

Lecture notes for “Analysis of Algorithms”: Global minimum cuts

Lecturer: *Uri Zwick* *

Spring 2008

Abstract

We describe deterministic and randomized algorithms for finding global minimum cuts in *undirected* graphs.

1 Global and s - t cuts in undirected graphs

Let $G = (V, E, w)$ be a weighted undirected graph, where $w : E \rightarrow \mathbb{R}^+$ is a weight (or cost, or capacity) function defined on its edges. A (global) cut (S, T) of G is a partition of V into two disjoint non-empty sets, i.e., $S, T \neq \phi$ and $S \cap T = \phi$. Edges in the set $E(S, T) = \{e \in E \mid |e \cap S| = 1\}$ are said to *cross* the cut. Let $s, t \in V$. A cut (S, T) is said to be an s - t cut if and only if $|\{s, t\} \cap S| = 1$, i.e., $s \in S$ and $t \in T$, or $s \in T$ and $t \in S$. The weight $w(S, T)$ of a cut (S, T) is defined to be the sum of the weights of the edges that cross the cut:

$$w(S, T) = \sum_{e \in E(S, T)} w(e).$$

A cut (S, T) of G is said to be a *global min-cut* if and only if the weight $w(S, T)$ of the cut is the smallest possible, i.e., for every other cut (S', T') of G we have $w(S, T) \leq w(S', T')$. An s - t *min-cut* is defined similarly.

Given $s, t \in V$, an s - t min-cut of G can be found using a network flow algorithm, relying on the celebrated max-flow min-cut algorithm. We show below that finding *global* min-cuts is easier.

If $A, B \subseteq V$ and $A \cap B = \phi$, we let $E(A, B) = \{e \in E \mid |e \cap A| = |e \cap B| = 1\}$ and $w(A, B) = \sum_{e \in E(A, B)} w(e)$. (Note that this is an extension of the definition above.) The following trivial facts would be used below: If $A = A_1 \cup A_2$ and $A_1 \cap A_2 = \phi$, then $w(A, B) = w(A_1, B) + w(A_2, B)$. If $A_1, A_2 \subseteq A$, $B_1, B_2 \subseteq B$ and $B_1 \cap B_2 = \phi$, then $w(A_1, B_1) + w(A_2, B_2) \leq w(A, B)$.

*School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: zwick@tau.ac.il

```

Function GlobalMinCut( $G$ )
  if  $V = \{a, b\}$  then
    | return ( $\{a\}, \{b\}$ )
  else
    |  $\langle C_1, s, t \rangle \leftarrow \text{stMinCut}(G)$ 
    |  $C_2 \leftarrow \text{GlobalMinCut}(G/\{s, t\})$ 
    | if  $w(C_1) \leq w(C_2)$  then return  $C_1$  else return  $C_2$ 

```

Figure 1: The global min-cut algorithm of Stoer and Wagner.

2 The Stoer-Wagner algorithm

Let $G = (V, E, w)$ be a weighted undirected graph. Let (S, T) be a global min-cut of G . Suppose that $s, t \in V$. If $|\{s, t\} \cap S| = 1$, then (S, T) is clearly also an s - t min-cut of G .

Stoer and Wagner [SW97] proposed the following method for finding a global min-cut of a graph G . Start by finding an s - t min-cut (S, T) of G , for some two vertices $s, t \in V$. Then, by the above observation, either (S, T) is also a global min-cut of G , or in any global min-cut of G vertices s and t must belong to the same side of the cut. The global min-cut in this case can be found by finding the global min-cut in the graph $G/\{s, t\}$ obtained by *merging* s and t into a new vertex st . If s and t are connected by an edge then this edge disappears. If s and t both have edges to some vertex v , then the weight of the edge from the new vertex st to v is $w(s, v) + w(t, v)$. (This is similar to the operation of contracting an edge, though in this case, we do not assume that s and t are connected by an edge.)

Two problems immediately come to mind. The first is how do we find an s - t min-cut of G ? It was claimed above that finding an s - t min-cut is harder than finding a global min cut. The second problem is how do we know whether the s - t min-cut found is also a global min-cut or not?

