A Simple Min-Cut Algorithm

MECHTHILD STOER

Televerkets Forskningsinstitutt, Kjeller, Norway

AND

FRANK WAGNER

Freie Universität Berlin, Berlin-Dahlem, Germany

Abstract. We present an algorithm for finding the minimum cut of an undirected edge-weighted graph. It is simple in every respect. It has a short and compact description, is easy to implement, and has a surprisingly simple proof of correctness. Its runtime matches that of the fastest algorithm known. The runtime analysis is straightforward. In contrast to nearly all approaches so far, the algorithm uses no flow techniques. Roughly speaking, the algorithm consists of about $|V|$ nearly identical phases each of which is a *maximum adjacency search*.

Categories and Subject Descriptors: G.L.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*

General Terms: Algorithms

Additional Key Words and Phrases: Min-Cut

1. *Introduction*

Graph connectivity is one of the classical subjects in graph theory, and has many practical applications, for example, in chip and circuit design, reliability of communication networks, transportation planning, and cluster analysis. Finding the minimum cut of an undirected edge-weighted graph is a fundamental algorithmical problem. Precisely, it consists in finding a nontrivial partition of the graphs vertex set V into two parts such that the *cut weight*, the sum of the weights of the edges connecting the two parts, is minimum.

A preliminary version of this paper appeared in *Proceedings of the 2nd Annual European Symposium on Algorithms*. Lecture Notes in Computer Science, vol. 855, 1994, pp. 141–147.

This work was supported by the ESPRIT BRA Project ALCOM II.

Authors' addresses: M. Stoer, Televerkets Forskningsinstitutt, Postboks 83, 2007 Kjeller, Norway; e-mail: mechthild.stoer@nta.no.; F. Wagner, Institut für Informatik, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Takustraße 9, Berlin-Dahlem, Germany; e-mail: wagner@inf.fu-berlin.de.

The usual approach to solve this problem is to use its close relationship to the maximum flow problem. The famous Max-Flow-Min-Cut-Theorem by Ford and Fulkerson [1956] showed the duality of the maximum flow and the so-called minimum s-t-cut. There, s and t are two vertices that are the source and the sink in the flow problem and have to be separated by the cut, that is, they have to lie in different parts of the partition. Until recently all cut algorithms were essentially flow algorithms using this duality. Finding a minimum cut without specified vertices to be separated can be done by finding minimum s-t-cuts for a fixed vertex s and all $|V| - 1$ possible choices of $t \in V \setminus \{s\}$ and then selecting the lightest one.

Recently Hao and Orlin [1992] showed how to use the maximum flow algorithm by Goldberg and Tarjan [1988] in order to solve the minimum cut problem in time $\mathcal{O}(|V||E|\log(|V|^2/|E|))$, which is nearly as fast as the fastest maximum flow algorithms so far [Alon 1990; Ahuja et al. 1989; Cheriyan et al. 1990].

Nagamochi and Ibaraki [1992a] published the first deterministic minimum cut algorithm that is not based on a flow algorithm, has the slightly better running time of $\mathcal{O}(|V||E| + |V|^2 \log|V|)$, but is still rather complicated. In the unweighted case, they use a fast-search technique to decompose a graph's edge set E into subsets $E_1, \ldots, E_\lambda$ such that the union of the first k E_i's is a k-edge-connected spanning subgraph of the given graph and has at most $k|V|$ edges. They simulate this approach in the weighted case. Their work is one of a small number of papers treating questions of graph connectivity by non-flow-based methods [Nishizeki and Poljak 1989; Nagamochi and Ibaraki 1992a; Matula 1992]. Karger and Stein [1993] suggest a randomized algorithm that with high probability finds a minimum cut in time $\mathcal{O}(|V|^2 \log|V|)$.

In this context, we present in this paper a remarkably simple deterministic minimum cut algorithm with the fastest running time so far, established in Nagamochi and Ibaraki [1992b]. We reduce the complexity of the algorithm of Nagamochi and Ibaraki by avoiding the unnecessary simulated decomposition of the edge set. This enables us to give a comparably straightforward proof of correctness avoiding, for example, the distinction between the unweighted, integer-, rational-, and real-weighted case.

This algorithm was found independently by Frank [1994].

Queyranne [1995] generalizes our simple approach to the minimization of submodular functions.

The algorithm described in this paper was implemented by Kurt Mehlhorn from the Max-Planck-Institut, Saarbrücken and is part of the algorithms library LEDA [Mehlhorn and Näher 1995].

2. *The Algorithm*

Throughout the paper, we deal with an ordinary undirected graph G with vertex set V and edge set E. Every edge e has nonnegative real weight $w(e)$.

