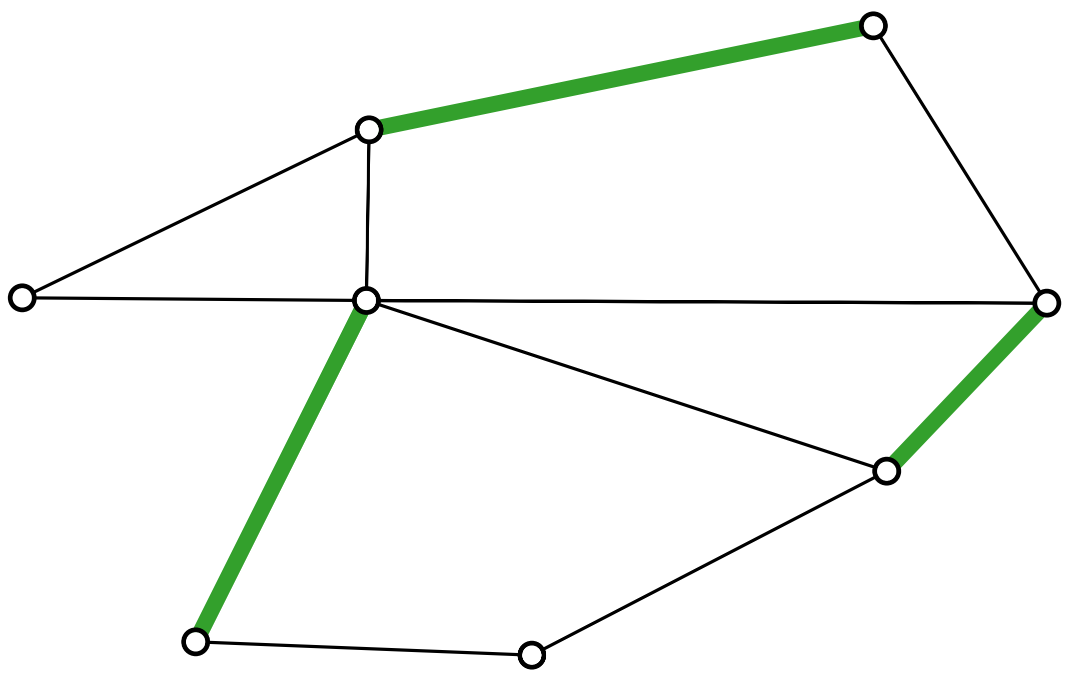
Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).



Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.

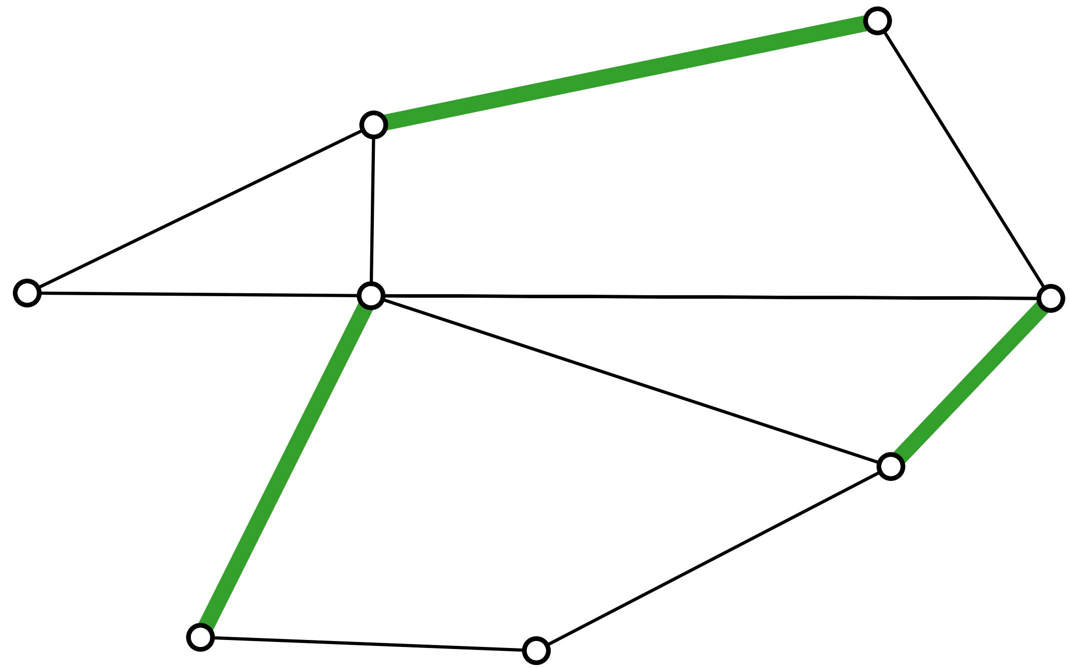


Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.

$\mathrm{OPT} \geq$



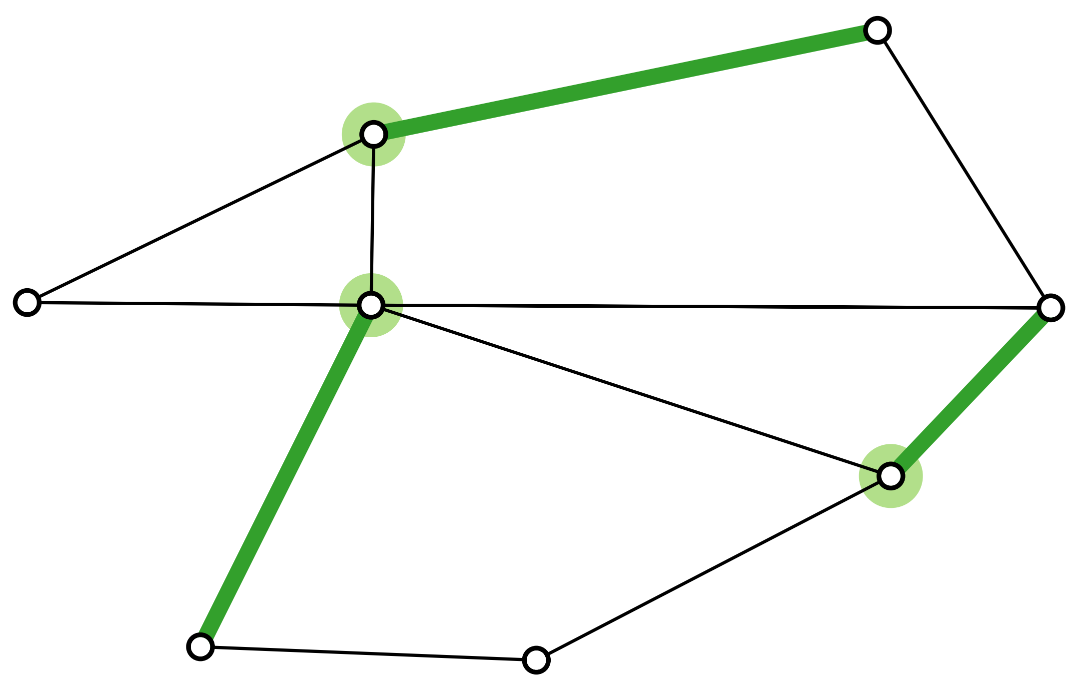
Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.

