E0 249: Approximation Algorithms

Naman Mishra

January 2024

Contents

Lecture 01: Mon 08 Jan '24

0 The Course

Course website: here
MS Teams: 091kg9h
Instructors: Prof. Arindam Khan and Prof. Anand Louis
TAs: Aditya Subramaniam (January)
Lecture Hours: MW 1400–1530 at CSA 112
Office Hours: Just email.

0.1 Grading

(30%) Homework

(20%) Project. Research papers from top conferences (STOC, FOCS, SODA, ICALP, SoCG). Around 10% on a report. 10% on a presentation (half an hour).

(20%) Midterm

(30%) Final

0.2 Texts

Primarily *The Design of Approximation Algorithms* by David Williamson and David Shmoys, Cambridge University Press, 2011. Available online for free.
Alternatively, *Approximation Algorithms* by Vijay Vazirani, Springer-Verlag, 2001.
For specific topics, Hochbaum, Sariel, etc.

1 Optimization Problems

Find the *best* solution from a set of *feasible* solutions. Richard Karp introduced the concept of NP-complete problems. Unless $\mathsf{P} = \mathsf{NP}$, the optimization version of the problems admit no algorithms that simultaneously (1) find optimal solution (2) in polynomial time (3) for all instances.

1.1 Optimization Version of Common NP-Complete Problems

Exact Decision Problem	Optimization Version
3-SAT Is a 3-CNF formula satifiable?	**Max 3-SAT** Find an assignment that satisfies as many clauses as possible.
3Col Is there a legal 3-coloring (all edges bichromatic) of a graph?	There are 2 natural corresponding optimization problems. • **Min-Coloring** Color legally with as few colors as possible. • **Max-3Cut** Color with 3 colors, make the coloring as legal as possible. Can be thought of as partitioning vertices into three sets, and maximizing the number of edges between the sets. (note: the usual **Max-Cut** is **Max-2Cut**)
Vertex Cover Input is a graph and an integer k. Is there a vertex cover (a subset of vertices such that every edge includes one of the vertices) of size less than k?	**Min-Vertex-Cover** Input is a graph. Output is a vertex cover. Value is the fraction of vertices in the cover.

1.2 Approximation Algorithms

Task: Solve NP-hard optimization problems A, but no efficient algorithm exists (unless $\mathsf{P} = \mathsf{NP}$).

Definition 1.1. Let Π be an optimization problem and let I be an instance of Π. Then $\mathrm{OPT}_\Pi(I)$ is the value of the optimal solution.

There might be several optimal solutions, but they all have the same value.

Definition 1.2. Let $\alpha \geq 1$. A is an α-approximation algorithm for a minimization problem Π if for every instance I of Π,

$$A(I) \leq \alpha \, \mathrm{OPT}_{\Pi}(I)$$

where $A(I)$ is the value of the solution that A returns for I.

Typically, α takes values like 1.5, 2, $O(1)$, $O(\log n)$, etc.
Usually we omit Π and I in $\mathrm{OPT}_{\Pi}(I)$.

For a maximization problem, $A(I) \geq \frac{1}{\alpha} \mathrm{OPT}_{\Pi}(I)$. (Sometimes in literature, $\alpha \leq 1$ is used for maximization problems.)

This is also called *absolute approximation*.

NP-hard problems are very similar to each other in terms of decidability, but can be very different in terms of approximability. For some problems, it is NP-hard to obtain any approximation (TSP) but for some (Knapsack) we can get $(1+\varepsilon)$-approximation in polynomial$(n, \frac{1}{\varepsilon})$ time.

Definition 1.3. A_ε is a polynomial time approximation scheme (PTAS) for a minimization problem Π if

$$A_\varepsilon(I) \leq (1+\varepsilon)\, \mathrm{OPT}(I)$$

and for every fixed $\varepsilon > 0$, the running time of A_ε is polynomial in n.

