CSE525: Randomized Algorithms and Probabilistic Analysis April 23, 2013

Lecture 7

Lecturer: Anna Karlin *Scribe: Svetoslav Kolev, Tanner Schmidt*

1 Congestion Minimization

In congestion minimization we are given a directed graph and a set of pairs of nodes that we wish to connect with (possibly non-disjoint) paths while minimizing the maximum use of any edge.

Formally we are given a directed graph $G = (V, E)$ and a set of pairs (s_i, t_i) for $i = 1, \ldots, k$. We need to compute a path P_i from s_i to t_i for $i = 1, \ldots, k$, such that the congestion C is minimized where $C = \max_{e \in E}$ number of P_i contain e.

The problem is NP-hard, so we naturally turn to a randomized algorithm. We will developed an approximation algorithm using randomized rounding.

2 Linear Program Formulation

First we formulate a fractional version of the problem (multicommodity flow) as a linear program.

Our variables are the congestion C and $f_i(e)$ where $e \in E$, $i = 1, \ldots, k$. We represent the flow of path P_i on edge e with $f_i(e)$.Let also OUT_v and IN_v denote the sets of outgoing and incoming edges to v. The linear program is:

$$\begin{aligned} & minimize \;\; C \\ \text{subject to} \quad & \sum_{e \in IN_v} f_i(e) = \sum_{e \in OUT_v} f_i(e) \quad \forall v \neq s_i, t_i \\ & \sum_{OUT_{s_i}} f_i(e) = 1 \\ & \sum_i f_i(e) \leq C \;\; \forall e \\ & f_i(e) \geq 0 \;\; \forall i, e \end{aligned}$$

The first constraint ensures conservation of flow. The second ensures we route 1 unit of flow from s_i to t_i. The third is the congestion bound. The fourth ensures positive flow.

Solve the Linear Program and obtain $f_i^*(e)$ and C^*. We note here that $C^* \leq OPT$ congestion for the integer version.

3 Decomposition into discrete paths

Next we decompose the flows f^* into discrete paths. Let

$$P_{s_i,t_i} = \{P | P \text{ is a path from } s_i \text{ to } t_i\}$$

be the set of all paths from s_i to t_i. For each i we find a set $P_i^* \subseteq P_{s_i,t_i}$ and $f_p^i \;\; \forall p \in P_i^*$, subject to:

$$\begin{aligned} & |P_i^*| \text{ is polynomial} \\ & \sum_{p \in P_i^* | e \in p} f_p^i = f_i^*(e) \end{aligned}$$

$$\sum_{p \in P_i^*} f_p^i = 1.$$

Basically we must decompose the fractional flow into a polynomial number of discrete paths (not necessarily disjoint). The flow on path p is f_p^i and we want the sum of the flow on all paths to be 1. Also, for each i the total flow on each edge e (summed over all paths containing it) must be equal to the one found from the linear program $f_i^*(e)$.

One way to find such a path is to repeatedly perform the following graph search. Consider all the edges with $f_i^*(e)$ and find a path p from s_i to t_i. Set $f_p^i = \min_{e \in p} f_i^*(e)$. Remove f_p^i units of flow from all edges in p. Then we repeat to find a new path.

We then have a set P_i^* of paths that connect the pairs $s_i \to t_i$. Next we pick path $p_i \in P_i^*$ with probability f_p^i. Output $p_i, \ldots, p_k$ as a solution to our original problem.

4 Analysis

4.1 Approximation ratio

Theorem 1.

$$Pr\ (\textit{any edge has congestion} \geq \frac{6 \log n}{\log \log n} C^*\ ) \ \leq \frac{1}{n}$$

Proof

Take $\alpha = \frac{6 \log n}{\log \log n}$. Then

$$\text{Fix} \quad e = (u \to v)$$

$$\text{Define} \quad X_i(e) = \begin{cases} 1 & \text{if} \quad e \in p_i \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Then} \quad E(X_i) = \sum_{p \in P_i | e \in p} f_p^i = f_i^*(e).$$

$$\text{Let} \quad X(e) = \sum_{i=1}^{k} X_i(e). \quad \text{Then} \quad E(X(e)) = \sum_{i=1}^{k} f_i^*(e) \leq C^*.$$

