Aalto University
School of Science

Department of Computer Science

Combinatorics of Efficient Computations

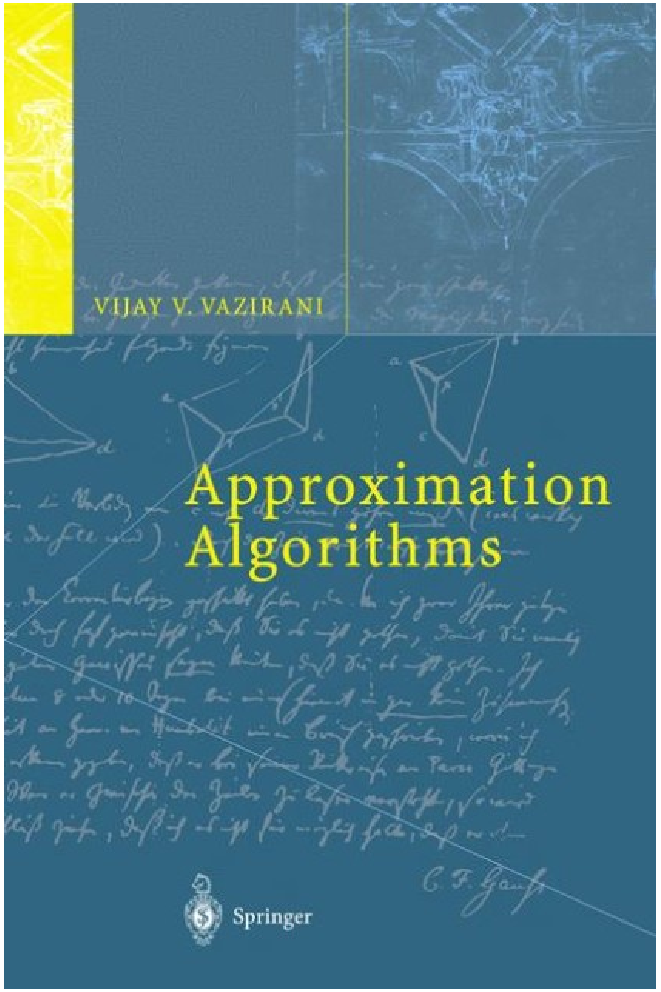
Approximation Algorithms

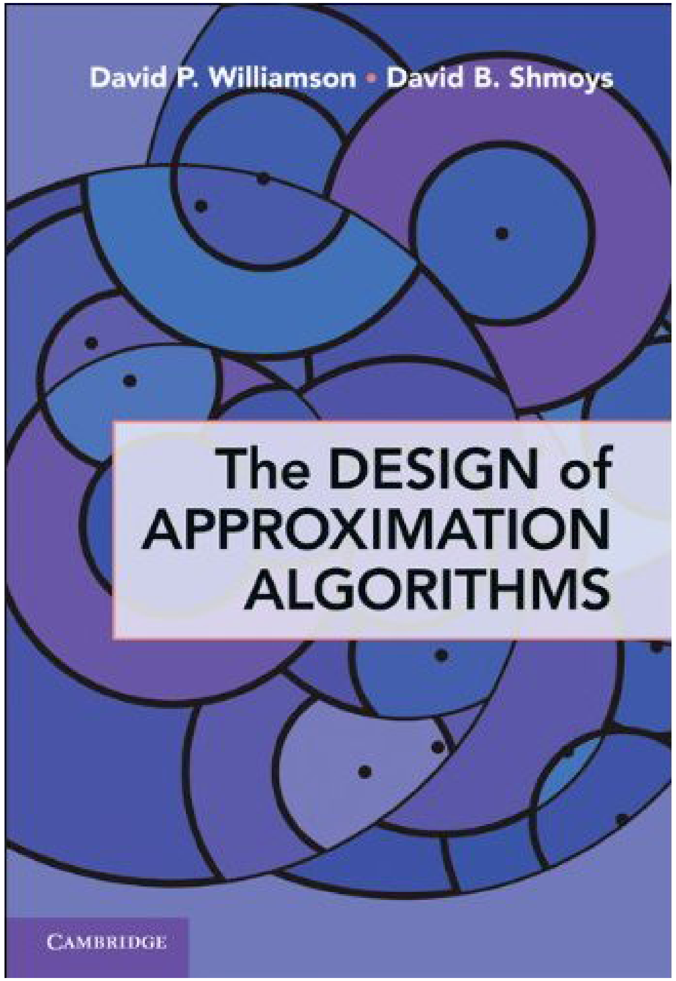
Lecture 1: Introduction & Vertex Cover

Joachim Spoerhase

2019

Textbooks





http://www.designofapproxalgs.com/

Vijay V. Vazirani
Approximation Algorithms
Springer-Verlag 2003

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

Approximation Algorithms

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

– Bertrand Russell

Overview of Possible Topics

Combinatorial Algorithms

- Introduction
- Set Cover
- Steiner Tree and TSP
- Multiway Cut
- k-Center
- Shortest Superstring
- Knapsack
- Bin Packing
- Minimum Makespan Scheduling
- Euclidean TSP

LP-Based Algorithms

- Introduction to LP-Duality
- Set Cover via Dual Fitting
- Rounding Applied to Set Cover
- Set Cover via the Primal–Dual Schema
- Maximum Satisfiability
- Facility Location
- $\ldots$

Approximation Algorithms

- Many optimization problems are NP-hard (e.g. the travelling salesman problem)
- $\rightsquigarrow$ 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 (currently) mostly from studying specific optimization problems.

