601.435 / 601.635 Approximation Algorithms **Spring 2024**
Homework #2 **Due:** February 29, 2024, 9:00am

Reminder: you may work in groups of up to three people, but must write up solutions entirely on your own. Collaboration is limited to discussing the problems – you may not look at, compare, reuse, etc. any text from anyone else in the class. Please include your list of collaborators on the first page of your submission. Many of these problems have solutions which can be found on the internet – please don't look. You can of course use the internet (including the links provided on the course webpage) as a learning tool, but don't go looking for solutions.

Please include proofs with all of your answers, unless stated otherwise.

Student: Ha Manh Bui (hbui13@jhu.edu)

1 Min-cost bounded path (50 points)

Suppose are are given a directed acyclic graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, a cost function $c : E \to \mathbb{N}$, a length function $\ell : E \to \mathbb{N}$, two nodes $s, t \in V$, and a length bound L. Our goal is to compute the minimum cost $s \to t$ path whose length is at most L. In other words, find the path P from s to t with $\sum_{e \in P} \ell(e) \leq L$ which minimizes $\sum_{e \in P} c(e)$.

(a) (25 points) Give a pseudopolynomial-time exact algorithm for this problem (i.e., an algorithm which solves the problem but runs in time polynomial in n, m and $\max_{e \in E} c(e), \max_{e \in E} \ell(e)$).

(b) (25 points) Using your algorithm from the previous part, give an FPTAS (a $(1+\varepsilon)$-approximation which runs in time polynomial in the size of the instance and $1/\varepsilon$, i.e., polynomial in $n, m, \log(\max_{e \in E} c(e)), \log(\max_{e \in E} \ell(e))$, and $1/\varepsilon$). Hint: Think about "guessing" the maximum cost of any edge that appears in the optimal path.

(a) The idea of Alg. 1 is finding the shortest path from $i, \forall i \in V$ to t with some cost. Then, return the smallest cost from s that satisfies the length bound L.
Notation. For all $i, j \in V$, let $c(i.j)$ be the cost from i to j and $l(i.j)$ is the length from i to j. Denote $out(i)$ be all node in V s.t. its destination is node i. Let $M = \max_{e \in E} c(e)$ and $f(i, v)$ be the shortest path from i to t with cost v. Following this notation, we obtain Alg. 1, which solves the problem and runs in time polynomial in n, m and $\max_{e \in E} c(e), \max_{e \in E} \ell(e)$.
Proof: By the notation, using dynamic programming, $\forall i \in V$ and $0 < v < mM$, we have

$$f(i,v) = \begin{cases} 0 & \text{if } i = t, v = 0 \\ \infty & \text{if } i = t, v > 0 \text{ or } i \neq t, v = 0 \\ \min(\{f(k,v) \text{ if } c(i,k) > v, \ell(i,k) + f(k, v - c(i,k) \text{ otw} | \forall k \in out(i))\}) & \text{if } i \neq t, v > 0. \end{cases}$$

We can see that $\forall v \in (0, nM)$, $f(s, v) \leq L$ is the feasible solution. Hence, we can solve the problem by finding the smallest v such that $f(s, v) \leq L$ (using extra $\mathcal{O}(mM)$ times). From Alg. 1, we can compute the function $f(i, v)$ in time $\mathcal{O}(n^2 mM)$, since there are at most n options for i, at most mM options for v, and evaluating a single table entry takes at most $\mathcal{O}(n)$ times. Because of the factor of M, we obtain the runtime of the Alg. 1 is time polynomial in n, m and $\max_{e \in E} c(e), \max_{e \in E} \ell(e)$.
■