Use Chernoff bound with C^* upper bound on μ.

$$Pr\ (\ X(e) \geq (1+\delta) C^*) \leq e^{-C^*[(1+\delta)\ln(1+\delta) - \delta]}.$$

Take $(1+\delta)C^* = \alpha C^*$ so that $1 - \alpha = -\delta$. Then

$$\begin{aligned} Pr(X(e) \geq \alpha C^*) &\leq e^{-C^*[\alpha \ln \alpha + 1 - \alpha]} \\ &\leq e^{-\alpha(\ln \alpha - 1)} \\ &\leq -(\frac{6}{2} \ln n) \\ &\leq \frac{1}{n^3}. \end{aligned}$$

Here we used $C^* \geq 1$ for the second inequality and $\alpha = \frac{6 \ln n}{\ln \ln n}$ so that

$$\alpha(\ln \alpha - 1) = \frac{6 \ln n}{\ln \ln n} [\ln 6 + \ln \ln n - \ln \ln \ln n - 1]$$

and

$$[\ln 6 + \ln\ln n - \ln\ln\ln n - 1] \geq \frac{\ln\ln n}{2}$$

as before.

$$Pr\ (\text{ any edge has congestion} \geq \alpha C^*) \leq \sum_e Pr(X(e) \geq \alpha C^*)$$

by the union bound. Then $\sum_e Pr(X(e) \geq \alpha C^*) \leq n^2 \frac{1}{n^3} = \frac{1}{n}$.

■

4.2 Optimality analysis

The algorithm achieves optimal approximation since there exist graphs with $\frac{OPT}{C^*} = \Omega(\frac{\log n}{\log\log n})$ (Integrality gap). Therefore we cannot do better with such an approach.

Hardness (directed graphs). Every polynomial time algorithm has $\Omega(\frac{\log n}{\log\log n})$ approximation ratio. Assuming $NP \not\subseteq BPTIME(n^{O(\log\log n)})$

If $C^* \geq c\ln n$ then with high probability congestion on every edge $\leq C^* + \sqrt{cC^*\ln n}$.

$$Pr(X(e) \geq C^* + \sqrt{cC^*\ln n}) \leq e^{-\frac{C^*\delta^2}{3}} = e^{-\frac{-c\ln n}{3}} = n^{-\frac{c}{3}}$$

$$(1+\delta)C^* \Rightarrow \delta = \sqrt{\frac{c\ln n}{C^*}}$$

Take $c = 9$. Then $Pr($ edge e is bad $) \leq \frac{1}{n^3}$. Therefore, Pr (there exists a bad edge) $\leq \frac{1}{n}$ by the union bound.

Observe that $C^* + \sqrt{cC^*\ln n} = O(C^*)$ if $C^* = \Omega\ln n$.

5 Bipartite Matching

The bipartite matching problem is a classic computer science problem in which we wish to find an optimal matching of vertices from opposing sides of a bipartite graph. More formally, let's call the graph $G = (L, R, E)$, where L represents the set of vertices in the 'left' partition, R represents the set of vertices in the 'right' partition, and E is the set of edges spanning the partition. We wish to find a subset of E such that each vertex lies on at most one edge, which defines a matching. Often, the matching is optimized by maximizing the size of the subset, or maximizing the total weight of the subset if the edges are weighted.

6 Online Bipartite Matching

The online version of the bipartite matching problem has practical (and high-value) applications to problems such as internet ad allocation, where the advertisers and demand for each advertiser is known a priori, but the page views are not.

In the online formulation, L is known ahead of time, but vertices in R arrive one at a time. When vertex $v_j \in R$ arrives, we learn which vertices in L are neighbors to v_j, and we must make an irrevocable decision as to which of the neighbors to match it to (if any).

7 Greedy Algorithm

Assume we are solving the online bipartite matching problem and we want to maximize the size of our matching. The most obvious greedy algorithm would simply match each incoming vertex to any of its neighbors at random.