Vertex Cover (cardinality)

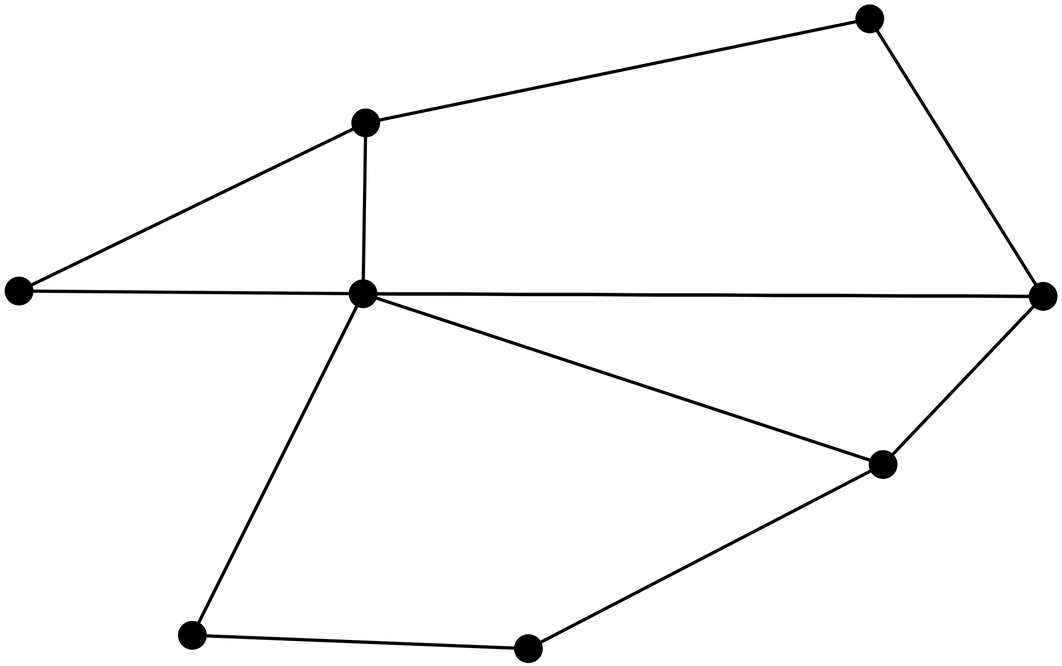
Input Graph $G = (V, E)$

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

VERTEX COVER (cardinality)

Input Graph $G = (V, E)$

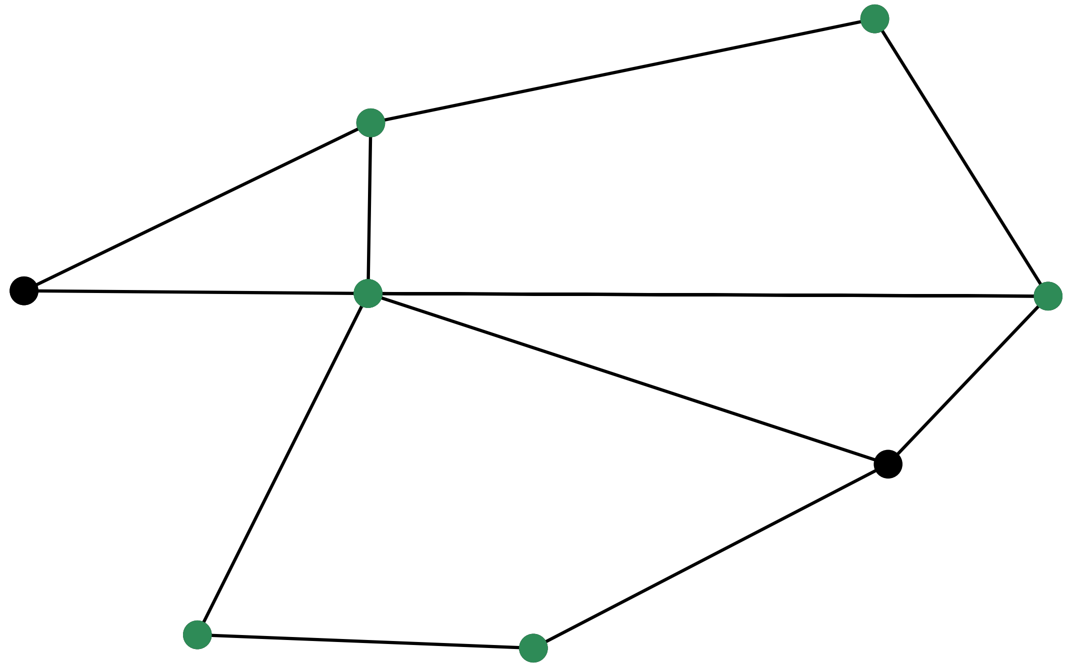
Output a minimum **vertex cover**: a minimum vertex set $V' \subseteq V$, such that every edge is **covered** by V' (i.e., for every $uv \in E$, either $u \in V'$ or $v \in V'$).



