CS369B: Take-Home Final

Due by 10PM on Thursday, March 20, 2008

Instructions: Same as the problem sets.

- Note that collaboration *is* allowed (and encouraged), like on the problem sets.
- Send Shaddin your solutions by email by Thursday, March 20th, at 10PM (the scheduled exam time for the course). As usual, **no extensions**.

Problem 21

Gomory-Hu Trees and Clustering. Let $G = (V, E)$ be an undirected graph, where each edge (u, v) has a nonnegative weight w_{uv}. Think of absent edges as having zero weight. For two subsets $A, B \subseteq V$, let $w(A, B) = \sum_{u \in A, v \in B} w_{uv}$ denote the total weight of edges with one endpoint in each of A and B.

We can use Gomory-Hu trees to cluster the nodes of G as follows. Add an auxiliary sink t to G and add an edge between t and every vertex of G. The new edges have weight α for a parameter $\alpha \geq 0$. Let T be a Gomory-Hu tree of this new graph G' (where we interpret the weights as capacities). Deleting t from T shatters it into components which partition V; these are the clusters in our clustering.

(a) [Do not hand in] Define a *community* as a set $S \subseteq V$ of nodes of G such that either $|S| = 1$, or else each node gives more weight to internal links than external links:

$$\forall s \in S, \sum_{v \in V \setminus S} w_{sv} \leq \sum_{v \in S} w_{sv}.$$

Convince yourself that the clustering method above produces a set of communities in this sense.

[Hint: Each cluster S corresponds to a min s-t cut in G' for the vertex s of S that is adjacent to t in the Gomory-Hu tree T. Interpret the inequality above as a "local optimality" condition for this cut.]

(b) Another nice property is that clusters must be internally rather dense. Precisely, for a cluster S, define its *expansion* by

$$\min_{\emptyset \neq Q \subset S} \frac{w(Q, S \setminus Q)}{\min\{|Q|, |S \setminus Q|\}}.$$

Show that every cluster has expansion at least α. (By convention, we define the expansion of a single node to be $+\infty$.)

(c) Prove also that the boundary of every cluster S is sparse in the following sense:

$$\frac{w(S, V \setminus S)}{|V \setminus S|} \leq \alpha.$$

(d) Convince yourself that a sufficiently small value of α, say 0, produces a single cluster containing all nodes. On the other hand, a sufficiently large value of α produces n singleton clusters. Prove more generally that as we increase α, we progressively refine the clustering. In other words, for $\alpha' > \alpha$, every cluster obtained when using α' is a subset of one obtained when using α.

[Hint: start by proving that for a fixed s, the s-side of a min s-t cut in G' can only shrink as α increases (assuming suitable tie-breaking among min cuts).]

(e) Part (d) implies that varying α leads to at most $n-1$ distinct clusterings, but it is not clear how to identify the relevant values of α. Suppose that G has only unit-weight edges. Identify a polynomial-size set of values of α that are guaranteed to produce all possible clusterings (up to tie-breaking issues).

Problem 22

More Applications of Gomory-Hu Trees.

(a) Suppose G is a (large) undirected graph, and H is a (small) undirected graph that is at least k-edge-connected. Choose $k/2$ nodes of G and H arbitrarily, and create a graph K by attaching these two sets of $k/2$ nodes to each other via an arbitrary matching. (So the vertex set of K is the union of those of G and H.) Compute a Gomory-Hu tree T of K and delete all edges of T with weight strictly less than k. Prove that one of the connected components that results is precisely the nodes of H.

[Aside: this idea was recently used to devise new attacks on anonymized social networks; here G is the anonymized network and H is a random graph chosen by the attacker.]

(b) Consider an undirected graph with nonnegative weights on the edges. A *k-cut* is a partition of the graph's vertices into k non-empty classes. The *weight* of a k-cut is the sum of the weights of the edges with endpoints in distinct vertex classes. Give a polynomial-time 2-approximation algorithm for the min-weight k-cut problem based on the idea of using the $k-1$ minimum-cost edges of a Gomory-Hu tree. Prove both that your algorithm outputs a feasible solution and that the weight of this solution is at most twice the minimum possible.

(c) [Extra Credit] Extend your algorithm and analysis in (b) to the more general problem in which there is a set $S \subseteq V$ of vertices with $|S| \geq k$, and the feasible solutions are only the k-cuts in which each vertex class contains at least one vertex of S. (So part (b) corresponds to the case of $S = V$.)

Problem 23

Counting Cuts and the Cut Sampling Lemma. We briefly recall Karger's Random Contraction Algorithm (RCA) for computing a global min-cut of a graph, covered in (for example) the books by Kleinberg & Tardos (Section 13.2) and Motwani & Raghavan (Section 10.2). For simplicity, throughout this problem we assume a graph with unit capacities (parallel edges are allowed). The algorithm is to repeatedly pick and contract a random edge (throwing out any resulting self-loops) until two (super)nodes remain, and finally returning the cut that corresponds to the final two nodes.

For the analysis, pick your favorite min cut (A, B), with k crossing edges. RCA returns this cut if and only if none of the k edges crossing it are chosen for contraction at any iteration. Throughout the algorithm, (super)nodes in the current contracted graph correspond to cuts in the original graph, each with at least k crossing edges. Thus every contracted graph has minimum degree at least k and hence at least $kn_i/2$ edges, where $n_i = n-i+1$ is the number of nodes of the contracted graph in the ith iteration. Thus the probability that none of (A, B)'s k crossing edges are chosen in the ith iteration, given that none were chosen earlier, is at least $1-2k/kn_i = 1-2/n_i$. It follows from a magically telescoping product that the probability that RCA outputs the cut (A, B) is at least $2/n(n-1)$. Obviously, this probability can be amplified via a polynomial number of repeated trials.

