Global Minimum Cut Problems

AU4606: Network Optimization

Xiaoming Duan
Department of Automation
Shanghai Jiao Tong University

November 13, 2023

Last few lectures

- Minimum cost flow problems: important concepts
  - Residual graphs
  - Negative cost cycles
  - Optimality conditions
- Minimum cost flow problems: algorithms
  - Generic cycle canceling algorithms $O(m^2 nCU)$
  - Minimum mean cost cycle canceling algorithms
    - Weak polynomiality analysis: $O(m^2 n^2 \log(nC))$
    - Strong polynomiality analysis: $O(n^2 m^3 \log(n))$
  - Network simplex method

Today

1. Global minimum cut problem: formulation
2. MA ordering
3. Random contraction algorithms

Today

1. Global minimum cut problem: formulation
2. MA ordering
3. Random contraction algorithms

What is a global minimum cut problem

Global min cut problem

Given an undirected graph $G = (N, A)$ and capacities $u_{ij} \geq 0$ for all arcs $(i, j) \in A$, find a subset of vertices $S \subset N$ and $S \neq \emptyset$, such that

$$u[S, \bar{S}] = \sum_{(i,j) \in A, i \in S, j \in \bar{S}} u_{ij}$$

is minimized.

What is a global minimum cut problem

Global min cut problem

Given an undirected graph $G = (N, A)$ and capacities $u_{ij} \geq 0$ for all arcs $(i, j) \in A$, find a subset of vertices $S \subset N$ and $S \neq \emptyset$, such that

$$u[S, \bar{S}] = \sum_{(i,j) \in A, i \in S, j \in \bar{S}} u_{ij}$$

is minimized.

A word on directed graphs

- Let $s \in N$, $t \in N$ and $s \neq t$, compute the maximum flow $n(n-1)$ times
- Can we be smarter?

Global minimum cut for directed graphs

s-cut problem

Given a directed graph $G = (N, A)$, capacities $u_{ij} \geq 0$ for all arcs $(i, j) \in A$, and a vertex $s \in S$, find a subset of vertices $S \subset N$ and $s \in S$, such that

$$u[S, \bar{S}] = \sum_{(i,j) \in A, i \in S, j \in \bar{S}} u_{ij}$$

is minimized.

- s-cut can be found by running s-t maximum flow $n - 1$ times

Global minimum cut for directed graphs

s-cut problem

Given a directed graph $G = (N, A)$, capacities $u_{ij} \geq 0$ for all arcs $(i, j) \in A$, and a vertex $s \in S$, find a subset of vertices $S \subset N$ and $s \in S$, such that

$$u[S, \bar{S}] = \sum_{(i,j) \in A, i \in S, j \in \bar{S}} u_{ij}$$

is minimized.

- s-cut can be found by running s-t maximum flow $n - 1$ times

How do we use s-cut to find a global minimum cut?

- A global minimum cut either contains s or not
- Need to find a minimum cut that does not contain s, how?

Global minimum cut for directed graphs

s-cut problem

Given a directed graph $G = (N, A)$, capacities $u_{ij} \geq 0$ for all arcs $(i, j) \in A$, and a vertex $s \in S$, find a subset of vertices $S \subset N$ and $s \in S$, such that

$$u[S, \bar{S}] = \sum_{(i,j)\in A, i\in S, j\in \bar{S}} u_{ij}$$

is minimized.

- s-cut can be found by running s-t maximum flow $n - 1$ times

How do we use s-cut to find a global minimum cut?

- A global minimum cut either contains s or not
- Need to find a minimum cut that does not contain s, how?
  Reverse the arcs!

Global minimum cut can be found by running max flow $2(n - 1)$ times

1. Global minimum cut problem: formulation
2. MA ordering
3. Random contraction algorithms

Using maximum flow for undirected graphs

Find a global min cut in undirected graphs by max flow in directed graphs

- Given undirected graph $G = (N, A)$, build a new graph $G' = (N, A')$:
  - if $(i, j) \in A$, then $(i, j) \in A'$ and $(j, i) \in A'$ with capacity u_{ij}
- Pick an arbitrary vertex $s \in N$
- Compute the maximum s-t flow for $t \in N$ and $t \neq s$ for $n - 1$ times
- Return the found minimum cut over all computations

Cuts are not directed in undirected graphs!