The answer to the second question is simple. We simply compute a global min-cut in the smaller graph $G/\{s, t\}$, by a recursive call to the algorithm, and compare its weight to the weight of the s - t min-cut found. The cut with the smaller weight among these two is a global min-cut of G . The overall structure of the algorithm of Stoer and Wagner, which we call **GlobalMinCut** is given in Figure 1.

That leaves us with the more challenging problem of finding an s - t min-cut. Finding an s - t min-cut of G does indeed seem to be a harder problem than finding a global min-cut, when s and t are *specified*. Note that here we do not care who s and t are, we are just interested in finding a cut (S, T) and two vertices s and t such that (S, T) is an s - t min-cut. Indeed, s and t are not passed to procedure **stMinCut** called by **GlobalMinCut**. They are *returned* by it.

Finding a cut (S, T) and a pair of vertices s, t such that (S, T) is a min s - t cut is a much easier problem for which Stoer and Wagner [SW97] found a simple and elegant solution. A high level description of their algorithm, called **stMinCut** is given in Figure 2.

Function <code>stMinCut</code> (G)
$A \leftarrow \{a\}$
while $A \neq V$ do
Let $v \in A$ be such that $w(A, \{v\})$ is maximized
$A \leftarrow A \cup \{v\}$
Let s and t be the last two vertices added to A
return $\langle (V - \{t\}, \{t\}), s, t \rangle$

Figure 2: High level description of `stMinCut`.

Lemma 2.1 *If s and t are the next to last and last vertices added to A by `stMinCut`, then $(V - \{t\}, \{t\})$ is an s - t min-cut of G .*

Proof: We prove the lemma by induction on the number of vertices in G . If $n = |V| = 2$, the claim is obvious. We assume, therefore, that the claim holds for all graphs with at most $n - 1$ vertices and show that it also holds for all graphs with n vertices.

Let $G = (V, E)$ be a graph on n vertices. Let $v_1 = a, v_2, \dots, v_{n-1} = s, v_n = t$ be the order in which the vertices of G are added to A . For every $1 \leq i \leq n$, let $V_i = \{v_1, v_2, \dots, v_i\}$ and let $G_i = G[V_i]$ be the subgraph of G induced by V_i . It is easy to check that v_1, v_2, \dots, v_i is also an order in which `stMinCut` can add vertices to A when `stMinCut` is run on $G[V_i]$.¹

Let $s = v_{n-1}$ and $t = v_n$ be the last two vertices added to A by `stMinCut`. Let (S, T) be an arbitrary s - t min-cut. We have to show that $w(V - \{t\}, \{t\}) \leq w(S, T)$. Let $t' = v_i$ be the last vertex from $T - \{t\}$ added to A , and let $s' = v_{i+1}$. Also let $V' = V_{i+1}$, $S' = S \cap V'$ and $T' = T \cap V'$. Also let $S'' = \{v_{i+1}, \dots, v_{n-1}\}$. By the definition of i we get that $S'' \subseteq S$. Also, $S' \subseteq S$ and $T - T' = \{v_n\}$. As $s' \in S'$ and $t' \in T'$, we get that (S', T') is an s' - t' cut in $G' = G[V']$. By the induction hypothesis, $w(V' - \{t'\}, \{t'\}) \leq w(S', T')$.

Consider now the iteration in which t' was added to A by `stMinCut` when run on G . At that stage $A = V' - \{t'\}$, and t' and t were both out of A . As t' was chosen by the algorithm, we get that $w(V' - \{t'\}, \{t\}) \leq w(V' - \{t'\}, \{t'\})$. (Recall that the algorithm chooses a vertex v that maximizes $w(A, \{v\})$.) Combined with the induction, hypothesis, we get that

$$w(\{v_1, v_2, \dots, v_i\}, \{v_n\}) = w(V' - \{t'\}, \{t\}) \leq w(V' - \{t'\}, \{t'\}) \leq w(S', T').$$

