Advanced Algorithms

Homework 2

Ari Feiglin

2.1 Exercise

Consider the following metric TSP problem:

- Construct an undirected graph $G = (V, E, w)$ which is a path of $n = 2k + 1$ vertices, such that for every $1 \leq i < n$, $(v_i, v_{i+1}) \in E$, $w(v_i, v_{i+1}) = 1$.
- Alter G such that for vertices two steps apart, $w(v_i, v_{i+2}) = 1 + \varepsilon$ for some $\varepsilon > 0$.
- Define $G' = (V, E)$ with metric d defined to be $d(u, v) = \delta_G(u, v)$.

(1) Show and compare the results of the twice-MST and Christofide's algorithms to the optimal solution's cost.

(2) Describe a function $f(n)$ such that when $\varepsilon = f(n)$, the analysis of Christofide's algorithm becomes tight as n grows.

(1) An MST of G' must connect n vertices, so it must have $n-1$ edges, and thus an MST of G' is the original path. So our MST is $v_1, v_2, \dots, v_n$ and so an Euler tour on the doubled MST after finding shortcuts is $v_1 \to v_2 \to \cdots \to v_n \to v_1$. This has a weight of

$$\sum_i d(v_i, v_{i+1}) + d(v_n, v_1) = n - 1 + d(v_n, v_1)$$

The lightest path from v_n to v_1 in G is $v_{2k+1} \to v_{2k-1} \to \cdots \to v_3 \to v_1$ which has a weight of $k(1+\varepsilon)$. Thus the weight of the lightest path, and thus $d(v_n, v_1)$, is

$$2k + 1 - 1 + k(1+\varepsilon) = k(3+\varepsilon) = \frac{n-1}{2}(3+\varepsilon) = n \cdot \frac{3+\varepsilon}{2} - \frac{3+\varepsilon}{2}$$

For Christofide's, the only vertices in the MST with odd degree are v_1, v_n. The minimum distance between them is $k(1+\varepsilon)$, as proven before. And an Euler tour in $T + M$ is just $v_1 \to \cdots \to v_n \to v_1$, which has weight $k(3+\varepsilon)$. So twice-MST and Christofide's have the same approximation here.

The optimal solution only uses edges of size 1 and $1+\varepsilon$, as otherwise we would have skipped a vertex for nothing. The optimal cycle is then

$$(v_1 \to v_3 \to \cdots \to v_n) \to (v_{n-1} \to v_{n-2} \to \cdots \to v_2) \to v_1$$

Which has a weight of

$$k(1+\varepsilon) + 2 + (k-2)(1+\varepsilon) = (1+\varepsilon)(2k-2) + 2 = (1+\varepsilon)(n-3) + 2 = n(1+\varepsilon) - 1 - 3\varepsilon$$

(2) We want

$$n\frac{3+\varepsilon}{2} - \frac{3+\varepsilon}{2} = \frac{3}{2}\big(n(1+\varepsilon) - 1 - 3\varepsilon\big)$$
$$\varepsilon n = 4\varepsilon$$

So there is no such $f(n)$.

2.2 Exercise

Consider the greedy algorithm we discussed in class for Set-Cover. Describe a family of instances (for arbitrarily large k) such that the analysis is tight.

Let us take k to be a power of 2, then we define $X = [1, 2k]^2$ (the set of all integer points (x_1, x_2) where $x_i \in [1, 2k]$). Then let us define the following the sets

$$S_1 = [1, 2k] \times [1, k], \quad S_2 = [1, 2k] \times [k, 2k]$$

Then let us define a sequence of values $a_0 = 0, a_1, \dots$ and sets

$$A_n = [a_{n-1}, a_n] \times [1, 2k]$$

The optimal solution is 2, but we want at each step to choose a set A_n. Let

$$U_n = X \setminus \bigcup_{i<n} A_n X \setminus [1, a_{n-1}] \times [1, 2k] = [a_{n-1} + 1, 2k] \times [1, 2k]$$

then we want A_n to maximize $S \cap U_n$. Note that the size of $S_i \cap U_n$ is $(2k - a_{n-1} - 1)k$, and the size of $A_n \cap U_n$ is $(a_n - a_{n-1} - 1)2k$, so we want

$$2k^2 - a_{n-1}k - k \leq 2a_n k - 2a_{n-1}k - 2k \iff 2k + a_{n-1} + 1 \leq 2a_n \iff a_n \geq k + \frac{a_{n-1}}{2} + \frac{1}{2}$$

So let us define $a_n = k + a_{n-1}/2 + 1/2$, and solving this gives

$$a_n = 2k + 1 - \frac{2k+1}{2^n}$$

In order for $a_n \leq 2k$ we must have

$$1 \leq \frac{2k+1}{2^n} \iff n \leq \log_2(2k+1)$$

So we have that the optimal solution is $\log_2(2k+1)$ as required.