Global minimum cuts: notation

Given an undirected graph $G = (N, A)$

- A cut is a nontrivial set of nodes $S \subset N$
- Given a cut S, the cut set $\delta(S)$ is defined by

$$\delta(S) = \{(i,j) \in A \,|\, i \in S, j \in \bar{S}\}$$

- The cut value $u(S)$ of a cut S is defined by

$$u(S) = \sum_{(i,j)\in\delta(S)} u_{ij}$$

- For two disjoint sets $A \subset N$ and $B \subset N$, $\delta(A, B)$ is defined by

$$\delta(A, B) = \{(i,j) \in A \,|\, i \in A, j \in B\}$$

Similarly, $u(A, B) = \sum_{(i,j)\in\delta(A,B)} u_{ij}$

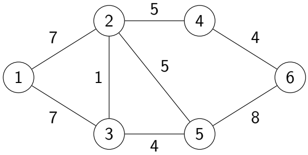
MA ordering: computation

MA: maximum adjacency

Algorithm MA ordering

Pick v_1 arbitrarily from N
$W_1 \leftarrow \{v_1\}$
$k \leftarrow 2$
while $k \leq |N|$ **do**
    Choose v_k from $N \setminus W_{k-1}$ to maximize $u(W_{k-1}, \{v_k\})$
    $W_k \leftarrow W_{k-1} \cup \{v_k\}$
    $k \leftarrow k + 1$
end while

Pick a vertex to maximize sum of arc capacities to previous vertices

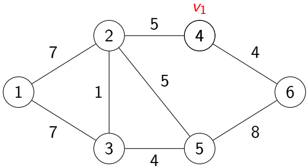


MA: maximum adjacency

Algorithm MA ordering

```
Pick v_1 arbitrarily from N
W_1 ← {v_1}
k ← 2
while k ≤ |N| do
    Choose v_k from N \ W_{k−1} to maximize u(W_{k−1}, {v_k})
    W_k ← W_{k−1} ∪ {v_k}
    k ← k + 1
end while
```

Pick a vertex to maximize sum of arc capacities to previous vertices



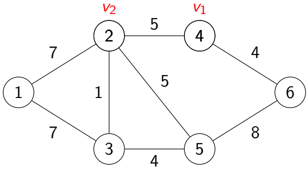
MA ordering: computation

MA: maximum adjacency

Algorithm MA ordering

```
Pick v_1 arbitrarily from N
W_1 ← {v_1}
k ← 2
while k ≤ |N| do
    Choose v_k from N \ W_{k−1} to maximize u(W_{k−1}, {v_k})
    W_k ← W_{k−1} ∪ {v_k}
    k ← k + 1
end while
```

Pick a vertex to maximize sum of arc capacities to previous vertices



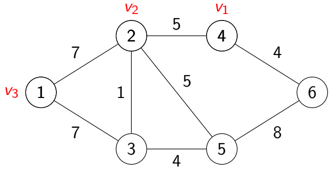
MA ordering: computation

MA: maximum adjacency

Algorithm MA ordering

1: Pick v_1 arbitrarily from N
2: $W_1 \leftarrow \{v_1\}$
3: $k \leftarrow 2$
4: **while** $k \leq |N|$ **do**
5: Choose v_k from $N \setminus W_{k-1}$ to maximize $u(W_{k-1}, \{v_k\})$
6: $W_k \leftarrow W_{k-1} \cup \{v_k\}$
7: $k \leftarrow k + 1$
8: **end while**

Pick a vertex to maximize sum of arc capacities to previous vertices



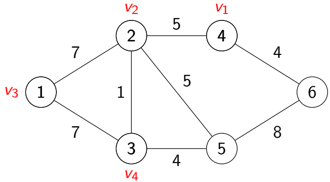
MA ordering: computation

MA: maximum adjacency

Algorithm MA ordering

Pick v_1 arbitrarily from N
$W_1 \leftarrow \{v_1\}$
$k \leftarrow 2$
while $k \leq |N|$ **do**
    Choose v_k from $N \setminus W_{k-1}$ to maximize $u(W_{k-1}, \{v_k\})$
    $W_k \leftarrow W_{k-1} \cup \{v_k\}$
    $k \leftarrow k + 1$
end while

Pick a vertex to maximize sum of arc capacities to previous vertices



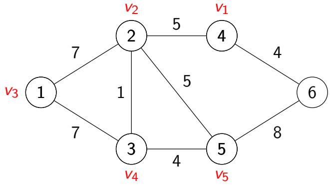
MA ordering: computation

MA: maximum adjacency

Algorithm MA ordering

Pick v_1 arbitrarily from N
$W_1 \leftarrow \{v_1\}$
$k \leftarrow 2$
while $k \leq |N|$ **do**
    Choose v_k from $N \setminus W_{k-1}$ to maximize $u(W_{k-1}, \{v_k\})$
    $W_k \leftarrow W_{k-1} \cup \{v_k\}$
    $k \leftarrow k + 1$
end while

