Algortithms for the Min-Cut problem

Hongwei Jin

Department of Applied Mathematics
Illinois Insititute of Technology

April 30, 2013

Outline

Problem Definition

Let $G = (V, E)$ be undirected graph with n vertices, and m edges. We are interested in the notion of a cut in a graph.

Definition

A **cut** in G is a partiontion of the vertices of V into two sets S and T, $T = V(G) \backslash S$, where the edges of the cut are

$$(S, T) = \{uv | u \in S, v \in T, S \cap T = V(G), uv \in E(G)\}$$

where $S \neq \emptyset$ and $T \neq \emptyset$. We will refer to the number of edges in the cut (S, T) as the *size* of the cut.

Problem Definition

We are intersted in the problem of computing the **minimum cut**, that is, the cut in the graph with minimum cardinality.

s-t minimum cut Require that the two specific vertices s and t be on opposite sides of the cut

gloable minimum cut No such requirement.

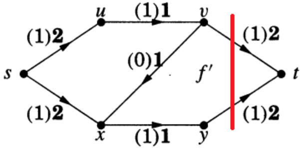
Previous Works

The oldest known way to compute min-cut is to use their well known duality with max-flow[1]. Now we should recall some definition and theorem from graph theorem.

Theorem (Max-flow Min-cut Theorem (Ford and Fulkerson,1956))

In every network, the maximum value of a feasible flow eqauls the minimum capacity of a source/sink cut.

For an example



[1] Lester R Ford and Delbert R Fulkerson. "Maximal flow through a network". In: *Canadian Journal of Mathematics* 8.3 (1956), pp. 399–404.

Previous Works

The best known sequential time bound is $O(mn\log(n^2/m))$, which is found by Glodberg and Tarjan[2] using Ford-Fulkerson algorithm.

Hao and Orlin algorithm shows how the max-flow computations can be pipelined so that together they take no more time than a single max-flow computation, requiring $O(mn\log(n^2/m))$[3].

[2] Andrew V Goldberg and Robert E Tarjan. "A new approach to the maximum-flow problem". In: *Journal of the ACM (JACM)* 35.4 (1988), pp. 921–940.

[3] Jianxiu Hao and James B. Orlin. "A faster algorithm for finding the minimum cut in a directed graph". In: *J. Algorithms* 17.3 (1994), pp. 424–446.

Previous Work

Gabow algorithm shows how to find the edge-connectivity c of a graph in time $O(cn\log(n^2/m))$, c denotes the min cut[4].

Algorithm developed by Nagamochi and Ibaraki is designed for weighted graph, undirected graphs. They showed it can be runned in time $O(mn + n^2\log(n))$[5].

[4] Harold N Gabow. "A matroid approach to finding edge connectivity and packing arborescences". In: *Proceedings of the twenty-third annual ACM symposium on Theory of computing*. ACM. 1991, pp. 112–122.

[5] Hiroshi Nagamochi and Toshihide Ibaraki. "Computing edge-connectivity in multigraphs and capacitated graphs". In: *SIAM Journal on Discrete Mathematics* 5.1 (1992), pp. 54–66.

Karger's Algorithm

The fundamental concept of Karger's Algorithm is "contraction(edge contraction)"

Definition

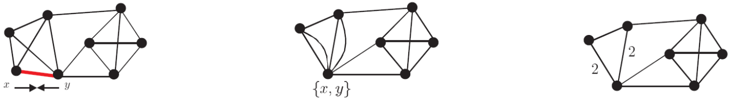
In a graph G, **contraction** of edge e with endpoints u, v is the replacement of u and v with single vertex whose incident edges are the edges other than e that were incident to u or v. the resulting graph,denoted as G/e.

Karger's Algorithm

The fundamental concept of Karger's Algorithm is "contraction(edge contraction)"

Definition

In a graph G, **contraction** of edge e with endpoints u, v is the replacement of u and v with single vertex whose incident edges are the edges other than e that were incident to u or v. the resulting graph,denoted as G/e.

