GEMS OF TCS

APPROXIMATION ALGORITHMS

Sasha Golovnev
August 28, 2023
APPROXIMATION ALGORITHMS

- Optimal exact solution $\text{OPT}$ (ex: shortest TSP cycle)
Approximation Algorithms

- Optimal exact solution \textbf{OPT} (ex: shortest TSP cycle)
- \textbf{OPT} is too hard to find (ex: \textbf{NP}-hard)
Approximation Algorithms

- Optimal exact solution $\text{OPT}$ (ex: shortest TSP cycle)
- $\text{OPT}$ is too hard to find (ex: $\text{NP}$-hard)
- A $k$-approximation algorithm finds a solution $\leq k \times \text{OPT}$
APPROXIMATION ALGORITHMS

- Optimal exact solution $\text{OPT}$ (ex: shortest TSP cycle)
- $\text{OPT}$ is too hard to find (ex: $\text{NP}$-hard)
- A $k$-approximation algorithm finds a solution $\leq k \times \text{OPT}$
- Possibly efficiently! (ex: poly time)
Approximation Algorithms

- Optimal exact solution OPT (ex: shortest TSP cycle)
- OPT is too hard to find (ex: NP-hard)
- A $k$-approximation algorithm finds a solution $\leq k \times \text{OPT}$
- Possibly efficiently! (ex: poly time)
- When do we use approximation algorithms?
Matchings

- A Matching in a graph is a set of edges without common vertices
MATCHINGS

- A **Matching** in a graph is a set of edges without common vertices

- A **Maximal Matching** is a matching which cannot be extended to a larger matching
**Matchings**

- A *Matching* in a graph is a set of edges without common vertices.

- A *Maximal Matching* is a matching which cannot be extended to a larger matching.

- A *Maximum Matching* is a matching of the largest size.
Matchings. Examples
Matchings. Examples
MATCHINGS. EXAMPLES
## Job Assignment

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Ben</th>
<th>Chris</th>
<th>Diana</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Programmer</td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Librarian</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Professor</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>
JOB ASSIGNMENT

adm

prog

libr

prof

A

B

C

D
JOB ASSIGNMENT

adm - A
prog - B
libr - C
prof - D
JOB ASSIGNMENT

adm - A
prog - B
libr - C
prof - D
## Room Assignment

<table>
<thead>
<tr>
<th></th>
<th>R# 1</th>
<th>R# 2</th>
<th>R# 3</th>
<th>R# 4</th>
<th>R# 5</th>
<th>R# 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aaron</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bianca</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Dana</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Emma</td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Francis</td>
<td></td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ROOM ASSIGNMENT

A 1
B 2
C 3
D 4
E 5
F 6
## Algorithms

<table>
<thead>
<tr>
<th>Maximal Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can be found in polynomial time by a greedy algorithm</td>
</tr>
<tr>
<td><strong>Algorithms</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td><strong>Maximal Matching</strong></td>
</tr>
<tr>
<td>Can be found in polynomial time by a greedy algorithm</td>
</tr>
<tr>
<td><strong>Maximum Matching</strong></td>
</tr>
<tr>
<td>Can be found in polynomial time by the blossom algorithm</td>
</tr>
<tr>
<td>Algorithm</td>
</tr>
<tr>
<td>-----------------------------------------</td>
</tr>
<tr>
<td><strong>Maximal Matching</strong></td>
</tr>
<tr>
<td><strong>Maximum Matching</strong></td>
</tr>
<tr>
<td><strong>Minimum Weight Perfect Matching</strong></td>
</tr>
</tbody>
</table>
ROOM ASSIGNMENT

A -> 1
B -> 2
C -> 3
D -> 4
E -> 5
F -> 6
Vertex Cover
Vertex Covers

- A **Vertex Cover** of a graph $G$ is a set of vertices $C$ such that every edge of $G$ is connected to some vertex in $C$. 
**Vertex Covers**

- A **Vertex Cover** of a graph $G$ is a set of vertices $C$ such that every edge of $G$ is connected to some vertex in $C$.

- A **Minimal Vertex Cover** is a vertex cover which does not contain other vertex covers.
**Vertex Covers**

- A *Vertex Cover* of a graph $G$ is a set of vertices $C$ such that every edge of $G$ is connected to some vertex in $C$.

- A *Minimal Vertex Cover* is a vertex cover which does not contain other vertex covers.

- A *Minimum Vertex Cover* is a vertex cover of the smallest size.
VERTEX COVERS: EXAMPLES
VERTEX COVERS: EXAMPLES
VERTEX COVERS: EXAMPLES
VERTEX COVERS: EXAMPLES
VERTEX COVERS: EXAMPLES
ANTIVIRUS SYSTEM
## Algorithms

<table>
<thead>
<tr>
<th>Minimal Vertex Cover</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can be found in polynomial time by a greedy algorithm</td>
</tr>
</tbody>
</table>
**Algorithms**

<table>
<thead>
<tr>
<th>Minimal Vertex Cover</th>
<th>Can be found in polynomial time by a greedy algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Vertex Cover</td>
<td>Is \textbf{NP}-hard. We only know exponential-time algorithms</td>
</tr>
</tbody>
</table>
APPROXIMATION ALGORITHM

