csce750 — Analysis of Algorithms
Fall 2019 — Lecture Notes: Approximation Algorithms

This document contains slides from the lecture, formatted to be suitable for printing or individual reading, and with some supplemental explanations added. It is intended as a supplement to, rather than a replacement for, the lectures themselves — you should not expect the notes to be self-contained or complete on their own.

1 Introduction

CLRS 35

Suppose the problem you want to solve is NP-hard.

How can you proceed?

Some choices:

- Give up.
- Restrict to only small problem sizes.
- Restrict to special cases that can be solved in polynomial time.
- Give up on optimality, and instead find *near-optimal* solutions in polynomial time.

An algorithm for an optimization problem that only guarantees to get **close** to the optimal solution is called an **approximation algorithm**.

2 Optimization problems

In an **optimization problem**, the goal is produce a solution that minimizes or maximizes some **cost function**.

- C — cost of solution produced by algorithm.
- $C^\star$ — optimal cost.

3 Approximation ratios

We can characterize how well an approximation algorithm works by studying the relationship between C and $C^\star$.

An algorithm has **approximation ratio** ρ if

$$\max\left(\frac{C}{C^\star}, \frac{C\star}{C}\right) \leq \rho$$

If $\rho = 1$, we have an **exact algorithm**.

As ρ increases further beyond 1, the approximation accuracy gets worse.

4 Example 1: Vertex Cover

VERTEX COVER:
Instance: A graph G and an integer K.
Question: Is there a set of K vertices in G that touches each edge at least once?

We showed last time that VERTEX COVER is NP-hard.

Can we find an efficient approximation algorithm?

First, we need to treat it as an optimization problem.

Cost of a solution: Number of vertices in the cover.

5 Approximating Vertex Cover

```
ApproxVertexCover(G)
  C = ∅
  E' = G.E
  while E' ≠ ∅ do
    (u, v) = some edge in E'
    C = C ∪ {u, v}
    Remove from E' every edge incident to u or v.
  end while
  return C
```

6 Correctness proof, part 1

Theorem: APPROXVERTEXCOVER returns a valid vertex cover.

Proof: Every edge is either selected as (u, v), or incident to the u or v of some edge that is.

7 Correctness proof, part 2

Theorem: Let C denote the cover returned by APPROXVERTEXCOVER and let C^* denote an optimal vertex cover. Then $|C| \leq 2|C^*|$.

Proof: Let A denote the set of edges selected as (u, v) in APPROXVERTEXCOVER.

- By construction, $|C| = 2|A|$.
- Since $C^\star$ is a vertex cover, it must contain at least one vertex from each edge in A. But the edges in A do not share any endpoints. Thus $|C^\star| \geq |A|$.

Combining these, we conclude that

$$|C| = 2|A| \leq 2|C^\star|$$

.

Theorem: APPROXVERTEXCOVER is a 2-approximation algorithm for VERTEX COVER.

8 Example 2: Traveling salesman problem

TSP:
Instance: A weighted complete undirected graph G and a number w.
Question: Does there exist a Hamiltonian cycle in G with total weight at most w?

Theorem 34.14: TSP is NP-complete.

Can we approximate the optimization version of TSP?

9 Some bad news

Theorem: If $P \neq NP$, then for any $\rho > 0$, there is no polynomial time ρ-approximation algorithm for TSP.

Proof: (continued over next several slides)

Suppose algorithm A is a polynomial time ρ-approximation algorithm for TSP.

We'll use A to construct a polynomial time algorithm for HAM-CYCLE, which is known to be NP-complete.

Note: HAM-CYCLE is not an optimization problem.

10 Constructing a TSP instance

Let $G = (V, E)$ denote an instance of HAM-CYCLE. Construct an instance of TSP $G' = (V, E')$:

- $E' = \{(u, v) \mid u \neq v\}$,
- $c(u, v) = \begin{cases} 1 & (u, v) \in E \\ \rho|V| + 1 & \text{otherwise} \end{cases}$

11 Solving HAM-CYCLE?

Here's an algorithm B for HAM-CYCLE.

- Use the construction above to form an instance of TSP.
- Use algorithm A to solve this instance of TSP with approximation ratio ρ. Let x denote the length of the TSP tour returned by A.
- If $x \leq |V|$, return True. Otherwise return False.

12 *Correctness of* B

Claim: Algorithm B correctly solves HAM-CYCLE.

Proof: There are two cases.

- If G has a Hamiltonian cycle, then the optimal TSP tour for G' uses only edges in E, and thus has cost $|V|$.

- If G does not have a Hamiltonian cycle, then the optimal TSP for G' must use at least one edge not in E. So its cost must be at least

$$\underbrace{(|V|-1)}_{\text{edges in } E} + \underbrace{\rho|V|+1}_{\text{one edge not in } E} = \rho|V| + |V| \geq |V|.$$

13 *Pulling everything together*

Thus, if algorithm A exists to approximate TSP with ratio ρ in polynomial time, we can use algorithm B to solve HAM-CYCLE in polynomial time.

This contradicts the presumption that $P \neq NP$.

Conclusion: TSP is hard to approximate.

14 *We've only scratched the surface*

We considered only **constant approximation ratio** algorithms.

1. There might be a tradeoff between approximation quality.

   - We can try to find an **polynomial-time approximation scheme (PTAS)**.
     - Include a number $\epsilon > 0$ in the input.
     - Find a $(1+\epsilon)$-approximation algorithm whose run time is polynomial in n for any fixed ϵ.
   - Even better, we might try for a **fully polynomial-time approximation scheme (FPTAS)**, a PTAS in which the run time is polynomial in both n and $1/\epsilon$.

2. For randomized algorithms, we can consider the **expected approximation ratio**.

3. *etc, etc, etc.*