UNIVERSITY OF LIVERPOOL

Department of Computer Science COMP523 Assignment 2

201672656

Tianxiang Jia

pstjia4@liverpool.ac.uk

A

(i) If a solution to the problem can be verified in polynomial time given a certificate, the problem is in NP.

  a) Check if T is a tree

    i. Verify that T covers all vertices V of the G. This can be done by ensuring that every vertex in V appears in T.

    ii. Verify that T has |V| - 1 edges, which is a characteristic of a tree.

    iii. Verify that T is acyclic. This can be achieved using a DFS, and the time is O(|V|+|E|)

  b) For each vertex in T, count the number of edges incident to it and verify that this count is less than or equal to d. This also can be completed in O(|V|+|E|) time since you just need to inspect each edge once.

  Both of these verification steps are polynomial in terms of the input size, therefore, the bounded degree spanning tree problem is in NP.

(ii) To show that the problem is NP-Hard for d=2, we can reduce the Hamiltonian Path problem to this problem, because Hamiltonian Path is an NP-Complete problem.

  a) If there is a Hamiltonian Path in G, this path itself is a spanning tree whose degree is less than or equal to 2.

  b) Conversely, if there is a spanning tree with a degree of no more than 2, and this tree covers all of the vertices in G. So, this tree is a Hamiltonian path, because the degree of any vertex is not more than 2.

  c) For example if we have a G, in which V = {1,2,3,4}, E={(1,2), (2,3), (3,4), (4,1), (1,3)}. The Hamiltonian path can be found easily, such as 1-2-3-4.

  d) If we want to find a bounded degree spanning tree in which d =2 since there is a Hamiltonian path 1-2-3-4, we can directly use this path as a spanning tree where each vertex has a degree of no more than 2.

  e) Because the Hamiltonian path can be reduced into the bounded degree spanning tree problem d=2. This means that this problem is at least as difficult as the Hamiltonian path, so it is an NP-Hard problem.

(iii) If we want to prove that the bounded degree spanning tree problem for d=10 is NP-Hard, we can reduce from the Hamiltonian Path to this problem.

a) Use the same graph G for both problems. The key point is that if a graph has a Hamiltonian Path, then it has a spanning tree where the maximum degree is 2.

b) Implication: If there is a method that can find a spanning tree with a maximum degree of 10 in any graph, then it can find a Hamiltonian path by checking if there is a path with a maximum node degree of 2.

c) Through the above reduction, it can be seen that the bounded degree-spanning tree can be reduced from the Hamiltonian Path problem. Because the Hamiltonian Path problem is an NP-Hard problem, this problem is at least as difficult as the Hamiltonian Path problem. So this problem is an NP-Hard problem.

B

(i) Due we use the Greedy $k-Explorer, we will buy the bag that has the most number of 'new' flavor candies. The x_i is the number of flavors in this bag, it must be bigger than or equal to the average of the 'new' flavor in each left bag. OPT can be seen as the maximum number of flavors, $\sum_{j=1}^{i-1} x_j$ can be seen as the number of 'old' flavors. So, the $OPT - \sum_{j=1}^{i-1} x_j$ is the number of the 'new' flavors left. Therefore the average of the 'new' flavor in each left bag is $\frac{OPT-\sum_{j=1}^{i-1} x_j}{k}$. So, the $x_i \geq \frac{OPT-\sum_{j=1}^{i-1} x_j}{k}$ can be proved.

(ii) At first, we need to know what is the fomular $OPT - \sum_{j=1}^{i} x_j \leq \left(1-\frac{1}{k}\right)^i * OPT$ means. $OPT - \sum_{j=1}^{i} x_j$ means that the number of the left 'new' flavors, and we need to prove that this number is less than or equal to $\left(1-\frac{1}{k}\right)^i * OPT$. It can be proved by induction that after I iteration of the algorithm, the number of flavors that we have not yet tried is at most $\left(1-\frac{1}{k}\right)^i * OPT$.

a) When $i = 0$, $OPT = OPT$, obviously satisfies.

b) Assume the inequality holds for $i-1$, according to the answer (i), we can trans the formula as below:

$$OPT-\sum_{j=1}^{i}x_j \le \mathrm{OPT}-\left(\sum_{\mathrm{j=1}}^{\mathrm{i-1}}\mathrm{x_j}+\frac{OPT-\sum_{\mathrm{j=1}}^{\mathrm{i-1}}\mathrm{x_j}}{k}\right)$$

$$=\left(1-\frac{1}{k}\right)*\left(OPT-\sum_{\mathrm{j=1}}^{\mathrm{i-1}}\mathrm{x_j}\right)$$