OPT $\geq$

Vertex cover of M



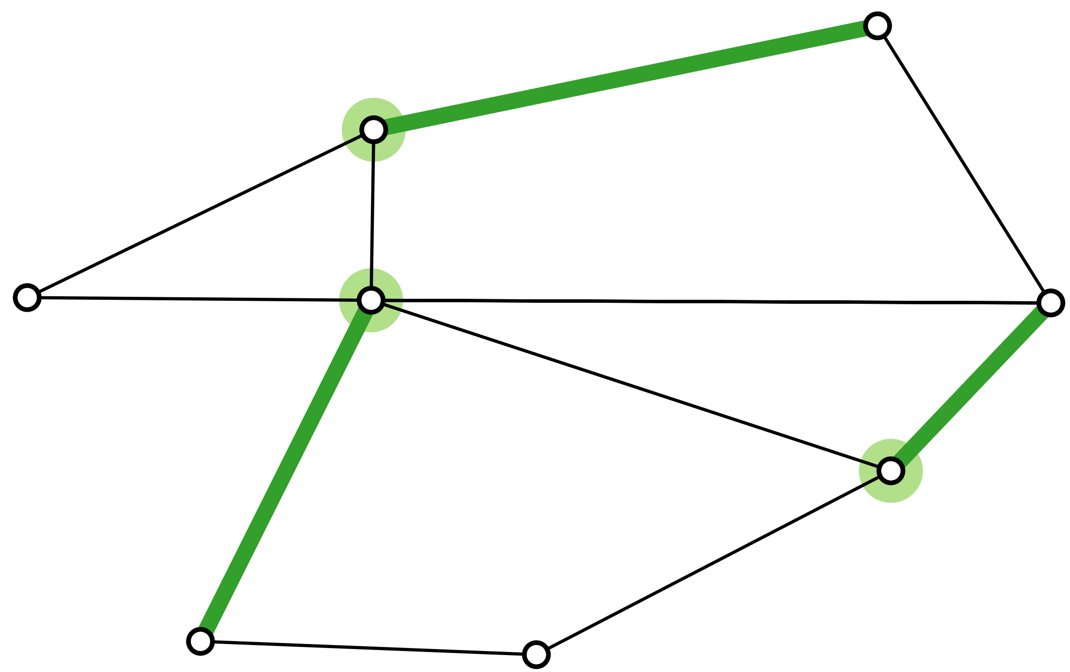
Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.

$\mathrm{OPT} \geq |M|$

Vertex cover of M



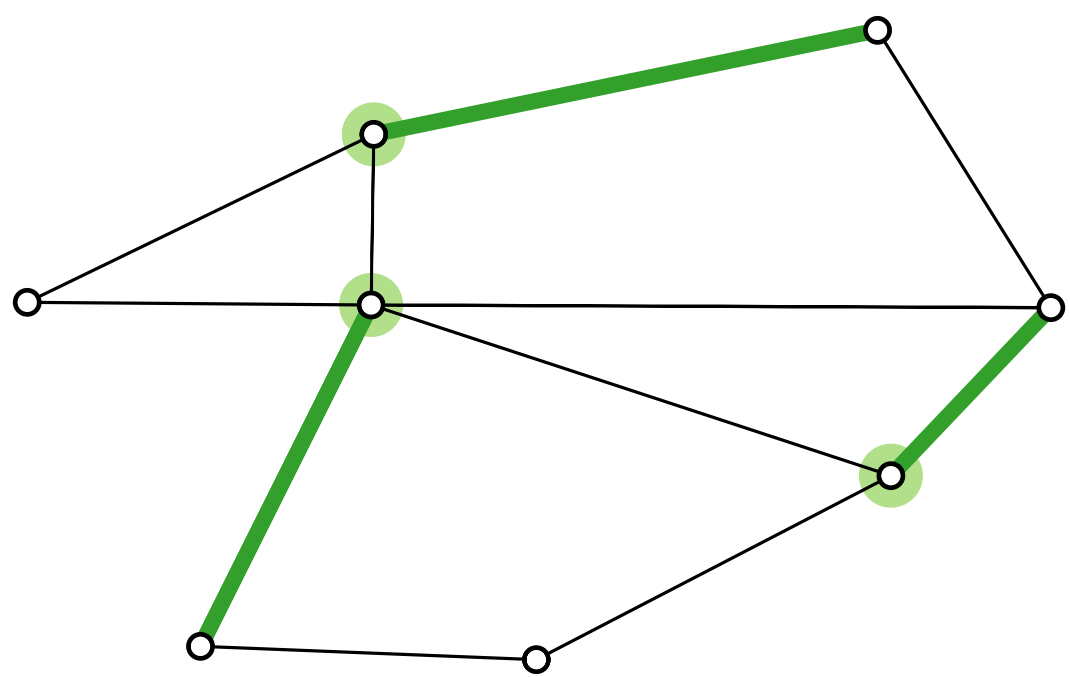
Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.

$\mathrm{OPT} \geq |M|$
$\mathrm{OPT} = |M|$?

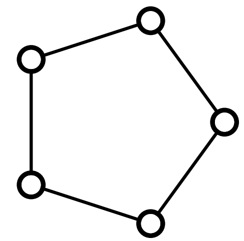
Vertex cover of M



Lower Bound by Matchings

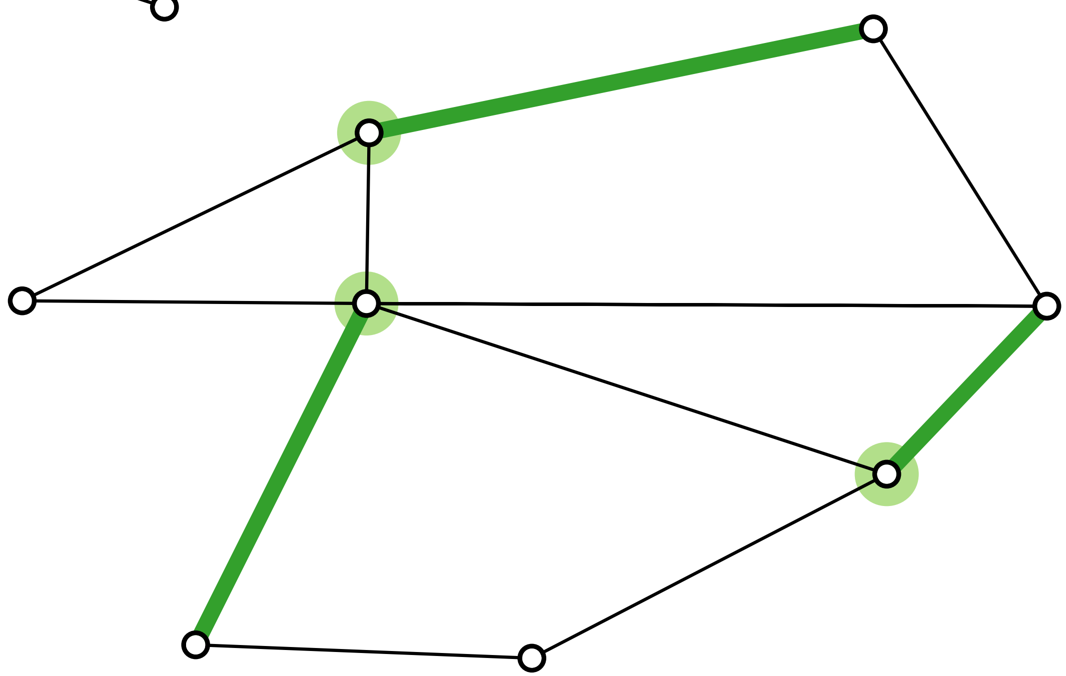
Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$
$\mathrm{OPT} = |M|$?