The simple key observation is that, if we know how to find two vertices s and t, and the weight of a minimum s-t-cut, we are nearly done:

THEOREM 2.1. *Let s and t be two vertices of a graph G. Let $G/\{s, t\}$ be the graph obtained by merging s and t. Then a minimum cut of G can be obtained by taking the smaller of a minimum s-t-cut of G and a minimum cut of $G/\{s, t\}$.*

The theorem holds since either there is a minimum cut of G that separates s and t, then a minimum s-t-cut of G is a minimum cut of G; or there is none, then a minimum cut of $G/\{s, t\}$ does the job.

So a procedure finding an *arbitrary* minimum s-t-cut can be used to construct a recursive algorithm to find a minimum cut of a graph.

The following algorithm, known in the literature as *maximum adjacency search* or *maximum cardinality search*, yields the desired s-t-cut.

```
MinimumCutPhase(G, w, a)
A ← {a}
while A ≠ V
add to A the most tightly connected vertex
store the cut-of-the-phase and shrink G by merging the two vertices added last
```

A subset A of the graphs vertices grows starting with an arbitrary single vertex until A is equal to V. In each step, the vertex outside of A *most tightly connected* with A is added. Formally, we add a vertex

$$z \notin A \text{ such that } w(A, z) = \max\{w(A, y) | y \notin A\},$$

where $w(A, y)$ is the sum of the weights of all the edges between A and y. At the end of each such phase, the two vertices added last are *merged*, that is, the two vertices are replaced by a new vertex, and any edges from the two vertices to a remaining vertex are replaced by an edge weighted by the sum of the weights of the previous two edges.

Edges joining the merged nodes are removed.

The cut of V that separates the vertex added last from the rest of the graph is called the *cut-of-the-phase*. The lightest of these cuts-of-the-phase is the result of the algorithm, the desired minimum cut:

```
MinimumCut(G, w, a)
while |V| > 1
  MinimumCutPhase(G, w, a)
  if the cut-of-the-phase is lighter than the current minimum cut
    then store the cut-of-the-phase as the current minimum cut
```

Notice that the starting vertex a stays the same throughout the whole algorithm. It can be selected arbitrarily in each phase instead.

3. *Correctness*

In order to proof the correctness of our algorithms, we need to show the following somewhat surprising lemma.

LEMMA 3.1. *Each cut-of-the-phase is a minimum s-t-cut in the current graph, where s and t are the two vertices added last in the phase.*

PROOF. The run of a MinimumCutPhase orders the vertices of the current graph linearly, starting with a and ending with s and t, according to their order of addition to A. Now we look at an arbitrary s-t-cut C of the current graph and show, that it is at least as heavy as the cut-of-the-phase.

We call a vertex $v \neq a$ *active* (with respect to C) when v and the vertex added just before v are in the two different parts of C. Let $w(C)$ be the weight of C, A_v the set of all vertices added before v (excluding v), C_v the cut of $A_v \cup \{v\}$ induced by C, and $w(C_v)$ the weight of the induced cut.

We show that for every active vertex v

$$w(A_v, v) \leq w(C_v)$$

by induction on the set of active vertices:

For the first active vertex, the inequality is satisfied with equality. Let the inequality be true for all active vertices added up to the active vertex v, and let u be the next active vertex that is added. Then we have

$$w(A_u, u) = w(A_v, u) + w(A_u \setminus A_v, u) =: \alpha$$

Now, $w(A_v, u) \leq w(A_v, v)$ as v was chosen as the vertex most tightly connected with A_v. By induction $w(A_v, v) \leq w(C_v)$. All edges between $A_u \setminus A_v$ and u connect the different parts of C. Thus they contribute to $w(C_u)$ but not to $w(C_v)$. So

$$\alpha \leq w(C_v) + w(A_u \setminus A_v, u) \leq w(C_u)$$

As t is always an active vertex with respect to C we can conclude that $w(A_t, t) \leq w(C_t)$ which says exactly that the cut-of-the-phase is at most as heavy as C.

4. *Running Time*

As the running time of the algorithm MinimumCut is essentially equal to the added running time of the $|V| - 1$ runs of MinimumCutPhase, which is called on graphs with decreasing number of vertices and edges, it suffices to show that a single MinimumCutPhase needs at most $\mathcal{O}(|E| + |V| \log |V|)$ time yielding an overall running time of $\mathcal{O}(|V||E| + |V|^2 \log |V|)$.

