CS 170 HW 10

Due **2021-04-13, at 10:00 pm**

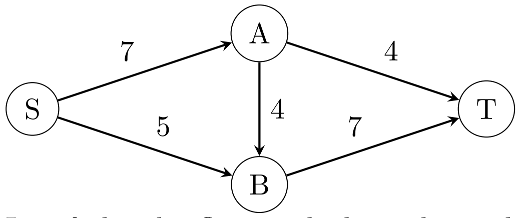
1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, write "none".

2 Max Flow, Min Cut, and Duality

In this exercise, we will demonstrate that LP duality can be used to show the max-flow min-cut theorem.

Consider this instance of max flow:



Let f_1 be the flow pushed on the path $\{S, A, T\}$, f_2 be the flow pushed on the path $\{S, A, B, T\}$, and f_3 be the flow pushed on the path $\{S, B, T\}$. The following is an LP for max flow in terms of the variables f_1, f_2, f_3:

$$\begin{aligned}
\max \quad & f_1 + f_2 + f_3 \\
& f_1 + f_2 \leq 7 && \text{(Constraint for } (S, A)\text{)} \\
& f_3 \leq 5 && \text{(Constraint for } (S, B)\text{)} \\
& f_1 \leq 4 && \text{(Constraint for } (A, T)\text{)} \\
& f_2 \leq 4 && \text{(Constraint for } (A, B)\text{)} \\
& f_2 + f_3 \leq 7 && \text{(Constraint for } (B, T)\text{)} \\
& f_1, f_2, f_3 \geq 0
\end{aligned}$$

The objective is to maximize the flow being pushed, with the constraint that for every edge, we can't push more flow through that edge than its capacity allows.

(a) Find the dual of this linear program, where the variables in the dual are x_e for every edge e in the graph.

(b) Consider any cut in the graph. Show that setting $x_e = 1$ for every edge crossing this cut and $x_e = 0$ for every edge not crossing this cut gives a feasible solution to the dual program.

(c) Based on your answer to the previous part, what problem is being modelled by the dual program? By LP duality, what can you argue about this problem and the max flow problem?

Solution:

(a) The dual is:

$$\begin{aligned} \min \quad & 7x_{SA} + 5x_{SB} + 4x_{AT} + 4x_{AB} + 7x_{BT} \\ & x_{SA} + x_{AT} \geq 1 \text{ - Constraint for } f_1 \\ & x_{SA} + x_{AB} + x_{BT} \geq 1 \text{ - Constraint for } f_2 \\ & x_{SB} + x_{BT} \geq 1 \text{ - Constraint for } f_3 \\ & x_e \geq 0 \quad \forall e \in E \end{aligned}$$

(b) Notice that each constraint contains all variables x_e for every edge e in the corresponding path. For any $s-t$ cut, every $s-t$ path contains an edge crossing this cut. So for any cut, the suggested solution will set at least one x_e to 1 on each path, giving that each constraint is satisfied.

(c) The dual LP is an LP for the min-cut problem. By the previous answer, we know the constraints describe solutions corresponding to cuts. The objective then just says to find the cut of the smallest size. By LP duality, the dual and primal optima are equal, i.e. the max flow and min cut values are equal.

3 How to Gamble With Little Regret

Suppose that you are gambling at a casino. Every day you play at a slot machine, and your goal is to minimize your losses. We model this as the experts problem. Every day you must take the advice of one of n experts (i.e. a slot machine). At the end of each day t, if you take advice from expert i, the advice costs you some c_i^t in $[0,1]$. You want to minimize the regret R, defined as:

$$R = \frac{1}{T}\left(\sum_{t=1}^{T} c_{i(t)}^t - \min_{1\leq i\leq n} \sum_{t=1}^{T} c_i^t\right)$$

where $i(t)$ is the expert you choose on day t. Your strategy will be probabilities where p_i^t denotes the probability with which you choose expert i on day t. Assume an all powerful adversary (i.e. the casino) can look at your strategy ahead of time and decide the costs associated with each expert on each day. Let C denote the set of costs for all experts and all days. Compute $\max_C(\mathbb{E}[R])$, or the maximum possible (expected) regret that the adversary can guarantee, for each of the following strategies.

(a) Choose expert 1 at every step, i.e. $p_1^t = 1$ for all t.

(b) Any deterministic strategy, i.e. for each t, there exists some i such that $p_i^t = 1$.

(c) Always choose an expert according to some fixed probability distribution at every time step. That is, fix some $p_1, \ldots, p_n$, and for all t, set $p_i^t = p_i$.

What distribution minimizes the regret of this strategy? In other words, what is $\text{argmin}_{p_1,\ldots,p_n} \max_C(\mathbb{E}[R])$?

