Deterministic edge-connectivity in near-linear time

Ken-ichi Kawarabayashi
National Institute of Informatics
JST ERATO Kawarabayashi Large Graph Project
Joint work with Mikkel Thorup

Preliminary version in STOC’15
Full version at ArXiv(different title)
``Deterministic global minimum-cut of a simple graph in near-linear time``
**Graph Cut**: One of the most famous problem

Partition $G$ into $(A,B)$ such that # of edges between $A$ and $B$ is as small as possible.

$G$: $n$ vertices and $m$ edges. $k$: cut size

Variants: Simple, multiple, weighted, directed....

- Ford and Fulkerson’56 \( O(mn^2) \) s-t cut
- Even and Tarjan’75 \( O(nm^{1.5}) \) s-t cut
- Podderyugin’73 \( O(kn^2) \) Deterministic (for simple graphs)
- Hao and Orlin’92 \( O(nm \log(n^2/m)) \) Deterministic (for directed weighted graphs)
- Ibaraki and Nagamochi’92 \( O(nm+n^2\log n) \) Deterministic
- Gabow’93 \( O(km \log n) \) Deterministic
- Karger’94 \( O(mn^{0.5}) \) (Randomize, Monte Carlo, Random Contraction)
- Karger and Stein’96 \( O(n^2 \log^3 m) \) (Randomize, Monte Carlo)
- Karger’99 \( O(m \log^3 n) \) (Randomize, Monte Carlo, Tree Packing)
- KK and Thorup(STOC’15) \( O(m \log^{12} n) \) Deterministic! (for simple graphs)
Randomize Algorithms: Two kinds

- **Monte Carlo Algorithm:**
  Guarantee the time complexity, and with high probability, the answer is correct.

  Kager’s mincut: NO WAY to check that the cut is min... (Need deterministic one to check....)

- **Las Vegas Algorithm:**
  Guarantee the correctness, but with small probability, time complexity takes forever..
Contents

- Known results for mincut and edge-connectivity
- Our main result and technical contribution
- Very high level sketch: Sparsest cut and contraction
- Sparsest cut in our business: more details
- PageRank and Endgame
- Our algorithm again
Theorem (KK, Thorup)

$O(m \log^{12} n)$ Deterministic algorithm for edge-connectivity for simple graphs.

Two Drawbacks:

1. Slower than Karger’s (well…)
2. Only works for simple graphs. Does not work for multigraphs/weighted graphs.
Cut is either trivial or non-trivial. Non-trivial is the problem!

G: n vertices, m edges, undirected, simple. k: edge-connectivity. So min. degree \( \geq k \).

Main Technical Contribution.
For any undirected simple graph \( G \), after some contractions in \( O(m \text{ poly}(\log n)) \) time,
Can find a graph \( G' \) such that
1. \( G' \) has \( O(m/k) \) edges, and
2. All non-trivial cuts are preserved in \( G' \)!

So we can apply Gabow’s result\( (O(km) \) time to find a min-cut) for \( G' \)!

NOTE: Hereafter, ignoring \( \text{poly}(\log n) \) for time complexity.....
Summary of our work

Theorem (KK, Thorup)
$O(m \log^{12} n)$ Deterministic algorithm for edge-connectivity for simple graphs.

Two Drawbacks:
1. Slower than Karger’s (well...)
2. Only works for simple graphs. Does not work for multigraphs/weighted graphs.

Main Technical Contribution.
For any undirected simple graph $G$, after some contractions in $O(m \text{ poly}(\log n))$ time,
Can find a graph $G'$ such that
1. $G'$ has $O(m/k)$ edges, and
2. All non-trivial cuts are preserved in $G'$!

Two technical components
Bring “sparsest cut” and “PageRank” business to edge-connectivity!
Note: This is linked to the second drawback.
Contents

- Known results for mincut and edge-connectivity
- Our main result and technical contribution
- Very high level sketch: Sparsest cut and contraction
- Sparsest cut in our business: more details
- PageRank and Endgame
- Our algorithm again
Key Observation 1: G:simple
Let \((C,G-C)\) be a ``minimum'' cut (of order at most \(k\)). Either
1. One side is trivial (note that min degree is at least \(k\)), or
2. Both sides contain at least \(k\) vertices.

