Approximation Algorithms

Department of Mathematics and Computer Science
Drexel University

What is going to happen in this course?

- **What I would do for you:**
  - **Positive results:** we will review well-known approximation algorithms for various optimization problems.
  - **Negative results:** we will identify the level of hardness for approximating the solution to some of the optimization problems.

- **What You would do for me:**
  - **A project:** (Nirav will tell you about it).
  - **Homeworks:** Will be posted on the course web-site.
  - **Final Exam:** A three hour brutal exam!
Projects:

- Implementation of one of the 12 approximation algorithms (will be posted on the course web-site).
- Projects will be done individually.
- Projects are due one week before final exam.
- All the coding should be done in C++.
- The protocol for project submission will be presented by Nirav.
- Projects will carry 40% of final grade.

Let us START!

- The main question we did not answer last semester:
  - How can we solve NP-Complete problems efficiently (assuming P is not equal to NP)?
- The promise of approximation algorithms for NP-hard problems is to
  - Find a good solutions (hopefully very close to optimal).
  - Work for every instance of the problem.
  - Finish the algorithm fast (in polynomial time).
- So many terms, so little time!
What is an NP-Optimization Problem?

- An NP-Optimization problem is defined by the following items:
  - \( I \): a set of possible inputs, where we can check for each \( x \) whether or not it belongs to \( I \) in polynomial time.
  - \( S(x) \): The set of feasible solutions for \( x \) in \( I \).
  - \( f(y) \): An evaluation function that assigns a quantity to each solution \( y \) in \( S(x) \).
  - A \textit{min} or \textit{max} objective: a solution with optimum value, for each instance of the problem.

Example:

- Multiprocessor scheduling (minimum makespan):
  - There are \( m \) processors,
  - There are \( n \) jobs, with required execution times \( t_1, \ldots, t_n \).
  - We would like to assign the job to processors such that the total execution time is minimum.
Approximation algorithm defined

- A polynomial time approximation algorithm \( A \) gets input \( x \) in \( I \), and outputs in polynomial time \( y_A \) in \( S(x) \) that is close to optimal subject to some measurement for closeness. We use \( A(x) = f(y_A) \).

How Good is good enough?

- An approximation algorithm is said to be an \( \alpha \)-approximation algorithm for an optimization problem \( \Pi \) if:
  - The algorithm runs in polynomial time.
  - The algorithm always produces a solution which is within a factor \( \alpha \) of the value of the optimal solution.
- For minimization \( \alpha > 1 \), while for maximization \( \alpha < 1 \).
Example:

- Input: a graph $G=(V,E)$.
- Goal: Find a subset of vertices $C$ such that each edge $(u,v)$ has one of its endpoint in $C$.
- Objective: Minimize $|C|$.
  - If $M$ is a maximal matching for $G$, then $|M|\leq\text{OPT}_{VC}$.
  - Form a sub-optimal solution $C$ by adding both endpoints of edges in $M$ to $C$. Observe that $C$ is a vertex cover.
  - The performance guarantee of this solution is at most $2\text{OPT}_{VC}$.

FPTAS

- A problem $\Pi$ has a fully polynomial time approximation scheme if there exist an algorithm $A$ that takes as input an instance $x$ of $\Pi$ and a parameter $\varepsilon$:
  - Runs in time $\text{poly}(|x|,1/\varepsilon)$.
  - Produces a $(1+\varepsilon)$-approximation solution if $\Pi$ is a minimization problem.
  - Produces a $(1-\varepsilon)$-approximation solution if $\Pi$ is a maximization problem.
Example

- As we will see later, many problems admit FPTAS, including Knapsack and Euclidian TSP (Arora 1996, Mitchell 1996).

Knapsack:

- **Input**: \( n \) objects. Object \( i \) has weight \( w_i \) and value \( v_i \), with \( w_i \leq C, \forall 1 \leq i \leq n \).
- **Goal**: Find a subset \( S \) of the \( n \) elements that does not exceed capacity \( C \).
- **Objective**: maximize
  \[
  \sum_{i \in S} v_i
  \]
Constant factor and FPTAS for Knapsack

- Will be presented in the class!