THE SAMPLING PARADIGM OF KARGER

Y. KOHAYAKAWA

§0. INTRODUCTION AND DISCLAIMER

In the past couple of years, two simple observations of Karger have been exploited by a few authors with quite surprising results. Our aim here is to say a few words on what the observations are and to discuss its consequences. We shall not try to be comprehensive; instead, we shall concentrate on the basics with the hope that the interested readers might be at a good starting point when turning to the literature, which seems to be growing quite rapidly.

At this point I should like to insert a disclamer: I have been actively interested in this area only quite recently, and I do not claim to present new ideas or insights here. The purpose of this short note is to stir up the readers' interest in Karger's method, with the hope that members of Project ProComb and possibly others will work on their problems with this new tool in hand.

In the sequel, we shall discuss both algorithms and purely combinatorial results. A common feature of the topics below is that probability will always be lurking in the background. In particular, we shall concentrate on randomised algorithms, although derandomisation has proved to fit in nicely in parts of what follows.

Let us recall the two most common types of randomised algorithms. A *Monte Carlo* type algorithm of time complexity $T(n)$ finishes its computation in time not exceeding $T(n)$ for inputs of size n, but might give an erroneous answer with some small probability, usually approaching 0 as $n \to \infty$. On the other hand, a *Las Vegas* type algorithm of time complexity $T(n)$ finishes its computation in time $T(n)$ with high probability (usually $1 - o(1)$ where the $o(1)$ term refers to $n \to \infty$), and its output is always correct.

For conciseness, we shall always assume that G is a graph of order $|G| = |V(G)| = n$ and size $e(G) = |E(G)| = m$. Our graphs will be allowed to have multiple edges.

§1. MINIMUM SPANNING TREES

A recent success in the theory of randomised computing was the development of a Las Vegas *linear* time algorithm for finding a minimum spanning tree in an edge-weighted graph G. The currently best deterministic algorithm, due to Gabow, Galil, and Spencer [3] (see also [4]), runs in time $O(m \log \beta(m, n))$, where $\beta(m, n) = \min\{i: \log^{(i)} n \leq m/n\}$ is the inverse Ackerman function. Thus here we have another piece of evidence that randomisation does help.

Work partially supported by FAPESP (Proc. 93/0603-1) and by CNPq (Proc. 300334/93-1 and Project ProComb).

The linear time algorithm above was developed by Klein and Tarjan [11], following a simple but powerful observation of Karger [6].

Theorem 1. *There is a Las Vegas algorithm that takes a connected edge-weighted graph G as input and outputs a minimum spanning tree for G in time $O(m)$ with probability $1 - \exp\{-m/\log^{O(1)} m\}$.*

In the result above, the model of computation is that of a unit-cost random-access machine with the restriction that only binary comparisons are allowed with the edge weights. Putting it very crudely (in fact, in an *oversimplified* manner), the algorithm of Klein and Tarjan behaves as follows: first, it randomly chooses a half of the edges of G to be discarded, and it concentrates its attention on the remaining edges. Thus the graph G is reduced to a graph G' of size roughly $e(G)/2$. The algorithm is then called recursively with input G'. The output T' of this recursive call is not necessarily a minimum spanning tree of G, and hence in the rest of the algorithm it is worked on to become one such tree T: here the crucial point is that with very high probability very little work will be needed to obtain T from T'. Thus, in short, the algorithm is based on *sampling* the edges of G with the aim of picking a substantially smaller subgraph G' that is a good 'picture' of G.

The key result connecting the cost of T' and the actual cost of a minimum spanning tree of G is a 'sampling lemma' of Klein and Tarjan. To state this result, let us say that an edge e of G is *T'-light* if it is either contaned in T' or else if it is not contained in T' but its weight is no greater than the weight of some edge of T' in the fundamental cycle of e with respect to T'. The sampling lemma of Klein and Tarjan [11] is as follows. We may and shall assume that the edge weights of our graph G are all distinct.

Theorem 2. *Let G' be obtained from the edge-weighted graph G by independently removing each edge of G with probability $1 - p$, and let F' be a minimum spanning forest of G'. Then, for any k, the probability that the number of F'-light edges in G exceeds k is at most*

$$\sum_{0 \le i \le n-2} \binom{k}{i} p^i (1-p)^{k-i}.$$

An immediate corollary of the sampling lemma is that the expected number of F'-light edges is at most n/p, and that the probability that the number of such edges exceeds $(1+\varepsilon)n/p$ is $\exp\{-\Omega(\varepsilon^2 n)\}$ for any $0 \le \varepsilon \le 1$.

