COMPSCI 590.1: Graph Algorithms February 12, 2015

Lecture 11

Lecturer: Debmalya Panigrahi *Scribe: Ilker Nadi Bozkurt*

1 Overview

The first major theme of the lectures was computing maximum *s*-*t* flows. The algorithms we covered in these lectures were deterministic. Later, we covered global min-cut algorithms, mostly relying on randomized algorithms which used sampling and contraction operations. In this lecture, we will revisit maximum flows using random sampling. We will present an algorithm for unit capacity, undirected graphs due to Karger and Levine [KL02]. The algorithm is useful when the maximum flow is small.

Remember the runtime of Ford-Fulkerson algorithm is $O(mv)$, where v is the value of the maximum flow. Even when we have a small value of v, this algorithm will be slow if $m = \Omega(n^2)$.

Example 1. Let's connect two copies of $K_{n/2}$ with a single edge. In figure 1, $v = 1$ for any s,t residing in a different complete graph. So, runtime of Ford-Fulkerson is $O(n^2)$ for this example. This example illustrates that the dependency of m and v should be removed to have a fast algorithm.

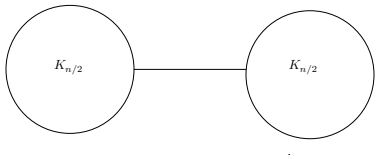

Figure 1: An example with a flow value of 1

Using the Nagamochi Ibaraki algorithm [NI92] to compute a spanning forest packing of G, we can reach the $O(m+nv^2)$ bound. First note that we can compute the spanning forests $F_1, F_2, \ldots$ in $O(m)$ time. We will use the following fact to justify running Ford Fulkerson algorithm on the spanning forests:

Fact 1. *Let $F_1, \ldots, F_l$ be a spanning forests of G computed using Nagamochi-Ibaraki algorithm and let C be any cut of size k. Then, $F_1, \ldots, F_l$ has at least* $\min(l,k)$ *edges from cut C.*

Proof. Let G_1 and G_2 be the two vertex induced subgraphs of G by the cut C and let there be k edges crossing the cut. There are two cases to consider:

- If G_1 and G_2 are disconnected in F_l, then all k edges in the cut appear in somewhere in F_1 to F_l to connect G_1 and G_2.
- If G_1 and G_2 are still connected in F_l, at least a total of l edges from this cut must have appeared in the forests $F_1, \ldots, F_l$.

□

Using this fact, it is obvious that we will find a maximum flow if we run Ford-Fulkerson on $\cup_{i=1}^{v} F_i$. Since the value of maximum flow, v, is not known beforehand, we can run Ford-Fulkerson first on F_1, then

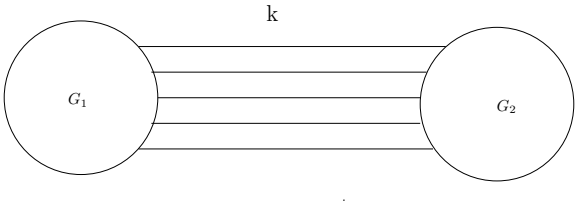

Figure 2: A cut of size k

on $F_1 \cup F_2$, and continue doubling the number of forests in each iteration (until the computed flow is less than the number of forests). This amounts to running Ford-Fulkerson on $O(nv)$ edges, hence the running time will be $O(m+nv^2)$.

Note that any augmenting paths algorithm cannot avoid the cost $O(nv)$. The maximum flow can be decomposed into v paths and each path can have length up to $n-1$. So, a natural target is $O(m+nv)$. The algorithm of Karger and Levine that we describe in this lecture reaches this bound (ignoring logarithmic factors).

2 Random Sampling Algorithm

In each step of the algorithm, a random sample of edges of the residual graph is used to find an augmenting path. If the remaining flow is large, sparsification is done aggressively, i.e. a small number of edges are sampled since it is easier to find an augmenting path. When the remaining flow is small, more edges are sampled to find an augmenting path. Next we present the algorithm of Karger and Levine.

Find (lower bounds on) edge strengths k_e
$\alpha \leftarrow 1$
while $\alpha n \leq m$ **do**
  Sample αn edges from the residual network with prob. proportional to $\frac{1}{k_e}$
  if The sample contains an $s-t$ path **then**
    Increment the flow along the $s-t$ path
    Recompute the residual network
  else
    $\alpha \leftarrow 2 * \alpha$;
  end if
end while
Run the augmenting paths algorithm on G_F

Correctness. The correctness of the algorithm is ensured by the last step, which uses Ford Fulkerson algorithm to compute the remaining flow. Hence, the computed flow will be a maximum flow.

