

CS-E4530

# Computational Complexity Theory

**Janne H. Korhonen**

Aalto University

Spring 2018

Lecture 12:

# Approximation

# Solving Hard Problems: Approximation

- **There are intractable problems that we don't know how to solve in polynomial time**
  - How to deal with such problems in practice?
- **Today's concept: approximation**
  - **Basic idea:** instead of looking for the best solution, look for a fairly good solution
  - When does this help?

# Optimisation Problems

# Optimisation Problems

## Definition

An **optimisation problem**  $\Pi$  is defined by

- a set of **valid instances**  $I \subseteq \{0, 1\}^*$ ,
- a set of **feasible solutions**  $F(x) \subseteq \{0, 1\}^*$  for all valid instances  $x \in I$ ,
- an integer **cost**  $c(s)$  for all feasible solutions  $s \in F(x)$ , and
- a **goal function**, which is either min or max.

• **Task is to compute optimal solution:**

- **Maximisation:**  $\text{OPT}(x) = \max_{s \in F(x)} c(x)$
- **Minimisation:**  $\text{OPT}(x) = \min_{s \in F(x)} c(x)$

# NP Optimisation Problems

## Definition

An optimisation problem  $\Pi$  is an **NP optimisation problem (NPO)** if it holds that

- there is a constant  $c > 0$  such that for all  $x \in I$  and  $s \in F(x)$ , we have  $|s| = O(|x|^c)$ ,
- languages  $\{x : x \in I\}$  and  $\{(x, s) : s \in F(x)\}$  are decidable in polynomial time, and
- the function  $s \mapsto c(s)$  is computable in polynomial time.

- **We restrict our attention to NP optimisation problems**

# Covering Problems

## Example (Vertex Cover)

- **Instance:** Graph  $G = (V, E)$ .
- **Feasible solution:** A vertex cover  $C \subseteq V$ .
- **Objective:** minimise  $c(C) = |C|$

## Example (Set Cover)

- **Instance:** A finite set  $U$  and a family  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  of subsets of  $U$ .
- **Feasible solution:** A subfamily  $\mathcal{T} \subseteq \mathcal{S}$  such that any element  $u \in U$  is contained in at least one set  $T \in \mathcal{T}$
- **Objective:** minimise  $c(\mathcal{T}) = |\mathcal{T}|$

# Travelling Salesman Problem

## Example (TSP)

- **Instance:** An undirected/directed weighted graph  $G = (V, E, w)$ .
- **Feasible solution:** A tour  $T = (v_1, v_2, \dots, v_n)$  visiting all vertices once.
- **Objective:** minimise  $c(T) = \sum_{i=1}^n w(v_i, v_{i+1 \bmod n})$ .



# Approximation Algorithms

# Approximation Algorithms

## Definition (Approximation algorithm)

Let  $\Pi$  be an optimisation problem. For  $\alpha > 1$ , we say that  $M$  is a **polynomial-time  $\alpha$ -approximation algorithm** if  $M$  runs in polynomial time,

- for all  $x \in I$ , we have  $M(x) \in F(x)$ , and
- for all  $x \in I$ , we have  $(1/\alpha) \text{OPT}(x) \leq c(M(x)) \leq \alpha \text{OPT}(x)$ .

- **Maximisation:**  $(1/\alpha) \text{OPT}(x) \leq c(M(x)) \leq \text{OPT}(x)$
- **Minimisation:**  $\text{OPT}(x) \leq c(M(x)) \leq \alpha \text{OPT}(x)$
- **Notations may vary between sources:**
  - e.g.  $1/\alpha$ -approximation instead

## 2-approximation for Vertex Cover

### Approximation algorithm for vertex cover

**Input:** graph  $G = (V, E)$ , **Output:** vertex cover  $C$

- Start from  $C = \emptyset$
  - Select arbitrary edge  $\{u, v\} \in E$
  - Add  $u$  and  $v$  to  $C$ , remove  $u, v$  and all incident edges from the graph
  - Repeat until no edges remain
- 
- **This is a 2-approximation algorithm:**
    - Let  $C'$  be an optimal vertex cover
    - For any edge  $\{u, v\}$  selected by the algorithm, at least one of  $u$  and  $v$  must be in  $C'$
    - Thus,  $c(C) \leq 2c(C') = 2 \text{OPT}(G)$

# $O(\log n)$ -approximation for Set Cover

## Approximation algorithm for set cover

**Input:** set family  $\mathcal{S}$  over  $U$ , **Output:** set cover  $\mathcal{T}$

- Start from  $\mathcal{T} = \emptyset$
  - Find the set  $S \in \mathcal{S}$  that covers most uncovered elements in  $U$
  - Add  $S$  to  $\mathcal{T}$
  - Repeat until all elements of  $U$  are covered
- 
- **Let  $u_1, u_2, \dots, u_n$  be the order in which the algorithm covered the elements of  $U$** 
    - Assume that  $u_i$  was covered by a set  $T \in \mathcal{S}$  picked by the algorithm, and that  $T$  covered  $a$  uncovered elements
    - We define the cost of  $u_k$  as  $c(u_k) = 1/a$

# $O(\log n)$ -approximation for Set Cover

## Lemma

We have  $c(u_k) \leq \text{OPT} / (n - k + 1)$ .