We close this section with the following remark. The brief description of the algorithm of Klein and Tarjan above is far from complete: the sampling lemma, the algorithm sketched above, and some standard ideas only give an algorithm of time complexity $O(m + n \log n \log\log n)$. The linear algorithm actually makes use of an operation known as the Borůvka reduction and the transformation of T' into T involves the use of sophisticated data structures. Our aim in the presentation above was to focus on the sampling paradigm of Karger, which in this particular case amounts to the use of Theorem 2.

§2. Cuts, Flows, and Network Design

2.1 Cuts. Edge-contraction has been used by Nagamochi and Ibaraki [12] and Matula [13] to develop fast algorithms for determining or estimating the edge

connectivity or, equivalently, the minimum cut of a given graph. A simple but important remark of Karger [5] connected to edge-contraction and minimum cuts is best illustrated by an algorithm for finding a minimum cut in a given unweighted graph G. The discussion that follows concentrates on the unweighted case, but similar results may be obtained for weighted graphs. In §2, we shall always assume that G has minimum cut c. Here is the algorithm of Karger: given a graph G, pick an edge e of G uniformly at random and contract e. Remove all loops that arise. Proceed contracting the edges of our graph and removing loops until only two vertices remain. The cut naturally corresponding to the resulting 2-vertex graph is the output of the algorithm.

Of course, the algorithm above does not have much chance of producing a minimum cut of G unless one is lucky, but in fact one need not to be *too* lucky. This is the key observation of Karger.

Theorem 3. *Fix a minimum cut C of G. The algorithm above produces C as output with probability at least $\binom{n}{2}^{-1}$.*

Proof. Note first that the cut C is the output of our algorithm if and only if no edge of C is contracted in the process. Suppose we are executing our procedure and we have so far contracted t edges ($0 \le t < n-2$). The current graph G_t has minimal degree at least $|C|$, as contraction of edges does not decrease the minimum cut of our graph. Thus G_t has at least $|C|(n-t)/2$ edges. The probability that we chose an edge from C to be contracted is thus not larger than $2/(n-t)$. Therefore the probability that an edge of C is never chosen to be contracted is at least

$$\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\cdots\left(1-\frac{2}{3}\right)=\binom{n}{2}^{-1},$$

as required. □

An immediate corollary to the result above is a polynomial time Monte Carlo algorithm for finding a minimum cut of G. More interestingly, one deduces from Theorem 3 that any graph G on n vertices has at most $\binom{n}{2}$ minimum cuts. Karger in fact proves the following general result.

Theorem 4. *Let G be a graph with minimum cut c, and let $\alpha \ge 1$ be a real number. Then the number of cuts in G of cardinality at most αc is at most $n^{2\alpha}$.*

A moment's thought suggests that the result above must be very useful in sampling. Suppose we delete each edge of a graph G independently with probability $1-p$, and let G_p be the resulting graph. In this process, *any* fairly small cut C of G is bound to be reduced to a cut of size precisely $(1+o(1))p|C|$ provided only that the minimum cut of G is a little more than of logarithmic size: this is an immediate consequence of Theorem 4 and the exponential decay of the tail of the binomial distribution. A more careful argument gives the following result of Karger [7].

Theorem 5. *Let G be an n-vertex graph whose minimum cut has value c, let $p = 2(d+2)(\log n)/\varepsilon^2 c$, where d is some fixed constant, and suppose $0 < p < 1$. Then the*

cardinality c_p of a minimum cut of G_p satisfies $|c_p - pc| \le \varepsilon pc$ with probability $1 - O(n^{-d})$.

Theorem 5 suggests a simple algorithm for approximating the value of a minimum cut of a given graph G. Clearly, results similar to the above theorem may be proved for minimum s–t cuts for any fixed s and $t \in G$.

Theorem 6. *Let G be an n-vertex graph and fix s and $t \in G$. Suppose that G has minimum cut c and that the minimum s–t cut in G has value v. Suppose $0 < p < 1$ satisfies $p = \Theta((\log n)/\varepsilon^2 c)$. Then, with probability $1 - o(1)$ as $n \to \infty$, the value v_p of the minimum s–t cut in G_p is such that $|v_p - pv| \le \varepsilon pv$.*

Theorem 6 gives a Monte Carlo algorithm for finding a $(1+\varepsilon)$-approximation for the minimum s–t cut in a given graph G that runs in time roughly $mv/\varepsilon^4 c^2$. Faster algorithms were found by Karger, elaborating on the ideas presented in the sequel. Finally, we remark that, using more sophisticated techniques, Karger has obtained a Las Vegas algorithm for computing the value c of a minimum cut in time $\tilde{O}(mc^{1/2})$. (Here and in the sequel, $\tilde{O}(f)$ stands for $O(f \operatorname{polylog} f)$.)