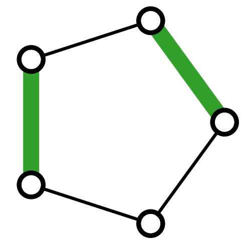
Vertex cover of M



Lower Bound by Matchings

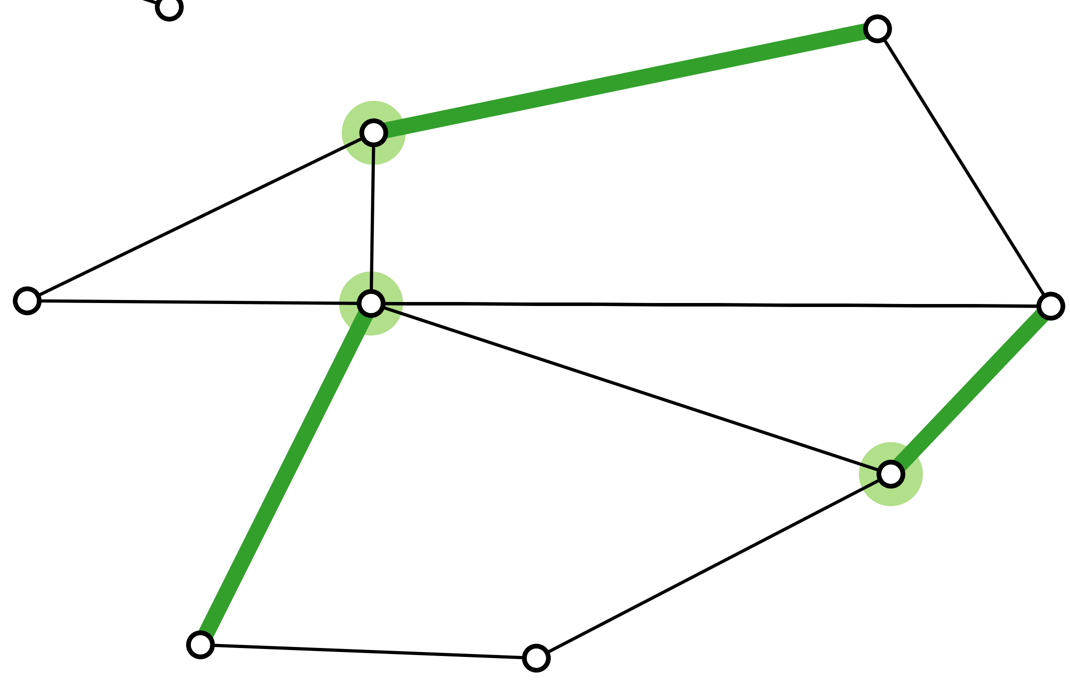
Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$
$\mathrm{OPT} = |M|$?

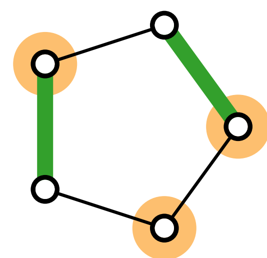
Vertex cover of M



Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

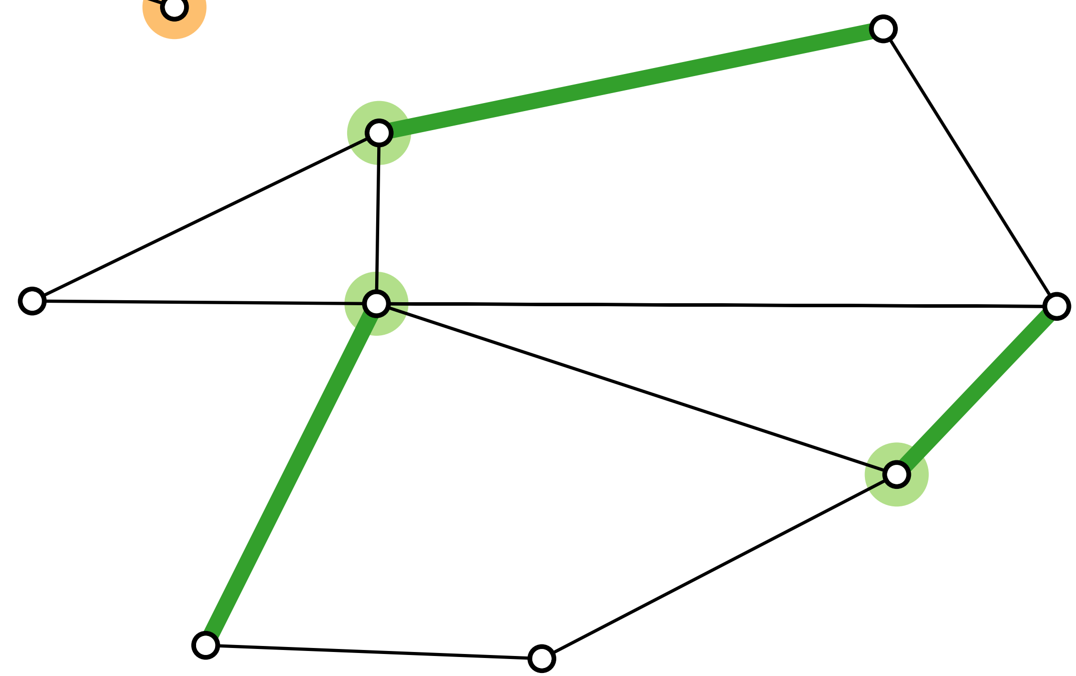
M is **maximal** if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

$\mathrm{OPT} = |M|$?

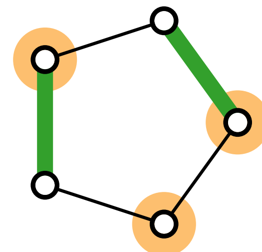
Vertex cover of M



Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

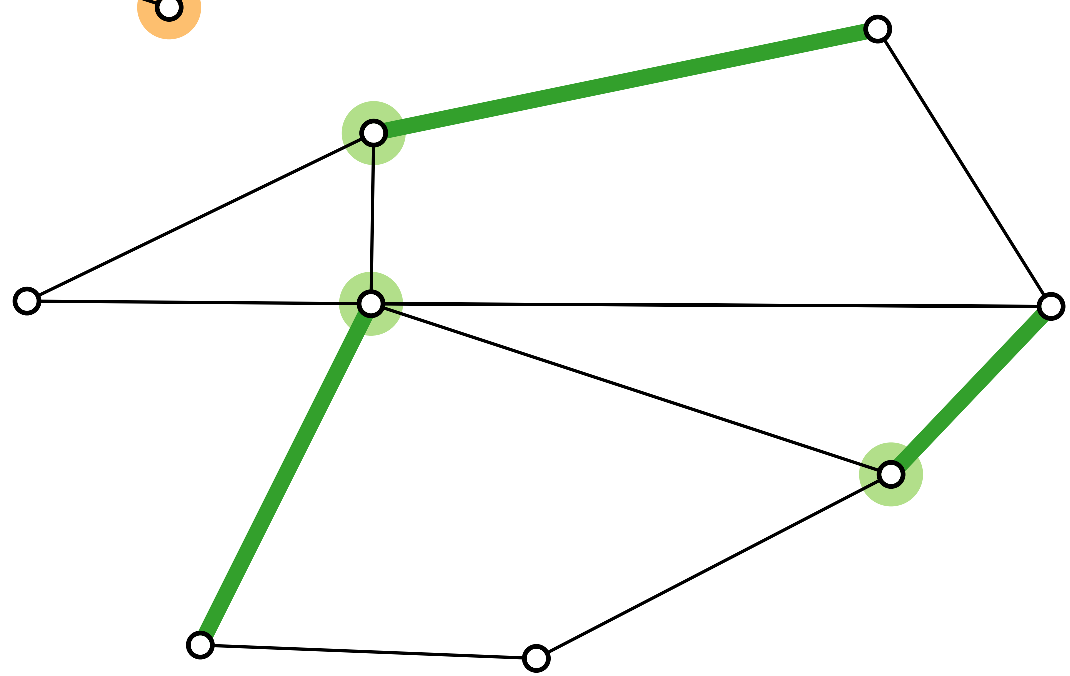
M is **maximal** if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$~~ ?