Vertex Cover (cardinality)

Input Graph $G = (V, E)$

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

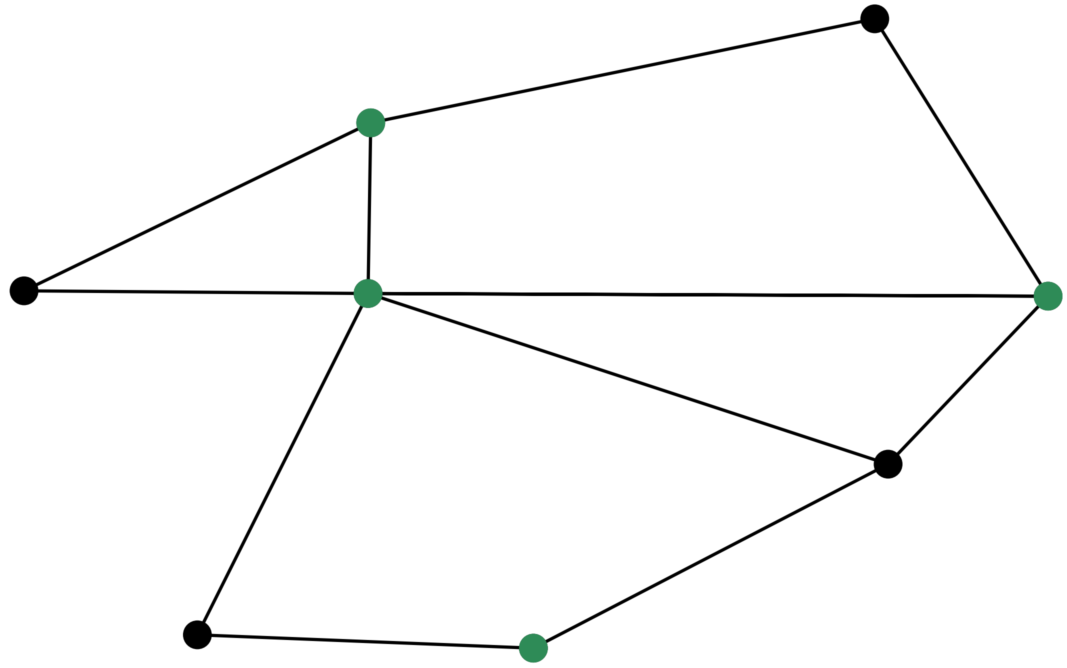


a vertex cover

Vertex Cover (cardinality)

Input Graph $G = (V, E)$

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

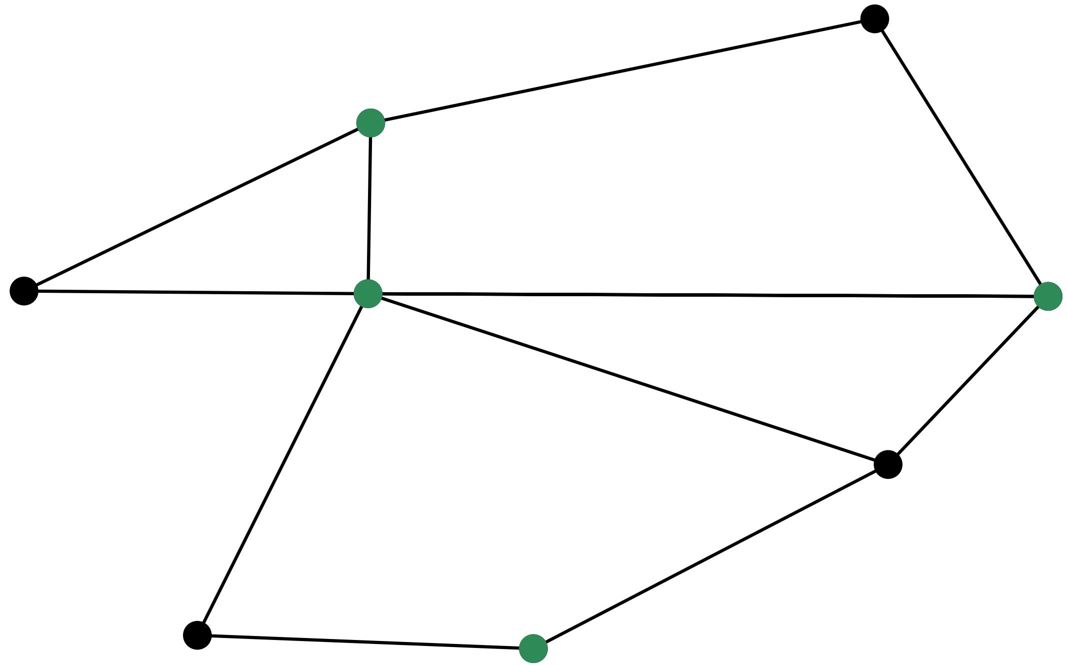


Optimum (OPT $= 4$)

Vertex Cover (cardinality)

Input Graph $G = (V, E)$

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



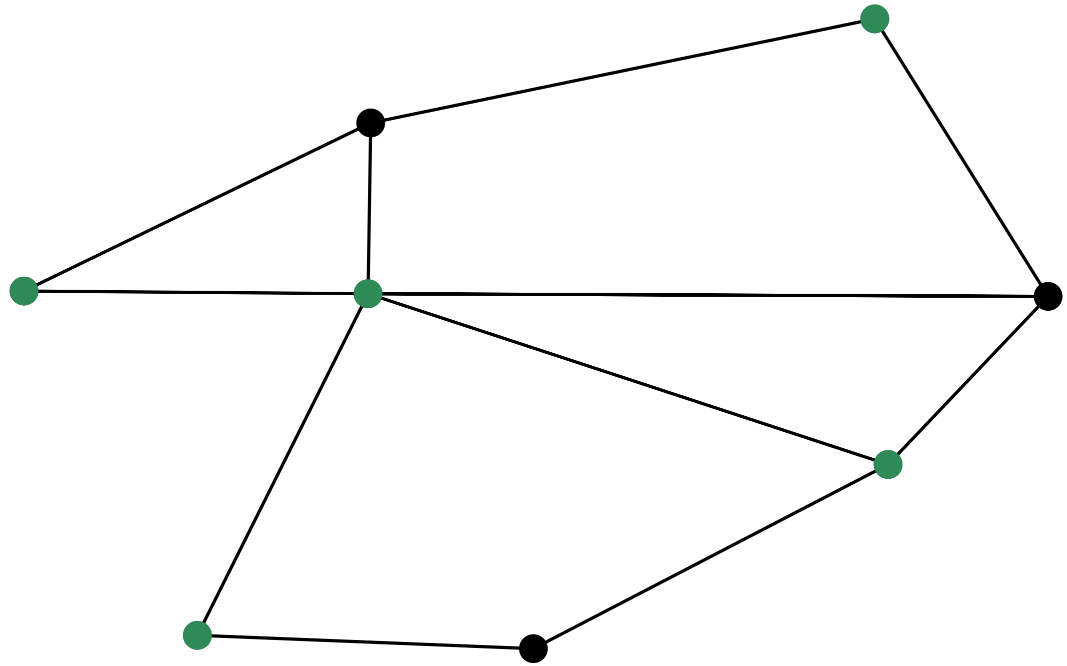
Optimum (OPT $= 4$)

