Tsinghua University

Discrete Optimistic

Fall 2024

**Homework 1**

Lin Zejin May 28, 2025

- **Collaborators:** I finish this homework by myself.

**Problem 1.** (a) When $\mathrm{OPT} \geq c$, assume with $\frac{1}{T}$ algorithm $A$ outputs a solution of value at least $s$. $T \in O(poly(n))$ Run algorithm $A$ for $T \cdot n$ iterations. Then with $(1-\frac{1}{T})^{Tn} < e^{-n}$ probability, the algorithm $A$ outputs a solution of value less than $s$.

So with at least $1 - e^{-n}$ probability, the algorithm $A$ outputs a solution of value at least $s$.

(b)

$$s = \mathbb{E}[outputs] \leq \Pr[outputs \geq s - \frac{1}{n^a}] \cdot poly(n) + (1 - \Pr[outputs \geq s - \frac{1}{n^a}]) \cdot (s - \frac{1}{n^a})$$

Then

$$\Pr[outputs \geq s - \frac{1}{n^a}] \geq \frac{\frac{1}{n^a}}{poly(n) - s + \frac{1}{n^a}} = \frac{1}{n^a(poly(n) - s) + 1}$$

Here we end the proof.

**Problem 2.** (a) Use the original greedy algorithm $\lceil \ln(n/\mathrm{OPT}) \rceil \cdot \mathrm{OPT}$ times, there are at most

$$(1 - 1/\mathrm{OPT})^{\lceil \ln(n/\mathrm{OPT}) \rceil \cdot \mathrm{OPT}} \cdot n \leq e^{-\ln(n/\mathrm{OPT})} \cdot n = \mathrm{OPT}$$

elements that do not cover.

So suffices to find at most OPT sets to cover those elements in polynomial time.

Here we obtain a $\lceil \ln(n/\mathrm{OPT}) \rceil + 1$-factor approximation algorithm.

(b) If the optimum solution covers $c \times 100\%$ elements, denote $S$ as the set of elements that optimum solution covers. For $t$-step, there exists a set in optimum solution that covers $(1 - \frac{1}{k}) \cdot$ elements in $S$ uncovered.

By induction, we can prove greedy algorithm covers $(1 - (1 - \frac{1}{k})^t) \cdot c$ elements in $t$-step.

Therefore, greedy algorithm returns a value larger than

$$(1 - (1 - \frac{1}{k})^k)c \geq (1 - \frac{1}{e})c$$

So greedy algorithm is a $(1 - \frac{1}{e})$-factor approximation algorithm.

(c) If $e_i$ is covered by $S_i$ in the solution, denote

$$p(e_i) = \frac{\omega(S_i)}{\text{number of uncovered elements that } S_i \text{ would cover}}$$

Then

$$\sum \omega(S_i) = \sum_{i=1}^{n} p(e_i)$$

We prove that if $e_i$ is covered in $k$-step, then $p(e_i) \leq \frac{\text{OPT}}{n-k+1}$.

If the optimal sets are $O_1, O_2, \cdots, O_p$, then $\text{OPT} = \sum_{i=1}^{p} \omega(O_i)$.

For any $O_i = \{x_k, x_{k-1}, \cdots, x_1\}$, wlog, we assume the algorithm covers $x_k, x_{k-1}, \cdots, x_1$ in order.

Then at the start of the iteration in which the algorithm covers element $x_j$ of $O_i$, at least $i$ elements that do not covered in $O_i$. Since $p(x_i)$ takes minimum in the equation,

$$p(x_i) \leq \frac{\omega(O_i)}{i}$$

Therefore, $\sum_{i=1}^{k} p(x_i) \leq H_D \omega(O_i)$

Then

$$\text{Val} = \sum_{i=1}^{D} p(e_i) \leq (1 + \frac{1}{2} + \cdots + \frac{1}{D})\text{OPT}$$

(d) Consider the set $[D]$ and $S_i = \{i\}, S_{D+1} = [D]$. Equipped with

$$\omega(S_i) = \frac{1}{i}, i = 1, 2, \cdots, D, \omega(S_{D+1}) = 1 + \epsilon$$

Then in $t$-step, $\dfrac{1}{D-t+1} < \dfrac{1+\epsilon}{D-t+1}$, so algorithm chooses $S_{D-t+1}$. Therefore, the value of algorithm is $H_D$ but $\text{OPT} = 1 + \epsilon$.

**Problem 3.** (a) Let $k = c$, $U = \{1, 2, \cdots, c\}^q$ where $q$ large enough. Introduce

$$S_{i,b} = \{e \in U : e_i = b\},\ i \in [q], b \in [c]$$

Choose $S_{i,t}, 1 \leq t \leq c$ and the coverage is 1.