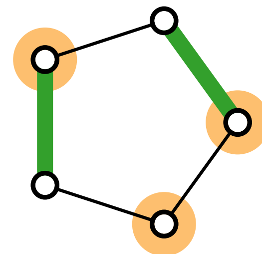
Vertex cover of M



Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.

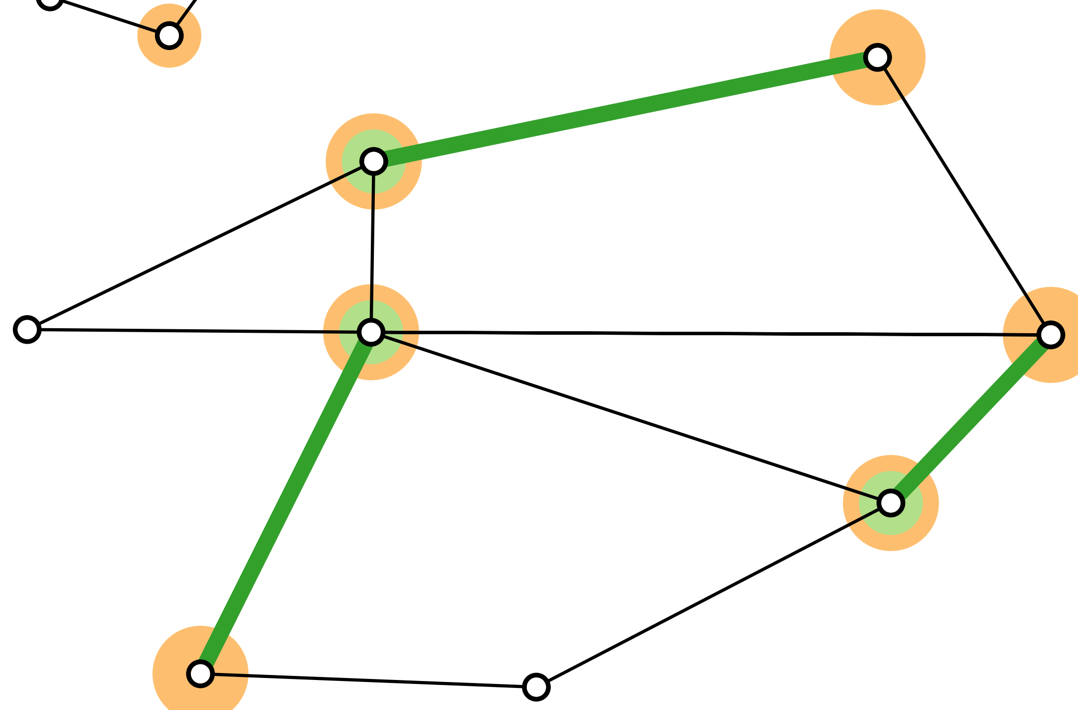


$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$?~~

Vertex cover of M

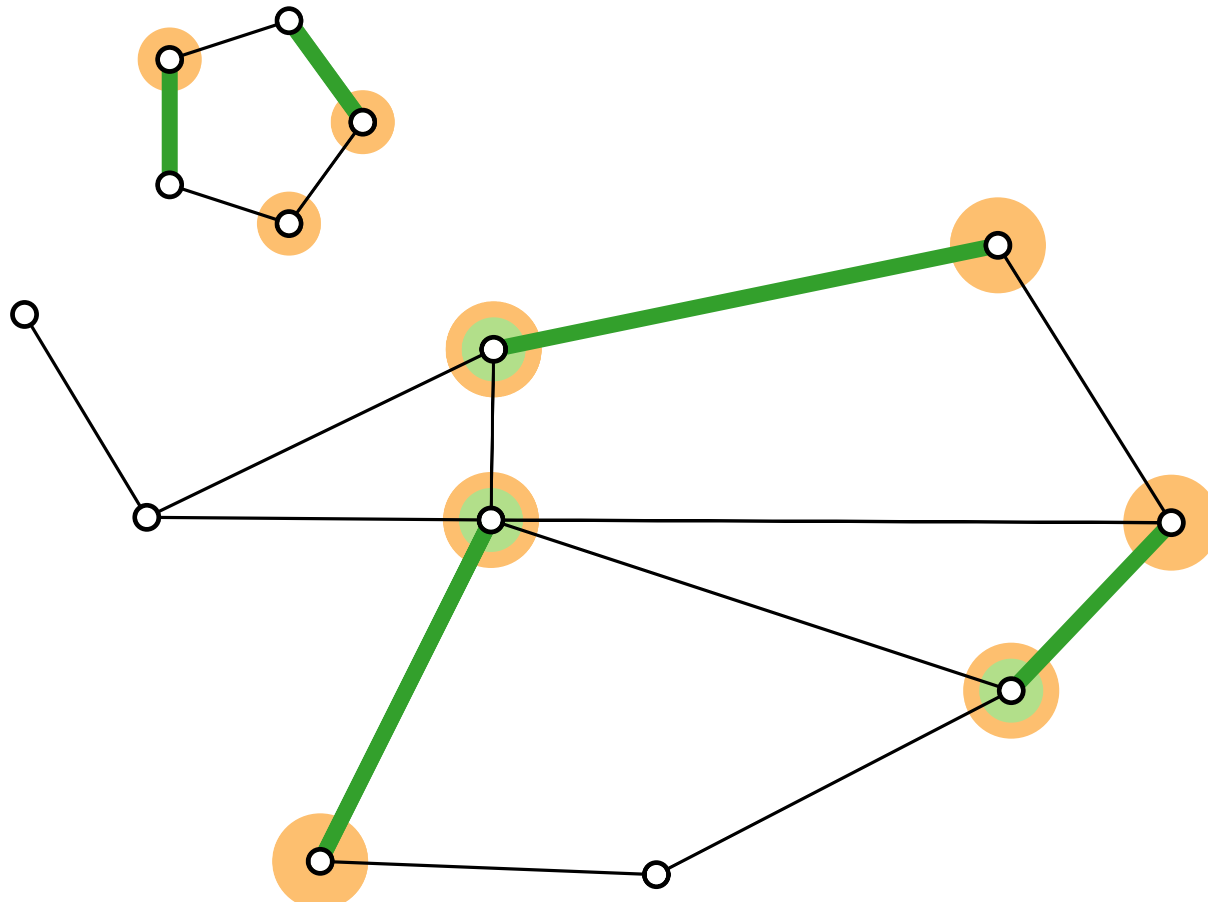
Vertex cover of E



Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$~~ ?