NP-hard to find :-(

Vertex Cover (cardinality)

Input Graph $G = (V, E)$

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



"good" approximate solution (5/4-Approximation)

NP-Optimization Problem

An NP-optimization problem Π is given by:

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

NP-Optimization Problem

An NP-optimization problem Π is given by:

- A set D_Π of **instances**.
  We use $|I|$ to denote the size of an instance $I \in D_\Pi$.
- For each instance $I \in D_\Pi$ there is a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I where:
  - For each solution $s \in S_\Pi(I)$, its size $|s|$ is polynomially bounded in $|I|$.
  - There is a polynomial time algorithm to decide 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 use $|I|$ to denote the size of an instance $I \in D_\Pi$.
- For each instance $I \in D_\Pi$ there is a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I where:
  - For each solution $s \in S_\Pi(I)$, its size $|s|$ is polynomially bounded in $|I|$.
  - There is a polynomial time algorithm to decide 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 $\mathrm{obj}_\Pi(I, s)$ to a given pair (I, s).

NP-Optimization Problem

An NP-optimization problem Π is given by:

- A set D_Π of **instances**.
  We use $|I|$ to denote the size of an instance $I \in D_\Pi$.
- For each instance $I \in D_\Pi$ there is a set $S_\Pi(I) \neq \emptyset$ of **feasible solutions** for I where:
  - For each solution $s \in S_\Pi(I)$, its size $|s|$ is polynomially bounded in $|I|$.
  - There is a polynomial time algorithm to decide 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 $\text{obj}_\Pi(I, s)$ to a given pair (I, s).
- Π is either a minimization or maximization problem.

Vertex Cover NP-Optimization Problem

Exercise

Vertex Cover NP-Optimization Problem

Exercise

What are the *instances*?

What are the *feasible solutions*?

What is the *objective function*?

Optimum, and optimal objective value.

Let Π be a minimization (maximization) problem and $I \in D_\Pi$ be an instance of Π. A feasible solution $s^* \in S_\Pi(I)$ is **Optimal** when $\text{obj}_\Pi(I, s^*)$ is the minimum (maximum) among objective values attained by the feasible solutions of I.

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

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 a feasible solution $s \in S_\Pi(I)$ for **any** instance $I \in D_\Pi$ such that:

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

Approximation Algorithms

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

Let Π be a minimization problem and ~~$\alpha \in \mathbb{Q}^+$~~.
A factor-α-approximation algorithm for Π is an efficient algorithm that provides a feasible solution $s \in S_\Pi(I)$ for **any** instance $I \in D_\Pi$ such that:

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

Approximation Algorithms

Let Π be a ~~minimization~~ maximization problem and $\cancel{\alpha \in \mathbb{Q}^+}$ $\alpha \colon \mathbb{N} \to \mathbb{Q}$.
A factor-α-approximation algorithm for Π is an efficient algorithm that provides a feasible solution $s \in S_\Pi(I)$ for **any** instance $I \in D_\Pi$ such that:

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

Approximation Alg. for VERTEX COVER

Ideas?

Approximation Alg. for Vertex Cover

Ideas?

- Edge-Greedy
- Vertex-Greedy (see Exercises)
- Inclusion-wise minimal vertex cover

Approximation Alg. for Vertex Cover

Ideas?

- Edge-Greedy
- Vertex-Greedy (see Exercises)
- Inclusion-wise minimal vertex cover

How can we measure the quality of a feasible solution?

Approximation Alg. for Vertex Cover

Ideas?

- Edge-Greedy
- Vertex-Greedy (see Exercises)
- Inclusion-wise minimal vertex cover

How can we measure the quality of a feasible solution?

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

Approximation Alg. for Vertex Cover

Ideas?

- Edge-Greedy
- Vertex-Greedy (see Exercises)
- Inclusion-wise minimal vertex cover

How can we measure the quality of a feasible solution?