In 2, we get a ``sparse'' cut (conductance \(1/k < o(1/\log m))\).
→ KEY IDEA! Keep applying sparsest cut!

Conductance is:
\[
\frac{\text{(# of edges between } G-C \text{ and } C)}{\text{Min}(e(G), e(G-C))}.
\]
What happens without sparsest cuts?  Contractions!

- Observation 2: 
  Q: a subgraph s.t. each vertex in Q has degree at least k in Q. 
  If no sparsest cut of conductance $1/k$ in Q
  \[ \rightarrow \] Q is contractible (preserving all min nontrivial cuts)!

Note: $k > \log^6 n$. 
A non-trivial min cut C
\[ \rightarrow \] a sparsest cut of conductance $< 1/\log^6 m$!

No small conductance cut
\[ \rightarrow \] we can contract (preserving non-trivial cuts)
Key Observation 1:
Let $C$ be a "minimum" cut (of order at most $k$). Either
1. One side is trivial (note that min degree is at least $k$), or
2. Both sides contain at least $k$ vertices.
In 2, we get a "sparse" cut (conductance $o(1/\log m)$) → KEY IDEA! Keep applying sparsest cut!

Key Observation 2:
Q: a subgraph s.t. each vertex in Q has degree at least $k$ in Q.
If no sparsest cut of conductance $1/k$ in Q → Q is contractible (preserving all min nontrivial cuts)!

Recursively, either
1. We delete a "sparse" cut with conductance $o(1/\log m)$, or
2. "Contract" a current graph if there is no sparse cut, preserving non-trivial mincuts.
We do 1 for at most $\log n$ depth and eventually 2 happens (average degree still $k-o(\log^2 m)$)
(then we get a graph that has $O(m/k)$ edges!)
Contents

- Known results for mincut and edge-connectivity
- Our main result and technical contribution
- Very high level sketch: Sparsest cut and contraction
- Sparsest cut in our business: more details
- PageRank and Endgame
- Our algorithm again
Sparsest Cut in Our Business: More details
Let \((C,G-C)\) be a cut.

Conductance is:
\[
\frac{\text{(# of edges between } G-C \text{ and } C)}{\text{Min}(e(G), e(G-C))}.
\]

Low conductance -> ``Sparsest'' cut.

\(O(\log n)\) approx. algorithm: Leighton-Rao (1999)
\(O(\sqrt{\log n})\) approx. algorithm: Arora-Rao-Vazirani (2009)
Reminder:

Key Observation 1:
Let \((G-C,C)\) be a ``minimum'' cut (of order at most \(k\)). Either
1. One side is trivial (note that min degree is at least \(k\)), or
2. Both sides contain at least \(k\) vertices..

Note: \(k \geq \log_6 m\) (otherwise, use \(O(km)\) algorithm by Gabow)
A non-triticale min cut \(C \rightarrow\) a sparsest cut of conductance \(< 1/\log^6 m\)

Recursively, either
1. We delete a ``sparse'' cut with conductance \(o(1/\log m)\), or
2. ``Contract'' a current graph if there is no sparse cut, preserving non-trivial mincuts.

We cannot find an OPT sparsest cut, but...
Nearly Linear time $\sqrt{(a \log m)}$ approximation algorithm is just needed!

This implies.. (in our case)
If a non-trivial min cut $C$ exists,
$\rightarrow$ Linear time algorithm to find a sparsest cut of conductance $< 1/\log^2 m$! (because conductance is $1/k$ and $k > \log^6 m$)

Talk to you later
1. We need to find such a cut in linear.
2. We do only $\log m$ depth.
Assuming an approximation algorithm, we are in a good shape!

Reminder:
Recursively, either
1. We delete a "sparse" cut with conductance $o(1/\log m)$, or
2. "Contract" a current graph if there is no sparse cut, preserving non-trivial mincuts.

We do 1 for at most $\log m$ depth and eventually 2 happens (average degree still $k-o(\log^2 m)$)

Nearly Linear time $\sqrt{a \log m}$ approximation algorithm is just needed!