So the question becomes to the relationshipn between $\left(1-\frac{1}{k}\right)*\left(OPT-\sum_{\mathrm{j=1}}^{\mathrm{i-1}}\mathrm{x_j}\right)$ and $\left(1-\frac{1}{k}\right)^{i}*OPT$. In order to simplify the comparison, it can divide by the common factors $\left(1-\frac{1}{k}\right)$ and OPT. So, it becomes to the $1-\frac{\sum_{\mathrm{j=1}}^{\mathrm{i-1}}\mathrm{x_j}}{OPT}$ and $\left(1-\frac{1}{k}\right)^{i-1}$. It can be seen that when the $i>1$ the $1-\frac{\sum_{\mathrm{j=1}}^{\mathrm{i-1}}\mathrm{x_j}}{OPT}$ always less than $\left(1-\frac{1}{k}\right)^{i-1}$. When $i=1$ $1-\frac{\sum_{\mathrm{j=1}}^{\mathrm{i-1}}\mathrm{x_j}}{OPT}=\left(1-\frac{1}{k}\right)^{i-1}=1.$

Therefore, $OPT-\sum_{j=1}^{i}x_j \le \left(1-\frac{1}{k}\right)^{i}*OPT$ is satisfied.

(iii)

(iv) To formulate the k-Explorer Problem as an ILP and give its LP-relaxation, some variables need to be defined. For each flavor j, set a binary variable y_i to show whether the flavors have been tried. For each bag i, set a binary variable x_i to show the binary decisions for buying bags.

a) Target: Maximize the number of different flavors tried:$max\sum_j y_j$

b) For each flavor j, $y_j \le \sum_{i:j\in B_i}x_i\,\forall j$

c) For x_i, $\sum_i x_i \le k, x_i \in \{0,1\}\forall i$

d) For each $y_j \in \{0,1\}\ \forall j$

The LP-relaxation involves relaxing the integer constraints on the variables to allow them to take any value in the continuous range [0,1]. The rest of ILP stays the same in the LP-relaxation. TDFShe LP-relaxation can be solved in polynomial time using linear programming techniques.

C

(i) At first, define the ILP as follows:

a) Variables: x_i is a binary variable that indicates whether request r_i is served in Direction 1 or Direction 2. When r_i is served in Direction 1, the value x_i is 0, otherwise the value is 2.

b) Objective: minimize the maximum makespan congestion C_{max} for all edges.

c) Constraints:

i. For each edge e, calculate the congestion C_e if the requests r_i pass through the edge e, and this congestion should not be bigger than C_{max}.

ii. Each x_i must be either 0 or 1, indicating the direction chosen for that request.

d) The whole ILP is

Minimize C_{max}

Subject to $C_{\max} \geq \sum_{r_i \in R_e}(1 - x_i)$ for each edge e in Direction 1

$C_{max} \geq \sum_{r_i \in R_e} x_i$ for each edge e in Direction 2

$x\_i \in \{0,1\}$ for each $r_i \in R$

And for the LP-relaxtion, change the variable x_i as below:

$x\_i \in [0,1]$ for each request $r_i \in R$

(ii) Assume that the x_i^* the solution of the i request in the LP relaxation. For each request r_i:

a) If the $x_i^* \leq 0.5$, $x_i = 0$, it means that the request will in Direction 1

b) If the $x_i^* > 0.5$, $x_i = 1$, it means that the request will in Direction 2

c) A formulaic description of the rounding scheme:

For each I, $x_i = \begin{cases} 0 if x_i^* \leq 0.5 \\ 1 if x_i^* > 0.5 \end{cases}$

(iii) In the LP-relaxation solution, for the edge e, C_e is calculated by weighting the requested fractional value of x_i^*. It means that for each request r_i in LP-relaxation solution, load by up to 0.5 is increased in one direction. In the worst case, if $x_i = 0.5$ in the LP-relaxation solution, rounding doubles the load on the edge e. Due to this load increase is uniform, for each request on every edge e:

$$C_e \leq \sum_{r_i \in R_e} x_i \leq \sum_{r_i \in R_e} (x_i^* + 0.5) \leq C_e^* + \frac{1}{2} \times |R_e|$$

In this formula, $|R_e|$ is the total number of the request which passes through the edge e. In LP-relaxation because we are minimizing the C_{max}^*. so for any edge e, $C_e^* \leq C_{max}^*$. In this way, the biggest congestion in the rounded solution is guaranteed:

$$C_{max} \leq C_{max}^* + \frac{1}{2} \times |R_e|$$

For each edge e, in the worst case, $|R_e| \leq 2C_e^*$, therefore the formula can be change to:

$$C_{max} \leq C_{max}^* + C_e^* \leq 2 \times C_{max}^*$$

Therefore, the rounding scheme results in a polynomial time 2-approximation algorithm for the original problem.