- **Proof:** Consider the iteration when  $u_k$  was covered
  - Let  $C$  denote the covered elements at the beginning of the iteration
  - Since remaining elements could be covered with  $\text{OPT}$  sets, there is a set that covers at least  $\frac{|U \setminus C|}{\text{OPT}}$  elements
  - Since  $u_k$  was covered in this iteration, we had  $|U \setminus C| \geq n - k + 1$
  - Since we picked the set that covered most elements, we have

$$c(u_k) \leq \frac{\text{OPT}}{|U \setminus C|} \leq \frac{\text{OPT}}{n - k + 1}$$

# $O(\log n)$ -approximation for Set Cover

- We have  $|\mathcal{T}| = \sum_{k=1}^n c(u_k)$
- By previous lemma, we have

$$\sum_{k=1}^n c(u_k) \leq \sum_{k=1}^1 \frac{\text{OPT}}{n - k + 1} = \text{OPT} \left( 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \right)$$

- **Fact:**  $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \leq \log n + 1$ , where  $\log$  is the natural logarithm
- Thus,  $|\mathcal{T}| \leq (\log n + 1) \text{OPT}$

# **PTAS and FPTAS**

# Polynomial-time Approximation Schemes

## Definition (PTAS)

Let  $\Pi$  be an optimisation problem. We say that  $M$  is a **polynomial-time approximation scheme (PTAS)** if for all  $\varepsilon > 0$ ,

- for all  $x \in I$ , we have  $M(x, \varepsilon) \in F(x)$ , and
- for all  $x \in I$ , we have
$$(1 - \varepsilon) \text{OPT}(x) \leq c(M(x, \varepsilon)) \leq (1 + \varepsilon) \text{OPT}(x),$$

and  $M(x, \varepsilon)$  runs in time  $p_\varepsilon(|x|)$  time, where  $p_\varepsilon$  is a polynomial that only depends on  $\varepsilon$ .

- **PTAS allows trading time for accuracy**
- **Dependence on  $\varepsilon$  can be very large**
- **Example:** Euclidean TSP has a PTAS



# Fully Polynomial-time Approximation

## Definition (FPTAS)

Let  $\Pi$  be an optimisation problem. We say that  $M$  is a **fully polynomial-time approximation scheme (FPTAS)** if for all  $\varepsilon > 0$ ,

- for all  $x \in I$ , we have  $M(x, \varepsilon) \in F(x)$ , and
- for all  $x \in I$ , we have
$$(1 - \varepsilon) \text{OPT}(x) \leq c(M(x, \varepsilon)) \leq (1 + \varepsilon) \text{OPT}(x),$$
and  $M(x, \varepsilon)$  runs in time  $p(|x|, 1/\varepsilon)$  time, where  $p$  is a polynomial.

- **Example:** Knapsack has an FPTAS

# Hardness of Approximation

# Hardness of Approximation

- **Which NP-complete problems are easy to approximate?**
  - Do **all** NP-complete problems have a PTAS or FPTAS?
  - FPTAS is almost as good as polynomial-time algorithm
- **Can prove ad-hoc inapproximability results**
- **Can also define complexity classes related to approximation and prove hardness**
  - **FPTAS:** problems with FPTAS
  - **PTAS:** problems with PTAS
  - **AXP:** problems with constant-factor approximation
- **However, proving inapproximability results is very difficult with the tools we have seen so far**

## Example: TSP

### Theorem

*TSP cannot be  $\alpha$ -approximated for any constant  $\alpha > 1$  unless  $P = NP$ .*

- **Proof:** consider the following reduction from Hamiltonian cycle instance  $G = (V, E)$ :
  - The TSP instance is a complete graph  $G' = (V, E')$
  - Edge  $e \in E'$  has weight 1 if  $e \in E$ , and weight  $\alpha |V|$  otherwise
  - If  $G$  has Hamiltonian cycle, then  $G'$  has TSP tour with weight  $|V|$
  - If  $G$  does not have Hamiltonian cycle, then the minimum TSP tour in  $G'$  has weight  $> \alpha |V|$
  - An  $\alpha$ -approximate algorithm can tell these two cases apart

# Example: TSP

## Example (Symmetric Metric TSP)

- **Instance:** A complete undirected graph  $G = (V, E, w)$  with a weight function satisfying the triangle inequality

$$w(u, v) \leq w(u, w) + w(w, v) \quad \text{for all } u, v, w \in V .$$

- **Feasible solution:** A tour  $T = (v_1, v_2, \dots, v_n)$  visiting all vertices once.
- **Objective:** minimise  $c(T) = \sum_{i=1}^n w(v_i, v_{i+1 \bmod n})$ .