Algorithm 1 A pseudopolynomial-time exact algorithm

Init: Matrix $f = [n, mM]$, Queue *vist_node*
Set $f(t, 0) = 0$, $f(t, v : 1 \to mM) = \infty$ ▷ $\mathcal{O}(mM)$
Push $out(t)$ to *vist_node*
while *vist_node* $\neq \emptyset$ **do**
  Pop i from *vist_node*
  Push $out(i)$ to *vist_node*
  Set $f(i, 0) = \infty$
  for $v : 1 \to mM$ **do**
    Set *length_path* = []
    for $k \in out(i)$ **do**
      if $c(i, k) > v$ **then**
        Add $f(k, v)$ to *length_path*
      else
        Add $\ell(i, k) + f(k, v - c(i, k))$ to *length_path*
      end if
    end for ▷ $\mathcal{O}(n)$
    Set $f(i, v) = \min(length\_path)$ ▷ $\mathcal{O}(n)$
  end for ▷ $\mathcal{O}(nmM)$
end while ▷ $\mathcal{O}(n^2mM)$
Return: $v^* = \arg\min_v\{f(s, v)|f(s, v) \le L\}$ ▷ $\mathcal{O}(mM)$

(b) Let $\delta = \frac{\epsilon M}{m \log(M)}$ and round $c'(e) = \lceil \frac{c(e)}{\delta} \rceil, \forall e \in E$ and $M' = \max_{e \in E} c'(e)$. Then, apply Alg. 1 by substituting c' for c, we obtain an FPTAS (a $(1 + \varepsilon)$-approximation which runs in time polynomial in the size of the instance and $1/\varepsilon$, i.e., polynomial in $n, m, \log(\max_{e \in E} c(e))$, $\log(\max_{e \in E} \ell(e))$, and $1/\varepsilon$).

Proof: Due to substituting c' for c in Alg. 1, using the proof for question (a), we obtain the new algorithm is at most

$$\mathcal{O}(n^2 mM') = \mathcal{O}(n^2 m \lceil \frac{M}{\delta} \rceil) = \mathcal{O}(n^2 m \lceil \frac{m \log(M)}{\epsilon} \rceil) = \mathcal{O}(n^2 m^2 \log(M) \frac{1}{\epsilon}) \tag{1}$$

On the other hand, let ALL, P_A be the solution returned from this new algorithm and its corresponding path, and OPT, P_O is the optimal solution from Alg. 1 and its corresponding path. Then, firstly, we still have ALL as a feasible solution since we do not change the length function ℓ, the total length in P_A must be at most L. Secondly, since $c(e) \le \delta c'(e)$, $\forall e \in E$, we get

$$ALL = \sum_{e \in P_A} c(e) \le \delta \sum_{e \in P_A} c'(e) \le \delta \sum_{e \in P_O} c'(e), \tag{2}$$

combining with the fact that $c'(e) \le \frac{c(e)}{\delta} + 1$, we obtain

$$\delta \sum_{e \in P_O} c'(e) \le \delta \sum_{e \in P_O} \left(\frac{c(e)}{\delta} + 1 \right) = \sum_{e \in P_O} c(e) + \delta |P_O| \le OPT + \delta(m \log(M)) = OPT + \epsilon M \le (1 + \epsilon) OPT, \tag{3}$$

i.e., the cost of the algorithm $ALL = \sum_{e \in P_A} c(e)$ is at most $(1 + \epsilon)OPT$. Combining with Eq. 1, we obtain a $(1 + \varepsilon)$-approximation which runs in time polynomial in the instance size and $1/\varepsilon$. ■

2 Min-degree Steiner Tree (50 points)

The Min-degree Steiner tree problem is the same as the Steiner tree problem, except instead of minimizing total cost our goal is to minimize the maximum degree. More formally, we are given an undirected graph $G = (V, E)$ and a subset $D \subseteq V$ of terminals. The goal is to find a tree T in G which spans all of D (but not necessarily all of V) and which minimizes the maximum degree.

Show how to modify the local search algorithm from class for min-degree spanning tree to get a local search algorithm for min-degree Steiner tree that runs in polynomial time and returns a tree with maximum degree at most $2\Delta^* + \log n$ (where Δ^* is the maximum degree of the optimal tree). If you're off by $+O(1)$ (like we were in class) that's OK, i.e., a bound of the form $2\Delta^* + \log n + c$ is OK for constant c.

