COMP4141 Theory of Computation

Lecture 19 Approximation & Optimisation

Ron van der Meyden

CSE, UNSW

Revision: 2016/05/18

(Credits: D Dill, K Engelhardt, M Sipser, W Thomas, T Wilke, R van Glabbeek, P Hofner)

VERTEX-COVER

If $G = (V, E)$ is an undirected graph, a *vertex cover* of G is a subset $C \subseteq V$ where every edge touches one of the nodes in C:

$$\forall (x, y) \in E \, (x \in C \vee y \in C)$$

Theorem

The problem

$$\mathit{VERTEX\text{-}COVER} = \{ \langle G, k \rangle \mid G \text{ has a } k\text{-node vertex cover} \}$$

is **NP***-complete.*

Proof.

We show $\mathit{3SAT} \leq_{\mathbf{P}} \mathit{VERTEX\text{-}COVER}$ by transforming 3cnf-formulas into undirected graphs with 2 nodes per variable and 3 nodes per clause. [Details: see Sipser] $\square$

"An **NP***-completeness proof is typically the first act of the analysis of a computational problem by the methods of the theory of algorithms and complexity, not the last. Once* **NP***-completeness has been established, we are motivated to explore possibilities that are less ambitious than solving the problem exactly, efficiently, every time."*

Papadimitriou 1994, Page 299

Optimisation Problems

In *optimisation problems* we seek the best solution among a collection of possible solutions.

Example

On input $\langle G \rangle$, where G is an undirected graph, find a smallest vertex cover.

Remark: This is **NP**-hard, too, but optimization problems require their own special notion of reduction that describes how solutions to the problem reduced to are mapped back to solutions to the original problem.

NP Optimization Problems

Definition

An *NP Optimization* problem consists of

- An input space $I \subseteq \Sigma^*$
- A solution space $S \subseteq \Sigma^*$
- A solution predicate $P(x, y)$, representing that $y \in S$ solves problem $x \in I$, such that
  - $\{(x, y) \mid P(x, y)\}$ is in **P**
  - there exists a polynomial q such $P(x, y)$ implies $|y| \leq q(|x|)$
- A cost function $c : I \times S \to \mathbb{N}$, computable in polynomial time

An *optimal* (minimal/maximal) solution $y \in S$ for $x \in I$ is one that satisfies $P(x, y)$ and

- (minimal case) for all y' such that $P(x, y')$, we have $c(x, y) \leq c(x, y')$
- (maximal case) for all y' such that $P(x, y')$, we have $c(x, y) \geq c(x, y')$

Approximation Algorithms

$\textsc{Apx}$ = "On input $\langle (V, E) \rangle$, where (V, E) is an undirected graph:

1. Set $M := \emptyset$.
2. While there exists an (uncovered) edge $(x, y) \in E$
    1. add both vertices to the cover:
       $M := M \cup \{x, y\}$
    2. remove the vertices from the graph:
       $V := V \setminus \{x, y\}$
       $E := E \cap (V \times V)$
3. Output $\langle M \rangle$".

generates *some* vertex cover for (V, E) in polynomial time. But how close is it to an optimal one?

Theorem

APX *produces a vertex cover no more than twice as large as a smallest one.*

Proof.

Correctness: With every pair of nodes removed in the body of the main loop, we add both nodes to M. This implies that when we then remove these nodes from the graph, all edges removed touch a node in M. So the final M is a cover.

Approximation: Let H be the set of edges (x, y) considered, and Y an optimal vertex cover. Then, at termination,

1. $|M| = 2|H|$,
2. Y touches every edge in H,
3. H contains no edges sharing a vertex, so
4. no element of Y touches *two* edges in H, so
5. $|H| \leq |Y|$

from which $|M| \leq 2|Y|$ follows. □

k-Optimality

Definition

An approximation algorithm for a minimisation problem is *k-optimal* if it always finds a solution that is at most k times the size of an optimal one.

An approximation algorithm for a maximisation problem is *k-optimal* if it always finds a solution that is at least $\frac{1}{k}$ times the size of an optimal one.

Example

We've just shown that $\textsc{Apx}$ is 2-optimal for *VERTEX-COVER*.

Traveling Salesman Problem

Given n cities $1, \ldots, n$, and a nonnegative symmetric integer distance $d_{i,j} = d_{j,i}$ between any two cities i and j, we're asked to find the *shortest tour* of the cities—that is, a permutation π such that

$$d_{\pi(1),\pi(2)} + d_{\pi(2),\pi(3)} + \ldots + d_{\pi(n-1),\pi(n)} + d_{\pi(n),\pi(1)}$$

is minimal. This problem is called *TSP*.

Theorem

Unless $\mathbf{P} = \mathbf{NP}$*, there is no* $k \in \mathbb{N}$ *for which there exists a* k*-optimal approximation algorithm for TSP.*

Proof.

Suppose to the contrary that T is k-optimal for TSP. Then we can use T to solve $HAMCYCLE$ (undirected $HAMPATH$ with a closing edge) in **P** by the machine

Given undirected graph $\langle G = (V, E) \rangle$, let $t(G)$ be the TSP instance with $|V|$ nodes and distances

$$d_{i,j} = \begin{cases} 1 & \text{if } (i, j) \in E \\ k \cdot |V| & \text{otherwise} \end{cases}$$

$H =$ "On input G

If $T(t(G))$ returns a total cost of $\leq k \cdot |V|$ then *accept*, otherwise *reject*."

□

Proof (ctd).

Note the following are equivalent

- $G \in HAMCYCLE$
- $t(G)$ has a solution with cost $\leq |V|$
- $T(t(G))$ returns a solution with cost $\leq k|V|$
- $H(G)$ accepts

So H solves $HAMCYCLE$ in polynomial time! □

NB

This looks bad. If however all distances satisfy the triangle inequality, $d_{i,j} + d_{j,k} \geq d_{i,k}$, there are $\frac{3}{2}$-optimal approximation algorithms.

—The End—