Since $k \geq \log^6 m$, and $\log m$ depth
The remaining degree is $k- (\log m \times k/(\log^2 m)) = k-O(\log^2 m) = k-o(k)$.
So if $Q$ has at most $k-o(k)$ vertices, then we can contract!

In the end:
Find contractible subgraphs $Q'$ such that after the contraction, the graph has at most $O(m/k)$ edges (but all nontrivial cuts are there).
→ By Gabow's, we get an $O(m)$ algorithm.
More technical components

1. Nearly Linear time for approximating a sparsest cut? We will discuss here! PageRank and Sparsest cut!

2. How do we make the whole thing nearly linear? How do we make ``log m'' depth?

Coming next.
Contents

- Known results for mincut and edge-connectivity
- Our main result and technical contribution
- Very high level sketch: Sparsest cut and contraction
- Sparsest cut in our business: more details
- PageRank and Endgame
- Our algorithm again
Linear time for a sparsest cut: PageRank (à la Brin & Page, Google?!) 

Andersson, Chung and Lang (ACL)'07 model (Local Partitioning).

Given two constants $0 < a, t < 1$. (note: in our case, typically $t=1/m$ and $a = 1/\log^6 m$)

Start from one single vertex $v$ with ``residual mass $r(v)=1$'' and ``pagerank score $p(v)=0$''. (Other vertices $u$ have $r(u)=0$ and $p(u)=0$ at the beginning).

Then ``push'' each vertex $u$ with the following rule:

Whenever $r(u)/d(u) \geq t$,
1. $p(u) \leftarrow p(u) + a \cdot r(u)$ (p(u) increases)
2. For each edge $su$, $r(s) \leftarrow r(s) + (1-a)r(u)/2d(u)$;
3. $r(u) \leftarrow (1-a)r(u)/2$. (r(u) decreases)

Final (converge) value is $p(u)$ (PageRank score). Density is $p(u)/d(u)$
PageRank (ACL’07) Results

Theorem 1:
The PageRank algorithm stops in $O(1/\alpha t)$ time.

Theorem 2:
Suppose there is a sparsest cut $(C, G-C)$ with
1. Conductance $< \alpha$ (in our case, $1/\log^6 m = 1/k$)
2. $s/2 \leq e(C) \leq 2s$.

If $v$ is in $C$ and whenever $r(u)/d(u) \geq t = 1/(2s)$, we push, then in $O(s/\alpha)$ time, we can find a cut $(C', G-C')$ with conductance $\sqrt{a \log m}$ and with $s/2 \leq e(C') \leq 2s$. Moreover, $v$ is in $C'$.

(Therefore, we get $O(\log^2 m)$-approximation!)
Big Side G-C

Residual $r$ can cross the cut only tiny. So $p$ in $C$ is big, but $p$ in $G-C$ is tiny. So large $p(u)/d(u)$ value vertices give a cut! We spend only $O(s)$ time!

Small side $C$

$s/2 \leq e(C) \leq 2s$

Almost same as "local partitioning"
End Game!

If $u$ gets small ($< (1-x)/2m$), then we can find a sparse cut!

Min cut

Small side $C$

Big Side $G-C$

Residual $r$ can cross only tiny across the cut.
So $p$ in $C$ is tiny and $p(u)$ is tiny for some $u$ in $C$
So tiny $p(z)/d(z)$ value vertices give a cut!
We spend $O(m)$ time..
PageRank Result 2: Endgame!

Assumption: No non-trivial min cut \((C', G-C')\) (with conductance < \(a\)) and with \(s/2 < e(C') \leq 2s\) with \(v\) in \(C'\),

Theorem 3:
Suppose there is a sparsest cut \((C, G-C)\) with conductance < \(a\).
If \(v\) is in \(G-C\), and
1. \(u\) is in \(C\),
2. the final density \(p(u)/d(u)\) is at most \((1-x)/2m\) (for some \(0 < x < 1\)),
3. whenever \(r(u)/d(u) \geq (1-2x)/(2m)\), we push.

