ECE385: Homework 04

Farbod Mohammadzadeh
(1008360462)
Connor Ludwig
(1008333028)

06 Decemberr 2023
6 Pages

Problem 1

Part a)

The sum of the weights of the parts of A is the total weight of T_{opt}. This is because every edge in T_{opt} is part of exactly one C_i therefore sum of the weights of these parts, $\sum_{i=1}^{l} w(C_i)$, is the total weight of T_{opt}.

$$\therefore \sum_{i=1}^{l} w(C_i) = w(T_{opt})$$

Part b)

The weight of C_{max} is greater than or equal to the average part weight.

$$\therefore w(C_{max}) \geq \frac{\sum_{i=1}^{l} w(C_i)}{l}$$

Part c)

Consider $P = \frac{A}{C_{max}}$. We want to derive a lower bound for the quantity $P \leq 2(1 - 1/l)w(T_{opt})$ using the answers to the questions a-b.

Since A is the total weight of the minimum Steiner tree T_{opt}, and C_{max} is the weight of the largest part, we can express A as $A = \sum_{i=1}^{l} w(C_i) \geq l \cdot w(C_{max})$. Therefore, $w(C_{max}) \leq \frac{A}{l}$.

Now, we have $P = \frac{A}{C_{max}} \geq \frac{A}{\frac{A}{l}} = l$. Rearranging, we get $l \leq P$.

Using the result from Part b, $w(C_{max}) \leq \frac{A}{l}$, we have $P \cdot w(C_{max}) \leq A$. Substituting $P \leq l$ into this inequality, we get $l \cdot w(C_{max}) \leq A$.

Now, we can use the fact that $A \geq l \cdot w(C_{max})$ to derive a lower bound for P:

$$P = \frac{A}{C_{max}} \leq \frac{A}{\frac{A}{l}} = l$$

$$P \leq l$$

Consider $P = \frac{A}{C_{max}}$. Using the results from parts a and b, we can say that $A = \sum_{i=1}^{l} w(C_i)$ and C_{max} is the largest among C_i's. Therefore, P is at most twice the average part weight, i.e., $P \leq 2 \cdot \frac{\sum_{i=1}^{l} w(C_i)}{l}$. Substituting the result from part a, we get $P \leq 2(1 - 1/l)w(T_{opt})$.

Part d)

The weight of P is at least half the weight of T_{opt}.

$$\therefore \; w(P) \geq \frac{1}{2} w(T_{opt})$$

Part e)

Consider the graph G_4 obtained in step 5 of the algorithm. In G_4, we iteratively delete all leaves that are not vertices in R. Therefore, the weight of G_4 is less than or equal to the weight of the minimum Steiner tree T_{opt}.

$$w(G_4) \leq w(T_{opt})$$

In the algorithm, G_1 is the complete distance graph, and G_2 is a minimum spanning tree of G_1. This means that the weight of G_2 is at most the weight of any spanning tree of G_1. Therefore, $w(G_2) \leq w(T_{opt})$.

Part f)

Now, combine the results from parts c, d, and e:

$$w(P) \geq w(T_{opt}) - \frac{\sum_{i=1}^{l} w(C_i)}{l} \geq \frac{1}{2} w(T_{opt})$$

Combine this with the result from part e:

$$w(P) \geq \frac{1}{2} w(T_{opt}) \geq \frac{1}{2} w(G_4)$$

Now, the weight of G_4 is related to the weight of T_{approx} by the factor of 2 (as mentioned in the algorithm):

$$w(T_{approx}) \leq 2w(G_4)$$

Combining the above inequalities:

$$w(T_{approx}) \leq 2w(G_4) \leq 4w(P)$$

Now, substitute the bound for P from part c:

$$w(T_{approx}) \leq 4w(P) \leq 4 \left(w(T_{opt}) - \frac{\sum_{i=1}^{l} w(C_i)}{l} \right)$$

Simplify the expression:

$$w(T_{approx}) \leq 4w(T_{opt}) - 4 \cdot \frac{\sum_{i=1}^{l} w(C_i)}{l}$$

Divide by 2:

$$w(T_{approx}) \leq 2w(T_{opt}) - 2 \cdot \frac{\sum_{i=1}^{l} w(C_i)}{l}$$

Now, notice that $\sum_{i=1}^{l} w(C_i) = w(T_{opt})$:

$$w(T_{approx}) \leq 2w(T_{opt}) - 2 \cdot \frac{w(T_{opt})}{l}$$

Factor out $w(T_{opt})$:

$$w(T_{approx}) \leq 2(1 - \frac{1}{l})w(T_{opt})$$

This completes the proof:

$$w(T_{approx}) \leq 2(1 - \frac{1}{l})w(T_{opt})$$

Combining the results from parts c, d, and e, we have:

$$w(T_{opt}) \leq w(G_2) \leq w(P) \leq 2(1 - 1/l)w(T_{opt})$$

This completes the proof. We have shown that $w(T_{opt}) \leq 2(1 - 1/l)w(T_{opt})$, which is the statement to be proven.