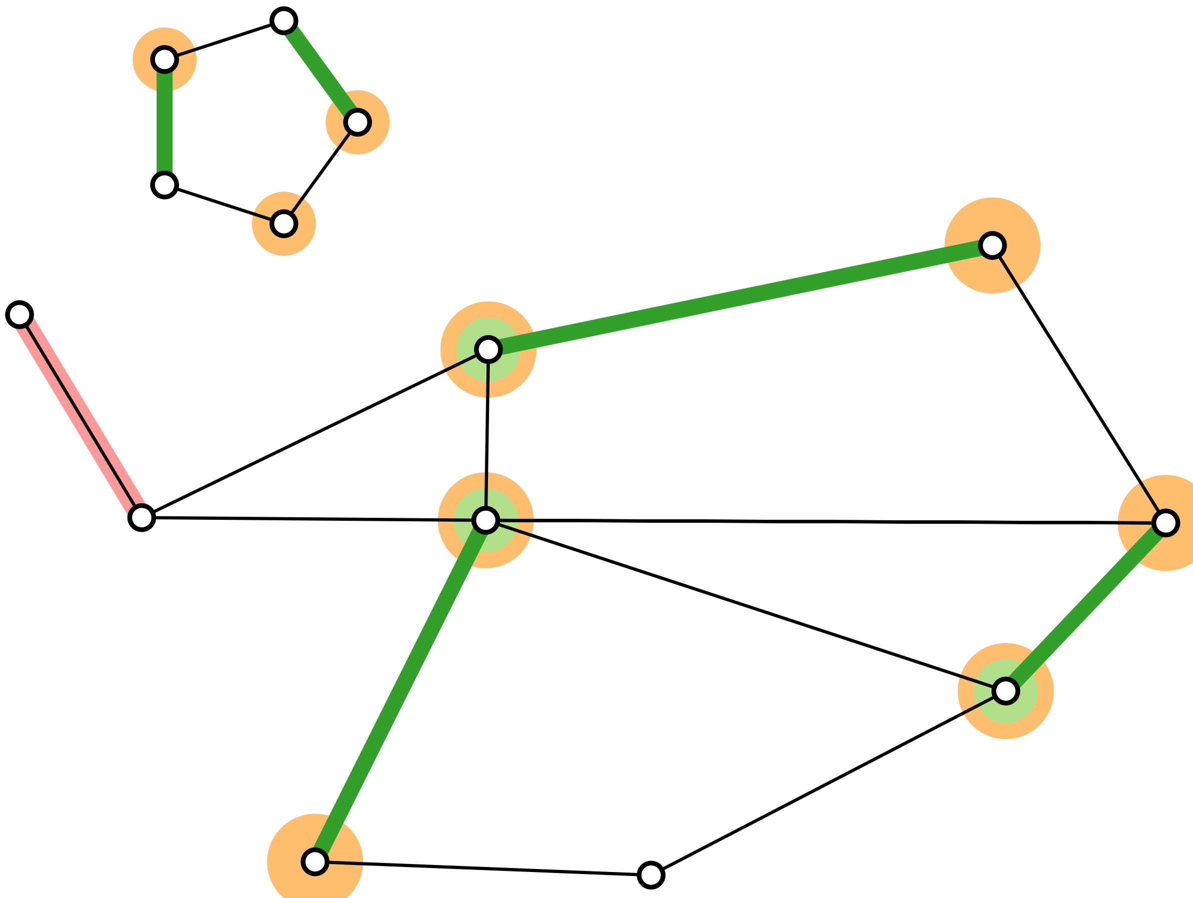
Vertex cover of M

Vertex cover of E

Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$~~ ?

