Lab11-Approximation Algorithm

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

$*$ If there is any problem, please contact TA Xinyu Wu.

$*$ Name:Juncheng Wan Student ID:516021910620 Email: 578177149@qq.com

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

Solution. For **Max k-Cover**:

(a) I is 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 $k < m$;

(b) $sol(U, S, k) = \{Z_1, Z_2, \cdots, Z_k | \forall Z_i, Z_i \subseteq S\}$;

(c) $m(U, Z) = |U \cap Z_1 \cap Z_2 \cap \cdots \cap Z_k|$;

(d) $goal$ =max.

For **Minimum Bin Packing**:

(a) $I = F = \{a_1, a_2, \cdots, a_n | \forall a_i, a_i \in (0, 1]\}$;

(b) $sol(F) = P = \{$ a partition $\{F_i\}_{i=1}^k$ of F $|\forall i \neq j, F_i \cap F_j = \emptyset, \cup_{i=1}^k F_i = F\}$;

(c) $m(F, P) = |P|$;

(d) $goal$ =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.

(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 $\gamma_i - \gamma_{i-1} \geq \frac{\beta_{i-1}}{k}$;

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

Algorithm 1: Greedy Max k-Cover

Input: U,$\{S_i\}_{i=1}^m$, k.
Output: k subsets from $\{S_i\}_{i=1}^m$.

$V \leftarrow U$; $W \leftarrow \emptyset$;
for $i = 1$ *to* k **do**
    Pick S_j that covers max number in V;
    $V \leftarrow V \backslash S_j$; $W \leftarrow W \cup S_j$;
return W;

Proof. For **Problem(2a)**, I prove by contradiction:

Assume that $\gamma_i - \gamma_{i-1} < \frac{\beta_{i-1}}{k}$.

Assume that $S^g = \{S_1^g, S_2^g, \cdots S_k^g\}$ where S_i^g is the subset the greedy algorithm picks at ith loop, $S^* = \{S_1^*, S_2^*, \cdots S_k^*\}$ where S_i^* is the subset the optimal condition picks at ith loop.

According to the knowledge of set theory, we have the following relationships:

$$\gamma_i - \gamma_{i-1} = |S_i^g \cap U| \tag{1}$$

$$|S_i^g \cap U| = \max_{1 \le t \le k} |S_t \cap (U - \gamma_{i-1})| \tag{2}$$

$$\beta_{i-1} = opt - \gamma_{i-1} = | \cup_{i=1}^k S_k^* - \gamma_{i-1}| \tag{3}$$

$$\cup_{i=1}^k S_k^* - \gamma_{i-1} = \cup_{i=1}^k (S_k^* \cap (U - \gamma_{i-1})) \tag{4}$$

Therefore, we can get that:

$$\gamma_i - \gamma_{i-1} < \frac{\beta_{i-1}}{k} \Longrightarrow |S_i^g \cap U| < \frac{1}{k}|S_i^g \cap U| \Longrightarrow$$
$$k \max_{1 \le t \le k} |S_t \cap (U - \gamma_{i-1})| < | \cup_{i=1}^k (S_k^* \cap (U - \gamma_{i-1}))|$$

However, it is obviously impossible. Therefore, $\gamma_i - \gamma_{i-1} \ge \frac{\beta_{i-1}}{k}$.

For **Problem(2b)**, I prove as follws:

$$\gamma_i - \gamma_{i-1} \ge \frac{\beta_{i-1}}{k} \Longrightarrow k\gamma_i - (k-1)\gamma_{i-1} \ge opt \Longrightarrow \frac{opt - \gamma_i}{opt - \gamma_{i-1}} \le \frac{k-1}{k} \Longrightarrow$$
$$\frac{opt - \gamma_k}{opt - \gamma_0} \le (\frac{k-1}{k})^k \Longrightarrow 1 - \frac{1}{r} \le (\frac{k-1}{k})^k \Longrightarrow r \le \frac{1}{1 - (1 - \frac{1}{k})^k}$$

Becasue $(1 - \frac{1}{k})^k \le \lim_{k \to \infty} (1 - \frac{1}{k})^k = e^{-1}$, therefore:

$$r \le \frac{1}{1 - (1 - \frac{1}{k})^k} \le \frac{1}{1 - (1 - e^{-1})} = 1 + \frac{1}{e-1}$$

□

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 $r = 2$.

   Proof. Set s the number of bins the *sequential* algorithm use, t the optimal number of bins.

   Because $r = \frac{s}{t}$. I prove $r \le 2$ by giving the uper bound of s and the lower bound of t.

   Set there are s bins generated by *sequential* algorithm sequentially and the sums of numbers in each bin are $b_1, b_2, \cdots, b_s$. Set there are t bins generated by optiaml solution sequentially and the sums of members in each bin are $c_1, c_2, \cdots, c_t$. Besides, set $d_1, d_2, \cdots, d_{s-1}$ are the immediate incompatiale number after each bin $b_1, b_2, \cdots, b_{s-1}$.

