Random Sampling in Residual Graphs

David R. Karger [*]
MIT Laboratory for Computer Science
Cambridge, MA 02138
karger@theory.lcs.mit.edu

Matthew S. Levine[*]
MIT Laboratory for Computer Science
Cambridge, MA 02138
mslevine@theory.lcs.mit.edu

ABSTRACT

Consider an n-vertex, m-edge, undirected graph with maximum flow value v. We give a new $\tilde{O}(m+nv)$-time maximum flow algorithm based on finding augmenting paths in random samples of the edges of residual graphs. After assigning certain special sampling probabilities to edges in $\tilde{O}(m)$ time, our algorithm is very simple: repeatedly find an augmenting path in a random sample of edges from the residual graph.

1. INTRODUCTION

In this paper we consider the problem of finding maximum flows in undirected graphs with small flow values. The history of the study of algorithms for small flow values is as old as the study of the general maximum flow problem. In fact, the original $O(mv)$-time Ford-Fulkerson algorithm [3] is still the best known deterministic algorithm for sparse graphs with sufficiently small flows. Here m is the number of edges, v is the value of the maximum flow, and n will be the number of nodes. Karger opened the question of how much easier it might be to find flows in undirected graphs, first by giving an $\tilde{O}(mv/\sqrt{c})$-time algorithm for graphs with connectivity c [9], and then by giving an $\tilde{O}(m^{2/3}n^{1/3}v)$-time algorithm for simple (unit-capacity, no parallel edges) graphs [7]. This lead to a number of results, the most recent being algorithms with the unpleasant time bounds $\tilde{O}(m+nv^{5/4})$ and $\tilde{O}(m+n^{11/9}v)$ [10]. See Table 1 for a history of algorithms for small flow values.

All of the recent papers on flows in undirected graphs make some attempt to avoid looking at all of the edges all of the time, so as to reduce the amortized time per augmenting path to $o(m)$. Our work closes a chapter in this research, finally achieving $\tilde{O}(m+nv)$ time, or amortized time per augmenting path of $\tilde{O}(n)$. Since a flow path can have as many as $n-1$ edges, this is the best result (up to logarithmic factors) that one can hope to achieve by sparsification alone.

Our key advance is taking random samples of the edges of residual graphs. Benczúr and Karger showed that sampling each edge of an undirected graph with probability inversely proportional to a quantity called *strength* yields a connected graph with only $O(n \log n)$ edges with high probability. This would be a great way to find a first augmenting path quickly, but once you have a non-zero flow the residual graph is directed, so it is no longer helpful to know that you can sample from an undirected graph. One way to interpret our result is that we show that a residual graph remains similar to the original undirected graph, which means that the Benczúr-Karger sampling can be applied iteratively. With the exception of computing the edge strengths at the beginning, our algorithm is just this simple iteration: sample according to strength, find an augmenting path, repeat until done.

In order to show which algorithms have the best performance for different values of m and v relative to n, we have drawn figures (Figures 1 and 2): one for deterministic algorithms only and one including randomized algorithms. A point in the figure represents the value of m and v relative to n. Specifically, (a,b) represents $v=n^a, m=n^b$. Each region is labeled by the best time bound that applies for values of m and v in that region. Note that the region $m>nv$ is uninteresting, because the sparsification algorithm of Nagamochi and Ibaraki [12] can always be used to make $m \le nv$ in $O(m)$ time. The shaded region corresponds to the algorithm given in this paper. Note that the $O(nm^{2/3}v^{1/6})$-time algorithm (which is the fastest algorithm for the region surrounded by a dashed line) is the only one in the picture that cannot handle capacities or parallel edges, so the picture looks strange at $v=n$. If capacities are being considered, then this algorithm should be removed from the picture; if only simple graphs are being considered, then the picture should end at $v=n$. The complexity of the diagram for deterministic algorithms suggests that more progress can be made there. With the addition of our result, the diagram for randomized algorithms is now fairly simple.

The rest of this paper is organized as follows. In Section 2 we review some notation and basic definitions. In Section 3 we give the new algorithm. In Section 4 we give the analysis of the algorithm. We conclude and discuss some open questions in Section 5.