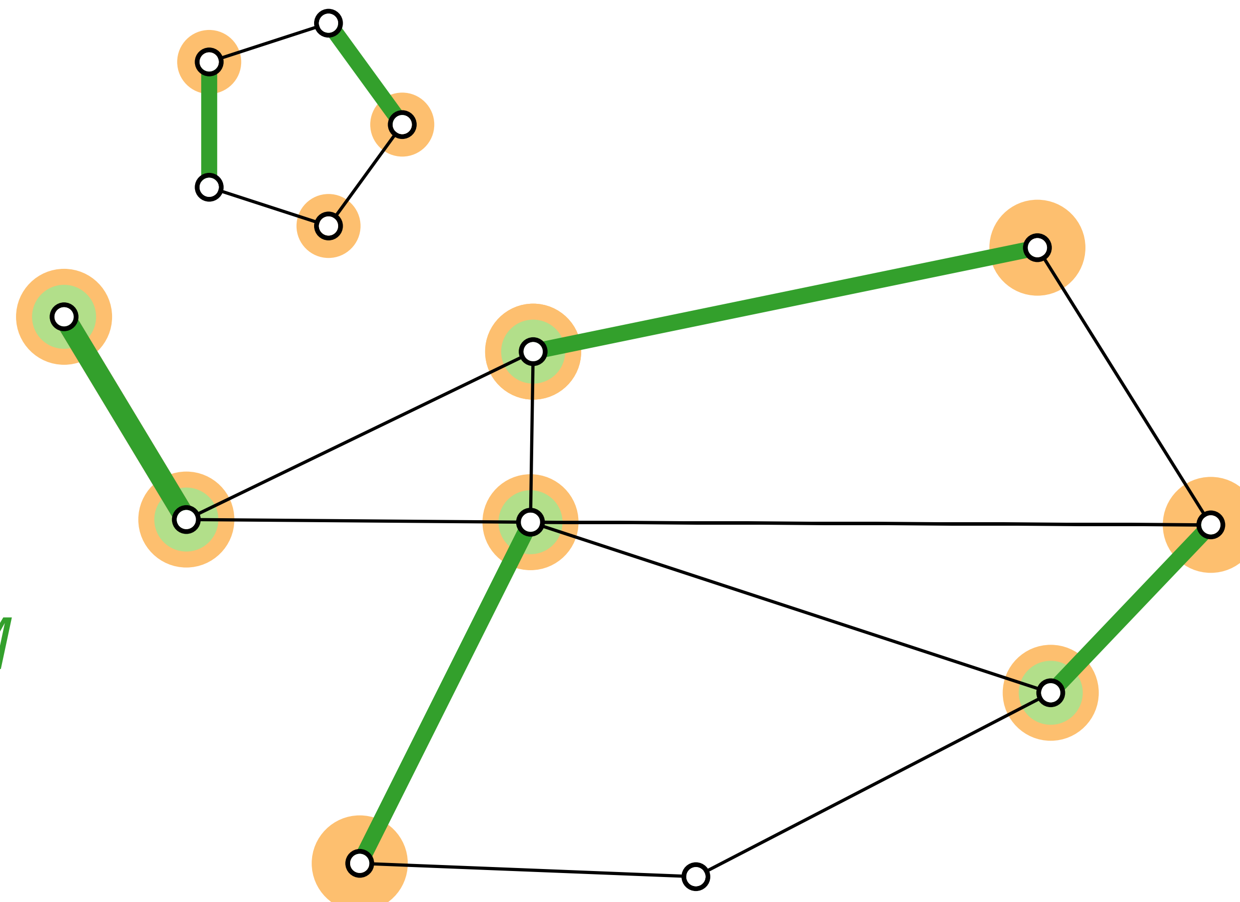
Vertex cover of M

Vertex cover of E

Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$?~~

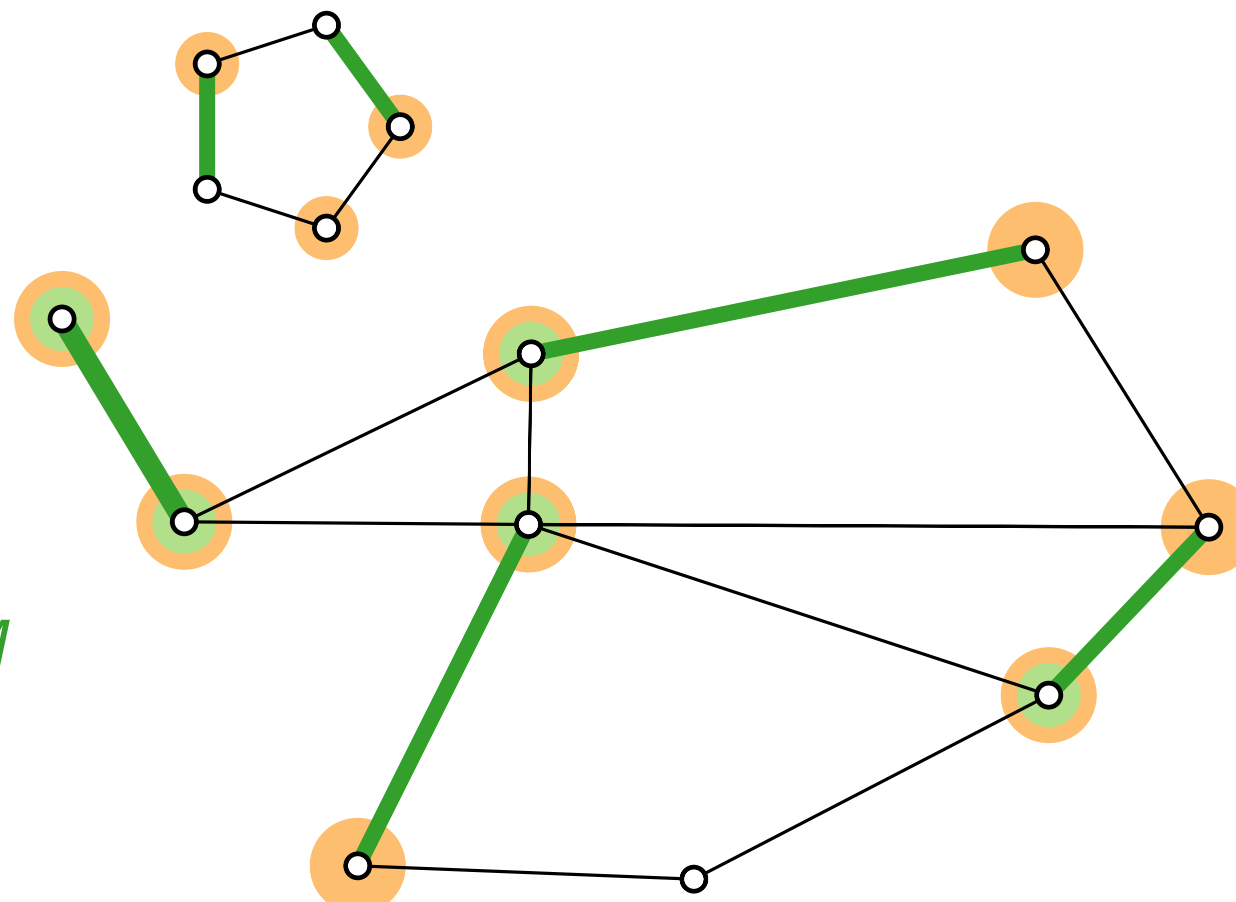
Vertex cover of M

Vertex cover of E

Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.



$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$~~ ?

Vertex cover of M

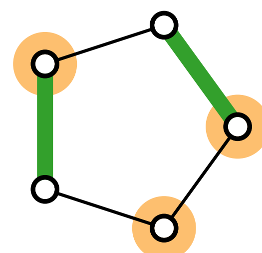
Vertex cover of E

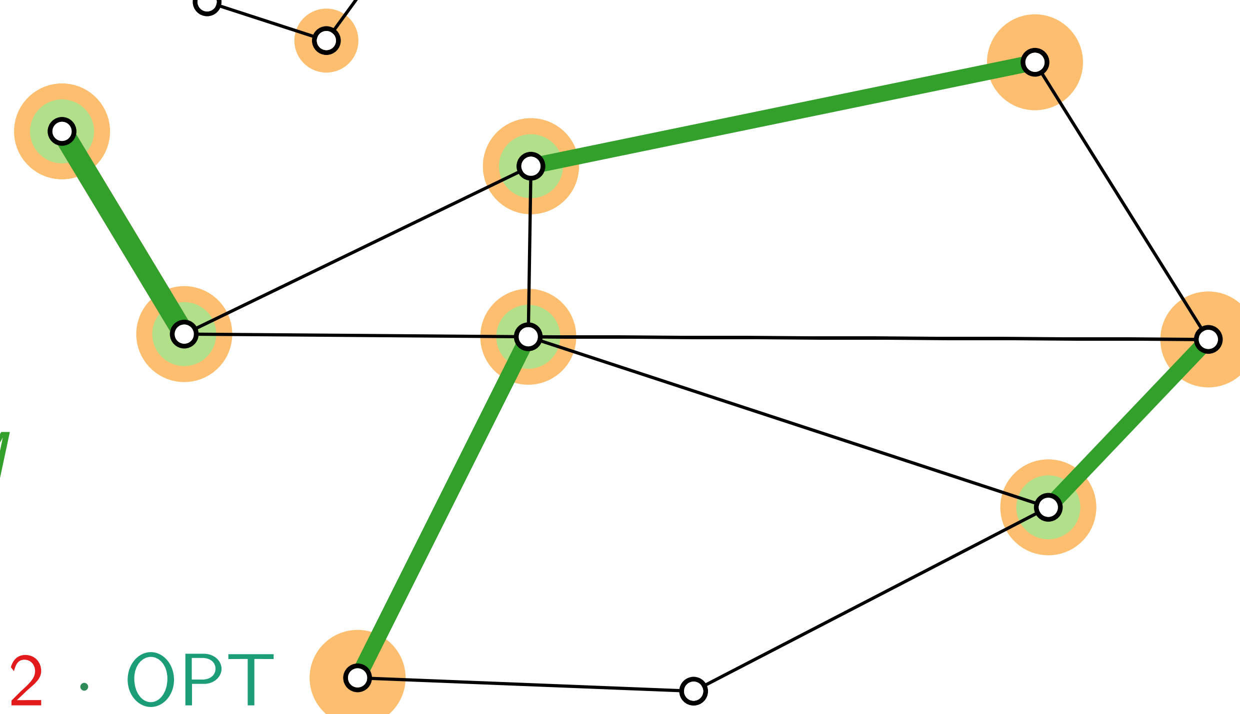
$\mathrm{ALG} = 2 \cdot |M| \leq$

Lower Bound by Matchings

Given a graph G, a set M of edges of G is a **matching** if no two edges of M are adjacent (i.e., share an end vertex).

M is **maximal** if there is no matching M' with $M' \supsetneq M$.





$\mathrm{OPT} \geq |M|$

~~$\mathrm{OPT} = |M|$~~ ?

Vertex cover of M

Vertex cover of E

$\mathrm{ALG} = 2 \cdot |M| \leq 2 \cdot \mathrm{OPT}$

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

  if e is not adjacent to any edge in M **then**

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

  if e is not adjacent to any edge in M **then**

    $M \leftarrow M \cup \{e\}$

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

  if e is not adjacent to any edge in M **then**

    $M \leftarrow M \cup \{e\}$

return $\{\, u, v \mid uv \in M \,\}$

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$

foreach $e \in E(G)$ **do**

  if e is not adjacent to any edge in M **then**

    $M \leftarrow M \cup \{e\}$

return $\{\, u, v \mid uv \in M \,\}$

Theorem. The above algorithm is a factor-2 approximation algorithm for VertexCover.

Approximation Alg. for VertexCover

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e is not adjacent to any edge in M **then**
    $M \leftarrow M \cup \{e\}$
return $\{\, u, v \mid uv \in M \,\}$

Theorem. The above algorithm is a factor-2 approximation algorithm for VertexCover.

Proof. $\text{ALG} = 2 \cdot |M| \leq$

Approximation Alg. for $\textsc{VertexCover}$

Algorithm VertexCover(G)

$M \leftarrow \emptyset$
foreach $e \in E(G)$ **do**
  if e is not adjacent to any edge in M **then**
    $M \leftarrow M \cup \{e\}$
return $\{\, u, v \mid uv \in M \,\}$

Theorem. The above algorithm is a factor-2 approximation algorithm for $\textsc{VertexCover}$.