This analysis should conclude that a good strategy for the problem must necessarily be randomized and adaptive.

Solution:

(a) 1. Consider the case where the cost of expert 1 is always 1 and the costs of all the other experts are always 0. Thus $\sum_{t=1}^{T} c_{i(t)}^t = \sum_{t=1}^{T} c_1^t = T$, and $\min_{1\le i\le n} \sum_{t=1}^{T} c_i^t = 0$, so the regret is $R = \frac{1}{T}(T-0) = 1$.

(b) $\frac{n-1}{n}$. Consider the case where the cost of the chosen expert is always 1, and the cost of each other expert is 0. Let k be the least-frequently chosen expert, and let m_k be the number of times that expert is chosen. This will result in a regret of $\frac{1}{T}(T-m_k)$

Since the best expert is the one that is chosen least often, the best strategy will try to maximize the number of times we choose the expert that is chosen least often. This means we want to choose all the experts equally many times, so expert k is chosen in at most T/n of the rounds. Therefore, $m_k \le \frac{T}{n}$, thus the regret is at least $\frac{1}{T}(T-\frac{T}{n}) = \frac{n-1}{n}$.

(c) $(1-\min_i p_i)$. Like in part (a), the distribution is fixed across all days, so we know ahead of time which expert will be chosen least often in expectation. Let $k = \text{argmin}_i p_i$ be the expert with least cost. Let $c_k^t = 0$ for all t, and let $c_i^t = 1$ for all $i \neq k$ and for all t. This way, $\mathbb{E}\left[\sum_{t=1}^{T} c_{i(t)}^t\right] = T\left(\sum_{i\neq k} p_i \cdot 1 + p_k \cdot 0\right) = T(1-p_k)$. We also have $\min_i \sum_{t=1}^{T} c_i^t = 0$, so we end up with an expected regret of $\frac{1}{T}(T(1-p_k)-0) = 1-p_k$.

To minimize the expectation of R is the same as maximizing $\min_i p_i$, which is achieved by the uniform distribution. This gives us regret $\frac{n-1}{n}$ (this is the same worst case regret as in part (b)).

4 Global Mincut to Min s-t Cut

In class, we showed how to use maximum s-t flow to solve the minimum s-t cut. Now we ask you to consider the *global mincut* problem. Given a weighted, undirected graph G, the problem asks you to find a minimum weight set of edges whose removal disconnects the graph. (Note that there is no notion of s and t).

Show how to use maxflow or min s-t cut algorithms to solve this problem efficiently. Prove that your reduction would lead to the optimal solution. How many times do you need to invoke the maxflow or min s-t cut subroutine?

Solution: Fix an arbitrary vertex s. Compute minimum s-t cut for all $t \neq s$. Output the minimum among all these s-t mincuts. We need to invoke the min s-t cut subroutine $n-1$ times.

The reduction works since for any choice of s, any graph cut would separate s from some other vertex $t \neq s$ (simply take any t on the other side of the cut). In particular, if the global mincut separates s from t^*, then it is the minimum s-t^* cut (otherwise, there would be a even better global mincut). Our reduction loops over all choices of $t \neq s$, which include t^*.

5 Dijsktra's Sort

Show how to use Dijsktra's algorithm to sort n real numbers (not necessarily non-negative or integral) in ascending order in $O(n \log n)$ time. You may use the intermediate outputs of Dijsktra's to do the sorting (as opposed to using it as a blackbox).

Argue that this means that improving upon the $O(m + n \log n)$ run-time of Dijsktra's algorithm (with Fibonacci heap) would lead to a faster sorting algorithm than merge and quick sort.

Hint: Given the numbers, construct a graph such that the order of vertices being extracted from priority queue by Dijsktra's algorithm corresponds to the right sorting order.

Solution: Consider the star graph with one center vertex s connected to n other vertices. Let the input numbers be $\{t_1, \cdots, t_n\}$. Let edge (s, i) have weight t_i. We consider each non-center vertex i as corresponding to the number t_i We claim that the order of vertices being extracted from the priority queue by the Dijsktra's algorithm is precisely the ascending order of the inputs. In the first iteration, s is extracted, and we can ignore that, since it doesn't correspond to any input number. Observe that then the Dijsktra's algorithm updates the vertex labels `dist` to be exactly the input numebers. Hence, the next iteration will extract exactly the vertex corresponding to the smallest number t_i. But since a non-center vertex is not adajacent to anything except the center, we do not update `dist`, so we move on. Then the algorithm proceeds to extract the vertex corresponding to the second smallest number t_i in the next iteration, and so on.

If there's any data structure that enables faster run-time for Dijsktra's, then we can always apply the reduction above to sort numbers faster. Essentially, we are just using the fact that Dijsktra's uses priority queue and priority queue can be used for sorting.