2. NOTATION AND DEFINITIONS

We use the following notation:

G	the graph
v	value of a maximum flow
n	number of nodes
m	number of edges
s	the source
t	the sink
f	a flow
G_f	residual graph of G with respect to f

[*]Research supported by NSF contract CCR-9624239 and a David and Lucille Packard Foundation Fellowship.

Source	Year	Time bound	Capacities?	Directed?	Deterministic?
Ford-Fulkerson [3]	1956	$O(mv)$	√	√	√
Even-Tarjan [2]	1975	$O(m\min\{n^{2/3},m^{1/2}\})$		√	√
Karger [7]	1997	$\tilde{O}(m^{2/3}n^{1/3}v)$			
Goldberg-Rao [4]	1997	$\tilde{O}(m\min\{n^{2/3},m^{1/2}\}\log v)$	√	√	√
Goldberg-Rao [5]	1997	$O(n\sqrt{nm})$			√
Karger [8]	1998	$\tilde{O}(v\sqrt{nm})$	√		
Karger-Levine[10]	1998	$O(nm^{2/3}v^{1/6})$			√
Karger-Levine[10]	1998	$\tilde{O}(m+nv^{3/2})$	√		√
Karger-Levine[10]	1998	$\tilde{O}(m+nv^{5/4})$	√		
Karger-Levine[10]	1998	$\tilde{O}(m+n^{11/9}v)$	√		
this paper	2001	$\tilde{O}(m+nv)$	√		

Table 1: Summary of algorithms. The long history of $\tilde{\Omega}(mn)$-time algorithms, which are still best for large v, have been omitted.

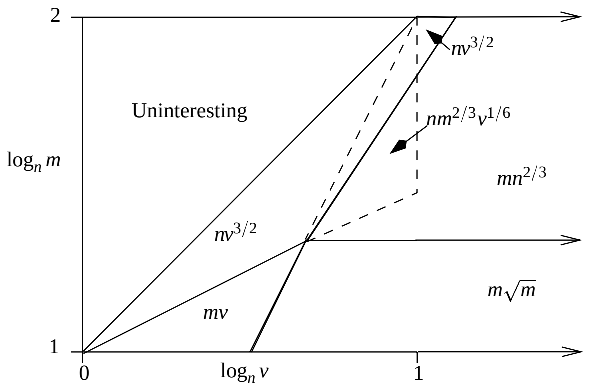

Figure 1: Pictures of the best deterministic bounds. (See text for explanation.)

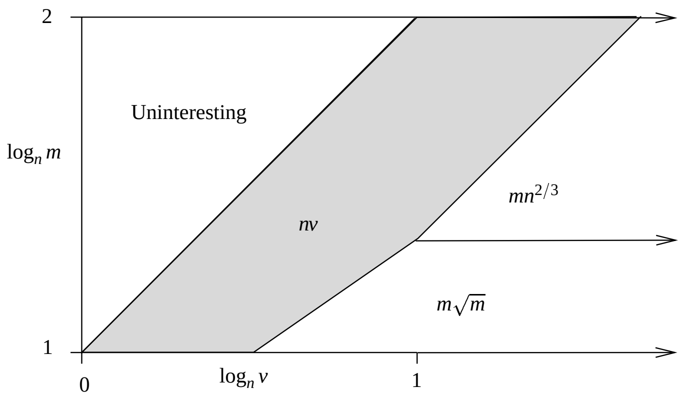

Figure 2: Pictures of the best randomized bounds. (See text for explanation.)

$\lvert f\rvert$	the value of flow f
e	an edge
k_e	the strength of edge e
X	the set of edges that cross a given cut

DEFINITION 2.1. *The* strength *of an edge e, denoted k_e, is the maximum value of k such that a k-connected vertex induced subgraph of G contains e. We say e is k*-strong *if its strength is k or more, and k*-weak *otherwise.*

It is common to refer to a cut as a non-empty set of vertices Y that is a proper subset of the vertices. For convenience, in our proofs we use a shorthand of referring to a cut by the set of edges that cross the cut, for which we have used the symbol X. In particular, in several cases where we wish to sum over the edges that cross a cut, we write $\sum_{e\in X}$.

