

# Homework 7: Network Flow and Linear Programming

Be sure to read the grading guidelines and style guidelines. Especially to see the suggested format for describing algorithms.

We sometimes describe how long are justifications or proofs are. These lengths are intended to help you estimate how much detail we're expecting, you should not take those estimates as hard length-limitations.

Our solutions for any individual problem will fit in approximately one page or less.

You are allowed (and encouraged!!) to collaborate with each other. Brainstorming is much easier to do in a group than alone! But you must follow the collaboration policy (which includes needing to write your submission on your own).

You will submit to Gradescope; we will have a different box for each problem, so please give yourself extra time to submit.

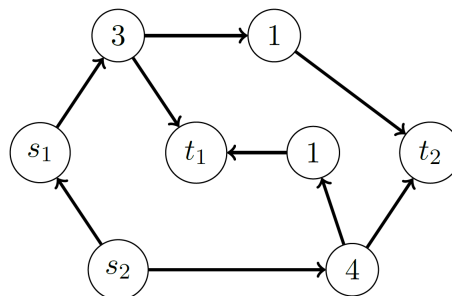
## 1. Dualling LPs [10 points]

Write a linear program in standard form for the following optimization problem and then write its LP dual: Maximize  $3x_1 - 2x_2 + x_3$  such that  $x_1 + 2x_2 - x_3 = 4$ ,  $5x_2 + 2x_3 \geq 6$ , and  $4x_3 - x_1 + x_2 \leq 7$ , and none of the  $x_i$  are negative.

## 2. $k$ -Flow [25 points]

Suppose that you are given an directed graph  $G$  with  $k$  vertices  $s_1, \dots, s_k$  as source vertices and another  $k$  vertices  $t_1, \dots, t_k$  as destination vertices for some  $k \geq 1$  and that no source vertex is directly connected to a sink vertex. Suppose further that each non-source, non-destination vertex  $v$  of  $G$  has an integer capacity  $c_v \geq 0$  which shows the maximum amount of flow that can go through  $v$ . The source and destination vertices have no capacity limit so each vertex either has an  $s_i$  label, a  $t_i$  label or a capacity label.

Design an algorithm that runs in polynomial time that determines the maximum total flow that can be sent from all sources to all sinks combined satisfying all capacity constraints of the vertices. (The amount of flow sent from  $s_i$  does *not* have to be the same as the amount received at  $t_i$ ; all we care about is the total flow.) For example, in the following picture the maximum amount of flow that can be sent from  $s_1, s_2$  to  $t_1, t_2$  is 7.



### 3. Allocating Cell Bandwidth [25 points]

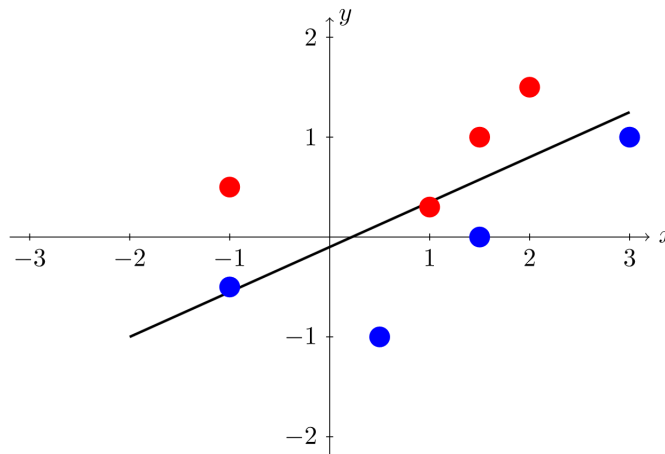
A service provider has deployed a wireless network in a city via a number of base station towers at fixed locations in the city. The stations were installed at different times so they have a variety of power/reach of their signal and can each support a variety numbers of clients. Each client they wish to serve is also assumed to be served via an antenna at a fixed and known location. The goal of the provider is to serve as many of their clients as possible by allocating their clients to talk with with specific based stations.

