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 & $|E'| = O\left(\frac{n \log n}{\varepsilon^2}\right)$
- 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

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

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

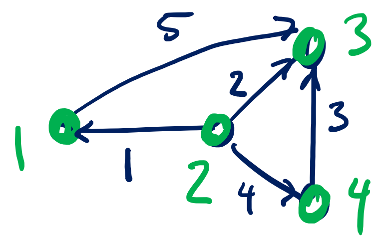
How to solve max s-t flow:

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 $e = \underline{(u, v)}$:
  - $f_e > 0$: 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 $w \notin \{u, v\}$

Example:



$$B = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 & +1 \\ +1 & +1 & 0 & +1 & 0 \\ 0 & -1 & -1 & 0 & -1 \\ 0 & 0 & +1 & -1 & 0 \end{pmatrix} \end{array}$$

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

$$(Bf)_v = \sum_{e=(v,w)} f_e + \sum_{e=(w,v)} -f_e \overset{!}{=} b_v$$

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:

$$C = \begin{pmatrix} c_1 & & & 0 \\ & c_2 & & \\ & & \ddots & \\ 0 & & & c_m \end{pmatrix}$$

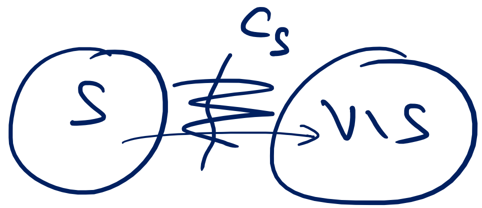
Goal: minimize $\|C^{-1}\boldsymbol{f}\|_\infty$ s.t. $\underline{B\boldsymbol{f} = \boldsymbol{b}}$

opt(b)

Dual Problem (Generalization of Min Cut)

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

- Capacity of cut c_S



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

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

$$\boxed{\mathrm{opt}(b) \geq \frac{b_S}{c_S}}$$

Max flow min cut theorem:

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

Dual Problem (Generalization of Min Cut)

Dual problem: maximum congested cut

- Vertex potentials $\boldsymbol{x} \in \mathbb{R}^n$, $x_v \in \mathbb{R}$
- 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 ($x_v = 1 \Leftrightarrow v \in S$)

$$b^T x = \sum_{v \in S} b_v = b_S$$

$$(CB^T x)_e = c_e (x_u - x_v) \quad \text{if } e = (u,v)$$

$$m \left(\; C \; \right)_{m} \left(\cdots \right)_{n} \left(| \right)_{n} = {}_{m}\left(\cdot \right)$$

$$\|CB^T x\|_1 = \sum_e c_e |x_u - x_v| = c_S$$

after scaling s.t. $\|CB^T x\|_1 = 1$

$$x'_v = \frac{1}{c_S} \Leftrightarrow v \in S$$

$$b^T x = \frac{b_S}{c_S}$$

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 $\boldsymbol{b}^\top \boldsymbol{x} = \mathrm{opt}(\boldsymbol{b})$

We need to show that for all $x \in \mathbb{R}^n$, $b^T x \leq \mathrm{opt}(b) \cdot \|CB^T x\|_1$

assume valid flow f

$$\boxed{\mathrm{opt}(b) \geq \frac{b^T x}{\|CB^T x\|_1}}$$

$$\rho_e := \frac{f_e}{c_e}$$

$$b^T x = \sum_{v \in V} x_v \cdot b_v = \sum_{e=(u,v)\in E} f_e (x_u - x_v)$$

$$= \sum_{e=(u,v)} \rho_e \cdot c_e (x_u - x_v)$$

$$= \rho^T \cdot (CB^T x) \leq \|\rho\|_\infty \cdot \|CB^T x\|_1 \qquad (*)$$

$$\Longrightarrow \|\rho\|_\infty \geq \frac{b^T x}{\|CB^T x\|_1}$$

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 1:

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

$$R_{S,v} = \frac{1}{c_S} \quad \text{for } v \in S \qquad R_{S,v} = 0 \text{ for } v \notin S$$

$$(Rb)_S = \sum_{v\in S} \frac{1}{c_S} \cdot b_v = \frac{b_S}{c_S} \leq \mathrm{opt}(b)$$

max flow ~~min cut~~ thm $\rightarrow \exists S: (Rb)_S = \mathrm{opt}(b)$

$\Longrightarrow$ we get $\alpha = 1$

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.

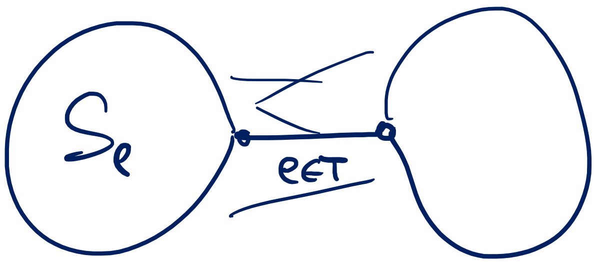
$$\forall \boldsymbol{b} \in \mathbb{R}^{\boldsymbol{n}}:\ \ \|R\boldsymbol{b}\|_\infty \leq \mathrm{opt}(\boldsymbol{b}) \leq \alpha \cdot \|R\boldsymbol{b}\|_\infty$$

Example 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 $m \Longrightarrow \alpha = m$



$\mathrm{rload}_T(e) = |b_{S_e}| / c_e$

$c_e \geq \frac{1}{m} \cdot c_{S_e}$

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 [Ma~~rd~~dry; 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 $\alpha = n^{o(1)}$