Proof. $\text{ALG} = 2 \cdot |M| \leq 2 \cdot \text{OPT}$ □

Approximability of Vertex Cover

The best known approximation factor for VertexCover is

Approximability of Vertex Cover

The best known approximation factor for VertexCover is

$$2 - \Theta(1/\sqrt{\log n}).$$

Approximability of Vertex Cover

The best known approximation factor for VertexCover is

$$2 - \Theta(1/\sqrt{\log n}).$$

If P $\neq$ NP, VertexCover cannot be approximated within a factor of 1.3606.

Approximability of Vertex Cover

The best known approximation factor for VertexCover is

$$2 - \Theta(1/\sqrt{\log n}).$$

If P $\neq$ NP, VertexCover cannot be approximated within a factor of 1.3606.

VertexCover cannot be approximated within a factor of $2 - \Theta(1)$ – if the *Unique Games Conjecture* holds.

Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part V:
An LP-based Algorithm for VertexCover

Task

Write an integer linear program (ILP) for VertexCover:

Task

Write an integer linear program (ILP) for VERTEXCOVER:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

Task

Write an integer linear program (ILP) for VertexCover:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Task

Write an integer linear program (ILP) for VertexCover:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Variables:

Objective:

Constraints:

Task

Write an integer linear program (ILP) for VertexCover:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Variables: for each vertex v of G, we introduce

Objective:

Constraints:

Task

Write an integer linear program (ILP) for VertexCover:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Variables: for each vertex v of G, we introduce $x_v \in \{0, 1\}$.

Objective:

Constraints:

Task

Write an integer linear program (ILP) for VERTEXCOVER:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Variables: for each vertex v of G, we introduce $x_v \in \{0, 1\}$.

v not in the solution (→ 0)
v in the solution (→ 1)

Objective:

Constraints:

Task

Write an integer linear program (ILP) for VERTEXCOVER:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Variables: for each vertex v of G, we introduce $x_v \in \{0, 1\}$.

v not in the solution (→ 0)
v in the solution (→ 1)

Objective: minimize

Constraints:

Task

Write an integer linear program (ILP) for VertexCover:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Variables: for each vertex v of G, we introduce $x_v \in \{0, 1\}$.

v not in the solution (→ 0)
v in the solution (→ 1)

Objective: minimize $\sum_{v \in V(G)} x_v$

Constraints:

Task

Write an integer linear program (ILP) for VertexCover:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Variables: for each vertex v of G, we introduce $x_v \in \{0, 1\}$.

v not in the solution (→ 0)
v in the solution (→ 1)

Objective: minimize $\sum_{v \in V(G)} x_v$

Constraints: for each edge uv of G, we require that

Task

Write an integer linear program (ILP) for VertexCover:

Using integer (and/or real) variables, express the problem using

- linear constraints and
- a linear objective function.

You can iterate over the vertices / edges of the given graph G.

Variables: for each vertex v of G, we introduce $x_v \in \{0, 1\}$.

v not in the solution (→ 0)
v in the solution (→ 1)

Objective: minimize $\sum_{v \in V(G)} x_v$

Constraints: for each edge uv of G, we require that

$$x_u + x_v \geq 1.$$

Standard ILP Format

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \in \{0, 1\} & \text{for each } v \in V(G) \end{array}$$

Standard ILP Format

$$\begin{aligned} &\text{minimize} \quad \textstyle\sum_{v \in V(G)} x_v \\ &\text{subject to} \quad x_u + x_v \geq 1 \qquad \text{for each } uv \in E(G) \\ &\qquad\qquad\quad x_v \in \{0, 1\} \qquad \text{for each } v \in V(G) \end{aligned}$$

Problem:

Standard ILP Format

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \in \{0, 1\} & \text{for each } v \in V(G) \end{array}$$

Problem: It's NP-hard to solve ILPs in general.

Standard ILP Format

minimize	$\sum_{v \in V(G)} x_v$	
subject to	$x_u + x_v \geq 1$	for each $uv \in E(G)$
	$x_v \in \{0, 1\}$	for each $v \in V(G)$

Problem: It's NP-hard to solve ILPs in general.

But: LPs can be solved efficiently (in $O(L \cdot n^{3.5})$ time),

Standard ILP Format

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \in \{0, 1\} & \text{for each } v \in V(G) \end{array}$$

Problem: It's NP-hard to solve ILPs in general.

But: LPs can be solved efficiently (in $O(L \cdot n^{3.5})$ time),
where $n = \#$ variables and $L =$ total bit complexity of coefficients.

Standard ILP Format

LP relaxation

minimize $\sum_{v \in V(G)} x_v$

subject to $x_u + x_v \geq 1$ for each $uv \in E(G)$

$x_v \geq 0$ ~~$x_v \in \{0, 1\}$~~ for each $v \in V(G)$

Problem: It's NP-hard to solve ILPs in general.

But: LPs can be solved efficiently (in $O(L \cdot n^{3.5})$ time),

where $n = \#$ variables and $L =$ total bit complexity of coefficients.

Standard ILP Format

LP relaxation

minimize $\sum_{v \in V(G)} x_v$

subject to $x_u + x_v \geq 1$ for each $uv \in E(G)$

$x_v \geq 0$ ~~$x_v \in \{0, 1\}$~~ for each $v \in V(G)$

Problem: It's NP-hard to solve ILPs in general.

But: LPs can be solved efficiently (in $O(L \cdot n^{3.5})$ time),

where $n = \#$ variables and $L =$ total bit complexity of coefficients.

Problem′: Now we can get fractional solutions, i.e., in $(0, 1)$.

Standard ILP Format

LP relaxation

minimize	$\sum_{v \in V(G)} x_v$	
subject to	$x_u + x_v \geq 1$	for each $uv \in E(G)$
	$x_v \geq 0$ ~~$x_v \in \{0, 1\}$~~	for each $v \in V(G)$

Problem: It's NP-hard to solve ILPs in general.

But: LPs can be solved efficiently (in $O(L \cdot n^{3.5})$ time),
where $n = \#$ variables and $L =$ total bit complexity of coefficients.

Problem′: Now we can get fractional solutions, i.e., in $(0, 1)$.

Task: Find a graph G with $\mathrm{OPT}_{\mathrm{LP}} \neq \mathrm{OPT}_{\mathrm{ILP}}$!

Standard ILP Format

LP relaxation

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 \quad \sout{x_v \in \{0,1\}} & \text{for each } v \in V(G) \end{array}$$

Problem: It's NP-hard to solve ILPs in general.

But: LPs can be solved efficiently (in $O(L \cdot n^{3.5})$ time),

where $n = \#$ variables and $L =$ total bit complexity of coefficients.

Problem′: Now we can get fractional solutions, i.e., in $(0,1)$.

Task: Find a graph G with $\mathrm{OPT}_{\mathrm{LP}} \neq \mathrm{OPT}_{\mathrm{ILP}}$!

Solution?

Standard ILP Format

LP relaxation

minimize $\sum_{v \in V(G)} x_v$

subject to $x_u + x_v \geq 1$ for each $uv \in E(G)$

$x_v \geq 0$ ~~$x_v \in \{0,1\}$~~ for each $v \in V(G)$

Problem: It's NP-hard to solve ILPs in general.

But: LPs can be solved efficiently (in $O(L \cdot n^{3.5})$ time),

where $n = \#$ variables and $L =$ total bit complexity of coefficients.

Problem′: Now we can get fractional solutions, i.e., in $(0, 1)$.

