15-251: Great Theoretical Ideas In Computer Science

Recitation 10 : Approximation Algorithms

Lecture Review

- The goal of an optimization problem is to find the minimum (or maximum) value under some constraints
- OPT(I) is the value of the optimal solution to an instance I of an optimization problem
- We say an algorithm $\mathcal{A}$ for an optimization problem is a factor-α approximation if for all instances I of the problem $\mathcal{A}$ outputs a solution that is at least as good as $\alpha \cdot \text{OPT}(I)$.

Max(ish)-Cut

We define the Max-Cut problem as follows:

Let $G = (V, E)$ be a graph. Given a coloring of the vertices with 2 colors, we say that an edge $e = \{u, v\}$ is *cut* if u and v are colored differently. In the *Max-Cut problem*, the input is a graph G, and the output is a coloring of the vertices with 2 colors that maximizes the number of cut edges.

Consider the following approximation algorithm for the Max-Cut problem:

def MaxCutApprox(G):

- cut = $\emptyset$
- improved = true
- **while** (improved):
  - improved = false
  - **for** v **in** G:
    - **if** adding v to cut increases cutEdges(cut):
      - cut = cut $\cup \{v\}$
      - improved = true
- **return** cut

(a) Prove that this algorithm is poly-time.

(b) Prove that this algorithm is a $\frac{1}{2}$-approximation for Max-Cut.

Pokémon Coverage

Consider a set of Pokémon and a set of trainers each having a subset of these Pokémon. Given k (assuming k is less than the number of trainers), the problem is to maximize the number of distinct Pokémon covered. Prove that there exists a polynomial-time $(1 - 1/e)$-approximation algorithm for this problem by considering the following greedy algorithm and by using the following steps:

On input $S_1, \ldots S_m$ (each set correponds to the Pokémon that a given trainer has) and k (the number of trainers chosen):

- Let $T = \emptyset$ (keeping track of trainers chosen)
- Let $U = \emptyset$ (keeping track of Pokémon covered)
- Repeat k times:
  - Pick j such that $j \notin T$ and $|S_j - U|$ is maximized.
  - Add j to T.
  - Update U to $U \cup S_j$.
- Output T.

(a) Show that the algorithm runs in polynomial time.

(b) Let T^* denote the optimum solution, and let $U^* = \cup_{j \in T^*} S_j$. Note that the value of the optimum solution is $|U^*|$. Define U_i to be set U in the above algorithm after i iterations of the loop. Let $r_i = |U^*| - |U_i|$. Prove that $r_i \leq (1 - \frac{1}{k})^i |U^*|$.

(c) Using the inequality $1 - \frac{1}{k} \leq e^{-\frac{1}{k}}$, conclude that the algorithm is a $(1 - \frac{1}{e})$-approximation algorithm for the problem.