A representation and query structure to approximate mincuts of a network

András A. Benczúr*
Department of Mathematics, M. I. T.

1 Introduction

We describe a data structure that handles all *approximate* mincuts of a network. Handling approximate mincuts can be very important in network reliability problems. Let us consider a network with various capacities over the connections, in which it is crucial that the minimum throughput between any pair of nodes be large. This minimum value equals the minimum capacity of edges through a cut, over all cuts of the network. The "ancient" cactus representation [3] of all mincuts contains precisely these cuts. Now assume some failure or congestion can block part or all of the capacity of an edge, independently for all edges, with some known probability. Karger [5] showed that in this case, the value of all cuts stays close to their expected value. However, by random fluctuations, non-minimal but close to minimal cuts may become mincuts after failures. Our representation contains all cuts that may become a connectivity bottleneck after edge failures.

Our representation is a simple geometric structure consisting of polygons with some of their diagonals drawn in the plane. Our approach is entirely new in relating edge connectivity to geometry. The representation show that the structure of possible bottlenecks in general networks with edge failures is close to that in planar networks. The representation can be very efficiently built, in $O(n^2 \log^3 n)$ time, which is faster than the time to find a single s–t mincut.

Considering approximate mincuts can also be useful in algorithms related to various network design problems. Often, mincuts have to be repeatedly found in a dynamically changing graph. Then by computing all approximate mincuts instead of mincuts in a step, we may not have to recompute them in subsequent steps. In our full paper, we improve on the best known algorithms for various problems, e.g. those of [1, 4]. A further example where the knowledge of a representation of approximate mincuts may be useful is the cutting-plane algorithms for solving the traveling salesman problem [7, 2].

*E-mail: `benczur@theory.lcs.mit.edu`. Address: Lab. for Computer Science #346, 545 Technology Square, Cambridge MA 02139. Author is a student in the Theory of Computation Group, supported by DARPA contract N00014-92-J-1799 and NSF contract 9302476-CCR.

2 Why are we better than classical flows?

A "classical" network flow algorithm finds a minimum capacity cut separating two given vertices, a source s and a sink t. We call such a cut an s–t mincut. In comparison, we measured network reliability by the *global* mincut capacity, which can be defined as the minimum of s–t mincuts, over all source–sink pairs. An algorithmic advantage of (approximate) mincuts is that finding even *all* of them seems to be easier than finding a single s–t mincut. The randomized algorithm of Karger and Stein [6] for finding approximate mincuts "...breaks the $\Omega(nm)$ max-flow barrier for finding mincuts ...[and] suggests that the [global] min-cut problem might be fundamentally easier to solve..." [6]. While there are efficient parallel algorithms for finding (approximate) mincuts [6], this is unlikely for s–t mincuts.

One may ask why our representation is necessary, when the Karger–Stein algorithm already "finds" all approximate mincuts. Notice however that there can be as many as $\Omega(n^2)$ mincuts of a graph [3, 6], requiring $\Omega(n^3)$ space to list all of them. Given this list, we have to scan the entire list to answer a non-trivial question about the cuts. The space of this list exceeds the roughly $O(n^2)$ running time of the Karger–Stein algorithm; consequently, that algorithm outputs an unstructured, data compressed version of the cut list. Algorithms for transforming the output of the Karger–Stein algorithm to a cactus representation of mincuts, without exceeding the asymptotic running time of the Karger–Stein algorithm, were given independently in [6] and [1]. By the results of this paper, applications requiring a structured knowledge of approximate mincuts can take advantage of the fast running time of the Karger–Stein algorithm as well.

3 The representation: Geometry and beyond

Let c be the minimum capacity of a cut in a network $G = (V, E)$. We represent the sides of all cuts with capacity less than $6/5 \cdot c$. The representation has the following geometric description. It consists of a collection of convex *representing polygons*, with some of their diagonals drawn by straight lines in the plane (see Fig. 1). Vertices of the graph are partitioned into *atoms*, and to each atom, there is a corresponding geometric position within the polygon. Each diagonal divides the

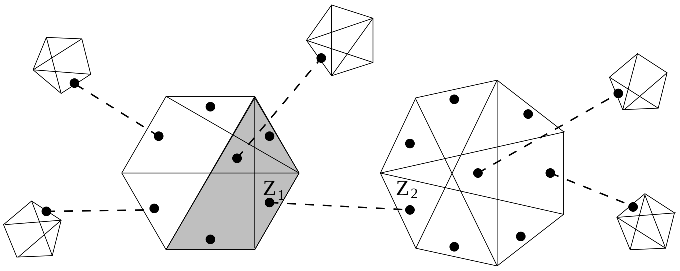

Figure 1: A collection of representing polygons. Dots denote the positions of the *atoms*. The darkened region is the side of a diagonal, representing the union of the four atoms in that region. Dashed edges are *hierarchy-edges*; one of them is between atoms Z_1 and Z_2.

polygon into two *sides*. The representation consists of diagonal sides; the diagonal sides represent sides of cuts as a Venn diagram. The arrangement of the diagonals divides this diagram into *cells*. Cells are equivalence classes for the positions of atoms; we may assign atoms to cells (some of which do not contain atoms).