   According to the *sequential* algorithm, we have the following inequations:

$$b_2 \geq d_1 > 1 - b_1$$
$$b_3 \geq d_2 > 1 - b_2$$
$$\cdots$$
$$b_s \geq d_{s-1} > 1 - b_{s-1}$$

Therefore, for any adjacent b_i, b_{i+1} ($\forall i \in \{1, 2, \cdots, s-1\}$), $b_i + b_{i+1} > 1$. If we sum two columns of the inequations above directly, we can only have:

$$\begin{aligned} s &< 2\sum_{i=1}^{s} b_i + (1 - b_1 - b_s) \\ &= 2\sum_{i=1}^{n} a_i + (1 - b_1 - b_s) \\ &= 2\sum_{i=1}^{t} c_i + (1 - b_1 - b_s) \end{aligned}$$

We are forced to discuss it by two situations ($1 - b_1 - b_s \geq 0$ and $1 - b_1 - b_s < 0$). However, we have the following relationship:

$$\sum_{i=1}^{t} c_i = t + \sum_{i=1}^{t} (c_i - 1)$$

I want to prove $r \leq 2$ by the equations above. However, I spend a lot of time to proving it and the situation $1 - b_1 - b_s \geq 0$ is hard to deal (the situation $1 - b_1 - b_s < 0$ is obvious to prove).

However, I think I misunderstand the question. If what the question means is that number a_i is not put into the last bin immediately. Instead, if what the question means is that we can try the number a_j after a_i to test if a_j fits the last bin, I think I can prove it.

In fact, it is obvious that for all the selected bins, at most one bin is less than $\frac{1}{2}$. Assume that $b_p < \frac{1}{2}$, $b_p + b_{p+1}$ or $b_{p-1} + b_p$ is larger than 1 according to the analysis above. And other b_i is all larger than $\frac{1}{2}$. Therefore, we always have $s < 2\sum_{i=1}^{n} a_i \leq 2\sum_{i=1}^{t} c_i \leq 2t$ because $\forall c_i \in (0, 1]$. And we finish the proof. Note that in the finite rational set F, we can only have $<$ instead of $\leq$. □

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

Solution. Assume that there are $4m$ rational numbers:

$$a_i = \begin{cases} \frac{1}{2} & \text{i is odd} \\ \frac{1}{2m} & \text{i is even} \end{cases}$$

In this case, we have $s = 2m$, $t = m + 1$. Thus, $\lim\limits_{m \to \infty} r = \lim\limits_{m \to \infty} \frac{2m}{m+1} = 2$.

Note that this set is infinite rational set and what the question is interested in is finite set. However, this instance shows that the r of finite set can be infinitely close to 2, which proves the tightness of $r = 2$. □

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 $\sum_{x_i \in Y} a_i \leq b$.

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

2 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; $temp \leftarrow 0$; $currenttemp \leftarrow val(Y_0)$;

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

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

5   ;

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

7   ;

8   **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 $temp \leftarrow \sum_{i=1}^k p_i + \sum_{i \in I} p_i$;

9   ;

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

11   ;

12 **return** $val(Y) \leftarrow currenttemp$;

(a) Time complexity: $O(f(n, \epsilon)) = O(n^{2+[\frac{2}{\epsilon}]})$. (Sort: $O(n \log n)$).

Proof. I prove it line by line.
Line1: the time complexity of Greedy Knapsack is $O(nlogn)$.
Line2: the time complexity of reorder by using sort is $O(nlogn)$ as the hint is.
Line3-11: if $[\frac{2}{\epsilon}] \leq (n-m)$, the iterations of the *for* loop is $\binom{n-m}{1} + \binom{n-m}{2} + \cdots + \binom{n-m}{[\frac{2}{\epsilon}]} = O(n^{[\frac{2}{\epsilon}]})$; if $[\frac{2}{\epsilon}] > (n-m)$, the iterations of the *for* loop is $\binom{n-m}{1} + \binom{n-m}{2} + \cdots + \binom{n-m}{n-m} = O(2^n)$.
In the *for* loop: line4 costs $O(1)$; line6 costs $O(n)$ to calculate $\sum_{i=1} ma_i$; line8 costs $O(n^2)$ to try n times and each time to calculate the sum costs $O(n)$; line10 costs $O(1)$.
Line12 to get the $val(Y)$ costs $O(n)$.
Therefore, if $[\frac{2}{\epsilon}] \leq (n-m)$, we have the total time complexity as follows:

$$
\begin{aligned}
&O(nlogn + nlogn + (\binom{n-m}{1} + \binom{n-m}{2} + \cdots + \binom{n-m}{[\frac{2}{\epsilon}]})(1 + n + n^2 + 1) + n) \\
=&O(nlogn + n^2(\binom{n-m}{1} + \binom{n-m}{2} + \cdots + \binom{n-m}{[\frac{2}{\epsilon}]})) \\
=&O(n^{2+[\frac{2}{\epsilon}]})
\end{aligned}
$$

If $[\frac{2}{\epsilon}] > (n-m)$, the total time complexity is $O(n^2 2^n)$.
Note that I think what the question is interested in is when $[\frac{2}{\epsilon}] \leq (n-m)$.

□

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

Solution. PTAS. Because the approximation ratio is $1 + \epsilon$ when $\epsilon < 1$ and the time complexity is not proportional to ϵ (ϵ appears on the exponent). □