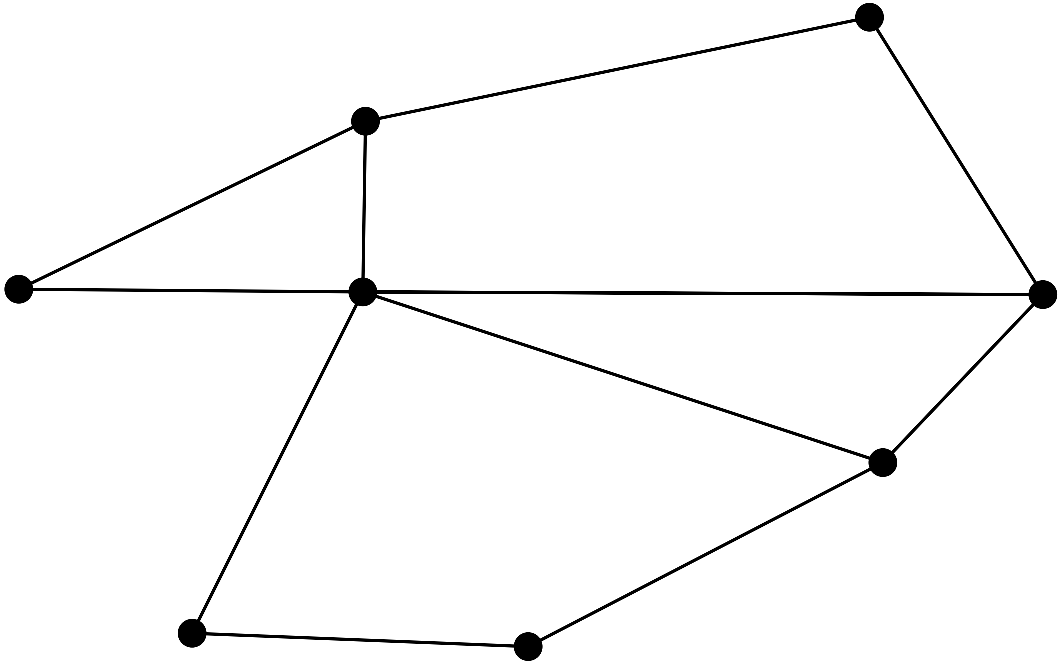
Problem: How can we estimate $\frac{\mathrm{obj}_\Pi(I,s)}{\mathrm{OPT}}$ when it is hard to calculate OPT?

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

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

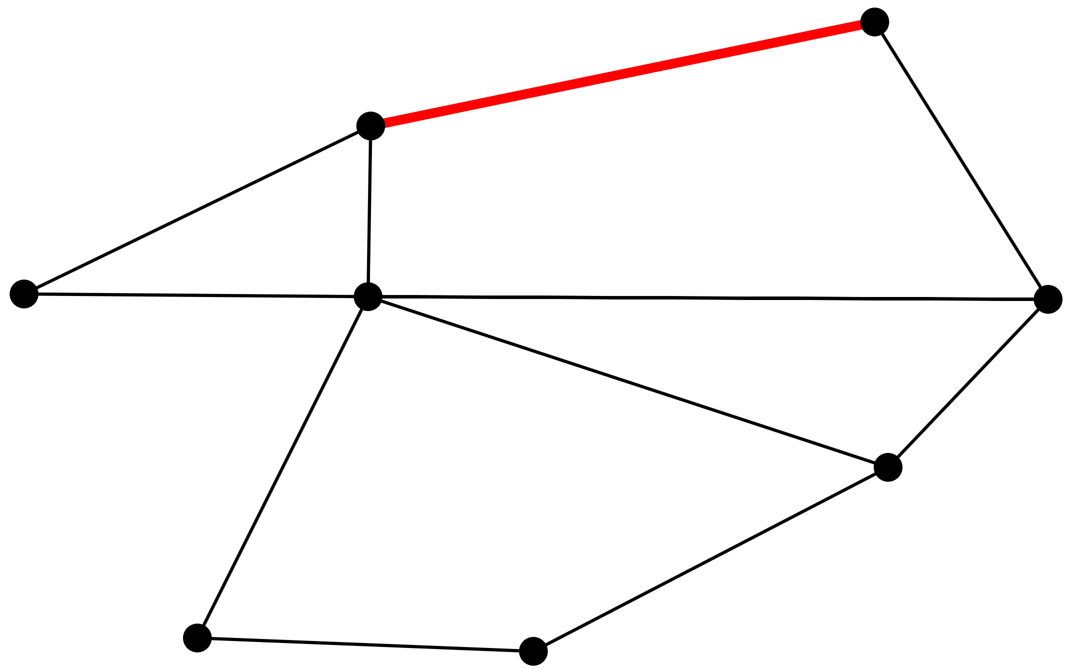
Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .



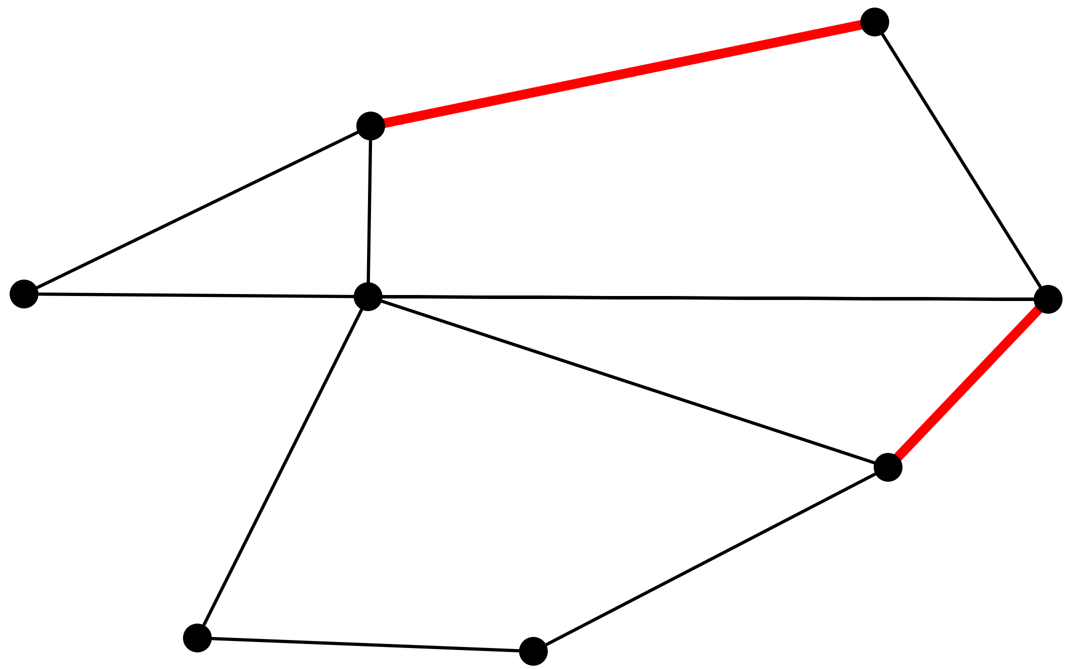
Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .



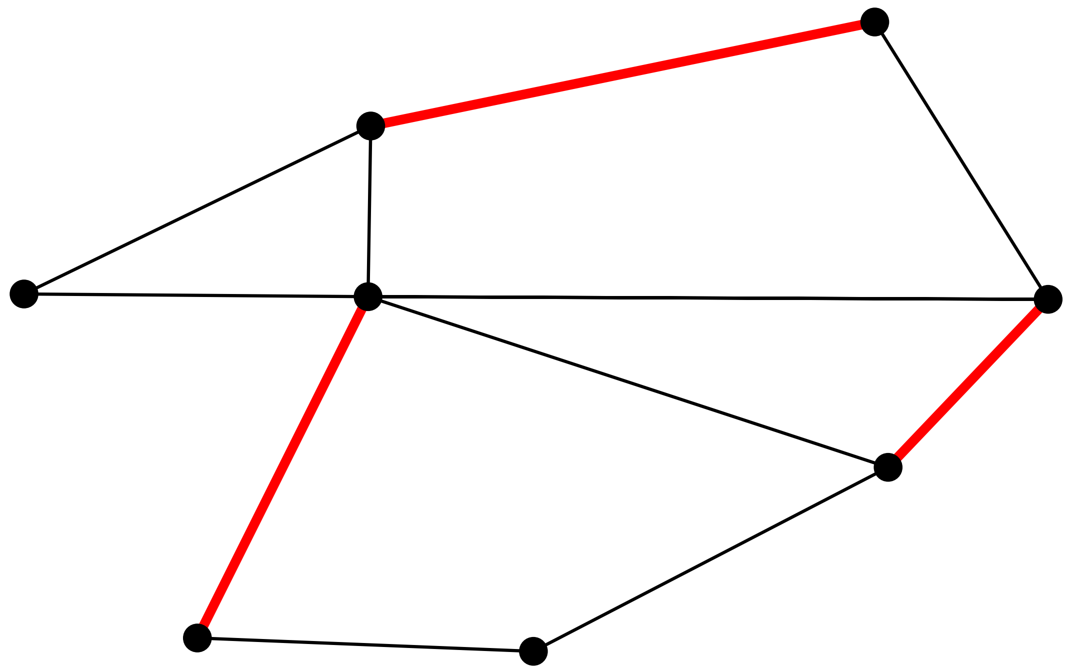
Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .



Lower Bound by Matchings

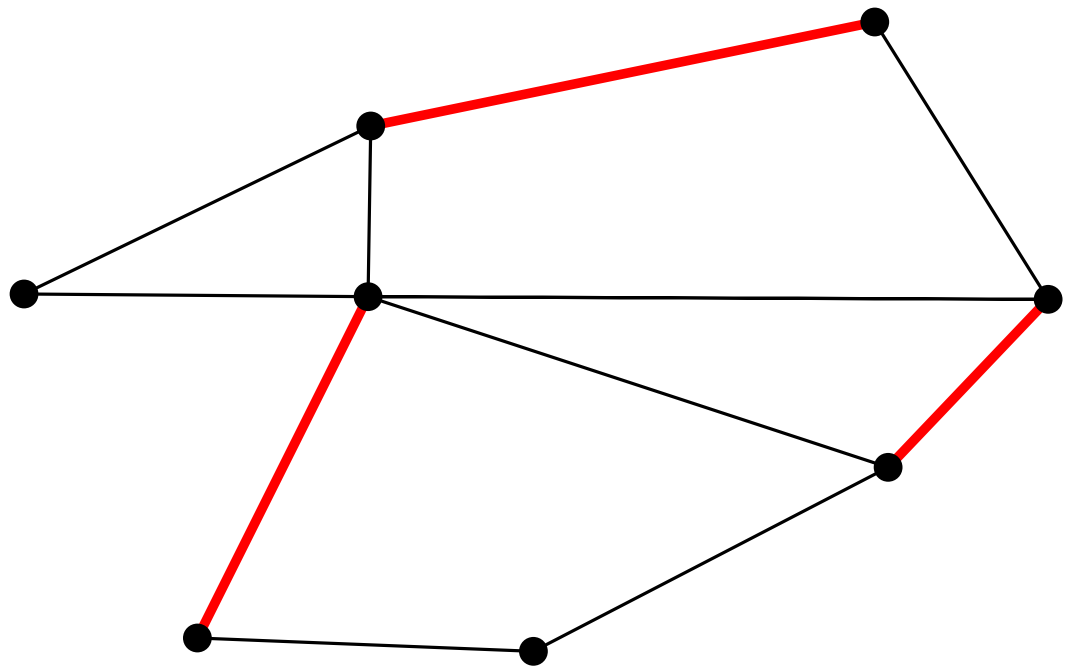
An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .

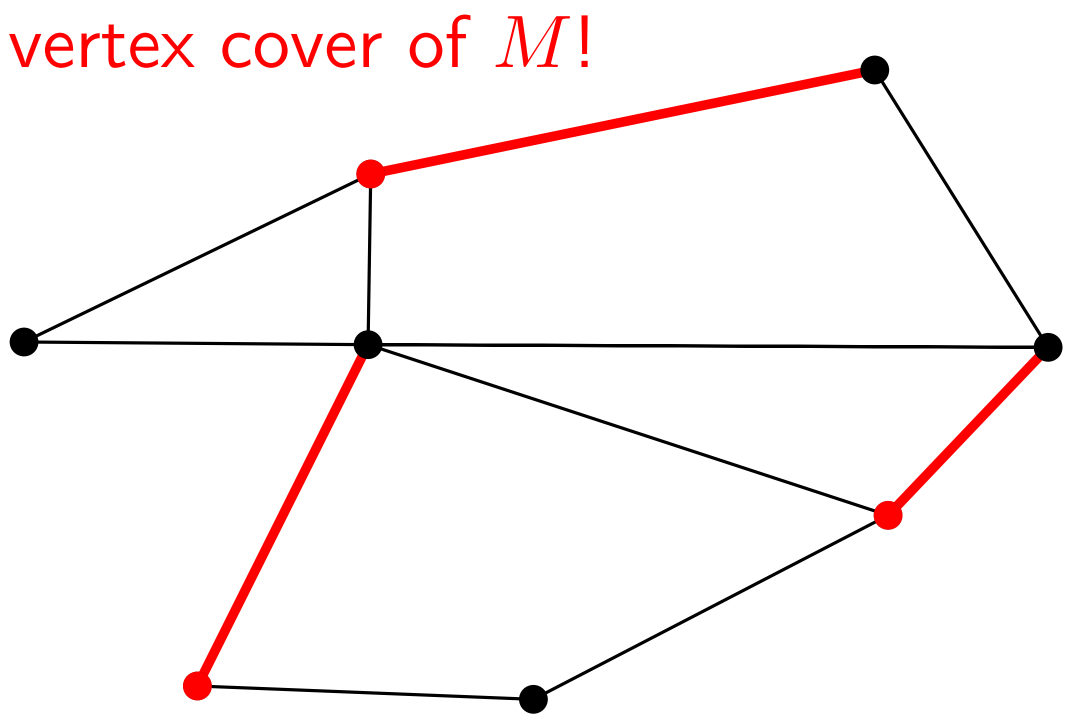
M is **maximal** when there is no matching M' with $M' \supsetneq M$.



Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .

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

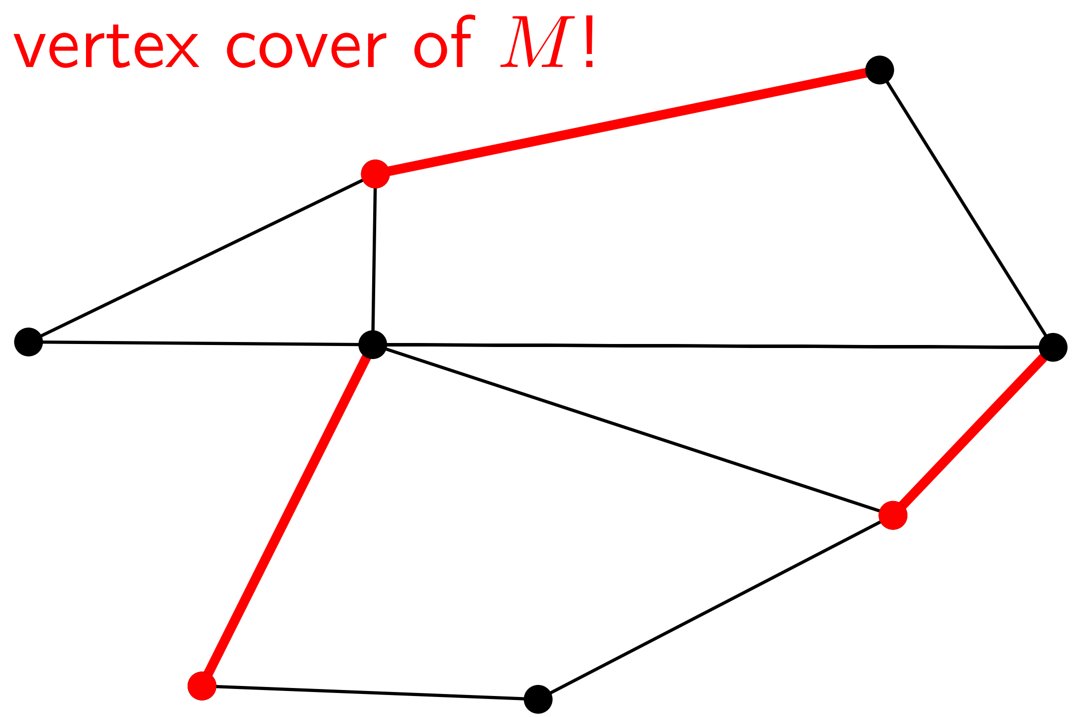


Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .

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

OPT ≥ 3



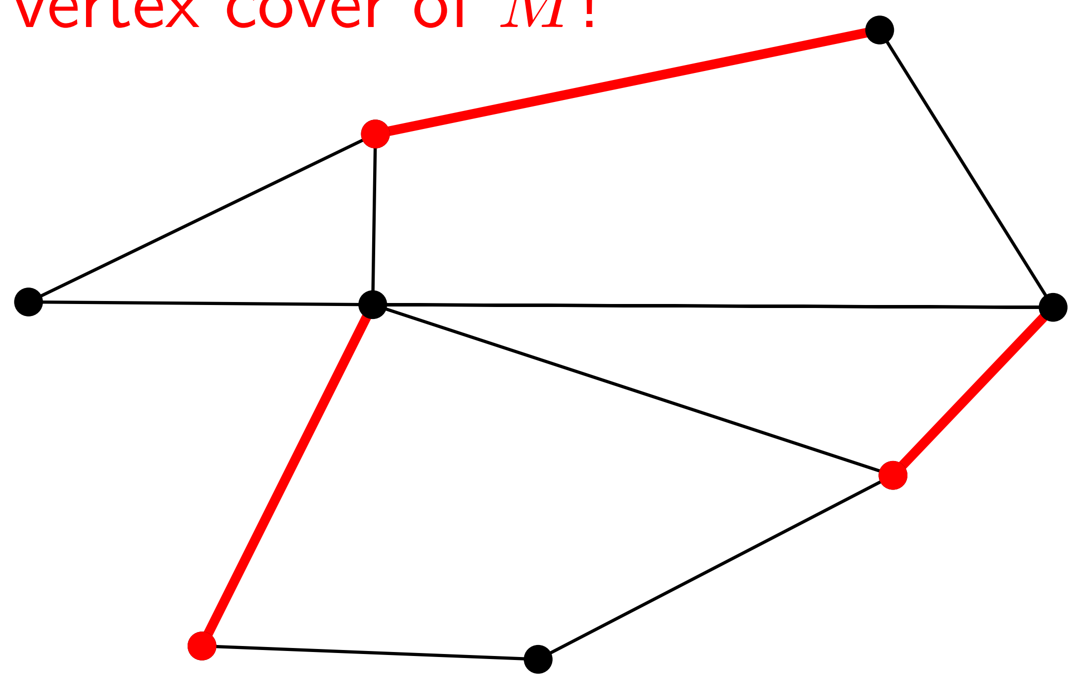
Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .

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

vertex cover of M!

OPT $\geq \overset{|M|}{\cancel{3}}$



Lower Bound by Matchings

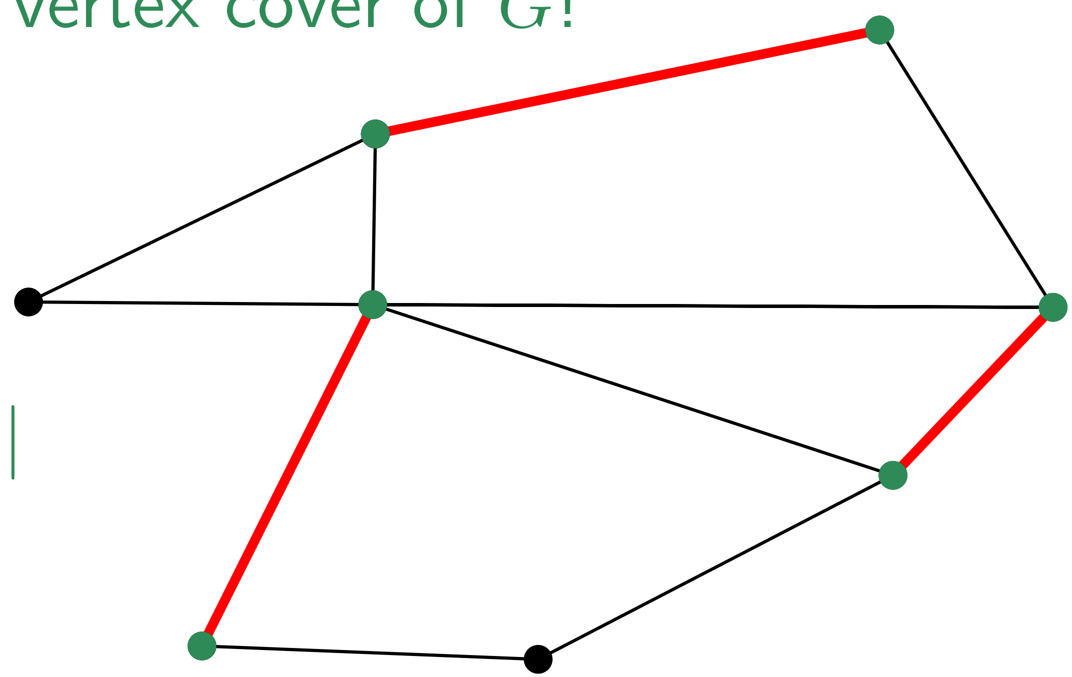
An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a **matching** when no vertex of G is incident to two edges in M .

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

vertex cover of G!

$\text{OPT} \geq \overset{|M|}{\cancel{3}}$

$\text{OPT} \leq 2 \cdot |M|$



Approximation Alg. for Vertex Cover

Algorithm for Vertex Cover (G)

$M \leftarrow \emptyset$

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

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

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

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

Approximation Alg. for Vertex Cover

Algorithm for Vertex Cover (G)

$M \leftarrow \emptyset$

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

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

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

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

Thm 1.1 The above algortihm is a factor-2 approximation algorithm for Vertex Cover

Approximation Alg. for Vertex Cover

Algorithm for Vertex Cover (G)

$M \leftarrow \emptyset$

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

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

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

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

Thm 1.1 The above algortihm is a factor-2 approximation algorithm for Vertex Cover

Next week: Set Cover