Putting everything together, we get that

$$\begin{aligned} w(V - \{t\}, \{t\}) &= w(\{v_1, \dots, v_i\}, \{v_n\}) + w(\{v_{i+1}, \dots, v_{n-1}\}, \{v_n\}) \\ &\leq w(S', T') + w(S'', T - T') \\ &\leq w(S, T) \end{aligned}$$

□

¹We say here that this is an order in which `stMinCut` can add vertices to A as the order is not uniquely determined when there are ties.

```

Function stMinCut( $G$ )
 $PQ \leftarrow \phi$ 
foreach  $u \in V$  do
     $key[u] \leftarrow 0$ 
     $insert(PQ, u, key[u])$ 
 $s, t \leftarrow nil$ 
while  $PQ \neq \phi$  do
     $u \leftarrow extractmax(PQ)$ 
     $s \leftarrow t ; t \leftarrow u$ 
    foreach  $(u, v) \in E$  do
        if  $v \in PQ$  then
             $key[v] \leftarrow key[v] + w(u, v)$ 
             $increasekey(PQ, v, key[v])$ 
return  $\langle (V - \{t\}, \{t\}), s, t \rangle$ 

```

Figure 3: Complete description of `stMinCut`.

An efficient implementation of `stMinCut` using a priority queue is given in Figure 3. Note the similarity of the algorithm to the algorithms of Dijkstra and Prim. The following two theorems are now immediate.

Theorem 2.2 *The running time of `stMinCut`, when implemented using Fibonacci heaps, is $O(m + n \log n)$.*

Theorem 2.3 *The running time of `GlobalMinCut` is $O(mn + n^2 \log n)$.*

3 The Karger-Stein random contraction algorithm

The Stoer-Wagner algorithm performs n iterations in each of which a pair of vertices s and t is selected and then merged. Selecting each such pair of vertices takes $O(m + n \log n)$ time. Is there a faster way of selecting pairs of vertices to be merged?

Karger and Stein [KS96] suggest the following intriguing way of selecting the next pair of vertices to be merged: Choose a *random edge* and contract it. For simplicity, we assume here that the input graph G is *unweighted*. Contracting an edge may create self-loops and parallel edges. Self-loops are discarded, but parallel edges are kept, and this is essential. (This corresponds to the operation of adding the weights in the previous section.) Each edge is then chosen by the algorithm with the same probability. The contraction of edges creates *super-vertices* which correspond to subset of vertices of the original graph. Edges are contracted until there are only two super-vertices left. These two remaining super-vertices are returned by the algorithm. A pseudo-code of the algorithm

Function RandMinCut(G)

```
while  $|V| > 2$  do  
   $\perp$  Pick a random edge and contract it  
return the remaining two super-vertices
```

Figure 4: The random contraction algorithm of Karger.

is given in Figure 4. What is the probability that the cut returned by `RandMinCut` is a global min-cut of G ?

If $G = (V, E)$ is a graph and $e \in E$, we let G/e be the graph obtained from G by contracting e . If $e = \{s, t\}$, then the two vertices s, t in G are replaced by the super-vertex st . Similarly, if $S \subseteq V$, we let S/e be $S \cup \{st\} - \{s, t\}$, if $s, t \in S$, and $S/e = S$, otherwise. We claim:

Lemma 3.1 *Let $G = (V, E)$ be an unweighted undirected graph. Let (S, T) be a global min-cut of G . Let $e \in E$ and let $G' = G/e$, $S' = S/e$ and $T' = T/e$. If $e \notin E(S, T)$, then (S', T') is a global min-cut of G' and $w(S', T') = w(S, T)$.*

Proof: Any cut of G' corresponds to a cut with the same weight of G . Thus, the weight of any cut of G' is at least $w(S, T)$. On the other hand, if $e \notin E(S, T)$, i.e., $e \subseteq S$ or $e \subseteq T$, then $w(S', T') = w(S, T)$, and hence (S', T') is a global min cut of G . \square

Lemma 3.2 *Let $G = (V, E)$ be an unweighted undirected multi-graph (i.e., a graph that may contain parallel edges.) Let (S, T) be a global min-cut of G . Let $e \in E$ be a random edge selected uniformly at random. Then, $\Pr[e \in E(S, T)] \leq \frac{2}{n}$, where $n = |V|$ is the number of vertices of G .*