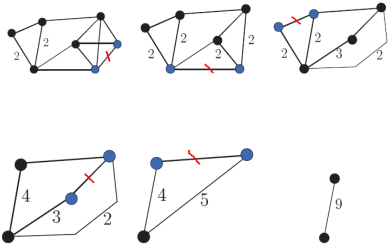


Karger's Algorithm

procedure MinCut $(G = (V, E))$
while $|V| > 2$
choose $e \in E$ uniformly and random
$G \rightarrow G/e$
return the only cut in G

Karger's Algorithm

procedure MinCut $(G = (V, E))$
while $|V| > 2$
choose $e \in E$ uniformly and random
$G \rightarrow G/e$
return the only cut in G



Karger's Algorithm

Obvervation

The size of the minimum cut in G/e is at least as large as the minimum cut in G (as long as G/e has at least one edge). Since any cut in G/e has a corresponding cut of the same cardinality in G.

Karger's Algorithm

Observation

The size of the minimum cut in G/e is at least as large as the minimum cut in G (as long as G/e has at least one edge). Since any cut in G/e has a corresponding cut of the same cardinality in G.

Observation

Let $e_1, \ldots e_{n-2}$ be a sequence of edges in G, such that none of them is in the minimum cut, and such that $G' = G/e_1, \ldots e_{n-2}$ is a single multi-edge. Then, this multi-edge correspond to the minimum cut in G.

Karger's Algorithm

Observation

The size of the minimum cut in G/e is at least as large as the minimum cut in G (as long as G/e has at least one edge). Since any cut in G/e has a corresponding cut of the same cardinality in G.

Observation

Let $e_1, ...e_{n-2}$ be a sequence of edges in G, such that none of them is in the minimum cut, and such that $G' = G/e_1, ...e_{n-2}$ is a single multi-edge. Then, this multi-edge correspond to the minimum cut in G.

Observation

The algorithm always output a cut, and the cut is not smaller than the minimum cut.

Algorithm Analysis

Lemma

A cut (S, T) is output by the MinCut algorithm if and only if no edge crossing (S, T) is contracted by the algorithm.

Algorithm Analysis

Lemma

A cut (S, T) is output by the MinCut algorithm if and only if no edge crossing (S, T) is contracted by the algorithm.

Lemma

If a graph G has a minimum cut of size k, and it has n vertices, then $|E(G)| \geq \frac{kn}{2}$

Algorithm Analysis

Lemma

A cut (S, T) is output by the MinCut algorithm if and only if no edge crossing (S, T) is contracted by the algorithm.

Lemma

If a graph G has a minimum cut of size k, and it has n vertices, then $|E(G)| \geq \frac{kn}{2}$

Lemma

If we pick in random an edge e from a graph G, then with probability at most 2/n it belong to the minimum cut.

Algorithm Analysis

Lemma

A cut (S, T) is output by the MinCut algorithm if and only if no edge crossing (S, T) is contracted by the algorithm.

Lemma

If a graph G has a minimum cut of size k, and it has n vertices, then $|E(G)| \geq \frac{kn}{2}$

Lemma

If we pick in random an edge e from a graph G, then with probability at most 2/n it belong to the minimum cut.

Lemma

MinCut algorithm runs in $O(n^2)$ time.

Algorithm Analysis

Theorem

MinCut algorithm outputs the min cut in probability $\mathbb{P} \geq \frac{2}{n(n-1)}$

Proof.

Let x_i be the event that edge e_i is not in the minimum cut of G_i. If the MinCut algorithm output a minimum cut, then all the event sequence $\{x_0, ... x_{n-3}\}$ will happen. Since at most with probability $2/n$ the edge will belong to the minimum cut. Thus we have the probability at least

$$(1 - \frac{2}{n})(1 - \frac{2}{n-1})...(1 - \frac{2}{3}) = (\frac{n-2}{n})(\frac{n-3}{n-1})...(\frac{1}{3}) = \frac{2}{n(n-1)}$$

$\square$

Algorithm Analysis

Lemma

The probability that repeat MinCut algorithm $T = \binom{n}{2} \log n$ times fails to return the minimum cut is $< \frac{1}{n}$

Proof.

The probability of failure is at most

$$(1 - \frac{2}{n(n-1)})^{\binom{n}{2} \log n} \leq \exp(-\log n) = \frac{1}{n}$$

$\square$

Algorithm Analysis

Theorem

In $O(n^4 \log n)$ time the minimum cut is returned with high probability.

Karger-Stein Algorithm

Obvervation

As the graph get smaller, the probability to make a bad choice increases. So, run the algorithm more times when the graph is smaller.

