15-451/651 Algorithm Design & Analysis, Spring 2026

Homework #5 Solutions

---

1. (**, disjoint short paths) Let G be a directed graph with m edges whose edges have positive lengths. Let s be a source vertex and let $v_1, v_2, \ldots, v_k$ be other vertices given as input. Your goal is to return edge-disjoint paths $P_1, P_2, \ldots, P_k$, where P_i is from $s \to v_i$, and minimize the total sum of their lengths. Give an algorithm to do this in time at most $O(m^{10})$.

   Can you give a simpler algorithm if the paths are not forced to be edge-disjoint?

   Solution: We reduce to minimum-cost flow problem.

   Construction of graph that we compute minimum cost flow on. The graph is the same as the original graph. Give all edges capacity 1 (and lower capacity 0). Let the demand $d \in \mathbb{R}^V$ be defined as $d_s = -k$ and $d_{v_i} = 1$ for all $i = 1, 2, \ldots, k$. The cost on an edge is its length.

   Correctness. Here we will argue two things. First is that the flow can be decomposed exactly into the paths $P_1, P_2, \ldots, P_k$. We know that the optimal minimum-cost flow is integral and thus can be decomposed into paths between positive and negative demand vertices (without cancellation), plus cycles. Cycles have positive costs since all edges have positive lengths/costs, so cannot exist in an optimal solution.

   Since only s has negative demand, all paths start from s. Because the flows are integral, and $d_{v_i} = 1$, the k paths end at the v_is.

   Runtime. The new graph has size $O(m)$. It was mentioned in Lecture 15 that min cost flow can be solved in $O(m^{10})$ time.

   Not-disjoint paths. If the paths are not forced to be disjoint, you could have just run Dijkstra's algorithm to find shortest paths from s and sum up their lengths.

2. (**, multicommodity flow) Let G be a directed graph, and let $(s_1, t_1), \ldots, (s_k, t_k)$ be pairs of sources and sinks. We want to send one unit of flow for each pair $s_i \to t_i$ (**Important: the flows need not be integral.**) in a way that minimizes the maximum congestion on any edge. Formally, let $f^{(1)}, \ldots, f^{(k)}$ be the flows. Then the congestion of edge e is defined as

$$\text{cong}_e = \sum_{i=1}^{k} f_e^{(i)},$$

   where $f_e^{(i)}$ denotes the amount of flow on edge e in $f^{(i)}$. You want to find flows $f^{(i)}$, each routing one unit from s_i to t_i, that minimize $\max_e \text{cong}_e$.

   Write a linear program that captures this problem.

   Solution: We will construct a linear program to model the problem as follows:

   Variables:

   - $f_e^{(i)}$ for the amount of flow on edge e in $f^{(i)}$
   - C for the maximum congestion over the edges

   Objective:

   - minimize C

Constraints:

- $\sum_{e \text{ out of } v} f^{(i)} - \sum_{e \text{ into } v} f^{(i)} = \begin{cases} -1 & \text{if } v = s_i \\ 1 & \text{if } v = t_i \\ 0 & \text{else} \end{cases}$ $\forall$ edge e and vertex v (flow conservation)
- $\sum_{i=1}^{k} f_e^{(i)} \leq C$ $\forall$ edge e (congestion bound)
- $f_e^{(i)} \geq 0$ $\forall$ edge e and flow i (non-negativity)

This LP is correct since C is constrained to be at least the congestion on each edge, and since we are minimizing this value, our solution will be equal to the maximum congestion. Since the flow does not need to be integral, an LP will find the optimal solution.

3. (***, Hall's Lemma modified / Tutte-Berge formula on Bipartite Graphs) Let G be a bipartite graph with bipartition $V = L \cup R$, where L is the set of left vertices and R is the set of right vertices. Let $a = |L|$ and $b = |R|$ (L has a vertices and R has b vertices). Define a 2-matching of G to be a subset of edges $F \subseteq E(G)$ such that:

   (a) Every vertex in L is incident to at most 2 edges of F, and

   (b) Every vertex in R is incident to at most one edge of F.

   Prove that the maximum number of edges in a 2-matching is given by the following formula:

   $$2a - \max_{S \subseteq L}(2|S| - |N(S)|),$$

   where $N(S) \subseteq R$ is the set of neighbors in R of the vertices in S.

   Hint. Set up the computation of a 2-matching as a flow problem. Use the max-flow min-cut theorem, and use the structure of the minimum cut to build the set S.

