Chapter 9

Fast Approximate Max Flow in Undirected Graphs

Advanced Algorithms

SS 2019

Fabian Kuhn

Cut Sparsifiers

Last week: cut sparsifiers by [Benczúr, Karger; '02]

Given: graph $G = (V, E) \Longrightarrow$ weighted graph $H = (V, E', w)$ with $E' \subseteq E$

- Such that all cuts are preserved up to a $(1 \pm \varepsilon)$-factor & $|E'| = O\left(\frac{n \log n}{\varepsilon}\right)$
- Can be computed in time $\tilde{O}(m)$
- Also works for weighted graphs

Gives an immediate algorithms to approximate cut problems:

- First, compute a cut sparsifier, then solve the problem on the sparsifier
- Outputs an almost optimal cut on G, faster if running time depends on m
- Example: $(1+\varepsilon)$-approximate minimum s-t cut in time $\tilde{O}\left(n^{3/2}/\varepsilon^3\right)$
  - in undirected graphs…

What about the max flow problem?

- We can approximate the value of the maximum flow, but it is not clear how to construct a flow (the sparsifier does not contain most of the edges of G)

Max Flow with Cut Sparsifiers

- Benczúr and Karger give a way to use their cut sparsifier also for the undirected max flow problem

The main ideas are:

- Replace "important" edges in G by multiple parallel edges (capacity is divided evenly among the multiple edges replacing an original edge)
- "important" $=$ small edge strength (strong edge connectivity k_e)
  - Small edge strength $\Longrightarrow$ large sampling probability in sparsifier algorithm
- This can be done such that the number of edges only grows by a constant factor and every edge e has sampling probability $\tilde{O}(n/m)$ in the sparsifier alg.
- One can then randomly partition the graph into $\tilde{O}(m/n)$ parts
  - All cuts are close to their expected size (same analysis as sparsifier analysis)
- We can then solve independent max flow problems in all the $\tilde{O}(n/m)$ parts and add the flows to get an $(1-\varepsilon)$-approximate max flow for G
- Allows to turn an existing $O\left(m^{3/2}\right)$ into an $\tilde{O}(m\sqrt{n}/\varepsilon)$ approximate alg.
- **Today:** Main ideas of a faster (more involved) way of solving max flow

A slightly more general Problem

Given:

- Undirected graph $G = (V, E, c)$ with edge capacities $c_e > 0$
- Every node $v \in V$ has a demand $b_v \in \mathbb{R}$ s.t. $\sum_{v \in V} b_v = 0$

Goal: Find a flow f such that

$$\forall v \in V : f_{out}(v) - f_{in}(v) = b_v$$

$$\text{minimize} \quad \max_{e \in E} \frac{|f_e|}{c_e}$$

How to solve max *s*-*t* flow:

1. Set $b_s = 1$, $b_t = -1$, $b_v = 0$ for $v \notin \{s, t\}$
2. Solve the above problem
3. Scale flow s.t. $\max_{e \in E} \frac{|f_e|}{c_e} = 1$

Matrix Representation of Max Flow

- Each edge $\{u, v\}$ added either as (u, v) or as (v, u), flow f_e on edge $e = (u, v)$:
  - $f_e > 0$: flow from u to v , $f_e < 0$: flow from v to u
- B: node-edge incidence matrix (B is an $n \times m$ matrix)
  - Edge $e = (u, v)$: $B_{u,e} = +1$, $B_{v,e} = -1$, $B_{w,e} = 0$ for $w \notin \{u, v\}$

Valid flow: $\boldsymbol{f}$ is valid $\Leftrightarrow$ $B\boldsymbol{f} = \boldsymbol{b}$

Matrix Representation of Max Flow

- B: node-edge incidence matrix (B is an $n \times m$ matrix)
  - Edge $e = (u, v)$: $B_{u,e} = +1$, $B_{v,e} = -1$, $B_{w,e} = 0$ for $w \notin \{u, v\}$

Valid flow: $\boldsymbol{f}$ is valid $\Leftrightarrow$ $B\boldsymbol{f} = \boldsymbol{b}$

Capacity matrix:

Goal: minimize $\|C^{-1}\boldsymbol{f}\|_\infty$ s.t. $B\boldsymbol{f} = \boldsymbol{b}$

Dual Problem (Generalization of Min Cut)

For every cut $(\boldsymbol{S}, \boldsymbol{V} \setminus \boldsymbol{S})$:

- Capacity of cut c_S
- Total amount of flow across cut $(S, V \setminus S)$ is at least

$$b_S := \sum_{v \in S} b_v$$

Max flow min cut theorem:

$$\exists \text{ cut } S : \text{opt}(\boldsymbol{b}) = \frac{b_S}{c_S}$$

Dual Problem (Generalization of Min Cut)