$x^*_{i,b} = \frac{1}{q}$ will also achieves coverage 1.

Then we cover each $j \in U$ with probability

$$1 - (1 - \frac{1}{c})^c$$

So the expected coverage of rounding is

$$1 - (1 - \frac{1}{c})^c$$

AS $c$ large enough, the expected coverage of rounding is $1 - \frac{1}{e}$.

(b) With instance $k = c$, $U = \{0, 1\}^q$, $n = 2^q$ and

$$S_{i,b} = \{e \in U : e_i = b\},\ i \in [q], b = 0, 1$$

The LP solution $x^*_{i,b} = \frac{1}{q}$.

$\alpha x^*_{i,b} = \frac{(1-\epsilon)\ln n}{q} = (1-\epsilon)\ln 2 < \ln 2$.

Then

$$\Pr[j \text{ is covered}] = 1-(1-\alpha x^*_{i,b})^q < 1-(1-\ln 2)^q < 1-(2^{-1.5})^{\log_2 n} = 1-n^{-3/2}$$

So

$$\Pr[U \text{ is all covered}] < (1-n^{-3/2})^n < 1-n^{-\frac{1}{2}+\epsilon}$$

as $n$ large enough. So the randomized rounding algorithm may not be able to find a feasible solutionwith probability at least $n^{-\frac{1}{2}+\epsilon}$.

**Problem 4.** (a)

(b) No, since the rounding algorithm gives a solution with expected value large than $(1-\frac{1}{e})$LP. So

$$\text{OPT} \geq (1-\frac{1}{e})\text{LP}$$

always holds.

**Problem 5.** (a) Suffices to prove the decision problem that if there exists a clique of size $k$ is whether or not NP-complete.

For an 3-SAT instance with clauses $c_1, \cdots, c_m$ and literals $x_1, \cdots, x_n$, we can construct a graph $G$ with vertices $c_1, \cdots, c_m, x_1, \bar{x}_1, x_2, \bar{x}_2, \cdots, x_n, \bar{x}_n$ and additional clauses $x_{j_1} \wedge x_{j_2} \wedge x_{j_3}$ if there is some $c_j$ is composed by $x_{j_1}, x_{j_2}, x_{j_3}$ or some of its nagation. Let $k = 2m+n$

First construct a complete graph.

Remove the edges between $x_i$ and $\bar{x}_i$

Remove those edges that connnects $c_j = a \vee b \vee c$ and $\bar{a} \wedge \bar{b} \wedge \bar{c}$.

For a clause $a \wedge b \wedge c$, which means $a, b, c$ holds in the same time, we remove the edges between $a \wedge b \wedge c$ and those additional clauses that contains one of $\bar{a}, \bar{b}, \bar{c}$

Remove the edges between $a \wedge b \wedge c$ and $\bar{a}, \bar{b}, \bar{c}$

Then, that a graph is clique is equivalent to this conditions:

(1) If $a \wedge b \wedge c$ belongs to it, then additional clauses that contains $\bar{a}, \bar{b}, \bar{c}$ cannot belong to the same clique, and $c_j = \bar{a} \vee \bar{b} \vee \bar{c}$ cannot belong if it exists.

(2) If $x_i$ belongs to it, then $\bar{x}_i$ cannot belong to the same clique and those additional clauses contains $\bar{x}_i$ cannot belong to the same clique.

Certainly, the size of the clique in this graph cannot be larger than $2m+n$ since each clauses $c_j$ corresponds to 8 additional clauses but at most one of them is contained.

Therefore, if $3-SAT$ is satifiable, then we choose all true literals and all clauses that is true in the solution. Then $c_j$ will be chosen all and exactly one of eight addtional clauses that corresponds to $c_j$ will be chosen, so $k = 2m+n$ is reachable.

If there exists a clique of size $k = 2m+n$, which means choosing $n$ of independent literals, $c_j$ and exactly one of eight additional clauses that corresponds to $c_j$. Those clauses will be TRUE when the literals that is chosen is assuemd TRUE.

So it is a feasible instance for max-clique problem.

Therefore, It is NP-Hard.

(b)

If there is a $1-\epsilon$-approximation polynomial algorithm for graph in (a) and returns a solution.

First, we prove that we can find a solution in polynomial time that contains $n$ independent literals. That's because, each additional clause $a \wedge b \wedge c$ implies that $a, b, c$ is TRUE, which will not cause a contradiction by the clique assumption. So we actually can choose those TRUE literals and other random literals that is not mentioned. Since we want to return a max-clique, we actually can return a solution that contains $n$ independent literals.

Second, we prove we can find a solution in polynomial time that contains $m$ additional clauses, *i.e.* exactly one of eight additional clauses that corresponds to $c_j$ is contained.