Hints:

(1) You will need to slightly redefine a u-improvement, since adding a single edge might not create a fundamental cycle anymore. What kind of structure "acts" in a way equivalent to a non-tree edge in spanning trees?

(2) You might want to use the following structural graph theoretic result: in any tree with n nodes in which all non-leaves have degree at least d, the number of leaves is at least $\frac{d-2}{d-1}n$. You may use this without proof, although it's a good idea to convince yourself that it's true.

(3) You will also have to change the potential function when analyzing the running time. How can you change it so that the same basic idea from class works?

Let $E \setminus T$ be set of edges E exclude edges in tree T, then define a u-improvement as follows: let $u \in T$, then (u, E') is a u-improvement if exist edge $\{u, x\}$ on the fundamental cycle of non-tree path $E' \subseteq E \setminus T$ s.t. we can swap E' for $\{u, x\}$ to get T' satisfy $\max_{i \in T'}\{d_{T'}(i)\} \leq d_{T'}(u) = d_T(u) - 1$.

Algorithm 2 Local search for Min-Degree Steiner Tree

Find a Steiner tree T in G which spans all of D
while There is u-improvement with $d_T(u) \geq \max_{v \in T} d_T(v) - \log n$ for the current T **do**
  Do the improvement
end while
Return: T

Remark: Since we start with finding a Steiner tree T in G which spans all of D, then do a local search on the fundamental cycle to update T, we can see T is still the feasible solution.
Theorem: The running time of Alg. 2 is polynomial.
Proof: The step of finding a Steiner tree T in $G = (V, E)$ which spans all of $D \subseteq V$ is polynomial in $n = |V|$ by computing a minimum spanning tree on the graph induced by the terminals D. The local search step is also polynomial per iteration because only needs to find a polynomial number of non-tree path $E' \subseteq E \setminus T$ and check the maximum degree of all vertices $i \in T'$ in the potential tree T'. Therefore, assuming a polynomial number of iterations in the local search, we obtain the Alg. 2 runs in polynomial time. ■
Theorem: Let T be an output of Alg. 2 and $\Delta(T) = \max_{v \in T} d_T(v)$, then $\Delta(T) \leq 2\Delta^* + \log n$.
Proof: Recall some results from the class works, using the fact that in any tree with n nodes in which all non-leaves have a degree at least d, the number of leaves is at least $\frac{d-2}{d-1}n$, then:

(a) Consider partitioning G into pieces $V_1, \cdots, V_k$ and let $E' \subseteq E$ be the edges with endpoints in different parts of the partition. Let $V' \subseteq V$ be a vertex cover for E' (every edge in E' has at least one endpoint in V'). Then $\Delta^* \geq \frac{k-1}{|V'|}$.

(b) Let S_i be the nodes with degree at least i in T, then there exists an $i \geq \Delta(T) - \log(n)$ s.t. $|S_{i-1}| \leq 2|S_i|$.

(c) Let E_i^T be edges of T incident on nodes in S_i, then $|E_i^T| \geq (i-1)|S_i| + 1$.

(d) Let $e = \{x, y\} \in E_i^{T - E_i^T}$, then either x or y in S_{i-1}.

Let i^* be the value from the result (b) applied to T. Consider the partition of V defined by the components of $T - E_{i^*}^T$. There are $|E_{i^*}^T|$ components in the decomposition, and by the result (d), the set of vertices incident on edges that cross this partition is a subset of S_{i^*-1}. Apply the result (a), we obtain

$$\Delta^* \geq \frac{|E_{i^*}^T| - 1}{|S_{i^*-1}|} \tag{4}$$

$$\geq \frac{(i^* - 1)|S_{i^*}|}{2|S_{i^*}|} \tag{5}$$

$$\geq \frac{i^* - 1}{2} \tag{6}$$

$$\geq \frac{\Delta(T) - \log n}{2}, \tag{7}$$

i.e., $\Delta(T) \leq 2\Delta^* + \log n$. ■