Pick a vertex to maximize sum of arc capacities to previous vertices



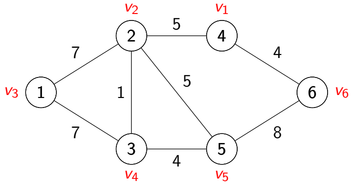
MA ordering: computation

MA: maximum adjacency

Algorithm MA ordering

Pick v_1 arbitrarily from N
$W_1 \leftarrow \{v_1\}$
$k \leftarrow 2$
while $k \leq |N|$ **do**
    Choose v_k from $N \setminus W_{k-1}$ to maximize $u(W_{k-1}, \{v_k\})$
    $W_k \leftarrow W_{k-1} \cup \{v_k\}$
    $k \leftarrow k + 1$
end while

Pick a vertex to maximize sum of arc capacities to previous vertices



MA ordering: computation

Algorithm MA ordering

Pick v_1 arbitrarily from N
$W_1 \leftarrow \{v_1\}$
$k \leftarrow 2$
while $k \leq |N|$ **do**
    Choose v_k from $N \setminus W_{k-1}$ to maximize $u(W_{k-1}, \{v_k\})$
    $W_k \leftarrow W_{k-1} \cup \{v_k\}$
    $k \leftarrow k + 1$
end while

The algorithm runs in $O(mn)$

MA ordering: computation

Algorithm MA ordering

```
Pick v_1 arbitrarily from N
W_1 ← {v_1}
k ← 2
while k ≤ |N| do
    Choose v_k from N \ W_{k-1} to maximize u(W_{k-1}, {v_k})
    W_k ← W_{k-1} ∪ {v_k}
    k ← k + 1
end while
```

The algorithm runs in $O(mn)$

Why is it useful?

MA ordering: computation

Algorithm MA ordering

1: Pick v_1 arbitrarily from N
2: $W_1 \leftarrow \{v_1\}$
3: $k \leftarrow 2$
4: **while** $k \leq |N|$ **do**
5: Choose v_k from $N \setminus W_{k-1}$ to maximize $u(W_{k-1}, \{v_k\})$
6: $W_k \leftarrow W_{k-1} \cup \{v_k\}$
7: $k \leftarrow k + 1$
8: **end while**

The algorithm runs in $O(mn)$

Why is it useful?

Properties of MA ordering

Given an MA ordering $v_1, \ldots, v_n$ of an undirected graph with n vertices, $\{v_n\}$ is a minimum v_n-v_{n-1} cut.

Global minimum cut from MA ordering

Properties of MA ordering

Given an MA ordering $v_1, \ldots, v_n$ of an undirected graph with n vertices, $\{v_n\}$ is a minimum v_n-v_{n-1} cut.

1. Either some global minimum cut S^* is a v_n-v_{n-1} cut
2. Or no global minimum cut is a v_n-v_{n-1} cut

- In case 1, we obtain a global minimum cut. Why?

Global minimum cut from MA ordering

Properties of MA ordering

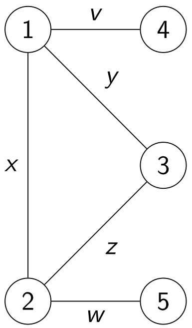
Given an MA ordering $v_1, \ldots, v_n$ of an undirected graph with n vertices, $\{v_n\}$ is a minimum v_n-v_{n-1} cut.

1. Either some global minimum cut S^* is a v_n-v_{n-1} cut
2. Or no global minimum cut is a v_n-v_{n-1} cut

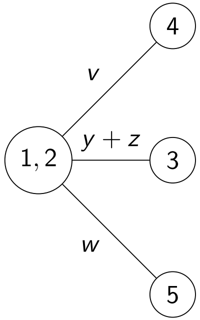
- In case 1, we obtain a global minimum cut. Why?
- In case 2, v_{n-1} and v_n must belong to same set for all global min cuts
  - Treat two nodes as one via contracting

Contracting two nodes v_n and v_{n-1}

- Remove v_n and v_{n-1} from N and add v'
- Remove arcs (i, v_n) and (j, v_{n-1}) for $i \neq j$, add (i, v') and (j, v') with corresponding capacities
- Remove arcs (i, v_n) and (i, v_{n-1}), add (i, v') with capacity $u_{iv'} = u_{iv_n} + u_{iv_{n-1}}$



(a) Graph G



(b) Contract 1 and 2

Global minimum cut from MA ordering: procedure

Algorithm MA ordering-based global minimum cuts

1: value $\leftarrow \infty$, $S \leftarrow \emptyset$, $\ell \leftarrow |N|$
2: **while** $\ell > 1$ **do**
3:     Compute MA ordering $v_1, \ldots, v_\ell$
4:     **if** $u(\{v_\ell\}) <$ value **then**
5:         value $\leftarrow u(\{v_\ell\})$
6:         $S \leftarrow \{v_\ell\}$
7:     **end if**
8:     Contract v_ℓ and $v_{\ell-1}$, $\ell \leftarrow \ell - 1$
9: **end while**