   Solution: (by TAs)

   Construct the graph with vertices $L \cup R \cup \{s, t\}$, edges:

   (a) From s to each vertex in L with capacity 2,

   (b) from each vertex in R to t with capacity 1,

   (c) from $l_i \to r_i$ with capacity ∞.

   A $s \to t$ integer flow in this graph corresponds to a matching in the original graph: each veretx on either side is used at most once, and all edges used are from the original graph.

   We first justify that a finite cut exists to claim no ∞ capacity edges will be cut. The cut $(V \setminus \{t\}, \{t\})$ exists and is of capacity $|R|$, therefore no ∞ capacity edges will ever be cut in a min-cut. (alternatively, since the max matching size is finite, the min-cut value is also finite)

   Now consider some arbitrary but finite valued cut. We define S to be the vertices in L on the s side of the cut. Since no infite capacity edges are cut, we get that all of $N(S)$ are on this side of the cut as well. So the numbers of edges cut are at least:

   (a) the $|N(S)|$ edges from $R \to t$, and

   (b) the $a - |S|$ edges from $s \to L$ all of capacity 2.

So the size of the cut is at least

$$2(a-|N(S)|)+|N(S)|=2a-2|S|+|N(S)|$$

with equality when we set the vertices in R on the s side of the cut to exactly $N(S)$. So the minimum over all cuts is equal to

$$\min_{S\subseteq L}(2a-2|S|+|N(S)|)=2a-\max_{S\subseteq L}(2|S|-|N(S)|)$$

Solution: (by Yang)

We will first construct a graph to represent the 2-matching problem. We will construct the graph with the following vertices:

- source node s
- sink node t
- nodes l_i for each $l\in L$
- nodes r_i for each $r\in R$

We will also add the following edges:

- add an edge from s to each l_i with capacity 2
- add an edge from each r_i to t with capacity 1
- add an edge from l_i to r_j for each edge (l_i,r_j) with capacity 1

According to the max-flow min-cut theorem, the max flow is equal to the minimum s-t cut.

Consider a cut defined by placing a subset $S\subseteq L$ of the left vertices on the t side of the cut and some subset $T\subseteq R$ of the right vertices of the s side of the cut. Then the s side contains $\{s\}\cup(L\setminus S)\cup T$ and the t-side contains $S\cup(R\setminus T)\cup\{t\}$.

The edges crossing this cut (from s-side to t-side) are:

(a) $s\to l_i$ for each $l_i\in S$: contributes $2|S|$ since each edge has capacity 2.

(b) $r_j\to t$ for each $r_j\in T$: contributes $|T|$ since each edge has capacity 1.

(c) $l_i\to r_j$ for each $l_i\in L\setminus S$, $r_j\in R\setminus T$ with $(l_i,r_j)\in E$: contributes $|E(L\setminus S,R\setminus T)|$ since each edge has capacity 1.

So the cut value is:

$$\text{cut}(S,T)=2|S|+|T|+|E(L\setminus S,R\setminus T)|.$$

For a fixed S, we want the $T\subseteq R$ to minimize the cut. Each vertex $r\in R$ can independently be placed on either side: if $r\in T$ (placed on the s-side), it contributes 1 to the cut via the $r\to t$ edge; if $r\notin T$ (placed on the t-side), it contributes $\deg_{L\setminus S}(r)$ to the cut via the crossing edges from $L\setminus S$. So the optimal choice for each r contributes $\min(1,\deg_{L\setminus S}(r))$. This equals 1 if r has at least one neighbor in $L\setminus S$, and 0 otherwise. Therefore the sum counts exactly $|N(L\setminus S)|$, giving us:

$$\min_T \text{cut}(S,T)=2|S|+|N(L\setminus S)|.$$

Now substituting $S'=L\setminus S$ (so $|S|=a-|S'|$):

$$\min_{S\subseteq L}(2|S|+|N(L\setminus S)|)=\min_{S'\subseteq L}\big(2(a-|S'|)+|N(S')|\big)=2a-\max_{S'\subseteq L}\big(2|S'|-|N(S')|\big).$$

Since the max flow equals the maximum 2-matching size (each unit of flow through $s\to l_i\to r_j\to t$ corresponds to an edge in the 2-matching, with capacities enforcing the degree constraints), we conclude:

$$\text{max 2-matching}=2a-\max_{S\subseteq L}(2|S|-|N(S)|)$$