Definition 1.4 (EPTAS)**.** Efficient PTAS. $(1+\varepsilon)$-approximation in runtime $f(1/\varepsilon)n^{O(1)}$, where the exponent of n is independent of ε.

Definition 1.5 (FPTAS)**.** Fully polynomial time approximation scheme. $(1+\varepsilon)$-approximation in runtime polynomial in both n and $\frac{1}{\varepsilon}$.

Definition 1.6 (QPTAS)**.** Quasi-polynomial time approximation scheme. $(1+\varepsilon)$-approximation in quasi-polynomial time in n.

A *quasi-polynomial* is between a polynomial and an exponential.

Definition 1.7 (PPTAS)**.** Pseudo-polynomial time approximation scheme. $(1+\varepsilon)$-approximation in pseudo-polynomial time in n.

For example, $(nB)^{O(1)}$, where B is the biggest numeric data.

1.3 Asymptotic Approximation

Definition 1.8. The asymptotic approximation ratio ρ_A^∞ of an algorithm A is

$$\lim_{n\to\infty} \rho_A^n, \quad \text{where} \quad \rho_A^n = \sup_{I\in\mathscr{I}} \left\{ \frac{A(I)}{\mathrm{OPT}(I)} \mid \mathrm{OPT}(I) = n \right\}$$

Lecture 02: Wed 10 Jan '24

2 Greedy Algorithms

Construct a solution iteratively by taking 'myopic' choices, *i.e.*, choose the best augmentation that optimizes the objective. Greedy algorithms that work are simple and fast. Greedy algorithms that don't work, don't work.

2.1 Set Cover

Problem 1. Given:

- A ground set of n elements $E = \{e_1, \ldots, e_n\}$.
- A collection of m subsets of E, $\mathscr{S} = \{S_1, \ldots, S_m\}$.
- A cost function $\mathrm{cost}\colon \mathscr{S} \to \mathbb{Q}_+$.

Goal: Find a minimum weight collection of subsets from $\mathscr{S}$ that covers E.

For instance,

$$\begin{aligned} E &= \{A, B, C, D, E, F\} & n &= 6 \\ S_1 &= \{A, B, C\} & c(S_1) &= 10 \\ S_2 &= \{C, F\} & c(S_2) &= 10 \\ S_3 &= \{E, F\} & c(S_3) &= 8 \\ S_4 &= \{D, E\} & c(S_4) &= 10 \\ S_5 &= \{B, D, E\} & c(S_5) &= 11 \end{aligned}$$

Brute force is exponential in m ($O(n2^m)$).

We have several greedy options:

- Select minimum cost set. Obviously fails. Consider singleton sets of cost 1, and universal set of cost $1 + \varepsilon$. Greedy gives cost n, optimal is $1 + \varepsilon$. This is an n-approximation.
- Select set that covers the most uncovered elements. Obviously fails. Consider the same sets as before, but cost of universal set being arbi-

trarily large.

- Choose set that covers the most uncovered elements per unit cost. This is a $O(\log n)$-approximation.

```
GreedySetCover(E, 𝒮, cost):
  C ← ∅
  while C ≠ E
      α_S ← cost(S)/|S\C| for each S ∈ 𝒮
      Select S with minimum α_S
      for e ∈ S \ C
          price(e) ← α_S
      C ← C ∪ S
  return C
```

Proposition 2.1. GreedySetCover is an $O(\log n)$-approximation.

Proof. We make two observations.

- Left over sets from OPT can cover the remaining items from $E \setminus C$ at cost at most OPT.
- Among these left over sets, at least one must have cost effectiveness at most $\frac{\text{OPT}}{|E \setminus C|}$.

WLOG, suppose that the elements are numbered in the order in which they are selected by the greedy algorithm.

Assume element e_k was covered by the most cost-effective set at iteration $i \leq k$. The numbering implies that at most $k-1$ elements were selected before iteration i.