3. THE ALGORITHM

Our algorithm is easy to state. With the possible exception of the first step, it ought to be correspondingly easy to implement. It is difficult to tell whether it might perform well in practice without actually implementing it.

```
1. run the Benczúr-Karger algorithm [1] to compute good lower
   bounds on the edge strengths, $k'_e$
2. $\alpha = 1$
3. while $\alpha n < m$
   (a) sampling according to weights $1/k'_e$, pick a sample of
       $\alpha n$ residual edges
   (b) search for an augmenting path in the sample
   (c) if no path is found, double $\alpha$
4. repeatedly search for augmenting paths in the residual graph
   until no more are found
```

The correctness of the algorithm is obviously guaranteed by the last step. Analysis of the running time is not obvious; it is the subject of the next section.

4. THE ANALYSIS

The foundation of our analysis is the following theorem, closely related to the main theorem of Benczúr and Karger:

THEOREM 4.1. *Let k'_e be a lower bound on the strength of edge e, as computed by the Benczúr-Karger algorithm. Let $\beta = \frac{6v\ln n}{v-|f|}$. If a sample of $4\beta n$ residual edges are chosen according to weights $1/k'_e$, then with high probability there is an augmenting path in the sample.*

Given the theorem, we can easily add up the times for each step to get the total running time. The first step requires $O(m\log^2 n)$ time, or $O(m\log^3 n)$ time if the input graph has capacities [1]. In each iteration of the loop we need to select $O(\beta n)$ edges and search for an augmenting path. Searching for an augmenting path takes only $O(\beta n)$ time. It is an easy matter to do the random sampling in $O(\beta n\log n)$ time. It is possible to give more elaborate schemes for sampling that achieve an amortized bound of $O(\beta n)$ time per iteration, but since we are not especially concerned about the logarithmic factors, we will not explain such a scheme here. So the time per iteration is $O(\beta n)$. This means that it takes $O(nv\log n)$ time to halve the remaining flow. We can double α only $\lg(m/n)$ times before we are simply finding augmenting paths in the entire residual graph, so the total time for the loop is $O(nv\log n\log m/n)$. When $\alpha = m/n$, it must be the case that $v/(v-|f|) = \Omega(m/n\log n)$, so $v-|f| = O((nv\log n)/m)$, which means that the time for the last step is only $O(nv\log n)$. Therefore, the total running time is $\tilde{O}(m+nv)$.

4.1 Supporting Lemmas

Before we can prove the main theorem, we need some supporting results. The first such is the result by Benczúr and Karger [1] that sampling an undirected graph with probability inversely proportional to strength preserves cut values well. The result that we need is slightly different from what they state, so we will state precisely what we need and provide a proof of it.

THEOREM 4.2. *In a connected undirected graph,*

$$\sum_{cutsX} \exp\left(-\sum_{e\in X}\frac{(d+2)\ln n}{k_e}\right) < \frac{d+2}{dn^d}$$

To prove this we also need a theorem due to Karger and Stein [11]:

THEOREM 4.3. *In an undirected graph with minimum cut value c, the number of cuts of value αc is at most $n^{2\alpha}$.*

We can now prove our restatement of the Benczúr-Karger result.

PROOF. Consider the weighted graph with the same vertices and edges as G and weight $w_e = (d+2)\ln n/k_e$ assigned to edge e. Order the cuts of this graph in increasing order, $c_1, c_2, \ldots, c_{2^{n-1}-1}$. So our goal is to bound

$$\sum_{i=1}^{2^{n-1}-1} e^{-c_i}$$

For any cut of the original graph, consider the maximum k_e of an edge crossing it. By definition of k_e there must be at least k_e such edges crossing the cut, so the value of the cut in the weighted graph is at least $(d+2)\ln n$. Since this is true of every cut (note that no cut has no edges), the minimum cut of the weighted graph must be at least $(d+2)\ln n$. By Theorem 4.3, $c_i \geq \frac{c_1\ln i}{2\ln n}$, so we can bound our sum as

$$\begin{aligned}
&\leq \sum_{i=1}^{n^2} e^{-c_1} + \sum_{i=n^2}^{2^{n-1}-1} e^{-\frac{c_1\ln i}{2\ln n}}\\
&\leq \sum_{i=1}^{n^2} e^{-(d+2)\ln n} + \sum_{i=n^2}^{\infty} e^{-\frac{(d+2)\ln i}{2}}\\
&\leq n^{2-(d+2)} + \sum_{i=n^2}^{\infty} i^{-1-d/2}\\
&\leq 1/n^d + \int_{x=n^2}^{\infty} x^{-1-d/2}dx\\
&\leq 1/n^d + \frac{n^{2(-d/2)}}{d/2}\\
&\leq \frac{d+2}{dn^d}
\end{aligned}$$