D

(i) If we want to find the probability that a random assignment is different on exactly k variables from a fixed satisfying assignment α, due to the probability of each variable being same or not same being equal. There are k variables picked from n variables and do not consider the order, so the combination number is $\binom{n}{k}$. For these k variables, each variable is different from the α. Therefore the probability $Pr[A_k]$ is :

$$\Pr[A_k] = \binom{n}{k}\left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{n-k} = \binom{n}{k}\left(\frac{1}{2}\right)^{\mathrm{n}}$$

(ii) The most k incorrect steps mean that after the algorithm switches the variables' value, the clause is still not satisfied, and the probability is $\frac{2}{3}$. According to the binomial distribution, there is:

$$P_k = \sum_{i=0}^{k} \binom{3k}{i}\left(\frac{2}{3}\right)^i \left(\frac{1}{3}\right)^{3k-i} = \sum_{i=0}^{k-1} \binom{3k}{i}\left(\frac{2}{3}\right)^i \left(\frac{1}{3}\right)^{3k-i} + \binom{3k}{k}\left(\frac{2}{3}\right)^k \left(\frac{1}{3}\right)^{2k}$$

Due to the probability is always positive, $\sum_{i=0}^{k-1}\binom{3k}{i}\left(\frac{2}{3}\right)^i \left(\frac{1}{3}\right)^{3k-i} \geq 0$ always hold. Therefore $P_k \geq \binom{3k}{k}\left(\frac{2}{3}\right)^k \left(\frac{1}{3}\right)^{2k}$ always hold.

(iii) $P_k \geq \frac{\sqrt{6\pi}}{e^2} \cdot \frac{2^{-k}}{\sqrt{k}}$

From the Stirling's inequality, n! can be bounded as:

$$\sqrt{2\pi} n^{n+1/2} e^{-n} \leq n! \leq e n^{n+1/2} e^{-n}$$

For 3k:

$$(3k)! \geq \sqrt{2\pi}(3k)^{3k+1/2} e^{-3k}$$

For k:

$$k! \leq \mathrm{e} k^{k+1/2} e^{-k}$$

For 2k:

$$(2k)! \leq e(2k)^{2k+1/2} e^{-2k}$$

Therefore:

$$\binom{3k}{k} \geq \frac{\sqrt{2\pi}(3k)^{3k+1/2} e^{-3k}}{\mathrm{e} k^{k+1/2} e^{-k} \cdot e(2k)^{2k+1/2} e^{-2k}}$$

Simplify:

$$\binom{3k}{k} \geq \frac{\sqrt{2\pi}(3k)^{3k+1/2}}{e^2\, k^{k+1/2} \cdot (2k)^{2k+1/2}}$$
$$= \frac{\sqrt{6\pi} 3^{3k}}{e^2 2^{2k+1/2} \sqrt{k}}$$

From:

$$P_k \geq \binom{3k}{k} \left(\frac{2}{3}\right)^k \left(\frac{1}{3}\right)^{2k}$$

We can get:

$$P_k \geq \binom{3k}{k} \left(\frac{2}{3}\right)^k \left(\frac{1}{3}\right)^{2k} = \frac{\sqrt{6\pi} 3^{3k}}{e^2 2^{2k+1/2} \sqrt{k}} \left(\frac{2}{3}\right)^k \left(\frac{1}{3}\right)^{2k}$$
$$= \frac{\sqrt{6\pi}}{e^2 2^{k+1/2} \sqrt{k}}$$
$$= \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{6\pi}}{e^2} \cdot \frac{2^{-k}}{\sqrt{k}}$$

I checked 4 times for these processes, and I think my calculation is right. There is always a $\frac{1}{\sqrt{2}}$ in the formula. The only explanation I can think of is when k is big enough, $\frac{1}{\sqrt{2}}$ can be ignored. So that $P_k \geq \frac{\sqrt{6\pi}}{e^2} \cdot \frac{2^{-k}}{\sqrt{k}}$ is hold.

(iv) Using the law of total probability and the expression $\sum_{k=0}^{n} \binom{n}{k} 2^{-k} = (3/2)^n$, show that if a solution exists, then $Pr[S] \geq \frac{\sqrt{6\pi}}{e^2 \sqrt{n}} \cdot (3/4)^n$

The total probability can be show as:

$$Pr[S] = \sum_{k=0}^{n} Pr[S|A_k]Pr[A_k]$$

From problem 1, it can be seen that the $Pr[A_k] = \binom{n}{k}\left(\frac{1}{2}\right)^{\mathrm{n}}$.

So the lower bounded of Pr[S] can be shown as:

$$Pr[S] \geq \sum_{k=0}^{n} \left(\frac{\sqrt{6\pi}}{e^2} \cdot \frac{2^{-k}}{\sqrt{k}}\right)\binom{n}{k}\left(\frac{1}{2}\right)^{\mathrm{n}}$$

$$= \frac{\sqrt{6\pi}}{e^2\sqrt{k}} \sum_{k=0}^{n} \binom{n}{k} (2)^{-k} \left(\frac{1}{2}\right)^{\mathrm{n}}$$

$$= \frac{\sqrt{6\pi}}{e^2\sqrt{k}} \left(\frac{3}{4}\right)^{n}$$

(v) To make the successful probability at least 1-1/n, restart the Random Guess 3-SAT multiple times. If the probability of success for each independent run is p. So, the probability that no solution is found after m times run is $(1-p)^m$. This question want this probability less than or equal to 1/n. So the formula can be structed as :

$$(1-p)^m \leq \frac{1}{n}$$

$$m \geq \frac{log\left(\frac{1}{n}\right)}{log(1-p)}$$

If every time the running time of Random 3-SAT is T, the total running time is O(mT).