Suppose that the locations of each of the  $B$  base station towers and each of the  $n$  client antennas are at known coordinates in the  $xy$ -plane. Suppose that base stations 1 through  $B$  have signal radius  $r_1, \dots, r_B$ , respectively (that is, base station  $b$  can communicate with any client within Euclidean distance  $r_b$  of its location) and, respectively, can handle a load of up to  $L_1, \dots, L_B$  clients at the same time,

Give a polynomial-time algorithm that will figure out the maximum number of clients that the service provider can communicate with at a time, and an allocation of clients to base station towers that will achieve this maximum.

### 4. What Dividing Line? [25 points]

You want to solve the following linear classification problem using linear programs. You are given  $n$  red points  $(x_1, y_1), \dots, (x_n, y_n)$  and  $n$  blue points  $(x'_1, y'_1), \dots, (x'_n, y'_n)$ . You want to find a half-plane that separates the red points from the blue points such that the sum the errors is minimized. Namely, for  $a, b \in \mathbb{R}$  we define the error of each a point  $(x, y)$  as  $\max(ax + y - b, 0)$  and the error of a blue point  $(x', y')$  is  $\max(b - ax' - y', 0)$ . Write a linear program to find  $a, b$  such that the sum of the errors over all points is minimized. You don't need a proof for this problem.



## 5. Even More Efficient Matchmaking [Extra credit]

We saw how repeatedly augmenting along shortest paths (given by a BFS each time) produced a better runtime analysis for general network flow. In this problem you will do something similar in the special case of finding a maximum matching in a bipartite graph  $G = (V, E)$  where the two sides of  $V$  are  $X$  and  $Y$ ,  $|V| = n$  and  $|E| = m$ . The idea here will be to augment along many shortest paths at the same time. In order for this to work, the paths better not share any edges. Your algorithm will add the slightly stronger condition that the paths do not share any vertices and will include as many shortest paths as possible at each step.

More precisely, at each round the new algorithm will simultaneously augment along all paths in some *maximal set*  $P_{short}$  of vertex-disjoint shortest augmenting paths in the flow graph associated with the bipartite matching problem. (In other words, if  $d = d_f(s, t)$  is the length of the shortest augmenting path in the residual graph  $G_f$  then every path in  $P_{short}$  has length  $d$ , no two paths in  $P_{short}$  share any vertices other than  $s$  or  $t$  and any other  $st$ -path of length  $d$  in  $G_f$  shares at least one vertex other than  $s$  and  $t$  with some path in  $P_{short}$ .)

- (a) Show how to find a set  $P_{short}$  and do all the augmentations on its paths to get a new flow  $f'$  in time  $O(m+n)$ .
- (b) Prove that  $d_{f'}(s, t) \geq d_f(s, t) + 2$ .
- (c) Given two matchings  $M'$  and  $M$  on graph  $G$  we can define the symmetric difference,  $M' \oplus M$ , of  $M'$  and  $M$  to be the graph consisting of edges that occur in exactly one of the two matchings.  $M' \oplus M$  consists of a collection of vertex-disjoint paths and cycles (why?).  
Show that if  $M'$  is a maximum matching and flow  $f$  corresponds to a matching of  $M$  then
  - i.  $M' \oplus M$  contains exactly  $|M'| - |M|$  vertex-disjoint odd-length paths,
  - ii. Any path of odd length  $k$  in  $M' \oplus M$  corresponds to an augmenting path of length  $k + 2$  in  $G_f$ .
  - iii. Use these two properties to prove that once all augmenting paths in  $G_f$  are of length at least  $k + 2$  then at most  $n/k$  additional augmentations will produce a maximum flow (and hence maximum matching).
- (d) Use the above analysis with a suitable value of  $k$  to show that this algorithm computes a maximum matching in  $O(m\sqrt{n})$  time