Representing polygons form a tree-like hierarchical collection (Fig. 1). We put *hierarchy edges* between pairs of atoms Z_1 and Z_2 of distinct polygons $\mathcal{P}_1$ and $\mathcal{P}_2$. We require that any pair of atoms Z_1 and Z_2 connected by a hierarchy-edge has $Z_1 \cup Z_2 = V$. We may think of $\mathcal{P}_2$ as a refinement within atom $Z_1 \in \mathcal{P}_1$, and similarly $\mathcal{P}_1$ within Z_2. Finally, we require that hierarchy edges form a tree over the collection of polygons as vertices.

In general, the arrangement of cells is dependent on the geometric position of the vertices of the polygon. By giving different coordinates, some cells described by intersecting diagonal sides may vanish, or new ones may appear. Thus it is not necessarily true that the positions of all atoms can be found, by arbitrarily positioning the polygon-vertices. However, we prove a special structure of approximate mincuts that force all atoms to belong to cells that cannot vanish. We could assume, for example, that the representing polygons are regular, or that all of their vertices have integer coordinates. Fixing such an actual geometric position of the polygons will be important in algorithmically handling the representation.

4 Algorithmic aspects

Our (randomized) algorithm for building our representation of approximate mincuts is based on a careful analysis of the Karger–Stein algorithm [6]. (Note that a deterministic, though not very efficient, algorithm can also be given.) The Karger–Stein algorithm builds a binary tree of graphs, by recursively contracting graphs as we move farther from the root. To achieve the very efficient running time, it is crucial that for an intermediate graph G' with n' vertices, we spend only $O({n'}^2)$ time for contracting G'. But in that case if $n' = o(n)$, we are not allowed to carry direct pointers to vertices of the *original* graph. The idea of our algorithm is to build the representation up the binary tree given by the Karger-Stein algorithm: in each step, the cut representations of the two different contracted graphs are merged as the vertices of the graph get "unwrapped".

Finally, we sketch a data structure to handle the polygon representation. The difficulty is that even within a single $\Omega(n)$-edge polygon, the entire cell arrangement may have size $\Omega(n^4)$. Thus using usual geometric search structures, scanning through all cells, is out of question. Instead, after fixing the vertices of the polygon, for each atom Z we give its coordinates in the plane, and a pointer to each of the $O(n)$ diagonals representing *minimal* sets containing Z.

Some examples of queries that we can efficiently answer are given next. Given a diagonal, we can obtain the sides of the corresponding cut in time $O(n)$ by comparing coordinates. Given two atoms, we can find a cut separating them by only looking at the list of minimal diagonal sides containing one of the two atoms. Edges entering an atom are ordered cyclically; small cuts may not cross edges that are too far apart in this ordering. Thus the simultaneous failure of far-apart edges are less dangerous that that of close ones.

References

[1] Benczúr, A.A., Augmenting edge connectivity in $\tilde{O}(n^3)$ time, *Proc. 26th Annual ACM Symp. on Theory of Comp.* (1994), 658–667.

[2] Cook, W., lecture, *Summer School on Combinatorial Optimization*, Maastricht, August 1993.

[3] Dinits, E.A., A.V. Karzanov and M.L. Lomonosov, On the structure of a family of minimal weighted cuts in a graph, *Studies in Discrete Optimization* (in russian), A.A. Fridman (Ed), Nauka, Moscow (1976), 290–306.

[4] Gabow, H.N., Efficient Splitting Off Algorithms for Graphs, *Proc. 26th Annual ACM Symp. on Theory of Comp.* (1994), 696–705.

[5] Karger, D.R., Random sampling in cut, flow and network design problems, *Proc. 26th Annual ACM Symp. on Theory of Comp.* (1994).

[6] Karger, D.R. and C. Stein, A new approach to the mincut problem, *draft journal version*. Also *Proc. 25th Annual ACM Symp. on Theory of Comp.* (1993), 757–765.

[7] Padberg, M. and G. Rinaldi, Facet identification for the symmetric traveling salesman polytope, *Math. Programming* **47** (1990), 85–136.