2.3 Exercise

Consider the following bin packing algorithm: given an item, pack it into the last opened bin. If this is impossible, open a new bin. Show that $|\text{alg}| \leq 2|\text{opt}| - 1$.

Let $U = \{\mu_1, \dots, \mu_n\}$ be our objects, with corresponding weights $\mathsf{s}(\mu_1), \dots, \mathsf{s}(\mu_n)$. Recall that

$$\sum_{i=1}^{n} \mathsf{s}(\mu_i) \leq |\text{opt}|$$

let $B_1^A, \dots, B_m^A$ be the bins used by any algorithm A, then

$$\sum_{i=1}^{n} \mathsf{s}(\mu_i) = \sum_{i=1}^{m} \mathsf{s}(B_i^A) \leq \sum_{i=1}^{m} 1 = m$$

so in particular, $\sum_{i=1}^{n} \mathsf{s}(\mu_i) \leq |\text{opt}|$. Now let $B_1, \dots, B_m$ be the bins used by our algorithm, notice that $\mathsf{s}(B_i) + \mathsf{s}(B_{i+1}) > 1$ as otherwise the algorithm would've just combined the two bins. So

$$2\sum_{i=1}^{m} \mathsf{s}(B_i) = \sum_{i=1}^{m-1} \big(\mathsf{s}(B_i) + \mathsf{s}(B_{i+1})\big) + \mathsf{s}(B_1) + \mathsf{s}(B_n) > m - 1 + \mathsf{s}(B_1) + \mathsf{s}(B_n)$$

So we have that

$$m - 1 + \mathsf{s}(B_1) + \mathsf{s}(B_n) < 2|\text{opt}| \implies m + \varepsilon < 2|\text{opt}| + 1 \implies m + \varepsilon \leq 2|\text{opt}| \implies m \leq 2|\text{opt}| - 1$$

As required.

2.4 Exercise

Consider an alternative algorithm for the bin-packing algorithm where all the items have weight at least δ. In this algorithm, we convert our set of items U to a new set U' where for every $u \in U$ we add an item u' to U' where $w_{u'}$ is w_u rounded up to the nearest multiple of $\frac{\delta}{k}$. I.e. $w_{u'} = \frac{\delta}{k} \cdot \left\lceil \frac{k}{\delta} w_u \right\rceil$.

(1) Show that $|\text{opt}(U)| \leq |\text{opt}(U')|$.

(2) Show that $|\text{opt}(U')| \leq t|\text{opt}(U)| + O(1)$ for as small of a t as possible.

(1) Since $w_u \leq w_{u'}$ if $B_1, \dots, B_m$ is a valid allocation of U', it is also a valid allocation of U. Thus the optimal solution to U' is a solution to U.

(2) Notice that

$$w_{u'} \leq \frac{\delta}{k} \cdot \left(\frac{k}{\delta} w_u + 1 \right) = w_u + \frac{\delta}{k}$$

and so now suppose that opt gives the bins $B_1, \dots, B_m$, we'd have that

$$s(B_i') \leq s(B_i) + \frac{\delta}{k} \cdot |B_i| \leq 1 + \frac{\delta}{k} \cdot |B_i|$$

and so if we require that $k \geq |B_i|$ for $1 \leq i \leq m$ we have that $s(B_i') \leq 1 + \delta$. Now for each B_i such that $s(B_i') > 1$ we can remove the minimal element (which is greater than δ), and this will reduce the bin's weight to at most 1. This minimal element's original weight is at most $\frac{1}{2}$ (since otherwise, it must be equal to 1, and we can just require that $\frac{k}{\delta} \in \mathbb{N}$ so $w' = w = 1$, and so $s(B_i) = 1$). If it is $\frac{1}{2}$, then this means that there is one other element in the bin which is also $\frac{1}{2}$, but then $w' = \frac{\delta}{k} \left\lceil \frac{k}{2\delta} \right\rceil$ and we can assume $\frac{k}{2\delta} \in \mathbb{N}$. So $w' = \frac{1}{2}$ and so $s(B_i) = 1$. Thus we have that $w < \frac{1}{2}$ and $w' \leq w + \frac{\delta}{k}$, and we want $w' \leq \frac{1}{2}$, so we require $w + \frac{\delta}{k} \leq \frac{1}{2}$, so $k \geq 2\delta \frac{1}{1-2w}$. Thus we also require

$$k \geq 2\delta \max_{w_i < \frac{1}{2}} \frac{1}{1 - 2w_i}$$

and then we have that $w' \leq \frac{1}{2}$ for all of the weights which we removed, and thus we can pack two weights into a single box. So for every two boxes, we create one extra box, thus

$$|\text{opt}'| \leq 1.5|\text{opt}| + O(1)$$