$n \cdot 2^{O(\sqrt{\log n})}$

An Unconstrained Version of the Problem

~~f~~ feasible $\Rightarrow$ $Bf = b$

- 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 $\boldsymbol{f}$

f valid $\Longleftrightarrow$ $\gamma(f) = \|C^{-1}f\|$

Assume general f

f': opt flow for the remaining demands $b - Bf$

$$2\,\mathrm{opt}(b-Bf) \leq 2\alpha \cdot \|R(b-Bf)\|_\infty$$

$$\gamma(f) \geq \|C^{-1}f\|_\infty + 2 \cdot \mathrm{opt}(b-Bf)$$

$$\|C^{-1}f\|_\infty \geq \mathrm{opt}(b) - \mathrm{opt}(b-Bf)$$

$= \mathrm{opt}(b-Bf)$

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

because $\|C^{-1}f\|_\infty + \|C^{-1}f'\|_\infty \geq \mathrm{opt}(b)$

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

$b_0 := b$, $f_0 :=$ AlmostRoute(b, ε) $\longrightarrow$ $\gamma(f_0) \leq (1+\varepsilon)\, opt(b)$

for $i := 1$ to T (for $T = \lceil \log_{4/3}(2m) \rceil$)

$\quad b_i := b_{i-1} - B f_{i-1}$ (rem. demands)

$\quad f_i :=$ AlmostRoute$(b_i, 1/2)$

$b_{T+1} = b_T - B f_T$

$f_{T+1} :=$ flow induced by routing b_{T+1} on max. weight sp. tree.

Thm1 gives $\gamma(f_0) = \|C^{-1} f_0\|_\infty + 2\alpha \|R b_1\|_\infty \leq (1+\varepsilon)\, opt(b)$

$\forall i \geq 1$: $\gamma(f_i) = \|C^{-1} f_i\|_\infty + 2\alpha \|R b_{i+1}\|_\infty \leq \frac{3}{2} \cdot opt(b_i) \leq \frac{3}{2} \cdot \alpha \|R b_i\|_\infty$

An Unconstrained Version of the Problem

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

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

$$(1+\varepsilon)\,\mathrm{opt}(b) \geq \|C^{-1} f_0\|_\infty + 2\alpha \|R b_1\|_\infty$$
$$\geq \|C^{-1} f_0\|_\infty + \|C^{-1} f_1\|_\infty + 2\alpha \|R b_2\|_\infty + \frac{\alpha}{2}\|R b_1\|_\infty$$
$$\vdots$$
$$\geq \|C^{-1} f_0\|_\infty + \ldots + \|C^{-1} f_T\|_\infty + 2\alpha \|R b_{T+1}\|_\infty + \frac{\alpha}{2}\|R b_1\|_\infty$$

$$\|C^{-1} f_{T+1}\| \leq m \cdot \mathrm{opt}(b_{T+1}) \leq m\alpha \|R b_{T+1}\|_\infty \leq m \cdot \alpha \cdot \left(\frac{3}{4}\right)^T \|R b_1\|_\infty \leq \frac{\alpha}{2}\|R b_1\|_\infty$$

$$(1+\varepsilon)\,\mathrm{opt}(b) \geq \sum_{i=0}^{T+1} \|C^{-1} f_i\|_\infty \geq \|C^{-1}(f_0 + \ldots + f_{T+1})\|_\infty$$

A Differentiable Objective Function

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

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

Properties of softmax:

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

$$\operatorname{lmax}(x) \le \ln\left(2\sum_i e^{|x_i|}\right) \le \ln\left(2 \cdot d \cdot e^{\|x\|_\infty}\right) = \|x\|_\infty + \ln(2d)$$

$$\|\nabla \operatorname{lmax}(\boldsymbol{x})\|_1 \le 1$$

$$\nabla \operatorname{lmax}(\boldsymbol{x})^\top \boldsymbol{x} \ge \operatorname{lmax}(\boldsymbol{x}) - \ln(2d)$$

$$\|\nabla \operatorname{lmax}(\boldsymbol{x}) - \nabla \operatorname{lmax}(\boldsymbol{y})\|_1 \le \|\boldsymbol{x} - \boldsymbol{y}\|_\infty$$

A Differentiable Objective Function

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

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

Replace

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

By

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

(# rows of R)

$$\gamma(f) \leq \phi(f) \leq \gamma(f) + \ln(2m) + \ln\left(2 \cdot \frac{n^2}{2}\right)$$
$$\leq \gamma(f) + 4\ln(n)$$

goal: $4\ln(n) \lesssim \varepsilon \cdot \gamma(f)$

$\rightarrow$ this is achieved by scaling problem

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 $\delta \coloneqq \|C \cdot \nabla\phi(\boldsymbol{f})\|_1$
  - 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 and output $\boldsymbol{f}$ after undoing all scalings

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

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

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

$$\cancel{\gamma(f)} \leq \quad \underline{\phi(\boldsymbol{f})} \leq (1+\varepsilon) \cdot \frac{\boldsymbol{b}^\top \boldsymbol{x}}{\|CB^\top \boldsymbol{x}\|_1} \leq (1+\varepsilon)\, opt(b)$$

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 $\delta := \|C \cdot \nabla\phi(\boldsymbol{f})\|_1$
  - 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

$\tilde{O}(m)$ time
$+$ 1 mult. with R
$+$ 1 " " R^T

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

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

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