2.1 Analysis

The following lemma is from Karger and Levine and is used for proving the main result which follows. The proof is given in [KL02].

Lemma 2. *Given positive real numbers $w_1, \ldots, w_l$ in decreasing order, for any real numbers $x_1, \ldots, x_l$ such that $\sum_{i=1}^{l} x_i \geq 0$ there exists a $j \in \{1, \ldots, l\}$ such that $\sum_{i=1}^{l} x_i w_i \geq 0$.*

The algorithm crucially depends on step 5, i.e. the existence of an $s-t$ path in the set of sampled edges. The next lemma shows that such a path exists with high probability if enough number of edges are sampled based on the remaining flow.

Lemma 3. *Suppose a set of $O(\beta n)$ edges with probability proportional to $1/k_e$ are sampled from the residual graph, where $\beta = \log n \frac{v}{v-|f|}$. Then, the sample has an $s-t$ path with high probability.*

Proof. We will give a sketch of the proof. For the full details see [KL02]. At any step of the algorithm, at least $\frac{v-|f|}{v}$ of each cut is preserved, but some edges from each cut might be saturated. The algorithm compensates the saturating edges by increasing the number of sampled edges by a factor of $\frac{v}{v-|f|}$. Even if the flow is not necessarily spread out evenly among edges with different strength, the existence of a $k-$strong edge implies the existence of at least k such edges and the likelihood of choosing one of these edges in the sample increases.

Counting the $k-$strong edges in the residual graph is difficult, so Karger and Levine compute the probability of choosing such an edge from the residual graph by counting k-strong edges in the original graph G. For each cut of the residual graph, the edges are grouped by strength and the groups are ordered in increasing order by strength. Let X be a cut of G and let the strength of the edges in group i be k_e. For group i, w_i is defined as β/k_e, and, x_i is defined as the number of residual edges (in the cut X) of strength k_e minus $\frac{v-|f|}{v}$ times the number of edges in G (in the cut X) of strength k_e. When the probability of picking an edge with strength at least k from this cut in the residual graph is evaluated, we can bound the resulting expression using lemma 2 and obtain $\exp(-\sum_{e\in X}\frac{\log n}{k_e})$. We have seen earlier while analyzing sparsifiers, that

$$\sum_{X\in \text{cuts}} \exp(-\sum_{e\in X}\frac{\log n}{k_e}) \leq \frac{1}{\text{poly}(n)} \tag{1}$$

holds. By proving a similar result, Karger and Levine show that at least one edge is chosen from each cut with high probability. This implies that there is an augmenting path between s and t in the residual graph. □

2.2 Running time analysis

1. Step 1 runs in $O(m\log^2 n)$ time if the algorithm in [BK96] is used.
2. α is doubled $O(\log(m/n))$ times in the while loop.
   - α is not doubled until the remaining flow is halved with high probability. This is due to lemma 3. Note that the remaining flow $v-|f|$ is the only variable part in the definition of β.
   - Let's assume the remaining flow is halved from 2^k from 2^{k-1} in an iteration of the while loop. 2^{k-1} augmentations will be performed on a graph of size $n\log n\frac{v}{2^k}$. So, time per iteration is $O(nv\log n)$.
   - Hence, the total running time of steps 3-11 is $O(nv\log n\log(m/n))$.
3. Step 12 is executed when $n\log n\frac{v}{v-|f|} = m$. So, $v-|f| = \frac{nv\log n}{m}$.
   - So, the running time of step 12 is $O(nv\log n)$.

3 Summary

We described an algorithm for the maximum flow problem which used sparsification (by sampling) on the residual graph. Even though the sampling techniques we have covered earlier are not applicable to directed graphs, Karger and Levine's algorithm shows that they can be applied on the residual graphs resulting from finding flows in undirected graphs. Note that the use of edge strengths for sampling is not essential, the algorithm and the analysis can be modified to work with other connectivity parameters.

References

[BK96] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in o(n2) time. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 47–55. ACM, 1996.

[KL02] David R. Karger and Matthew S. Levine. Random sampling in residual graphs. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 63–66. ACM, 2002.

[NI92] Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(1-6):583–596, 1992.