

Lecture 1

Lecturer: Rachit Chhaya

Scribe/s: Meet Desai (202311031)

1 Introduction

We discussed a fundamental overview of approximation methods in this lecture. We begin by defining approximation algorithms and outlining the necessity for their study. Then, using a classic optimization problem as an example, we define some algorithms addressing the problem.

Decision Problem: A computational problem known as a decision problem maybe expressed as a yes or no question of the input values.

Optimization Problem: A computational problem is known as an optimization problem in which the object is to find the best of all possible solutions.

Polynomial Time Reduction: Make A and B two decision-making problems. Let's say the initial algorithm solves B. In other words, depending on the input for problem B, the first algorithm will provide a yes or no response. Finding a transformation from A to B so that the first algorithm (that solves B) can

$$A \leq B ; \forall A \text{ in } NP$$

1.1 Polynomial Time Problems Class

The complexity class P contains the set of problems that can be resolved in polynomial time. This describes the class of problems that can be effectively solved through a theoretical point of view.

1.2 Non-deterministic Polynomial Time Problems Class

The class of problems that can be solved in non-deterministic polynomial time, or additionally, problems for which a solution can be verified in polynomial time, defines the class NP.

$$P \subseteq NP$$

1.2.1 NP-Complete problem set

Problem B is called NP-complete if and only if

- problem B is complete in Non-deterministic polynomial time,
 - Every problem in NP is reducible to problem B in polynomial time.
- For every problem A in NP;

$$A \leq B ; \forall A \text{ in } NP$$

1.2.2 NP-Hard Problem set

Problem B is called NP-hard if

- Every problem in NP is reducible to problem B in polynomial time. As an outcome, the NP-Hard set contains the NP-Complete set as a subset.

$$NP - Complete \subseteq NP - Hard$$

2 Approximation (Why?)

Let us say we are given an optimization problem to solve. The ideal algorithm would be one that always finds an optimal solution. However, implementing such an algorithm is unrealistic or impossible for many problems. NP-hard problems are a classic example of this type of problem since they can never be solved effectively, despite the exceptional case where $P = NP$. It is also possible that we need to be given more details at the beginning of the problem to handle it effectively.

2.1 α -Approximation Algorithm

Approximation Ratio (α): The approximation ratio (factor) of an algorithm is the ratio between the solution obtained by the algorithm and the optimal solution.

α - Approximation for maximization problem:

An algorithm is considered approximation for a maximization problem if, for all instances,

$$SOL \geq \alpha \cdot OPT \quad (\alpha < 1)$$

where SOL and OPT stand for the costs of the solution and the best solution, respectively.

α - Approximation for minimization problem:

An algorithm is considered approximation for a minimization problem if, for all instances,

$$SOL \leq \alpha \cdot OPT \quad (\alpha > 1)$$