□

We also need a results from Benczúr and Karger [1] about the k'_e:

LEMMA 4.4.

$$\sum_e \frac{1}{k'_e} \leq 4n$$

The final supporting result we need is a little combinatorial lemma:

LEMMA 4.5. *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 $\sum_{i=j}^{l} x_i w_i \geq 0$.*

PROOF. Consider the largest j such that $\sum_{i=j}^{l} x_i \geq 0$. Rewrite $\sum_{i=j}^{l} x_i w_i$ as

$$\begin{aligned}
&w_i(x_i + \ldots + x_l)\\
&+ (w_{i+1} - w_i)(x_{i+1} + \ldots + x_l)\\
&+ (w_{i+2} - w_{i+1})(x_{i+2} + \ldots + x_l)\\
&+ (w_{i+3} - w_{i+2})(x_{i+3} + \ldots + x_l)\\
&+ \ldots
\end{aligned}$$

Or more succinctly, as

$$w_j \sum_{i=j}^{l} x_i + \sum_{k=j}^{l-1}\left((w_{k+1} - w_k)\sum_{i=k+1}^{l} x_i\right)$$

The first term is clearly non-negative by the choice of j. Observe that j being largest means that for all $k > j$ we have $\sum_{i=k}^{l} x_i < 0$, so the inner sum of the second term is always negative. Since the w_i are in decreasing order $w_{k+1} - w_k$ is also non-negative, which means that the second term is a sum of non-negative numbers. So the entire expression is non-negative. □

4.2 Proof of the Main Theorem

We now have all the pieces necessary to prove the main theorem. The basic idea is to compare the residual graph to the original graph, which Theorem 4.2 gives us a handle on. As a result of being a residual graph, at least a $\frac{v-|f|}{v}$ fraction of each cut is still available. What we are trying to say is that we can compensate for the edges saturated by the flow by increasing the sampling probability by a factor of $\frac{v}{v-|f|}$. Looking at an individual cut, this is not immediately obvious because the flow is not necessarily spread out evenly among the edges of different strengths. In particular, the flow might use up the weakest edges, in which case the probability that at least one edge crossing the cut is chosen can decrease significantly. However, if there is one edge that is k-strong then there must be at least k, so if the flow only uses up the weak edges, the many strong edges that remain will be sufficient to make it very likely that some edge crossing the cut is chosen. We will use Lemma 4.5 to formalize this idea. We now give the full details.

PROOF. Consider a cut X in the residual graph. Group the edges of the cut by strength, and order the groups in increasing order by strength. Associate with the ith group $w_i = \beta/k_e$ and $x_i =$ the number of residual edges of that strength minus $\frac{v-|f|}{v}$ times the original number of edges of that strength. Thus the sum of the x_i is the residual capacity of the cut minus $\frac{v-|f|}{v}$ times the original capacity of the cut. The residual capacity of every cut is at least a $\frac{v-|f|}{v}$ fraction of the original capacity, so this quantity is always non-negative. Applying Lemma 4.5 we find that there exists a j such that $\sum_{i=j}^{l} x_i w_i \geq 0$; or rephrased in terms of the graph, we find that there exists a k such that the sum over residual k-strong edges of $\beta/k_e \geq$ sum over original k-strong edges of $(6\ln n)/k_e$.