Then we can find a cut \((C', G-C')\) with conductance \(\sqrt{(a \log m/x)}\) in \(O(m/a)\) time with the following property:

\(v\) is in \(G-C'\) and \(C'\) consists of all the vertices \(v'\) of density \(p(v')/d(v') < (1-x)/2m\).
More technical components

1. Linear time for approximating a sparsest cut?
   We will discuss here!
   PageRank and Sparsest cut!

2. How do we make the whole thing nearly linear?
   How do we make ``log m'' depth?

   Coming next!
Contents

- Known results for mincut and edge-connectivity
- Our main result and technical contribution
- Very high level sketch: Sparsest cut and contraction
- Sparsest cut in our business: more details
- PageRank and Endgame
- Our algorithm again
Reminder:
No small conductance cut
-> we can contract (preserving non-trivial cuts)
Indeed, if Q is too small (i.e., has at most k vertices),
then contractible! (provided that the average degree is good..)

The whole algorithm:
Fix one vertex v.
We alternate “pagerank” and endgame
Try s= m/4, m/16,.., k^2 /4
(shooting ``small side” between s/2 and 2s).
→ giving log m depth!
If there is a cut \((C, G-C)\) with \(m/8 \leq e(C) \leq m/2\), then we recurse to \(C\) and \(G-C\). We only pay \(O(e(C))\)!
If no cut \((C,G-C)\) with \(m/8 \leq e(C) \leq m/2\), then we apply endgame (with \(O(1)\) vertices). If there is a cut \((C',G-C')\), then we recurse to \(C'\), and we apply \(G-C'\) with \(s=m/16\).

We spent \(m\) time, but in the next iteration, we spend only \(m/8\) time for the large side!
Recursion to big side! (Pagerank $s=m/16$)!

If there is a cut $(C,G-C)$ with $m/32 \leq e(C) \leq m/8$, then we recurse to $C$ and $G-C$. We only pay $O(e(C))$!
Recursion to big side (endgame)!

If no cut \((C,G-C)\) with \(m/32 \leq e(C) \leq m/8\), then we apply endgame. If there is a cut \((C',G-C')\), then we recurse to \(C'\), but we apply \(G-C'\) with \(s=m/64\).

We paid \(m/8\) time, but in the next iteration, we only pay \(m/32\) time for the large side!
Key Observation:
Q: a subgraph s.t. each vertex in Q has degree at least k in Q. If no sparsest cut of conductance 1/k in $Q \rightarrow Q$ is contractible (preserving all min nontrivial cuts).

So if $e(Q) \leq k^2$, no sparsest cut.

Algorithm:
Try $s = m/4, m/16, \ldots, k^2$.
After at most $\log m$ iterations, we either
1. Find a contractible graph for case $s$, or
2. Divide into many smaller pieces of having at most half of edges of $G$ by deleting sparsest cuts.

If we divide into smaller pieces, then we recurse (so only $\log m$ depth)
Another point:
Only apply $O(\log m)$ iterations to the small side $C$
→ average degree is $k - O(\log^4 m) = k-o(k) \gg k/2$...

Can do each "small" side separately..
→ spend only $O(m)$ in total. Then we can find contractible subgraphs!

In the end, we have a graph with $O(m/k)$ edges.
Main Technical Contribution.
For any undirected graph $G$, after some contractions, can find a graph $G'$ such that
1. $G'$ has $O(m/k)$ edges, and
2. All non-trivial cuts are preserved in $G'$!

So we can apply Gabow's result ($O(km)$ time to find a min-cut) to $G'$!

Well, cheating.....
How do we deal with ``multiple edges''??

Well, they are not problem, because we can always contract $k$ multiple edges, and this allows us to play the game...

More details in the full paper...
Theorem (KK, Thorup)
$O(m \log^{12} n)$ Deterministic algorithm for edge-connectivity (mincut) for simple graphs.
Can do the same thing for the "cactus" representation.

Two Drawbacks:
1. Slower than Karger's (well...)
2. Only works for simple graphs. Does not work for multigraphs/weighted graphs.

New Contribution:
Bring "sparsest cut" and "PageRank" business to edge-connectivity! This restricted to "simple" graphs.

Future Work:
Non-simple graphs? (even for Las Vegas?)
Many Thanks for your Attention!

Question??  
Comment? I am cute, ya?