2.2 Las Vegas algorithms for maximum flow. The results of the previous section have direct consequences in the study maximum flows in graphs. Consider the following randomised divide-and-conquer algorithm for finding a maximum family of edge-disjoint paths between two fixed vertices s and t of a given graph G. We first split G into two edge-disjoint spanning subgraphs G_1 and G_2, randomly colouring the edges of G red and blue. We then recursively run the algorithm with input $(G_i; s, t)$ $(i \in \{1,2\})$, put together the two families of s–t paths to form a family $\mathcal{F}'$, and use an algorithm for augmenting paths to obtain an optimal family $\mathcal{F}$. The crucial observation is that $\mathcal{F}'$ will actually be very close to being optimal as long as G has reasonably high edge-connectivity. Indeed, Theorem 6 implies that $|\mathcal{F}| - |\mathcal{F}'|$ is bounded by $\tilde{O}(|\mathcal{F}|c^{-1/2})$.

Theorem 7. *The procedure above gives a Las Vegas algorithm that finds a maximum edge-disjoint family of s–t paths in the graph G in time $\tilde{O}(m|\mathcal{F}|c^{-1/2})$.*

2.3 Network design and other problems. We start with 'other problems'. We only wish to point out that, given a $2k$-edge-connected graph G, a random orientation of G is very likely to have edge-connectivity $(1 - o(1))k$, provided only that k is superlogarithmic. The following is a more precise statement that may be proved with the aid of Theorem 4.

Theorem 8. *There exists a linear time Monte Carlo type algorithm for finding a $(k - O\{(k \log n)^{1/2}\})$-edge-connected orientation of any n-vertex $2k$-edge-connected graph.*

The algorithm above may be used as a component of a Las Vegas algorithm that runs faster than the currently best deterministic algorithm for this problem, due to Gabow [2]. The speeding up factor is of order $\tilde{O}(k^{1/2})$.

We now turn to network designs. It is best if we restrict our attention on one specific problem. Suppose a graph G is given. We wish to find a spanning k-edge-connected subgraph of G with the smallest possible number of edges. This problem is NP-complete even for $k = 2$. However, a *fractional* solution to our problem may

be found in polynomial time through the use of interior point algorithms, as shown by Williamson, Goemans, Mihail, and Vazirani [14]. In the fractional solution, we assign to each edge $e \in E(G)$ a weight $0 \le p_e \le 1$. Interpreting the edge weights as probabilities, we may obtain a random subgraph of G that is likely to be close to an optimal solution as long as k is superlogarithmic. Specifically, this 'randomised rounding' procedure gives the following result.

Theorem 9. *There exists a polynomial time Las Vegas algorithm for finding a* $(1 + O\{((\log n)/k)^{1/2}\})$*-approximation to the minimum* k*-connected subgraph.*

The proof of Theorem 9 above is based on an obvious variant of Theorem 5, on a simple 2-approximation algorithm for our problem (to transform the almost optimal solution given by the random rounding procedure into an optimal one), and, of course, on the interior point method of [14].

§3. Matroids

Theorem 2 has an immediate generalisation to matroids. Thus, the problem of finding a minimum cost basis in a matroid may be approached through sampling. Since the ideas involved in this more general setting are quite similar to the ones briefly discussed in §1, we shall not dwell on them. We shall, however, mention two further results of Karger [6] concerning random submatroids of a matroid.

Let M be a matroid on S, and write $\rho_M = \rho(M)$ for the rank of M. Suppose $0 \le p \le 1$ is given, and let M_p be the random matroid obtained from M by restricting M to S_p, the random subset of S in which each element of S is independently present with probability p. Thus to obtain M_p we simply delete elements from M independently with probability $1 - p$. Let $\pi(M)$ be the *packing number* of M, *i.e.*, the maximal number of pairwise disjoint bases it contains.

Theorem 10. *Let* $0 < p < 1$ *be given and suppose the matroid* M *has packing number* $\pi(M) = a + 2 + p^{-1}\log\rho(M)$. *Then* M_p *contains a basis of* M *with probability* $1 - O(e^{-pa})$.