Proof: Let $m = |E|$ and $k = |E(S, T)|$. As (S, T) is a global min-cut, the degree of every vertex of G must be at least k . Thus $m \geq \frac{kn}{2}$ or $k \leq \frac{2m}{n}$. Finally,

$$\Pr[e \in E(S, T)] = \frac{k}{m} \leq \frac{\frac{2m}{n}}{m} = \frac{2}{n}.$$

\square

Theorem 3.3 *Let $G = (V, E)$ be an unweighted undirected graph. Let $\{S, T\}$ be a global min-cut of G . Then, the probability that algorithm `RandMinCut` returns the cut (S, T) is at least $\frac{1}{\binom{n}{2}}$.*

Proof: The algorithm returns the cut (S, T) (or (T, S)) if and only if no edge selected by the algorithms belongs to this cut. By Lemma 3.1 as long as this condition holds, the cuts corresponding to (S, T) in the contracted graphs are all global min-cuts. By Lemma 3.2, we get that when the number of (super-)vertices in the graph is i , the probability that the next edge selected by the

Function FastRandMinCut($G = (V, E)$)

```

if  $|V| \leq 6$  then
   $\lfloor$  return GlobalMinCut( $G$ )
 $t \leftarrow \lceil n/\sqrt{2} + 1 \rceil$ 
for  $i \leftarrow 1$  to 2 do
   $\lfloor$   $G_i \leftarrow$  RandContract( $G, t$ )
   $\lfloor$   $(S_i, T_i) \leftarrow$  FastRandMinCut( $G_i$ )
if  $w(S_1, T_1) \leq w(S_2, T_2)$  then
   $\lfloor$  return  $(S_1, T_1)$ 
else
   $\lfloor$  return  $(S_2, T_2)$ 

```

Figure 5: The recursive random contraction algorithm of Karger and Stein.

algorithm belongs to the cut (S, T) is at most $\frac{2}{i}$. Thus, the probability that the cut (S, T) is the cut returned by the algorithms is at least

$$\left(1 - \frac{2}{n}\right) \left(1 - \frac{2}{n-1}\right) \left(1 - \frac{2}{n-2}\right) \cdots \frac{2}{4} \cdot \frac{1}{3} = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n(n-1)}.$$

□

Corollary 3.4 *An undirected graph on n vertices has at most $\binom{n}{2}$ global min-cuts.*

4 The Karger-Stein recursive contraction algorithm

A randomized algorithm for computing global min-cuts using random edge contractions whose success probability is much larger than that of algorithm RandMinCut is given in Figure 5.

Let $T(n)$ be the running time of algorithm FastRandMinCut on an n -vertex graph and let $P(n)$ be the probability that it returns a specific global min-cut. We have

$$T(n) = 2T(\lceil \frac{n}{\sqrt{2}} + 1 \rceil) + O(n^2).$$

It is not difficult to see that $T(n) = O(n^2 \log n)$. We also have

$$P(n) = 1 - \left(1 - \frac{1}{2}P(\lceil \frac{n}{\sqrt{2}} + 1 \rceil)\right)^2.$$

Let $p(k) = P(2^{k/2})$. We then have $P(n) = p(2 \log n)$. The recurrence for $p(k)$ becomes

$$p(k+1) = p(k) - \frac{1}{4}p(k)^2.$$

If we now define

$$q(k) = \frac{4}{p(k)} - 1,$$

then a simple manipulation gives:

$$q(k+1) = q(k) + 1 + \frac{1}{q(k)}.$$

We can now show by induction that

$$k < q(k) < k + H_{k-1} + 3,$$

where $H_j = \sum_{i=1}^j \frac{1}{i}$. Putting everything together, we get that $P(n) = \Theta(\frac{1}{\log n})$.

Thus to obtain a global min-cut with a probability of, say, $1 - \frac{1}{n}$, we need to run `FastRandMinCut` only $\Theta(\log^2 n)$ times, giving us a total running time of $O(n^2 \log^3 n)$.

References

- [KS96] D.R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43:601–640, 1996.
- [SW97] M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.