Since, by Lemma 4.4 the total weight of edges is at most $4n$, the probability that we fail to choose any of these k-strong edges is

$$\begin{aligned}
&\leq (1 - \frac{\sum_{\text{residual } k\text{-strong } e\in X} 1/k'_e}{4n})^{4\beta n} \\
&\leq \exp\left(-\sum_{\text{residual } k\text{-strong } e\in X} \frac{\beta}{k'_e}\right) \\
&\leq \exp\left(-\sum_{\text{residual } k\text{-strong } e\in X} \frac{\beta}{k_e}\right) \\
&\leq \exp\left(-\sum_{\text{original } k\text{-strong } e\in X} \frac{6\ln n}{k_e}\right)
\end{aligned}$$

So with each cut we associate a strength k and a k-strong component such that the probability we fail to choose an edge from the k strong component is at most $\exp(-\sum_{e\in X} \frac{6\ln n}{k_e})$. By Theorem 4.2, summing this quantity over all cuts that have the same k and component is at most $1.5/n^4$. Since there can be only m distinct values of k, and at most n components associated with any k, the probability that we fail to choose an edge from any cut is at most $1.5/n$

If every s-t cut in the residual graph has a residual edge crossing it, then by the maxflow-mincut theorem there is an augmenting path. $\square$

5. CONCLUSION

Our result of $\tilde{O}(m + nv)$ time is a natural stopping point, but it is not necessarily the end of progress on algorithms for small maximum flows in undirected graphs. For one thing, it is randomized, so there is still the question of how well a deterministic algorithm can do. Perhaps there is some way to apply Nagamochi-Ibaraki sparse certificates [12] in a residual graph. For another, Galil and Yu showed that flows need only use $O(n\sqrt{v})$ edges on simple graphs. Therefore, while some augmenting paths can require $n-1$ edges, most of them are much shorter. Thus $\tilde{O}(m + n\sqrt{v})$ would be another natural time bound to hope to achieve. And of course one can always hope for linear time.

Another open question is whether it is possible to give a faster algorithm for small flows in directed graphs. In sampling from residual graphs, we have shown that random sampling in directed graphs is not entirely hopeless. Perhaps there is a suitable replacement for edge strength in a directed graph that would allow random sampling in general.

Finally, it is still an open question whether there are better algorithms for finding flows in undirected graphs with large flow values. One obvious goal would be to replace m by n in general. We note that it is possible to prove a compression theorem for a residual graph—that sampling with probabilities proportional to $1/k_e$ and multiplying up the capacity of sampled edges by k_e preserves all cut values reasonably well—but it is not clear to us that this is actually helpful.

6. REFERENCES

[1] A. A. Benczúr and D. R. Karger. Approximate s–t min-cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the* 28th *ACM Symposium on Theory of Computing*, pages 47–55. ACM, ACM Press, May 1996.

[2] S. Even and R. E. Tarjan. Network Flow and Testing Graph Connectivity. *SIAM Journal on Computing*, 4:507–518, 1975.

[3] L. R. Ford, Jr. and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[4] A. Goldberg and S. Rao. Beyond the flow decomposition barrier. In *Proceedings of the* 30th *Annual Symposium on the Foundations of Computer Science* [6], pages 2–11.

[5] A. Goldberg and S. Rao. Flows in undirected unit capacity networks. In *Proceedings of the* 30th *Annual Symposium on the Foundations of Computer Science* [6], pages 32–35.

[6] IEEE. *Proceedings of the* 30th *Annual Symposium on the Foundations of Computer Science*. IEEE Computer Society Press, Oct. 1997.

[7] D. R. Karger. Using random sampling to find maximum flows in uncapacitated undirected graphs. In *Proceedings of the* 29th *ACM Symposium on Theory of Computing*, pages 240–249. ACM, ACM Press, May 1997.

[8] D. R. Karger. Better random sampling algorithms for flows in undirected graphs. In H. Karloff, editor, *Proceedings of the* 9th *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 490–499. ACM-SIAM, Jan. 1998.

[9] D. R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24(2):383–413, May 1999. A preliminary version appeared in Proceedings of the 26th ACM Symposium on Theory of Computing.

[10] D. R. Karger and M. Levine. Finding maximum flows in simple undirected graphs seems faster than bipartite matching. In *Proceedings of the* 29th *ACM Symposium on Theory of Computing*, pages 69–78, New York, May 23–26 1998. ACM, ACM Press.

[11] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, July 1996. Preliminary portions appeared in SODA 1992 and STOC 1993.

[12] H. Nagamochi and T. Ibaraki. Linear time algorithms for finding k-edge connected and k-node connected spanning subgraphs. *Algorithmica*, 7:583–596, 1992.