Karger-Stein Algorithm

Obvervation

As the graph get smaller, the probability to make a bad choice increases. So, run the algorithm more times when the graph is smaller.

procedure MinCut $(G, \mathbf{t})$
while $|V| > \mathbf{t}$
choose $e \in E$ uniformly and random
$G \rightarrow G/e$
return the only cut in G

Karger-Stein Algorithm

Lemma

The probability that MinCut$(G, n/\sqrt{2})$ had NOT contracted the minimum cut is at least $1/2$.

Proof.

Let $l = n - t = n - \lceil 1 + n/\sqrt{2} \rceil$, we will get

$$\mathbb{P}[x_0 \cap \ldots \cap x_{n-t}] \geq \frac{t(t-1)}{n(n-1)} = \frac{(\lceil 1 + n/\sqrt{2} \rceil)(\lceil 1 + n/\sqrt{2} \rceil - 1)}{n(n-1)} \geq \frac{1}{2}$$

$\square$

Karger-Stein Algorithm

They introduced a recursive way to find the minimum cut

procedure FastMinCut (G)
if $|V| < 6$
MinCut(G,2)
else
$t = 1 + |V|/sqrt(2)$
G1 = MinCut(G,t)
G2 = MinCut(G,t)
return min (FastMinCut(G1), FastMinCut(G2))

Algorithm Analysis

Theorem

The running time of FastMinCut(G) is $O(n^2 \log n)$, *where* $n = |V(G)|$.

Proof.

Well, we perform two calls to MinCut(G,t) which takes $O(n^2)$ time. And then we perform two recursive calls, on the resulting graphs. We have:

$$T(n) = O(n^2) + 2T(\frac{n}{\sqrt{2}})$$

The solution to this recurrence is $O(n^2 \log n)$ as one can easily verify. □

Algorithm Analysis

Theorem

Running FastMinCut finds the minimum cut with probability larger that $\frac{2\log 2}{\log n}$*,which can be notated as* $\Omega(1/\log n)$

The probability to succeed in the first call on G_1 is the probability that contract did not hit the minimum cut (this probability is larger than $1/2$), times the probability that the algorithm succeeded on G_1 in the recursive call (those two events are independent). Thus, the probability to succeed on the call on G_1 is at least $\frac{1}{2}P(\frac{n}{\sqrt{2}})$. Thus, the probability to fail on G_1 is $\leq 1 - \frac{1}{2}P(\frac{n}{\sqrt{2}})$.

The probability to fail on both G_1 and G_2 is smaller than

$$(1 - \frac{1}{2}P(\frac{n}{\sqrt{(2)}}))^2$$

And thus, the probability for the algorithm to succeed is

$$P(n) \geq 1 - (1 - \frac{1}{2}P(\frac{n}{\sqrt{(2)}}))^2 = P(\frac{n}{\sqrt{(2)}}) - \frac{1}{4}(P(\frac{n}{\sqrt{(2)}}))^2$$

Algorithm Analysis

Theorem

With high probability we can find all min cuts in the running time of $O(n^2 \log^3 n)$.

Proof.

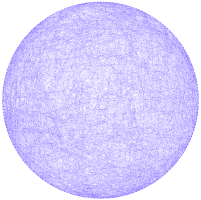
Since We know that $P(n) = O(\frac{1}{\log n})$, therefore after running this algorithm $O(\log^2 n)$ times, the probability of missing a specific min-cut is

$$\mathbb{P} = (1 - P(n))^{O(\log^2 n)} \leq (1 - \frac{c}{\log n})^{3 \log^2 n / c} \leq \exp(-3 \log n) = \frac{1}{n^3}$$

And there are at most $\binom{n}{2}$ min-cuts, hence the probability of missing any min-cut is

$$\mathbb{P}[\text{miss any min} - \text{cut}] \leq \binom{n}{2} \frac{1}{n^3} = O(\frac{1}{n})$$

Example



Total edges: 2517
Total vertices: 200
Maximum degree: 39
Minimum degree: 20
average degree: 25
Mincut is **17**