(a) Note that we lower bounded the probability that RCA outputs a *specific* minimum cut, rather than an *arbitrary* minimum cut. Explain why this implies that an undirected graph can possess only $\binom{n}{2}$ different minimum global cuts. Provide a graph for which this bound is tight.

(b) Extend your reasoning in (a) to show that, for every integer α, an undirected graph has only $O(\alpha n^{2\alpha})$ distinct α-approximate global cuts (i.e., cuts whose crossing edges number at most α times the minimum possible). (If you can't quite prove this bound, prove the best one you can.)

[Hint: You might find it helpful to modify RCA to stop a little early and then output a random cut.]

(c) Prove the following form of the Cut Sampling Lemma: let $G = (V, E)$ be an undirected graph with unit capacities and possibly parallel edges. Obtain G' from G by sampling each edge $e \in E$ with some probability p. Prove that if the expected number of edges in every cut of G' is at least $(a \log n)/\epsilon^2$ for a suitable constant $a > 0$, then the value of every cut of G' is within $(1 \pm \epsilon)$ of its expected value with high probability.

[Hint: For a fixed cut, apply the Chernoff bound. This should give you an inverse polynomial failure probability, but unfortunately there are exponentially many cuts. But (b) implies that most cuts have size must more than the minimum, and Chernoff's error bound becomes exponentially small for such cuts. This idea suggests a more refined Union Bound approach.]

Problem 24

Graph Sparsification and Max Flow. Consider the s-t max flow problem in undirected graphs with unit-capacity edges, with no parallel edges permitted. Recall that Dinitz's algorithm runs in $O(m \cdot \min\{\sqrt{m}, n^{2/3}\})$ time in this case (even if the graph is directed). This problem outlines how to do better in dense (undirected) graphs using graph sparsification. The idea is to sparsify the residual graph before each blocking flow computation. Two issues arise: we need to know what level of connectivity to preserve (for the sparsification subroutine); and reverse arcs in the residual network are inherently directed, while sparsification techniques work only for undirected graphs.

(a) [Do not hand in.] Recall the argument that in a unit-capacity network, if every augmenting path in the residual graph has length at least ℓ, then the maximum flow in the residual network (and hence the distance between the current flow value and the maximum in the original graph) is at most $a \cdot \min\{\sqrt{m}, n^{2/3}\}$ for a suitable constant $a > 0$.

(b) Consider a unit-capacity network with no parallel edges. Prove that if the Ford-Fulkerson augmenting path algorithm is implemented so that a shortest path (minimum number of hops in the current residual network) is chosen at every iteration (i.e., as in Edmonds-Karp), then at termination only $O(n^{3/2})$ edges carry flow. Draw an identical conclusion for the usual implementation of Dinitz's algorithm.

[Hints: Rearrange one of the bounds in (a). Note that the max flow value is $O(n)$. It might be convenient to estimate a sum by an integral.]

(c) [Do not hand in.] Suppose we know that the max flow value in the current residual network is at most μ. Let E_0 denote the unused and therefore undirected edges of G, and E_1 the (directed) edges of G that have been used in at least one direction on at least one augmenting path so far. Suppose the Nagamochi-Ibaraki subroutine, when applied to (V, E_0) with parameter $k = \mu$, returns the sparse certificate $F \subseteq E_0$. Convince yourself that we can safely delete the edges of $F \setminus E_0$ from G for the rest of the algorithm.

As a consequence, repeatedly alternating the following two steps correctly computes a max flow: (i) using $\mu = (an/\ell)^2$ as an upper bound on the max flow value in the current residual network, where ℓ is the s-t distance in this network and a is a constant, run the sparsification algorithm above to remove some of the (undirected) edges from the current network; (ii) route a blocking flow through the current residual network.

Note that this algorithm performs the same work as Dinitz's algorithm except on sparser graphs, and thus (since the sparsification subroutine is linear-time) can only have a better asymptotic run time.

(d) For the algorithm from part (c), use (b) and properties of the Nagamochi-Ibaraki algorithm to give an upper bound (in terms of n and ℓ) on the number of edges left in the residual graph when its current s-t distance is ℓ.

(e) Using part (d), prove that the above algorithm runs in $O(n^{3/2}\sqrt{m})$ time. [This bound is smaller than that for Dinitz's algorithm when m is sufficiently large relative to n. As noted in (c), it is also true that the running time of the algorithm in (c) is asymptotically at least as good as Dinitz's algorithm for all values of m and n.]

[Hint: Consider separately the iterations for which the s-t distance ℓ in the current residual graph is at most $n^{3/2}/\sqrt{m}$, the iterations for which $\ell \geq n^{3/4}$, and the remaining iterations.]

Problem 25

Max-Flow and Min-Cut in Planar Graphs. In this problem we consider undirected capacitated planar graphs. Since planar graphs are sparse ($m = O(n)$), the Goldberg-Rao algorithm would compute an s-t max flow and min cut in $\tilde{O}(n^{3/2})$ time in such graphs. But we can do even better.

(a) Give an $O(n \log n)$-time algorithm for computing a minimum s-t cut in undirected planar graphs.

[Hint: reduction to shortest paths in the planar dual.]

(b) Use the information provided by your algorithm in (a) to extract a max flow with only $O(n)$ further work.

[Hint: make heavy use of properties of shortest-path distances: to define the amount of flow on each edge, to show that capacity constraints are obeyed (use the Triangle Inequality), and to show that node conservation constraints are obeyed (use the correspondence between cuts in the original graph and cycles in the dual graph).]