CS 580 Algorithm Design, Analysis, And Implementation
Fall 2021 Vassilis Zikas HW 5

Due Wednesday Nov. 17 at 11:59 p.m.

1. (50 points, 10 + 5 + 5 + 15 + 15) Consider the *shortest s-t path problem* in which the input is an undirected graph $G = (V, E)$ with nonnegative edge weights and vertices $s, t \in V$. The goal is to find the shortest path (minimum cost) *s*-*t* path in G.

   Consider the following primal and dual LP for the shortest *s*-*t* path problem. Here $S' = \{S \subseteq V : s \in S, t \notin S\}$ i.e. S' is the set of all *s*-*t* cuts in G. $\delta(S)$ represents edges in the cut S i.e. $\delta(S) = |\{(u,v) \in E : |\{u,v\} \cap S| = 1\}|$. The LPs are given below:

minimize	$\sum_{e \in E} c_e x_e$		maximize	$\sum_{S \in S'} y_S$	
subject to	$\sum_{e \in \delta(S)} x_e \geq 1$	$\forall S \in S'$	subject to	$\sum_{S \in S' : e \in \delta(S)} y_S \leq c_e$	$\forall e \in E$
	$x_e \geq 0$	$\forall e \in E.$		$y_S \geq 0$	$\forall S \in S'$

   Let $G : (V, E)$ be the input graph. Consider the following algorithm based on these LPs.

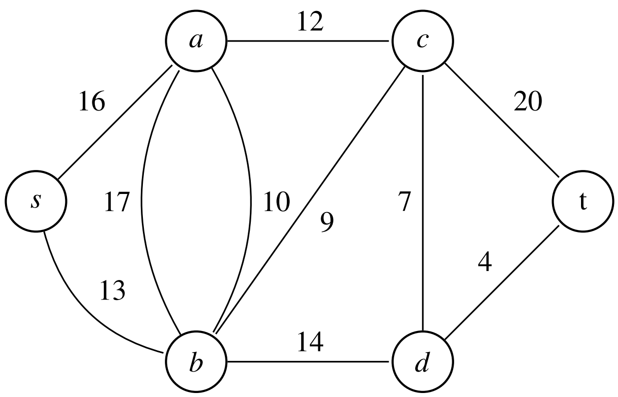
   - Initialize $y_S = 0$, $x_e = 0$ for all edges $e \in E$ and all cuts $S \in S'$.
   - **While** there is no *s*-*t* path formed in G using edges in $\{e : x_e > 0\}$ **do**
     Let $C \subseteq V$ be the connected component formed by edges whose $x_e = 1$ (C always contains s).[1]
     Increase y_C until there is an edge $e' \in \delta(C)$ such that $\sum_{S \in S' : e' \in \delta(S)} y_S = c_{e'}$
     Set $x_{e'} = 1$.
   - Return an *s*-*t* path in G using edges in $\{e : x_e > 0\}$.

   Let the path returned be denoted by P.

   (a) What is an interpretation of the primal and dual LP – explain in sentences, as was done in class for the Chicago vs. Detroit Pizza example? In particular, explain what each variable, the objective and each constraint means.

   (b) Consider the following graph. We execute the above algorithm on this graph. In the first iteration, some set C_1 is chosen, and some edge e_1 has $x_{e_1} = 1$ at the end of the first iteration. Similarly we choose sets C_2, C_3 and edges e_2, e_3 in the second and third iteration, respectively. What are $C_1, C_2, C_3, e_1, e_2, e_3$, as well as $y_{C_1}, y_{C_2}, y_{C_3}$?

[1] In the first step, when $x_e = 0$ for all e, C is just the node s.



(c) Prove that at any point of the algorithm the set F of edges with $x_e = 1$, i.e. $F_t = \{e \in E : x_e = 1\}$ at the end of iteration t of the algorithm, forms a tree .

(d) Prove that for any $S \in S'$, if $y_S > 0$ at the end of the algorithm, then $|P \cap \delta(S)| = 1$ (where P is the path returned by the algorithm).

(e) Prove that the given algorithm returns a shortest s-t path in G. You can use parts (c) and (d) as stated (even if you didn't provide a solution for them).

Answer: Let F denote the set of edges $\{e : x_2 = 1\}$. Notice that F changes at each iteration in the algorithm.

(a) Primal LP: We define variables x_e for each edge. x_e takes value 1 if it is part of a shortest path else 0. The first set of constraints say that we should select atleast 1 edge in each s-t cut. If not, then s and t will be connected by a path (formed by the edges whose x_e is set to 1).