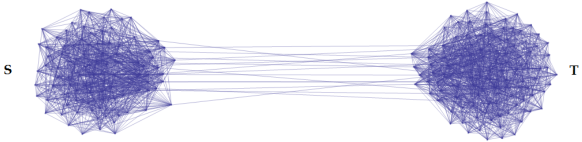
Karger's Algorithm

```
def MinCut(graph, t):
    while len(graph) > t:
        start = random.choice(graph.keys())
        finish = random.choice(graph[start])

    # Adding the edges from the absorbed node:
        for edge in graph[finish]:
            if edge != start:  # this stops us from making a self-loop
                graph[start].append(edge)

    # Deleting the references to the absorbed node
        # and changing them to the source node:
        for edge1 in graph[finish]:
            graph[edge1].remove(finish)
            if edge1 != start:  # this stops us from re-adding all the edges in start.
                graph[edge1].append(start)
        del graph[finish]

    # Calculating and recording the mincut
    mincut = len(graph[graph.keys()[0]])
    cuts.append(mincut)
```

...

```
# Running times
count = len( graph ) * len( graph ) * int( math.log(len( graph )))
while i < count:
    graph1 = copy.deepcopy(graph)
        g = MinCut(graph1,2)
```

Result



It gets the number of edges between vertex set S, T is **17**.

Karger-Stein Algorithm

For Karger-Stein Algorithm

```
def FastMinCut(graph):

    if len(graph) < 6:
        return MinCut(graph, 2)
    else :
        t = 1 + int(len(graph) / math.sqrt(2))
        graph_1 = MinCut(graph, t)
        graph_2 = MinCut(graph, t)
        if len(graph_1) > len(graph_2):
            return FastMinCut(graph_2)
        else:
            return FastMinCut(graph_1)
```

...

```
# Running times
count = int( math.log(len( graph ))) * int( math.log(len( graph )))
while i < count:
    graph1 = copy.deepcopy(graph)
        g = FastMinCut(graph1)
        i += 1
```

It will get the same result as Karger's algorithm.

Summing up

Comparision of Karger's algorithm and Karger-Stein algorithm.

Bound	**Karger algorithm**	**Karger-Stein algorithm**
Probability	$O(1/n^2)$	$O(1/\log n)$
Cost	$O(n^2)$	$O(n^2 \log n)$
Running times	$\binom{n}{2} \log n$	$\log^2 n$
Totol Order	$O(n^4 \log n)$	$O(n^2 \log^3 n)$

Improvement

- Parallel Algorithms
  - The parallel version of contraction algorithm $\mathcal{RNC}$ runs in polylogarithmic time using n^2 processors on a PRAM[6].

[6] David R Karger and Clifford Stein. "A new approach to the minimum cut problem". In: *Journal of the ACM (JACM)* 43.4 (1996), pp. 601–640.

[7] Timo Kötzing et al. "Ant colony optimization and the minimum cut problem". In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation.* ACM. 2010, pp. 1393–1400.

[8] Frank Neumann, Joachim Reichel, and Martin Skutella. "Computing minimum cuts by randomized search heuristics". In: *Algorithmica* 59.3 (2011), pp. 323–342.

Improvement

- Parallel Algorithms
  - The parallel version of contraction algorithm $\mathcal{RNC}$ runs in polylogarithmic time using n^2 processors on a PRAM[6].
- Random Algorithems
  - Timo Ktzing et al. apply "Ant colony optimization method" can obtain the solution in expected polynomial time[7].
  - Frank Neumann et al. apply "Randomized Search Heuristics" method to obtain the solution in expected polynomial time[8].

---

[6] David R Karger and Clifford Stein. "A new approach to the minimum cut problem". In: *Journal of the ACM (JACM)* 43.4 (1996), pp. 601–640.

[7] Timo Kötzing et al. "Ant colony optimization and the minimum cut problem". In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM. 2010, pp. 1393–1400.

[8] Frank Neumann, Joachim Reichel, and Martin Skutella. "Computing minimum cuts by randomized search heuristics". In: *Algorithmica* 59.3 (2011), pp. 323–342.

Further Reading

Classical textbook: *Randomized Algorithms*[9].

An supplementary reading material is class notes organized by Sariel Har-Peled[10].

There is a paper analysis most recent algorithm to find min-cut by conducting experimental evaluation the relative performance of these algorithms[11].

---

[9] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge university press, 1995.

[10] Sariel Har-Peledx. *Class notes for Randomized Algorithms*. 2005.

[11] Chandra S Chekuri et al. "Experimental study of minimum cut algorithms". In: *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 1997, pp. 324–333.

Thanks

Any Questions?