Lab10-Approximation & Randomized Algorithm

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2019.

$*$ If there is any problem, please contact TA Mingran Peng.

$*$ Name: KylinChen Student ID: 517030910155 Email: k1017856853@icloud.com

1. Given a CNF Φ with n boolean variables $\{x_i\}_{i=1}^n$ and m clauses, with each clause consisting of 3 boolean variables. For example $\Phi = C_1 \wedge C_2 = (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$. Assume that Φ is satisfiable, the goal is to find the feasible assignment of $\{x_i\}_{i=1}^n$ with **fewest true boolean variables**.

   (a) Please formulate it into integer programming.

   (b) Design an approximation algorithm based on deterministing rounding. Choose its approximation ratio and explain. Pseudo code is needed.

Proof.

(a) For every variable x_i, it equals 1 iff x_i is assigned to $True$, otherwise it's $False$. We can give a universe form for x_i and $\overline{x_i}$: z_i, it satisfy:

$$z_i = \begin{cases} x_i & , x_i, \\ 1 - x_i & , \overline{x_i}. \end{cases}$$

Then we can give the IP formulate below:

minimize.

$$\sum_{i=1}^{n} x_i$$

subject.

$$\forall x_i,\ x_i \in \{0, 1\}$$

$$\forall (i, j, k) \in clause,\ z_i + z_j + z_k \geq 1$$

(b) **LP-Relaxation:** We first convert IP-problem to LP-problem, we still use z_i to represent variable x_i and $\overline{x_i}$ while z_i can be any number between 0 and 1. The we can give the LP-formulation:

minimize.

$$\sum_{i=1}^{n} x_i$$

subject.

$$\forall x_i,\ 0 \leq x_i \leq 1$$

$$\forall (i, j, k) \in clause,\ z_i + z_j + z_k \geq 1$$

Deterministic Rounding Bound: For deterministic rounding bound, we can find that for every inequality(clause), there are exactly have three items(elements),it means with the constrain $z_i + z_j + z_k \geq 1$, there exists at least one item in $\{z_i, z_j, z_k\}$ which is greater than $\frac{1}{3}$. So we can use $\frac{1}{3}$ as the deterministic rounding bound, which make every clause $True$ obviously.

Algorithm: Therefore, we can use deterministic rounding to design an agorithm as follow:

Algorithm 1: 3-SAT via LP-Rounding (Deterministic)

Input: n boolean variables $\{x_i\}_{i=1}^n$; m clause;

Output: values of variables $\{x_i\}_{i=1}^n$ which make every clause $True$;

Find an optimal solution X to the LP-relaxation;

for $\forall x_i \in X$ **do**

  if $x_i \geq \frac{1}{3}$ **then**

    round $x_i = 1$;

  else

    round $x_i = 0$;

return $\{x_i\}_{i=1}^n = \{x_i | x_i = True \text{ iff } x_i = 1\}$;

Proof:

(1) **Feasible Solution:** For every clause $c \in C$, must have at least one element z_i which is greater than $\frac{1}{3}$(no matter which form it represent, x_i or $1 - x_i$). So with the deterministic rounding algorithm, it must have all the m clauses is $True$, therefore it get a feasible solution.

(2) **Approximation Ratio:** We first assume that m clause have $3m$ different variables, which means we at most define $3m$ $True$ variables at the worst case. but the OPT is m variables, which satisfies 3-Approximation. Once the frequency of some variable(s) are greater than others, LP-Rounding will first choose these variables $True$, since each clause just have 3 variables, it makes 3-Approximation, too.

2. (Bonus)Suppose there is a sequence of pearls of different color. Color is denoted as $1 - m$ and the total number of pearls is n. After you read these information and conduct some pre-processing, you need to face lots of of queries.

   A query gives two positions $1 \leq l \leq r \leq n$, and ask whether there exists a color, that at least half of pearls in $[l, r]$ is such color.

   (a) Design a random algorithm to solve this problem. Space complexity of your algorithm should be strictly better than $O(mn)$. Explain your idea briefly, give time complexity for pre-processing and per query, and give space complexity. Your accuray should be better than 99.9%.

   For example, a naive algorithm just read in all pearls as pre-processing. And naively iterate every color and every postion for query. This case, the pre-processing complexity is $O(n)$. For query, it will execute $(r-l)*m$ times, since $r-l$ can achieve $n-1$, so time complexity per query is $O(mn)$. No extra space needed.

(Hint: Random choose some color and examine.)

(b) **Remark:** This question involves a little bit knowledge about online algorithm. The ddl for this lab is 5/27/2019.

Now there are extra operation besides query.

Append(c): Put a peral with color c at the end of sequence.

Erase: Take out the last pearl.

Colouration(p,c): Choose pearl of postion p and change its color to c.

Assume that no operation will involve a new color. You may modify your algorithm and show time complexity for each type of operation(include query).

(Hint: Consider Balanced Binary Tree. Given an element e, they can find whether e exists in tree, and how many elements in tree are smaller than e, in $O(logn)$ time.)

Proof.

(a) We design an algorithm by a random prime number sequence P(for example, P can be $\{2,3,5,7,11,\cdots\}$, which must contains m elements). Then we can achieve this algorithm by steps below:

(1) **Pre-Processing:** Execute sequential scanning the pearls from l to r, once we scan a new color, we define it as a new prime number(for example, if we first scan blue pearl in location l, wedefine it as first prime in P, it says 2). Define a hush-check S, initiated with $S=1$, once we scan a pearl, we multiply the corresponding prime number to S, and finally store the multiply result in S. The complete scan get a $color-P$ relation(store in hush table) and hush-check S.
(2) **Romdom Algorithm:** We randomly generate a color sequence, which must be corresponding to a prime sequence. For each prime $p_i \in P$, We check the certifier equals to 0 or not:

$$S \ \% \ p_i^{(r-l-1)}$$

If the certifier equals to 0, it means p_i's corresponding color is half satisfied, then we can stop the process.

Analysis: In this Algorithm, we use $O(m)$ to store $color-P$ hush table and a $O(1)$ variable S to store hush-check, which means space complexity is $O(m)$, which is better than $O(mn)$. For time complexity, pre-process takes $O(r-l)$ and random check takes $O(2(r-l))$ (because hush check and mod operation takes 1 each). it means the Time complexity is $O(r-l)$, which is better than $O(n)$.

(b) We can use the algorithm in part(a) to complete the operations required. Now that we get the hush-check S and a $color-P$ hush table of origin sequence, we can achieve the operation below:
Append(c): Multiply c's corresponding prime number p_i by $color-P$ hush table with $O(1)$ time. Then multiply p_i to S. total operation take $O(1)$.
Erase: We assume the last pearl with color c, we let S devide c's corresponding prime number p by $color-P$ hush table with $O(1)$ time.
Colouration(p,c): We check position p's color and let its corresponding prime p is devided by S, then let S multiply c's correspond prime p. Since we should check position

p's color, it takes $O(n)$ time totally.
Query(c): it have the same operation as part(a).

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.