Otherwise, if all of eight additional clauses that contains $a, b, c$ or their negation are not contained. Then since we find a solution that contains $n$ independent literals, we can choose an additional clause $a' \wedge b' \wedge c'$ such that $a' \in \{a, \bar{a}\}, b' \in \{b, \bar{b}\}, c' \in \{c, \bar{c}\}$ and $a', b', c'$ are chosen. And we can remove vertices $\bar{a'} \vee \bar{b'} \vee \bar{c'}$ if exists. Then it remains a clique of the same size. After $O(m)$, we actually obtain a solution that contains $n$ independent literals and $m$ additional clauses.

The graph we obtain actually finds a solution in $3-SAT$, with value

$$\frac{(1-\epsilon)(n+2m)-n-m}{m} = \frac{(1-2\epsilon)m-\epsilon n}{m} > 1-3\epsilon$$

if $n < m$.

3-SAT for $n \geq m$ is P-solved. So we find a $(1-3\epsilon)$-approximation polynomial algorithm for $3-SAT$.

PCP theorem implies $3\epsilon > \epsilon_0$. So $\exists \epsilon_1 > 0$ such that $(1-\epsilon_1)$-approximation polynomial algorithm for max-clique is NP-hard.

For a graph $G = (V, E)$, define

$$G^{\otimes t} = (V^{\otimes t}, E^{\otimes t})$$

where

$$V^{\times t} = \{(v_1, v_2, \cdots, v_t) : v_i \in V\}$$

$(v_1, \cdots, v_t), (u_1, \cdots, u_t)$ are connected iff $(v_i, u_i) \in E$, $\forall i = 1, 2, \cdots, t$.

Then $S \subset G^{\times t}$ is a clique iff

$$S^i = \{v_i : (v_1, \cdots, v_i, \cdots, v_t) \in S\}$$

are all clique.

Since

$$|S| \leq \prod_{i=1}^{t} |S_i|$$

$\Rightarrow \exists\ |S_i| \geq |S|^{1/t}$.

Therefore, if we find a solution with value $(1-\epsilon)^t$ in $G^{\otimes t}$, then we find a solution with value $1-\epsilon$.

$(1-\epsilon)$-approximation is NP-Hard $\Rightarrow$ $(1-\epsilon)^t$-approximation is NP-Hard.

So $\forall \delta > 0$, $\delta$-approximation is NP-Hard.

**Problem 6.** Consider the verifier reads one of $3-CNF$ uniformly and returns the result of this 3-CNF under the value given by prover.

If there exists a GAP-3SAT solution with value 1, then verifier always accepts.

If there is a GAP-3SAT solution with value less than $s$, then verifier accepts with probability less than $s$.

So $\mathrm{GAP}-3\mathrm{SAT}_{1,s} \in \mathrm{PCP}_{1,s}[O(\log n), 3]$.

Therefore, for any $NP$ problem $\mathcal{L}$, $\mathrm{GAP}-3\mathrm{SAT}_{1,s}$ NP-Hard $\Rightarrow \mathcal{L} \leq_p \mathrm{GAP}-3\mathrm{SAT}_{1,s}$.

So

$$\mathcal{L} \in \mathrm{PCP}_{1,s}[O(\log n), 3] \leq_p \mathrm{PCP}_{1,\frac{1}{2}}[O(\log n), O(1)]$$

In the lecture we prove that $NP \geq_p \mathrm{PCP}_{1,\frac{1}{2}}[O(\log n), O(1)]$.

So $NP = \mathrm{PCP}_{1,\frac{1}{2}}[O(\log n), O(1)]$. *i.e.* PCP theorem holds.

**Problem 7.** There is a counter-example for $U = \{u_1, u_2\}, V = \{v_1, v_2\}, K = L = 4$ and $G$ is fully connected.

$[K], [L] \leftrightarrow \{u, v\} \times \{1, 2\}$

$\pi_{(u_i,v_j)} = \{((u,i),(u,i)), ((u,i'),(u,i)), ((v,j),(v,j)), ((v,j'),(v,j))\}$ where $\{i,i'\} = \{j,j'\} = \{1,2\}$

Clearly, $\mathrm{OPT}(H) = \frac{1}{2}$ since two edges from vertices in $V$ cannot be satisfied both.

However, $\mathrm{OPT}(H^{\otimes 2}) = \frac{1}{2}$.

Let $\sigma((u_{i_1}, u_{i_2})) = (((u, i_1), (v, i_1)))$, $\sigma((v_{j_2}, v_{j_2})) = ((u, j_1), (v, j_2))$.

So verifier accepts if $i_1 = j_2$.