Dual LP: We can interpret the variables y_S as shells in the cut S. Any path from s-t must cross this shell and incur cost y_S. The sum over all the shells that cover an edge cannot exceed the weight/capacity of the edge i.e. c_e. Any s-t path must cross all these shells and has length $\sum_S y_S$.

(b) At first iteration we add the edge with weight 13 and $y_C = 13$ where $C = \{s\}$. In the second iteration we add the edge with weight 16 and y_C where $C = \{s, b\}$ is 3. In the third iteration, we add the edge with weight 9 and set $y_C = 6$ where $C = \{s, a, b\}$.

(c) We can prove this using induction on the edges added to F. Suppose that at iteration i, F is a tree. Let C be the connected component containing s. Then, we add an edge that goes across the cut C. Therefore the edge added connects a vertex in C to a vertex not in C. Therefore, it cannot form a cycle.

(d) For purpose of contradiction, suppose that $|P \cap \delta(S)| > 1$. Then there must a subgraph P' of P such that the only the start and end vertex of P' lies in S. When y_S was increased F was a tree spanning vertices in S. $F \cup P'$ must contain a cycle because both the start and end vertices lie in S. Therefore, $F \cup P$ must contain a cycle which is not possible.

(e) Let d be the optimum of the primal LP i.e. d is the costs of the shortest s-t path. For any edge e in F, $c_e = \sum_{S:e\in\delta(S)} y_S$. Therefore the costs of edges in P are

$$\begin{aligned}
\sum_{e\in P} c_e &= \sum_{e\in P} \sum_{S:e\in\delta(S)} y_S \\
&= \sum_{S:s\in S, t\notin S} |P \cap \delta(S)| y_S && \text{(Rearranging summations)} \\
&= \sum_{S:s\in S, t\notin S} y_S \leq d && \text{(Weak duality)}
\end{aligned}$$

Since our path cannot have cost less than d, the cost of P must be d.

2. (50 points, 10 + 10 + 10 + 20 respectively) For each of the following problems either design and analyze an efficient algorithm (i.e. atleast polynomial time), or prove that they are NP-hard.

   (a) Given an undirected graph G, with positive edge weights, does G have a spanning tree of weight atmost 42? c

   (b) Given an undirected graph G, does G have a spanning tree with exactly 2 leaves?

   (c) Given an undirected graph G, does G have a spanning tree with maximum degree 2?

   (d) Given an undirected graph G, does G have a spanning tree with at most 42 leaves?

   Any vertex with degree one in a tree is a referred to as a leaf.

   Answer:

   (a) We calculate the minimum spanning tree of G. If weight of MST is less than or equal to 42 we output Yes else output No. To prove correctness, we can argue that G has a spanning tree of weight atmost 42 if and only if it has a minimum spanning tree of weight less than or equal to 42. For the forward direction, if there is a spanning tree of weight less than or equal to 42, then the minimum spanning tree must also have weight less than or equal to 42. Other direction follows similarly. Time complexity is $O(m \log n)$.

   (b) A (undirected) tree with exactly two leaves is a path. A spanning tree with two leaves is thus a Hamiltonian path by definition. Thus this problem is an equivalent characterization of the Hamiltonian Path problem, which is known to be at least as hard as SAT.

   (c) Any connected subgraph having maximum degree 2 is a path. Since a spanning tree is a connected subgraph, a spanning tree of degree two is a spanning path i.e. Hamiltonian path. Thus finding a spanning tree of degree at most 2 is at least as hard as solving SAT.

   (d) We reduce problem $2(b)$ to our problem.

   EDITED:
   Construction of $G' = (V', E')$. Given a graph instance $G = (V, E)$ for problem $2(c)$, we construct a new graph G' as follows:

   - Create a new vertex v.
   - Create 41 new vertices $u_1 \cdots u_{41}$ and connect them to v.
   - Connect v to all vertices in V.
   - Define V' as $V \cup \{v, u_1, \cdots u_{41}\}$ and E' as $E \cup$(the set of all newly added edges).

   We will prove the following: G has a spanning tree with at most 2 leaves if and only if G' has a spanning tree of at most 42 leaves.

   Proof. Assume that G has a spanning tree T with at most 2 leaves say ℓ_1 and ℓ_2. Then, T' defined as $T \cup \{(u, v), (v, u_1) \cdots (v, u_{41})\}$ such that $u \in V$ and $u \in \ell_1, \ell_2$ is a spanning tree of G' with at most 42 leaves.

   Now assume that G' has a spanning tree T' with at most 42 leaves. Then $\{u_1 \cdots u_{41}\}$ have to be leaves in T'. Removing $v, u_1 \cdots u_{41}$ and all incident edges from T' gives us a spanning tree of G with at most 2 leaves. □

   Note that all the reductions above can be done in polynomial time.