- **Symmetric metric TSP has  $3/2$ -approximation algorithm**
- **Symmetric metric TSP cannot be approximated with factor  $c$  for any  $c < 123/122$  unless  $P = NP$**

# On the PCP Theorem

# PCP Theorem

- **PCP theorem** is the one of the great celebrated result in theoretical computer science
  - Central tool for inapproximability results
- **What does the PCP theorem say?**
  - **Verification view:** every language in NP has a verifier that can verify the correctness of a certificate with constant number of queries
  - **Hardness of approximation view:** MAX-3SAT cannot be approximated within arbitrarily good constant

# PCP: Verification View

## Definition (Restricted verifiers)

- An  $(r(n), q(n))$ -**restricted verifier** is a polynomial-time probabilistic Turing machine  $V$  that takes as an input a string  $x \in \{0, 1\}$  and has a *random access* to a **proof**  $y$  of length at most  $q(n)2^{r(n)}$  such that
  - $V$  uses at most  $r(n)$  random bits, and
  - $V$  queries at most  $q(n)$  symbols of  $y$ .
- Given a random bit string  $z$  of at most  $r(n)$  bits, the verifier  $V$ :
  - (1) computes  $Q(x, z)$ , a set of  $k \leq q(n)$  indices,
  - (2) chooses  $k$  symbols  $y_1, \dots, y_k$  from  $y$  according to indices in  $Q(x, z)$ , and
  - (3) outputs 1 or 0 depending on  $x, z$ , and  $(y_1, \dots, y_k)$ .



# PCP: Verification View

## Definition (Probabilistically checkable proofs (PCP))

We say that a language  $L \subseteq \{0, 1\}^*$  is in class  $\mathbf{PCP}(r(n), q(n))$  if there is a  $(O(r(n)), O(q(n)))$ -restricted verifier  $V$  such that

- if  $x \in L$ , then there is a proof  $y \in \{0, 1\}^*$  such that  $\Pr[V(x, y) = 1] = 1$ , and
- if  $x \notin L$ , then for all proofs  $y \in \{0, 1\}^*$  we have  $\Pr[V(x, y) = 1] \leq 1/2$ .

## Theorem (The PCP theorem (Arora & Safra 1992))

$\mathbf{NP} = \mathbf{PCP}(\log n, 1)$ .

# PCP: Hardness of Approximation View

## Example (MAX-3SAT)

- **Instance:** A CNF formula  $\varphi$  with at most 3 literals per clause
- **Feasible Solution:** An assignment  $x$  into variables of  $\varphi$
- **Objective:** maximise  $c(x)$ , where  $c(x)$  is the number of clauses of  $\varphi$  satisfied by  $x$

# PCP: Hardness of Approximation View

## **Theorem (The PCP theorem, alternative form)**

*There is a constant  $\alpha > 1$  such that if there is a  $\alpha$ -approximation algorithm for MAX-3SAT, then  $P = NP$ .*

## **Theorem (Håstad 1997)**

*If there there is a  $(8/7 - \epsilon)$ -approximation algorithm for MAX-3SAT for any  $\epsilon > 0$ , then  $P = NP$ .*

# Inapproximability from PCP Theorem

## Theorem

*There are constants  $\alpha, \alpha' > 1$  such that maximum independent set cannot be  $\alpha$ -approximated and minimum vertex cover cannot be  $\alpha'$ -approximated unless  $P = NP$ .*

- **Proof sketch:**

- Apply the standard reduction from 3SAT to maximum independent set
- Observe that the size of the independent sets in the resulting graph is connected to the maximum amount of satisfiable clauses in the original formula

# Inapproximability from PCP Theorem

## Theorem

*Maximum independent set cannot be  $\alpha$ -approximated for any constant  $\alpha > 1$  unless  $P = NP$ .*

- **Proof:** boosting via graph products
- **The PCP theorem is analogous to Cook-Levin theorem for hardness of approximation**
  - Provides a starting point for further results

# Lecture 12: Summary

- Optimisation problems
- Approximation algorithms
- PTAS and FPTAS
- Inapproximability
- PCP theorem