1. Either some global minimum cut S^* is a v_n-v_{n-1} cut
2. Or no global minimum cut is a v_n-v_{n-1} cut

- Unknown which of the above two cases is true
- Contracting nodes and keep track of the minimum cuts
- Correctness can be proved via inductive arguments

Properties of MA ordering

Given an MA ordering $v_1, \ldots, v_n$ of an undirected graph with n vertices, $\{v_n\}$ is a minimum v_n-v_{n-1} cut.

Today

1. Global minimum cut problem: formulation
2. MA ordering
3. Random contraction algorithms

Random contraction algorithm: procedure

What edges are likely in cut set $\delta(S^*)$?

Random contraction algorithm: procedure

What edges are likely in cut set $\delta(S^*)$? Edges with small capacities!

Random contraction algorithm: procedure

What edges are likely in cut set $\delta(S^*)$? Edges with small capacities!

Basic idea of randomization

- Pick arcs with probability proportional to the arc capacities
- Contract the end points of these arcs
- Repeat recursively until two nodes left, and return the cut

When contracting two nodes, arc between them will not be in the cut set!

Random contraction algorithm: procedure

What edges are likely in cut set $\delta(S^*)$? Edges with small capacities!

Basic idea of randomization

- Pick arcs with probability proportional to the arc capacities
- Contract the end points of these arcs
- Repeat recursively until two nodes left, and return the cut

When contracting two nodes, arc between them will not be in the cut set!

Algorithm Random contraction

```
1: while |N| > 2 do
2:     Pick (i, j) with probability proportional to u_ij
3:     Contract i and j, update capacities
4: end while
```

The algorithm runs in $O(n^2)$.

Random contraction algorithm: analysis

Cut value and total arc capacities

Let S^* be a global minimum cut and $W = \sum_{(i,j)\in A} u_{ij}$, then

$$W \geq \frac{n}{2} u(S^*).$$

Corollary: S^* survives the first contraction with probability at least $1 - \frac{2}{n}$

Let W_k be arc capacities after k contractions, then $W_k \geq \frac{(n-k)}{2} u(S^*)$.

Probability of returning a minimum cut

The probability that the random contraction algorithm returns a global minimum cut is at least $\frac{1}{C_n^2}$.

Obtaining a minimum cut with high probability

The random contraction algorithm finds a global minimum cut in $O(n^4 \log n)$ with high probability.

Recursive random contraction: procedure

Can we do better?

- In the first few contractions, S^* survives with fairly high probability

Probability of surviving

The probability that a given global minimum cut S^* survives after $n-t$ contractions is at least $\frac{C_t^2}{C_n^2}$.

If we run the algorithm for $t = \lceil \frac{n}{\sqrt{2}} + 1 \rceil$, then S^* survives w.p. at least $\frac{1}{2}$.

Algorithm RecursiveRandomcontraction(G,n)

```
if n ≤ 6 then
    Find a global minimum cut in G by exhaustive search
else
    for i = 1 : 2 do
        H_i ←random contraction of G down to ⌈n/√2 + 1⌉ nodes
        S_i ←RecursiveRandomcontraction(H_i, ⌈n/√2 + 1⌉)
    end for
    if u(S_1) ≤ u(S_2) then
        return S_1
    else
        return S_2
    end if
end if
```

Recursive random contraction: analysis

Running time

The recursive random contraction algorithm runs in $O(n^2 \log n)$

Probability of returning correct cut

The recursive random contraction algorithm returns a global minimum cut S^* with probability $\Omega(\frac{1}{\log n})$.

Obtaining a minimum cut with high probability

The recursive random contraction algorithm finds a global minimum cut in $O(n^2 \log^3 n)$ with high probability.

Upcoming

Week 1-8 (AU4606 & AI4702):

- Introduction (1 lecture)
- Preparations (3 lectures)
  - basics of graph theory
  - algorithm complexity and data structure
  - graph search algorithm
- Shortest path problems (3 lectures)
- Maximum flow problems (5 lectures)
- Minimum cost flow problems (3 lectures)
- Introduction to multi-agent systems (1 lecture)
- Introduction to cloud networks (1 lecture)

Week 9-16 (AU4606):

- Simplex and network simplex methods (1 lecture)
- Global minimum cut problems (this and next few lectures)
- Minimum spanning tree problems (3 lectures)