Dual problem: maximum congested cut

- Vertex potentials $\boldsymbol{x} \in \mathbb{R}^n, \quad x_v \in \mathbb{R}$
- Goal:

$$\max \boldsymbol{b}^\top \boldsymbol{x} \quad s.t. \ \|CB^\top \boldsymbol{x}\|_1 \leq 1$$

- Example: consider a cut $(S, V \setminus S)$:
  Vector $\boldsymbol{x}_S$ is characteristic vector of set S ($x_v = 1 \Leftrightarrow v \in S$)

Dual Problem (Generalization of Min Cut)

Goal: vertex potentials $x_v \in \mathbb{R}$:

$$\max \boldsymbol{b}^\top \boldsymbol{x} \quad s.t. \ \|CB^\top \boldsymbol{x}\|_1 \leq 1$$

Claim: Opt. solution of above problem has value $\mathrm{opt}(\boldsymbol{b})$.

- We have seen that there exists $\boldsymbol{x}$ with $\boldsymbol{b}^\top \boldsymbol{x} = \mathrm{opt}(\boldsymbol{b})$

Congestion Approximator

A method that allows to get a good approximation of $\text{opt}(\boldsymbol{b})$

Definition: α-congestion approximator is a matrix $R \subseteq \mathbb{R}^{\ell \times n}$ s.t.

$$\forall \boldsymbol{b} \in \mathbb{R}^{\boldsymbol{n}}: \ \ \|R\boldsymbol{b}\|_\infty \leq \text{opt}(\boldsymbol{b}) \leq \alpha \cdot \|R\boldsymbol{b}\|_\infty$$

Example 1:

- One row for each possible cut $(S, V \setminus S)$:

$$R_{S,v} = \frac{1}{c_S}$$

Congestion Approximator

A method that allows to get a good approximation of $\text{opt}(\boldsymbol{b})$

Definition: α-congestion approximator is a matrix $R \subseteq \mathbb{R}^{\ell \times n}$ s.t.

$$\forall \boldsymbol{b} \in \mathbb{R}^{\boldsymbol{n}}: \; \|R\boldsymbol{b}\|_\infty \leq \text{opt}(\boldsymbol{b}) \leq \alpha \cdot \|R\boldsymbol{b}\|_\infty$$

Example 2:

- Assume T is a maximum weight spanning tree
- Add one row for each edge e of T, let S_e be the induced cut of e:

$$R_{e,v} = \frac{b_v}{c_{S_e}}$$

- Measures exactly the cost of routing the flow on the tree T
- Routing on the tree incurs at most a factor $m \Longrightarrow \alpha = m$

Congestion Approximator

A method that allows to get a good approximation of $\mathrm{opt}(\boldsymbol{b})$

Definition: α-congestion approximator is a matrix $R \subseteq \mathbb{R}^{\ell \times n}$ s.t.

$$\forall \boldsymbol{b} \in \mathbb{R}^{\boldsymbol{n}}: \ \|R\boldsymbol{b}\|_\infty \leq \mathrm{opt}(\boldsymbol{b}) \leq \alpha \cdot \|R\boldsymbol{b}\|_\infty$$

Example 3:

- Use all the trees of a low-congestion tree embedding
  - As considered in the lectures on May 17 and June 7
  - When picking a random tree, expected congestion of each edge is at most $O(\log n)$ times the congestion for an optimal solution of an arbitrary multicommodity flow problem
- Add one row for each tree T and each edge e of T, let S_e be the induced cut of e:

$$R_{e,v} = \frac{b_v}{c_{S_e}}$$

- Gives $\alpha = O(\log n)$

Congestion Approximator

A method that allows to get a good approximation of $\mathrm{opt}(\boldsymbol{b})$

Definition: α-congestion approximator is a matrix $R \subseteq \mathbb{R}^{\ell \times n}$ s.t.

$$\forall \boldsymbol{b} \in \mathbb{R}^{\boldsymbol{n}}: \ \|R\boldsymbol{b}\|_\infty \leq \mathrm{opt}(\boldsymbol{b}) \leq \alpha \cdot \|R\boldsymbol{b}\|_\infty$$

Example 4:

- Add one row for each tree T and each edge e of T of a low-congestion tree embedding, let S_e be the induced cut of e
- The construction required $\tilde{O}(m)$ trees $\Longrightarrow R$ has $\tilde{O}(mn)$ rows
- Can be improved by first computing a cut sparsifier
  - Now, we only need $\tilde{O}(n)$ trees $\Longrightarrow R$ has $\tilde{O}(n^2)$ rows
- In [Sherman; 2013], a recursive variant of this is described:
  - Based on a construction of [Mardy; 2010]
  - Instead of trees, embed into more complicated structures (needs less of them)
  - Gives a congestion approximator R with $n^{1+o(1)}$ rows that can be computed in time $m \cdot n^{o(1)}$ and with $\alpha = n^{o(1)}$

An Unconstrained Version of the Problem

- Assume that an α-congestion approximator R with $\leq n^2/2$ rows is given

Using it, we can turn max flow into an unconstrained optimization problem:

$$\min_{\text{flow } \boldsymbol{f}} \gamma(\boldsymbol{f}) \coloneqq \|C^{-1}\boldsymbol{f}\|_\infty + 2\alpha \cdot \|R(\boldsymbol{b} - B\boldsymbol{f})\|_\infty$$

Intuition:

- Optimal solution is an optimal flow $\boldsymbol{f}$

- Approximate solution will give an almost valid flow $\boldsymbol{f}$

An Unconstrained Version of the Problem

- Assume that an α-congestion approximator R with $\leq n^2/2$ rows is given

Using it, we can turn max flow into an unconstrained optimization problem:

$$\min_{\text{flow } \boldsymbol{f}} \gamma(\boldsymbol{f}) \coloneqq \|C^{-1}\boldsymbol{f}\|_\infty + 2\alpha \cdot \|R(\boldsymbol{b} - B\boldsymbol{f})\|_\infty$$

Theorem 1:

There is an algorithm `AlmostRoute`$(\boldsymbol{b}, \varepsilon)$ that returns a flow f for which

$$\gamma(\boldsymbol{f}) \leq (1+\varepsilon) \cdot \text{opt}(\boldsymbol{b}).$$

The algorithms requires $O\left(\frac{\alpha^2 \cdot \log \alpha \cdot \log n}{\varepsilon^3}\right)$ iterations that require time $\tilde{O}(m)$ plus a multiplication by R and by $R^\top$.

An Unconstrained Version of the Problem

$$\min_{\text{flow } \boldsymbol{f}} \gamma(\boldsymbol{f}) \coloneqq \|C^{-1}\boldsymbol{f}\|_\infty + 2\alpha \cdot \|R(\boldsymbol{b} - B\boldsymbol{f})\|_\infty$$

Theorem 2: There is an algorithm that computes a valid $(1 + \varepsilon)$-approximate flow that applies `AlmostRoute` $O(\log n)$ times.

An Unconstrained Version of the Problem

$$\min_{\text{flow } \boldsymbol{f}} \gamma(\boldsymbol{f}) \coloneqq \|C^{-1}\boldsymbol{f}\|_\infty + 2\alpha \cdot \|R(\boldsymbol{b} - B\boldsymbol{f})\|_\infty$$

Theorem 2: There is an algorithm that computes a valid $(1+\varepsilon)$-approximate flow that applies `AlmostRoute` $O(\log n)$ times.

A Differentiable Objective Function

Softmax function (on a vector $\boldsymbol{x} \in \mathbb{R}^d$):

$$\text{lmax}(\boldsymbol{x}) := \ln\left(\sum_{i=1}^{d}(e^{x_i} + e^{-x_i})\right)$$

Properties of softmax:

$$\|\boldsymbol{x}\|_\infty \leq \text{lmax}(\boldsymbol{x}) \leq \|\boldsymbol{x}\|_\infty + \ln(2d)$$

$$\begin{aligned} \|\nabla \text{lmax}(\boldsymbol{x})\|_1 &\leq 1 \\ \nabla \text{lmax}(\boldsymbol{x})^\top \boldsymbol{x} &\geq \text{lmax}(\boldsymbol{x}) - \ln(2d) \\ \|\nabla \text{lmax}(\boldsymbol{x}) - \text{lmax}(\boldsymbol{y})\|_1 &\leq \|\boldsymbol{x} - \boldsymbol{y}\|_\infty \end{aligned}$$

A Differentiable Objective Function

Softmax function (on a vector $\boldsymbol{x} \in \mathbb{R}^d$):

$$\operatorname{lmax}(\boldsymbol{x}) \coloneqq \ln\left(\sum_{i=1}^{d}(e^{x_i} + e^{-x_i})\right)$$

Replace

$$\gamma(\boldsymbol{f}) \coloneqq \|C^{-1}\boldsymbol{f}\|_\infty + 2\alpha \cdot \|R(\boldsymbol{b} - B\boldsymbol{f})\|_\infty$$