Claim 2. *If OPT is the optimal online matching, the greedy algorithm will always find a matching of size* $\geq \frac{1}{2} OPT$.

Proof Let's analyze the algorithm by attributing \$1 to each matching. Therefore, if the optimal solution to the offline algorithm matches m vertices, it has a value of $\$m$.

For the greedy algorithm, when incoming vertex $v_j \in R$ is matched to vertex $v_i \in L$, imagine putting half of the \$1 for edge (i, j) on v_i and half on v_j, such that each has a value of \$0.50.

Now consider only edges (i, j) such that vertices v_i and v_j were matched by the optimal offline solution, which thus contributed \$1 to OPT. There are only two possible outcomes for the greedy algorithm. The first is that v_i and v_j each have \$0.50, and thus the value for the edge is the same for the greedy algorithm and the optimal offline algorithm. The second is that only one of the two vertices has \$0.50, in which case the value for the edge is half that of the optimal offline algorithm. We know that it is not possible for both to have no money, because in that case, the greedy algorithm could have matched v_j. Therefore, since the worst case value for each edge is half that of the optimal value, the solution of the greedy algorithm must be $\frac{1}{2}$OPT.

■

8 Competitive Ratio

When analyzing the performance of online algorithms, it's often useful to talk about the competitive ratio of the algorithm, which compares the performance of the online algorithm to an optimal offline algorithm. If I is the set of all possible instances of the problem, the competitive ratio is defined by:

$$\text{c.r.} = \min_{I} \frac{\text{value of solution to online algorithm}}{\text{value of optimal offline solution}}$$

Therefore, what we have shown above is that the greedy algorithm for online bipartite matching has a competitive ratio of $\frac{1}{2}$.

9 Fractional Matching

We can recast the integral online bipartite matching problem as a fractional problem. Define variable x_{ij} for each edge (i, j). In the integral problem, x_{ij} takes value 1 if vertex v_i is matched to v_j, and 0 otherwise. In the fractional case, we allow x_{ij} to take fractional values, subject to the following additional constraints:

$$\forall_{i,j} x_{ij} \geq 0$$

$$\forall_{j|v_j \in R} \sum_{i \in N(j)} x_{ij} \leq 1$$

$$\forall_{i|v_i \in L} \sum_{j \in N(i)} x_{ij} \leq 1$$

where $N(j)$ defines the neighborhood of v_j.

Claim 3. *If A is a randomized algorithm for integral online bipartite matching, $\exists$ a deterministic fractional algorithm D s.t. $\forall$ instance I, $\sum_{i,j} x_{ij}^D(I) = E(\sum_{i,j} x_{ij}^A(I))$.*

Proof We wish to define a deterministic fractional algorithm D which simulates the performance of our randomized integral algorithm A. To do this, when v_j arrives, we simply assign it fractionally according to the probability distribution of the assignment in the randomized algorithm, i.e. set $x_{ij}^D = \Pr(X_{ij}^A = 1)$. We, of course, have:

$$\forall_i \sum_{j \in N(i)} X_{ij}^A \leq 1,$$

which means:

$$\forall_i E[\sum_{j \in N(i)} X_{ij}^A] \leq 1$$

which in turn tells us:

$$\forall_i \sum_{j \in N(i)} x_{ij}^D \leq 1.$$

$$\sum_{(i,j) \in E} x_{ij}^D = E[\sum_{(i,j) \in E} X_{ij}^A] = E[\text{performance of } A]$$

In other words, the expected value of the integral randomized solution falls within the constraints of the deterministic fractional algorithm, and therefore this strategy is valid, and the deterministic fractional algorithm finds a solution with value equal to the expected value of the randomized algorithm. ■

Given that we now know that, for any randomized algorithm for online integral bipartite matching, we can define a fractional deterministic algorithm that 'simulates' it's behavior, we know that any upper bound on the competitive ratio of any deterministic fractional algorithm will also provide an upper bound on the competitive ratio of any randomized algorithm.