We remark that the result above is strong enough to give the right order of magnitude for the connectivity threshold for the binomial random graph $G_{n,p}$. Indeed, Theorem 10 shows that $G_{n,p}$ is almost surely connected if $p = p(n) = (2 + \varepsilon)(\log n)/n$ and $\varepsilon > 0$ is a fixed constant.

In view of Theorem 10, one may ask whether if M has a large packing number, then M_p has a large packing number as well or, more precisely, whether M_p contains many pairwise disjoint bases of M. The following result of Karger answers this question.

Theorem 11. *Let* M *and* p *be as above, and write* $\pi'(M_p)$ *for the maximal number of pairwise disjoint bases of* M *that* M_p *contains. Then, for any* $0 \le \varepsilon \le 1$, *we have*

$$\mathbb{P}\{|\pi'(M) - p\pi(M)| > \varepsilon p\pi(M)\} < \rho(M)\exp\left\{-\frac{1}{2}\varepsilon^2 p\pi(M)\right\}.$$

An immediate corollary to Theorem 11 is the following.

Theorem 12. *Let $0 < p = p(n) < 1$ be given and suppose the n-vertex graph G contains π disjoint spanning trees. If π' denotes the maximal number of pairwise disjoint spanning trees of G that G_p contains, then we have $|\pi' - p\pi| \le 2(p\pi \log n)^{1/2}$ with probability $1 - o(1)$ as $n \to \infty$.*

§4. Concluding Remarks

We hope that the usefulness of Theorems 2 and 4 have been well illustrated by the above discussion. Further applications and extensions may be found in the work of Karger and others. In particular, many of the algorithms based on sampling are parallelisable (see, *e.g.*, Cole, Klein, and Tarjan [1]) and, furthermore, derandomisation has been achieved for a minimum cut algorithm, showing that determining a minimum cut in a graph is in NC (see Karger and Motwani [9]).

Perhaps more purely combinatorial results along the lines of Theorems 10 and 11 should be sought. Moreover, perhaps difficult algorithmic problems for matroids could be tackled with these methods; *e.g.*, can the intersection problem for three matroids be solved through sampling?

References

1. Cole, R., Klein, P.N., Tarjan, R.E., *A linear-work parallel algorithm for finding minimum spanning trees*, SPAA, 1994.
2. Gabow, H.N., *A frame-work for cost-scaling algorithms for submodular flow problems*, 34th FOCS, 1993, pp. 449–458.
3. Gabow, H.N., Galil, Z., Spencer T., *Efficient implementation of graph algorithms using contraction*, 25th FOCS, 1984, pp. 347–357.
4. Gabow, H.N., Galil, Z., Spencer T., Tarjan, R.E., *Efficient algorithms for finding minimum spanning trees in undirected and directed graphs*, Combinatorica **6** (1986), 109–122.
5. Karger, D.R., *Global min-cuts in $\mathcal{RNC}$, and other ramifications of a simple min-cut algorithm*, 4th SODA, 1993, pp. 21–30.
6. Karger, D.R., *Random sampling in matroids, with applications to graph connectivity and minimum spanning trees*, 34th FOCS, 1993, pp. 84–93.
7. Karger, D.R., *Random sampling in cut, flow, and network design problems*, 26th STOC, 1994, pp. 648–657.
8. Karger, D.R., *Using randomised sparsification to approximate minimum cuts*, 4th SODA, 1994.
9. Karger, D.R., Motwani, R., *Derandomization through approximation: an $\mathcal{NC}$-algorithm for minimum cuts*, 26th STOC, 1994, pp. 497–506.
10. Karger, D.R., Stein, C., *An $\tilde{O}(n^2)$ algorithm for minimum cuts*, 25th STOC, 1993, pp. 757–765.
11. Klein, P.N., Tarjan, R.E., *A randomized linear-time algorithm for finding minimum spanning trees*, 26th STOC, 1994, pp. 9–15.
12. Nagamochi, H., Ibaraki, T., *Computing edge connectivity in multigraphs and capacitated graphs*, SIAM J. Discrete Math. **5** (1992), 54–66.
13. Matula, D.W., *A linear $2+\varepsilon$ approximation algorithm for edge connectivity*, 4th SODA, 1993, pp. 500–504.
14. Williamson, D., Goemans, M.X., Mihail, M., Vazirani, V.V., *A primal-dual approximation algorithm for generalized Steiner problems*, 25th STOC, 1993, pp. 708–717.

Instituto de Matemática e Estatística, Universidade de São Paulo, Caixa Postal 20570, 01452-990 São Paulo SP, Brazil

E-mail address: <yoshi@ime.usp.br>