By

$$\phi(\boldsymbol{f}) \coloneqq \operatorname{lmax}(C^{-1}\boldsymbol{f}) + \operatorname{lmax}\big(2\alpha \cdot R(\boldsymbol{b} - B\boldsymbol{f})\big)$$

AlmostRoute$(\boldsymbol{b}, \varepsilon)$

- Initialize $\boldsymbol{f} = 0$, scale $\boldsymbol{b}$ so $2\alpha \cdot \|R\boldsymbol{b}\|_\infty = 16\varepsilon^{-1}\ln n$
- Repeat:
  - While $\phi(\boldsymbol{f}) < 16\varepsilon^{-1}\ln n$, scale $\boldsymbol{f}$ and $\boldsymbol{b}$ up by $17/16$
  - Set $\delta := \|C \cdot \nabla\phi(\boldsymbol{f})\|_1$
  - If $\delta \geq \varepsilon/4$, set $f_e := f_e - \frac{\delta}{1+4\alpha^2} \cdot \operatorname{sgn}\left(\left(\nabla\phi(\boldsymbol{f})\right)_e\right) \cdot c_e$
  - Otherwise, terminate and output $\boldsymbol{f}$ after undoing all scalings

    Also, output vertex potentials $\boldsymbol{x} := \mathrm{R}^\top \cdot \nabla \mathrm{lmax}\big(2\alpha \cdot R(\boldsymbol{b} - B\boldsymbol{f})\big)$

AlmostRoute$(\boldsymbol{b}, \varepsilon)$

Lemma: When `AlmostRoute`$(\boldsymbol{b}, \varepsilon)$ terminates, we have

$$\phi(\boldsymbol{f}) \leq (1+\varepsilon) \cdot \frac{\boldsymbol{b}^\top \boldsymbol{x}}{\|CB^\top \boldsymbol{x}\|_1}$$

AlmostRoute$(\boldsymbol{b}, \varepsilon)$

Lemma: When `AlmostRoute`$(\boldsymbol{b}, \varepsilon)$ terminates, we have

$$\phi(\boldsymbol{f}) \leq (1+\varepsilon) \cdot \frac{\boldsymbol{b}^\top \boldsymbol{x}}{\|CB^\top \boldsymbol{x}\|_1}$$

AlmostRoute$(\boldsymbol{b}, \varepsilon)$

Lemma: When `AlmostRoute`$(\boldsymbol{b}, \varepsilon)$ terminates, we have

$$\phi(\boldsymbol{f}) \leq (1+\varepsilon) \cdot \frac{\boldsymbol{b}^\top \boldsymbol{x}}{\|CB^\top \boldsymbol{x}\|_1}$$

AlmostRoute$(\boldsymbol{b}, \varepsilon)$

Lemma: When `AlmostRoute`$(\boldsymbol{b}, \varepsilon)$ terminates, we have

$$\phi(\boldsymbol{f}) \leq (1+\varepsilon) \cdot \frac{\boldsymbol{b}^\top \boldsymbol{x}}{\|CB^\top \boldsymbol{x}\|_1}$$

AlmostRoute$(\boldsymbol{b}, \varepsilon)$

Lemma: The number of iterations of `AlmostRoute`$(\boldsymbol{b}, \varepsilon)$ is at most

$$O\left(\frac{\alpha^2 \cdot \log \alpha \cdot \log n}{\varepsilon^3}\right).$$

- Initialize $\boldsymbol{f} = 0$, scale $\boldsymbol{b}$ so $2\alpha \cdot \|R\boldsymbol{b}\|_\infty = 16\varepsilon^{-1} \ln n$
- Repeat:
  - While $\phi(\boldsymbol{f}) < 16\varepsilon^{-1} \ln n$, scale $\boldsymbol{f}$ and $\boldsymbol{b}$ up by $17/16$
  - Set $\delta \coloneqq \|C \cdot \nabla\phi(\boldsymbol{f})\|_1$
  - If $\delta \geq \varepsilon/4$, set $f_e \coloneqq f_e - \frac{\delta}{1+4\alpha^2} \cdot \operatorname{sgn}\left(\left(\nabla\phi(\boldsymbol{f})\right)_e\right) \cdot c_e$
  - Otherwise, terminate

AlmostRoute$(\boldsymbol{b}, \varepsilon)$

Lemma: The number of iterations of `AlmostRoute`$(\boldsymbol{b}, \varepsilon)$ is at most

$$O\left(\frac{\alpha^2 \cdot \log \alpha \cdot \log n}{\varepsilon^3}\right).$$