Approximation Algorithms

Lecture 1: Introduction and Vertex Cover

Part I: Organizational
Organizational

Lectures: on site (in German :-)
Fri, 10:15–11:45 (ÜR I)
possibly some lectures via inverted classroom

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

Questions/Tasks during the lecture

Most slides are due to Joachim Spoerhase. Thanks!
Textbooks

Vijay V. Vazirani: Approximation Algorithms

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.)
- 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

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:** 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** ($\text{OPT} = 4$) – but in general NP-hard to find :-(
Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part III:
NP-Optimization Problem

Alexander Wolff
Winter 2022/23
NP-Optimization Problem

An **NP-optimization problem** $\Pi$ is given by:

- A set $D_\Pi$ 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** $\text{obj}_\Pi$ which assigns a positive objective value $\text{obj}_\Pi(I, s) \geq 0$ to any given pair $(s, I)$ with $s \in S_\Pi(I)$.

- $\Pi$ is either a minimization or maximization problem.
**VertexCover**: NP-Optimization Problem

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

$D_\Pi = \text{Set of all graphs}$

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

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

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

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

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

$\Pi$ is a minimization problem.
Optimum and Optimal Objective Value

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

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

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

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

$$\frac{\text{obj}_{\Pi}(I, s)}{\text{OPT}_{\Pi}(I)} \leq \alpha(|I|)$$
Approximation Algorithms

Lecture 1:
Introduction and Vertex Cover

Part IV:
Approximation Algorithm for Vertex Cover

Alexander Wolff
Winter 2022/23
Approximation Alg. for VertexCover

Ideas?
- Edge-Greedy
- Vertex-Greedy

Quality?

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

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

\[ \frac{\text{obj}_\Pi(l, s)}{\text{OPT}} \leq \frac{\text{obj}_\Pi(l, s)}{L} \]
Lower Bound by Matchings

An edge set $M \subseteq E$ of a graph $G = (V, E)$ 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 $M' \supseteq M$.

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

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 $M' \supseteq M$.

\[
\text{OPT} \geq |M|
\]

\[
\text{OPT} = |M|\quad ?
\]

**Vertex cover of $M$**

**Vertex cover of $E$**

\[
\text{ALG} = 2 \cdot |M| \leq 2 \cdot \text{OPT}
\]
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

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

end if

end foreach

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

Theorem. The above algorithm is a factor-2 approximation algorithm for VertexCover.
Approximability of Vertex Cover

The best known approximation factor for Vertex Cover is $2 - \Theta(1/\sqrt{\log n})$.

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

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