At the beginning of iteration i, $|E \setminus C| \geq n-k+1$. From our observation, we have that

$$\text{price}(e_k) \leq \frac{\text{OPT}}{|E \setminus C|} \leq \frac{\text{OPT}}{n-k+1}$$

The way price is defined, the cost of the set cover is the same as the sum of the prices of the elements. Thus we have

$$
\begin{aligned}
\operatorname{cost}(C) &= \sum_{j=1}^{n} \operatorname{price}(e_j) \\
&\leq \sum_{j=1}^{k} \frac{\mathrm{OPT}}{n-j+1} \\
&\leq \sum_{j=1}^{n} \frac{\mathrm{OPT}}{j} \\
&\leq H_n \, \mathrm{OPT}\,.
\end{aligned}
$$

From $H_n \leq 1 + \ln n$, we have that the cost of the greedy algorithm is at most $(1+\ln n)\,\mathrm{OPT}$.

Is this bound tight? Yes! Consider the following instance:

$$
\begin{aligned}
E &= \{1, \ldots, n\} \\
S_i &= \{i\} \\
\operatorname{cost}(S_i) &= 1 \text{ for } i = 1, \ldots, n \\
S_{n+1} &= E \\
\operatorname{cost}(S_{n+1}) &= 1 + \varepsilon
\end{aligned}
$$

The optimal solution is the set cover $\{S_{n+1}\}$ with cost $1+\varepsilon$. The greedy selects the sets $S_1, \ldots, S_n$ with total cost H_n. Thus, the approximation ratio r lies in

$$
\left[\frac{H_n}{1+\varepsilon}, H_n \right]
$$

for every $\varepsilon > 0$?

Current literature has an upper bound of $\ln(n/\ln n) + 0.78$ and a lower bound of $\ln(n/\ln n) - 0.31$. $\square$

Theorem 2.2 (Dinur-Steurer)**.** It is NP-hard to approximate set cover within $(1-\varepsilon)\ln n$ for all $\varepsilon > 0$.

2.1.1 Vertex Cover

Problem 2 (Vertex Cover). **Given:**

- a graph $G = (V, E)$.
- node weights $C : V \to \mathbb{Q}^+$.

Goal: A subset $U \subseteq V$ such that each edge is incident to at least one node in U and $\sum_{u \in U} C(u)$ is minimized.

This is a special case of SetCover where E is the ground set, and S_i is the set of edges incident to node i.

Homework: Prove that a maximal matching is a 2-approximation for VertexCover in the unweighted case.

2.2 Max-Cut

Definition 2.3 (Cut). Given an undirected graph $G = (V, E)$, a *cut* is a partition of V into S and $V \setminus S$.

Problem 3 (Max-Cut). **Given:**

- An undirected complete graph $G = (V, E)$.
- A weight function $w : E \to \mathbb{Q}$.

Goal: Find a cut $[S, V \setminus S]$ that maximizes the sum of the weights of the edges crossing the cut. That is,

$$\mathrm{OPT} = \max_{S \subseteq V} \sum_{(u,v) \in E} w(u, v)[u \in S \oplus v \in S]$$

Randomized algorithm:

<u>TBD($G = (V, E), w$):</u>
  $S \leftarrow \varnothing$
  for $v \in V$
    add v to S with probability $\frac{1}{2}$
  return $(S, V \setminus S)$

Proposition 2.4. The expected value of the cut returned by the above algorithm is $\frac{1}{2}\,\mathrm{OPT}$.

Proof. Define

$$X_i := \begin{cases} 1 & \text{if edge } i \text{ is a crossing edge} \\ 0 & \text{otherwise} \end{cases}$$

Then $E[X_i] = \frac{1}{2}$. Expected size of the cut is $\frac{|E|}{2} \geq \frac{\text{OPT}}{2}$ since $|\text{OPT}| \leq |E|$. $\square$