Lab11-Approximation Algorithm

Algorithm and Complexity (CS214), Xiaofeng Gao, Spring 2018.

* Name: Hongyi Guo   Student ID: 516030910306   Email: guohongyi@sjtu.edu.cn

1. Write $(I, sol, m, goal)$ for **Max k-Cover** and **Minimum Bin Packing** (refer Q2, Q3 below).

Solution.

name	I	sol	m	$goal$
Max k-Cover	$U = \{e_1, \cdots, e_n\}$ and a collection of m subsets $S = \{S_1, \cdots, S_m\}$ of U	k subsets from S $S_{i_1} \dots S_{i_k}$	number of e_i's s.t.$\exists j, e_i \in S_{i_j}$	max
Minimum Bin Packing	a finite rational set $F = \{a_i\}_{i=1}^n$, where $a_i \in (0, 1]$	$\{\{F_i\}_{i=1}^k \mid \cup_{i=1}^k F_i = F,$ $\forall i, j \in k, i \neq j, F_i \cap F_j = \emptyset$ $\forall i, 1 \leq i \leq k, \sum_{a \in F_i} a \leq 1\}$	k	min

□

2. **Max k-Cover:** Given a universe $U = \{e_1, \cdots, e_n\}$ of n elements, a collection of m subsets $S = \{S_1, \cdots, S_m\}$ of U, and a positive integer $k < m$. Our goal is to pick k subsets to *maximize* the number of covered elements. One greedy approach is shown in Algorithm 1.

Algorithm 1: Greedy Max k-Cover

```
Input: U,{S_i}_{i=1}^m, k.
Output: k subsets from {S_i}_{i=1}^m.
V ← U; W ← ∅;
for i = 1 to k do
    Pick S_j that covers max number in V;
    V ← V\S_j; W ← W ∪ S_j;
return W;
```

(a) Denote *opt* as the max number of covered elements; γ_i as the number of elements covered by greedy after i iterations; $\beta_i = opt - \gamma_i$. Show that $\gamma_i - \gamma_{i-1} \geq \frac{\beta_{i-1}}{k}$;

Solution. We denote W_i as the sets chosen by greedy algorithm after i iterations, and $W^\star$ as all sets chosen by optimal algorithm. It's clear that sets in $W^\star \backslash W_i$ can cover at least $opt - \gamma_i$ elements. On average, each of them can cover at least $\frac{opt-\gamma_i}{|W^\star \backslash W_i|} > \frac{opt-\gamma_i}{k}$ elements.

In greedy algorithm, S_j can cover the most number of elements among the remaining sets, so it definitely exceeds the average level. Therefore, $\gamma_i - \gamma_{i-1} \geq \frac{opt-\gamma_i}{k}$. □

(b) Prove that Algorithm 1 is an r-approximation where $r \leq 1 + \frac{1}{e-1}$, based on Problem 2a.

Solution. From problem 2a, we already proved that $\gamma_i - \gamma_{i-1} \geq \frac{\beta_{i-1}}{k}$.
So,

$$\gamma_k \geq \frac{opt - \gamma_{k-1}}{k} + \gamma_{k-1} = \frac{opt}{k} + (1 - \frac{1}{k})\gamma_{k-1}$$

.

Then we get

$$opt - \gamma_k \leq (1 - \frac{1}{k})(opt - \gamma_{k-1})$$

which is like a geometric sequence, then we can get following inequality equation via the property of geometric sequence.

$$opt - \gamma_k \le (1 - \frac{1}{k})(opt - \gamma_0) = (1 - \frac{1}{k})^k \times opt$$

Therefore,

$$\gamma_k \ge opt \times \left[1 - (1 - \frac{1}{k})^k\right] > opt \times (1 - \frac{1}{e})$$

.

So, the greedy algorithm is an r-approximation where $r = \frac{1}{1-\frac{1}{e}} = \frac{e}{e-1} = 1 + \frac{1}{e-1}$. □

3. **Minimum Bin Packing:** Given a finite rational set $F = \{a_i\}_{i=1}^n$, where $a_i \in (0, 1]$. We need to find a partition $\{F_i\}_{i=1}^k$ of F with no intersection and $\cup_{i=1}^k F_i = F$ with *minimum* k. The numbers in F_i are put into a bin, and the sum of numbers in each bin is at most 1. An idea to design a *sequential* algorithm is that for each number a_i, if a_i can fit into the last open bin then assign a_i to this bin, or else we open a new bin and assign a_i to it. Note that $\{a_i\}_{i=1}^n$ are NOT sorted in this algorithm.