- $M \leftarrow$ maximal matching in $G$
APPROXIMATION ALGORITHM

• $M \leftarrow$ maximal matching in $G$

• return all vertices in $M$
• $C \leftarrow \emptyset$
EQUIVALENT ALGORITHM

• $C \leftarrow \emptyset$

• while $E \neq \emptyset$
Equivalent Algorithm

• $C \leftarrow \emptyset$

• while $E \neq \emptyset$
  • $\{u, v\} \leftarrow$ any edge from $E$
Equivalent Algorithm

- $C \leftarrow \emptyset$

- while $E \neq \emptyset$
  - $\{u, v\} \leftarrow$ any edge from $E$
  - add $u, v$ to $C$
Equivalent Algorithm

- $C \leftarrow \emptyset$

- while $E \neq \emptyset$
  - $\{u, v\} \leftarrow$ any edge from $E$
  - add $u, v$ to $C$
  - delete from $E$ all edges incident to $u$ or $v$
- return $C$
**Proof**

**Lemma**

This algorithm runs in polynomial time and is 2-approximate: it returns a vertex cover that is at most twice larger then a minimum vertex cover.
Final Remarks

- The analysis is tight: there are graphs with matchings twice larger than vertex covers
Final Remarks

• The analysis is tight: there are graphs with matchings twice larger than vertex covers

• No 1.99-approximation algorithm is known
Break

Matchings:

Vertex covers:
Traveling Salesman
Approximation

- If $P \neq NP$, then there is no $k$-approximation algorithm for the general version of TSP for any constant $k$.
**Approximation**

- If $\mathbf{P} \neq \mathbf{NP}$, then there is no $k$-approximation algorithm for the general version of TSP for any constant $k$

- **Euclidean TSP**: $w(u, v) = w(v, u)$ and $w(u, v) \leq w(u, z) + w(z, v)$
APPROXIMATION

• If $P \neq NP$, then there is no $k$-approximation algorithm for the general version of TSP for any constant $k$

• **Euclidean TSP**: $w(u, v) = w(v, u)$ and $w(u, v) \leq w(u, z) + w(z, v)$

• We will design a 2-approximation algorithm: it quickly finds a cycle that is at most twice longer than an optimal one
DEFINITION

• A tree is a connected graph without cycles
Definition

• A tree is a connected graph without cycles
• A tree is a connected graph on $n$ vertices with $n - 1$ edges
**Definition**

- A **tree** is a connected graph without cycles.
- A **tree** is a connected graph on $n$ vertices with $n - 1$ edges.
- A **Spanning Tree** of a graph $G$ is a subgraph of $G$ that (i) is a tree and (ii) contains all vertices of $G$. 
**Definition**

- A **tree** is a connected graph without cycles
- A **tree** is a connected graph on $n$ vertices with $n - 1$ edges
- A **Spanning Tree** of a graph $G$ is a subgraph of $G$ that (i) is a tree and (ii) contains all vertices of $G$
- A **Minimum Spanning Tree** of a weighted graph $G$ is a spanning tree of the smallest weight
Minimum Spanning Tree: Examples

A — B (3) — G (1) — F (3) — E (4) — D (5)
A — B (3) — C (3) — G (2) — D (5)
A — B (3) — C (3) — E (7) — G (2) — D (5)
Minimum Spanning Tree: Examples
**Lemma**

Let $G$ be an undirected graph with non-negative edge weights. Then $\text{MST}(G) \leq \text{TSP}(G)$. 

**Minimum Spanning Trees**
# Minimum Spanning Trees

## Lemma

Let $G$ be an undirected graph with non-negative edge weights. Then $\text{MST}(G) \leq \text{TSP}(G)$.

## Proof

By removing any edge from an optimum TSP cycle one gets a spanning tree of $G$. 
An **Eulerian cycle (or path)** visits every edge exactly once
Eulerian Cycle

An Eulerian cycle (or path) visits every edge exactly once

Criteria

A connected undirected graph contains an Eulerian cycle, if and only if the degree of every node is even
EXAMPLE

Non-Eulerian graph
EXAMPLE

Non-Eulerian graph

Eulerian graph
ALGORITHM

• $T \leftarrow$ minimum spanning tree of $G$
Algorithm

- $T \leftarrow$ minimum spanning tree of $G$
- $D \leftarrow T$ with each edge doubled
**Algorithm**

- \( T \leftarrow \text{minimum spanning tree of } G \)
- \( D \leftarrow T \) with each edge doubled
- find an Eulerian cycle \( C \) in \( D \)
ALGORITHM

- $T \leftarrow$ minimum spanning tree of $G$
- $D \leftarrow T$ with each edge doubled
- find an Eulerian cycle $C$ in $D$
- return a cycle that visits the nodes in the order of their first appearance in $C$
### Approximation Guarantee

<table>
<thead>
<tr>
<th>Lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>The algorithm is 2-approximate.</td>
</tr>
</tbody>
</table>
**Approximation Guarantee**

<table>
<thead>
<tr>
<th>Lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>The algorithm is 2-approximate.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The total length of the MST $T \leq \text{OPT}$</td>
</tr>
</tbody>
</table>
## Approximation Guarantee

**Lemma**

The algorithm is 2-approximate.

**Proof**

- The total length of the MST $T \leq \text{OPT}$
- We start with Eulerian cycle of length $2|T|$
# Approximation Guarantee

## Lemma

The algorithm is 2-approximate.

## Proof

- The total length of the MST $T \leq \text{OPT}$
- We start with Eulerian cycle of length $2|T|$
- Shortcuts can only decrease the total length