Task: Find a graph G with $\mathrm{OPT}_{\mathrm{LP}} \neq \mathrm{OPT}_{\mathrm{ILP}}$!

Solution? Round the LP solution to get an integral solution!

Rounding the LP Solution

minimize	$\sum_{v \in V(G)} x_v$	
subject to	$x_u + x_v \geq 1$	for each $uv \in E(G)$
	$x_v \geq 0$	for each $v \in V(G)$

For each $v \in V(G)$:

Rounding the LP Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Rounding the LP Solution

$$
\begin{array}{lll}
\text{minimize} & \sum_{v \in V(G)} x_v & \\
\text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\
& x_v \geq 0 & \text{for each } v \in V(G)
\end{array}
$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check:

Rounding the LP Solution

$$\begin{aligned} &\text{minimize} && \textstyle\sum_{v \in V(G)} x_v && \\ &\text{subject to} && x_u + x_v \geq 1 && \text{for each } uv \in E(G) \\ & && x_v \geq 0 && \text{for each } v \in V(G) \end{aligned}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

Rounding the LP Solution

minimize	$\sum_{v \in V(G)} x_v$	
subject to	$x_u + x_v \geq 1$	for each $uv \in E(G)$
	$x_v \geq 0$	for each $v \in V(G)$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

In other words:

Rounding the LP Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

In other words: Is $\{v \in V(G) \colon x'_v = 1\}$ a vertex cover of G?

Rounding the LP Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

In other words: Is $\{v \in V(G) \colon x'_v = 1\}$ a vertex cover of G?

Need to make sure that every edge uv of G is covered.

Rounding the LP Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

In other words: Is $\{v \in V(G) \colon x'_v = 1\}$ a vertex cover of G?

Need to make sure that every edge uv of G is covered.

Is $x'_u = 0 = x'_v$ possible?

Rounding the LP Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

In other words: Is $\{v \in V(G) \colon x'_v = 1\}$ a vertex cover of G?

Need to make sure that every edge uv of G is covered.

Is $x'_u = 0 = x'_v$ possible? But then $x_u < 0.5$ and $x_v < 0.5$.

Rounding the LP Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

In other words: Is $\{v \in V(G) \colon x'_v = 1\}$ a vertex cover of G?

Need to make sure that every edge uv of G is covered.

Is $x'_u = 0 = x'_v$ possible? But then $x_u < 0.5$ and $x_v < 0.5$.

This contradicts $x_u + x_v \geq 1$.

Rounding the LP Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

In other words: Is $\{v \in V(G) \colon x'_v = 1\}$ a vertex cover of G?

Need to make sure that every edge uv of G is covered.

Is $x'_u = 0 = x'_v$ possible? But then $x_u < 0.5$ and $x_v < 0.5$.

This contradicts $x_u + x_v \geq 1. \Rightarrow x'_u = 1$ or $x'_v = 1$

Rounding the LP Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

Need to check: Is $(x'_v)_{v \in V(G)}$ a feasible solution?

In other words: Is $\{v \in V(G) \colon x'_v = 1\}$ a vertex cover of G?

Need to make sure that every edge uv of G is covered.

Is $x'_u = 0 = x'_v$ possible? But then $x_u < 0.5$ and $x_v < 0.5$.

This contradicts $x_u + x_v \geq 1$. $\Rightarrow$ $x'_u = 1$ or $x'_v = 1$ $\Rightarrow$ (x'_v) feasible!

Cost of the Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

ALG $=$

Cost of the Solution

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{v \in V(G)} x_v \\ \text{subject to} \quad & x_u + x_v \geq 1 && \text{for each } uv \in E(G) \\ & x_v \geq 0 && \text{for each } v \in V(G) \end{aligned}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$\text{ALG} = \sum_{v \in V(G)} x'_v \leq$

Cost of the Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$$\text{ALG} = \sum_{v \in V(G)} x'_v \leq \quad \sum_{v \in V(G)} x_v$$

Cost of the Solution

$$\begin{aligned} &\text{minimize} \quad \textstyle\sum_{v \in V(G)} x_v \\ &\text{subject to} \quad x_u + x_v \geq 1 \qquad \text{for each } uv \in E(G) \\ &\qquad\qquad\quad x_v \geq 0 \qquad \text{for each } v \in V(G) \end{aligned}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$\text{ALG} = \sum_{v \in V(G)} x'_v \leq \quad \sum_{v \in V(G)} x_v$

Cost of the Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$$\text{ALG} = \sum_{v \in V(G)} x'_v \leq 2 \cdot \sum_{v \in V(G)} x_v$$

Cost of the Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$$\text{ALG} = \sum_{v \in V(G)} x'_v \leq 2 \cdot \sum_{v \in V(G)} x_v = 2 \cdot \text{OPT}_{\text{LP}}$$

Cost of the Solution

minimize	$\sum_{v \in V(G)} x_v$	
subject to	$x_u + x_v \geq 1$	for each $uv \in E(G)$
	$x_v \geq 0$	for each $v \in V(G)$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$$\text{ALG} = \sum_{v \in V(G)} x'_v \leq 2 \cdot \sum_{v \in V(G)} x_v = 2 \cdot \text{OPT}_{\text{LP}} \leq 2 \cdot \text{OPT}_{\text{ILP}}$$

Cost of the Solution

minimize	$\sum_{v \in V(G)} x_v$	
subject to	$x_u + x_v \geq 1$	for each $uv \in E(G)$
	$x_v \geq 0$	for each $v \in V(G)$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$$\text{ALG} = \sum_{v \in V(G)} x'_v \leq 2 \cdot \sum_{v \in V(G)} x_v = 2 \cdot \text{OPT}_{\text{LP}} \leq 2 \cdot \text{OPT}_{\text{ILP}}$$

Theorem. The LP rounding algorithm is a factor-2 approximation algorithm for VertexCover.

Cost of the Solution

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V(G)} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for each } uv \in E(G) \\ & x_v \geq 0 & \text{for each } v \in V(G) \end{array}$$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$$\mathrm{ALG} = \sum_{v \in V(G)} x'_v \leq 2 \cdot \sum_{v \in V(G)} x_v = 2 \cdot \mathrm{OPT}_{\mathrm{LP}} \leq 2 \cdot \mathrm{OPT}_{\mathrm{ILP}}$$

Theorem. The LP rounding algorithm is a factor-2 approximation algorithm for WeightedVertexCover.

Cost of the Solution

minimize	$\sum_{v \in V(G)} x_v \cdot w(v)$	
subject to	$x_u + x_v \geq 1$	for each $uv \in E(G)$
	$x_v \geq 0$	for each $v \in V(G)$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$$\text{ALG} = \sum_{v \in V(G)} x'_v \leq 2 \cdot \sum_{v \in V(G)} x_v = 2 \cdot \text{OPT}_{\text{LP}} \leq 2 \cdot \text{OPT}_{\text{ILP}}$$

Theorem. The LP rounding algorithm is a factor-2 approximation algorithm for WeightedVertexCover.

Cost of the Solution

minimize	$\sum_{v \in V(G)} x_v \cdot w(v)$	
subject to	$x_u + x_v \geq 1$	for each $uv \in E(G)$
	$x_v \geq 0$	for each $v \in V(G)$

For each $v \in V(G)$: Set $x'_v = \begin{cases} 1 & \text{if } x_v \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$

$$\text{ALG} = \sum_{v \in V(G)} x'_v \cdot w(v) \leq 2 \cdot \sum_{v \in V(G)} x_v \cdot w(v) = 2 \cdot \text{OPT}_{\text{LP}} \leq 2 \cdot \text{OPT}_{\text{ILP}}$$

Theorem. The LP rounding algorithm is a factor-2 approximation algorithm for WeightedVertexCover.