   (a) Show that the approximation ratio of the algorithm by the idea above is at most $r = 2$.

   Solution. In any solution, the number of bins should be non-less than the sum of all numbers in F, otherwise they can't hold all numbers.

   In the greedy algorithm, the first number in the jth$(j > 1)$ bin is larger than the remaining place in the $j - 1$th bin. So the sum of number in two adjacent bin is larger than 1.

   Denote the number of bins in the sequential algorithm by m_s and in the optimal algorithm, $m^\star$. Suppose in the sequential algorithm, sum of number in the jth bin is b_j.

   Whether m_s is odd or even will not make much effect to r, we may as well assume m_s is even, then

$$\sum_{j=1}^{m_s} b_j = \sum_{j=1}^{m_s/2} (b_{j*2-1} + (b_{j*2})) > \sum_{j=1}^{m_s/2} = m_s/2$$

   So,

$$m_s < 2 \times \sum_{j=1}^{m_s} b_j = 2 \times \sum_{i=1}^{n} a_i \le 2m^\star$$

   Therefore, the approximation ratio of the sequential algorithm is at most $r = 2$. □

   (b) (Optional Subquestion with Bonus) Give an input instance to show the tightness of $r = 2$.

   Solution. For any given $0 < \epsilon < 1$. We can construct an instance as following.

$$F = \underbrace{\{\epsilon, 1, \epsilon, 1, \ldots, \epsilon\}}_{\frac{1}{\epsilon}\ \epsilon's,\ \frac{1}{\epsilon}-1\ 1's}$$

   The result given by greedy algorithm is $\frac{2}{\epsilon} - 1$.

   The result given by optimal algorithm is $\frac{1}{\epsilon}$.

   Therefore, $r = \frac{2/\epsilon - 1}{1/\epsilon} = 2 - \epsilon$. As long as ϵ is small enough, r is close to 2 infinitely. □

4. Consider the *Revised Greedy Knapsack* Algorithm (refer *GreedyKnapsack* in *Slide*17).

Algorithm 2: Revised Greedy Knapsack

Input: X with n items; b; $\{p_i\}_{i=1}^n$; $\{a_i\}_{i=1}^n$; $\epsilon > 0$.

Output: $val(Y)$: The total value of Y where $Y \subseteq X$ such that $\sum_{x_i \in Y} a_i \leq b$.

$Y_0 \leftarrow GreedyKnapsack(X, b, \{p_i\}_{i=1}^n, \{a_i\}_{i=1}^n)$; $h \leftarrow \epsilon \cdot val(Y_0)$;

Set $I_h \leftarrow \{i \mid 1 \leq i \leq n, p_i \leq h\}$, and reorder them as $I_h = \{1, 2, \cdots, m\}$ $(m \leq n)$, where $\{\frac{p_i}{a_i}\}_{i=1}^m$ is nonincreasing; $temp \leftarrow 0$; $currenttemp \leftarrow val(Y_0)$;

foreach $I \subseteq \{m+1, m+2, \cdots, n\}$ *such that* $|I| \leq \frac{2}{\epsilon}$ **do**

  if $\sum_{i \in I} a_i > b$ **then** $temp \leftarrow 0$;

  else if $\sum_{i=1}^m a_i \leq b - \sum_{i \in I} a_i$ **then** $temp \leftarrow \sum_{i=1}^m p_i + \sum_{i \in I} p_i$;

  else Find max k s.t. $\sum_{i=1}^k a_i \leq b - \sum_{i \in I} a_i < \sum_{i=1}^{k+1} a_i$ and $temp \leftarrow \sum_{i=1}^k p_i + \sum_{i \in I} p_i$;

  if $currenttemp < temp$ **then** $currenttemp \leftarrow temp$ and update Y;

return $val(Y) \leftarrow currenttemp$;

(a) Time complexity: $O(f(n, \epsilon)) = O(\_\_\_\_\_)$. (Sort: $O(n \log n)$).

Solution.

$$\begin{aligned} O(f(n,\epsilon)) &= O\left(n \log n + m * \binom{n-m}{1} + m * \binom{n-m}{2} + \cdots + m * \binom{n-m}{\lfloor \frac{2}{\epsilon} \rfloor}\right) \\ &= O\left(n \log n + m * (n-m)^{\lfloor \frac{2}{\epsilon} \rfloor}\right) \end{aligned} \quad (1)$$

□

(b) The approximation ratio is $1 + \epsilon$ when $\epsilon < 1$, then is it a log-APX? an APX? a PTAS? an FPTAS?

Solution.

log-APX	No
APX	Yes
PTAS	Yes
FPTAS	No

□