The key to implementing a phase efficiently is to make it easy to select the next vertex to be added to the set A, the most tightly connected vertex. During execution of a phase, all vertices that are not in A reside in a priority queue based on a key field. The key of a vertex v is the sum of the weights of the edges connecting it to the current A, that is, $w(A, v)$. Whenever a vertex v is added to A we have to perform an update of the queue. v has to be deleted from the queue, and the key of every vertex w not in A, connected to v has to be increased by the weight of the edge vw, if it exists. As this is done exactly once for every edge, overall we have to perform $|V|$ ExtractMax and $|E|$ IncreaseKey operations. Using Fibonacci heaps [Fredman and Tarjun 1987], we can perform an ExtractMax operation in $\mathcal{O}(\log|V|)$ amortized time and an IncreaseKey operation in $\mathcal{O}(1)$ amortized time.

Thus, the time we need for this key step that dominates the rest of the phase, is $\mathcal{O}(|E| + |V|\log|V|)$.

5. *An Example*

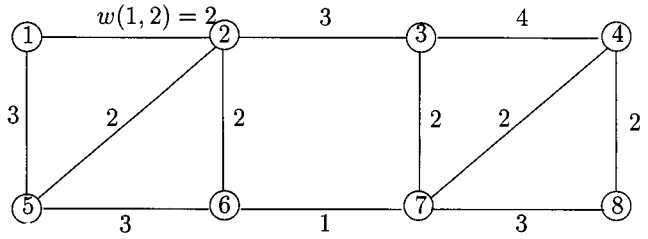

FIG. 1. A graph $G = (V, E)$ with edge-weights.

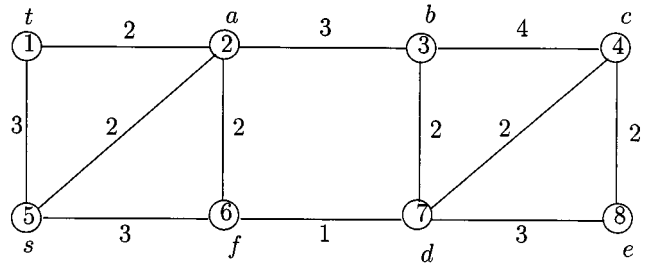

FIG. 2. The graph after the first MINIMUMCUTPHASE(G, w, a), $a = 2$, and the induced ordering a, b, c, d, e, f, s, t of the vertices. The first *cut-of-the-phase* corresponds to the partition {1}, {2, 3, 4, 5, 6, 7, 8} of V with weight $w = 5$.

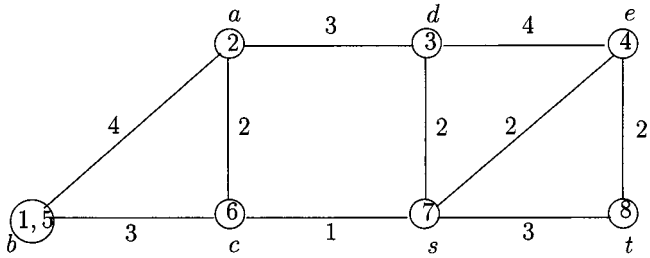

FIG. 3. The graph after the second MINIMUMCUTPHASE(G, w, a), and the induced ordering a, b, c, d, e, s, t of the vertices. The second *cut-of-the-phase* corresponds to the partition {8}, {1, 2, 3, 4, 5, 6, 7} of V with weight $w = 5$.

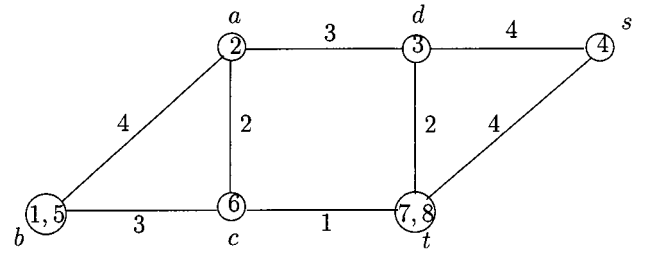

FIG. 4. After the third MINIMUMCUTPHASE(G, w, a). The third *cut-of-the-phase* corresponds to the partition {7, 8}, {1, 2, 3, 4, 5, 6} of V with weight $w = 7$.

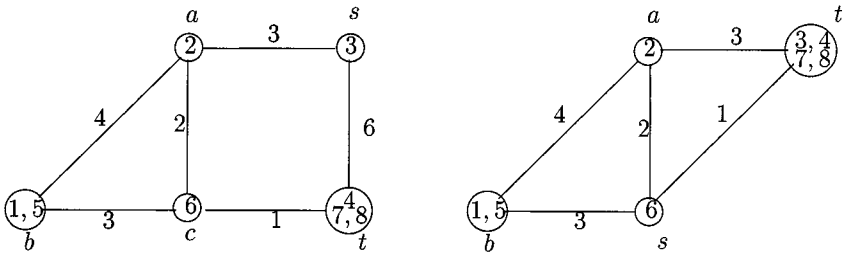

FIG. 5. After the fourth and fifth MINIMUMCUTPHASE(G, w, a), respectively. The fourth *cut-of-the-phase* corresponds to the partition {4, 7, 8}, {1, 2, 3, 5, 6}. The fifth *cut-of-the-phase* corresponds to the partition {3, 4, 7, 8}, {1, 2, 5, 6} with weight $w = 4$.