Problem 2

Part a)

The certificate is the graph $G = (V, E)$ with at least k edges that is a subgraph of both G_1 and G_2.

To verify that this is a valid solution, iterate over every vertex and edge in G to check if the vertex/edge belongs to both G_1 and G_2. Additionally, while performing this operation, sum the number of edges in G to verify that there are at least k edges.

Using a linear search, each vertex can be verified in $O(|V_1 + V_2|)$ time, similarly, each edge can be verified in $O(|E_1 + E_2|)$ time. There are $O(|V|)$ vertices, and $O(|E|)$ edges, putting this all together, the runtime can be computed to be:

$$O(|V|) * O(|V_1 + V_2|) + O(|E|) * O(|E_1 + E_2|) \tag{1}$$
$$= O(|V|\,|V_1 + V_2| + |E|\,|E_1 + E_2|) \tag{2}$$

Therefore, a given solution can be verified in polynomial time, meaning that $B \in NP$.

Part b)

Let the input to problem A be G'. The output is true if G' contains a clique of size k. Let G_1 be a complete graph of with k' vertices such that $k\prime \leq n$, and $G' = G_2$ be any graph with n vertices, where G_1 and G_2 are the inputs to problem B, additionally stipulate that the minimum number of edges for B is $k = \binom{k\prime}{2}$. Pass G_2 as the input to A, if A returns true, it means that there exists a clique of size k in G_2.

Creating the graph G_1 takes $O(V + E)$ time. $O(E) = O(V^2)$ because the graph is fully connected. Therefore, this operation takes $O(V + V^2) = O(V^2)$ time, which is polynomial time.

A clique of size k' is a complete graph with $k = \frac{k\prime(k\prime-1)}{2} = \binom{k\prime}{2}$ edges, meaning that if A returns true, G_1 must be a subgraph of G_2 and B returns true as well.

With the described inputs, if B returns true, it means that G_2 must have a subgraph that is a complete graph with $k = \frac{k\prime(k\prime-1)}{2} = \binom{k\prime}{2}$ edges which is the same as a clique of size k', meaning that A will also return true.

Therefore, $A \leq_p B$, meaning that B is $NP - hard$. Because $B \in NP$, and B is $NP - hard$, B is $NP - complete$.

Problem 3

Part a)

The certificate is the collection of numbered items $C = w_j,\ v_j \subseteq S$ such that $\sum_j w_{j\in C} \leq W$ and $\sum_j v_{j\in C} \geq V$.

To verify this is a valid solution, iterate over all j to compute the sum of all w_j and the sum of all v_j, then compare these sums to the values W and V respectively to determine if the solution is valid.

This operation iterates through a list of items a singular time, meaning the operation takes linear time. The number of items in the list is upper bound by the total number of items in the set S, which is n. Giving a runtime of $O(n)$.

Therefore, a given solution can be verified in polynomial time, meaning that $B \in NP$.

Part b)

Reduce a to b if x answers a, $f(x)$ answers b.

Let the set $S = a_1, a_2, \ldots, a_n$ be the input passed to problem A. The output is true if $\sum_i a_{i\in C} = k$ for some $C \subseteq S$. Let $w_i = v_i = a_i\ \forall i \in n$ be the inputs to problem B. Additionally, stipulate that $W = V = k$.

This operation is a linear operation through the set of items, meaning that it can be done in O(n) time, which is polynomial time.

If A returns true, it means that there exists a set C such that $\sum_i a_{i\in C} = k$, letting problem B take the same inputs as A, the following result is also obtained

$$\sum_i a_{i\in C} = \sum_i w_{i\in C} = W = k = V = \sum_i v_{i\in C}$$

Because equality still satisfies the condition of problem B, this means that if A returns true, B must also return true.

If B returns true, it requires that $\sum_i w_{i\in C} \leq W,\ \sum_i v_{i\in C} \geq V$, because W and V are equal to each other and k, this can be stated as $\sum_i v_{i\in C} \leq k \leq \sum_i w_{i\in C}$. We also know that, $\sum_i a_{i\in C} = \sum_i v_{i\in C} = \sum_i w_{i\in C}$, substituting in these value, the following statement is obtained, $\sum_i a_{i\in C} \leq k \leq \sum_i a_{i\in C}$, this tightly bounds k, yielding the result $\sum_i a_{i\in C} = k$. Therefore, if B returns true, A must also return true.

Therefore, $A \leq\ _p B$, meaning that B is $NP-hard$. Because $B \in NP$, and B is $NP-hard$, B is $NP-complete$.