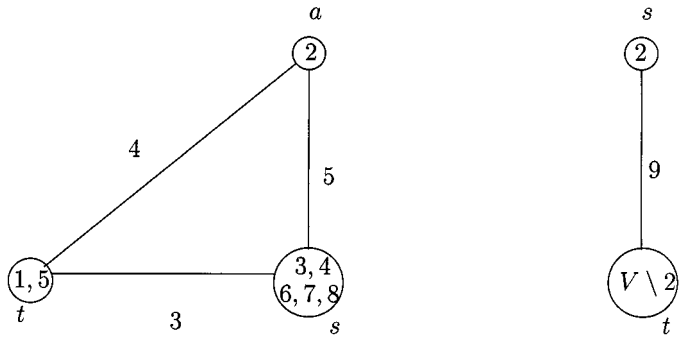

FIG. 6. After the sixth and seventh MINIMUMCUTPHASE(G, w, a), respectively. The sixth *cut-of-the-phase* corresponds to the partition {1, 5}, {2, 3, 4, 6, 7, 8} with weight $w = 7$. The last *cut-of-the-phase* corresponds to the partition {2}, $V \setminus \{2\}$; its weight is $w = 9$. The minimum cut of the graph G is the fifth *cut-of-the-phase* and the weight is $w = 4$.

ACKNOWLEDGMENT. The authors thank Dorothea Wagner for her helpful remarks.

REFERENCES

AHUJA, R. K., ORLIN, J. B., AND TARJAN, R. E. 1989. Improved time bounds for the maximum flow problem. *SIAM J. Comput. 18*, 939–954.

ALON, N. 1990. Generating pseudo-random permutations and maximum flow algorithms. *Inf. Proc. Lett. 35*, 201–204.

CHERIYAN, J., HAGERUP, T., AND MEHLHORN, K. 1990. Can a maximum flow be computed in $o(nm)$ time? In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*. pp. 235–248.

FORD, L. R., AND FULKERSON, D. R. 1956. Maximal flow through a network. *Can. J. Math. 8*, 399–404.

FRANK, A. 1994. *On the Edge-Connectivity Algorithm of Nagamochi and Ibaraki*. Laboratoire Artemis, IMAG, Université J. Fourier, Grenoble, Switzerland.

FREDMAN, M. L., AND TARJAN, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM 34*, 3 (July), 596–615.

GOLDBERG, A. V., AND TARJAN, R. E. 1988. A new approach to the maximum-flow problem. *J. ACM 35*, 4 (Oct.), 921–940.

HAO, J., AND ORLIN, J. B. 1992. A faster algorithm for finding the minimum cut in a graph. In *Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms* (Orlando, Fla., Jan. 27–29). ACM, New York, pp. 165–174.

KARGER, D., AND STEIN, C. 1993. An $\tilde{O}(n^2)$ algorithm for minimum cuts. In *Proceedings of the 25th ACM Symposium on the Theory of Computing* (San Diego, Calif., May 16–18). ACM, New York, pp. 757–765.

MATULA, D. W. 1993. A linear time $2 + \epsilon$ approximation algorithm for edge connectivity. In *Proceedings of the 4th ACM–SIAM Symposium on Discrete Mathematics* ACM, New York, pp. 500–504.

MEHLHORN, K., AND NÄHER, S. 1995. LEDA: a platform for combinatorial and geometric computing. *Commun. ACM 38*, 96–102.

NAGAMOCHI, H., AND IBARAKI, T. 1992a. Linear time algorithms for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica 7*, 583–596.

NAGAMOCHI, H., AND IBARAKI, T. 1992b. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM J. Disc. Math. 5*, 54–66.

NISHIZEKI, T., AND POLJAK, S. 1989. Highly connected factors with a small number of edges. Preprint.

QUEYRANNE, M. 1995. A combinatorial algorithm for minimizing symmetric submodular functions. In *Proceedings of the 6th ACM–SIAM Symposium on Discrete Mathematics* ACM, New York, pp. 98–101.

RECEIVED APRIL 1995; REVISED FEBRUARY 1997; ACCEPTED JUNE 1997