CSE 599: Interplay between Convex Optimization and Geometry **Winter 2018**

Lecture 9: Case Study - Maximum Flow Problem

Lecturer: Yin Tat Lee

Disclaimer: *Please tell me any mistake you noticed.*

In the past few lectures, we have covered some first-order methods. They seems all rely on different parameters of the functions and hence incomparable. In both practice and theory, you should try different methods to see which is the best. Although this looks obvious, finding the best way to use first-order methods is a black magic to me. It requires lots of practice and creativity. The goal of this lecture is to use maximum flow problem as an example.

9.1 Background on Maximum Flow Problem

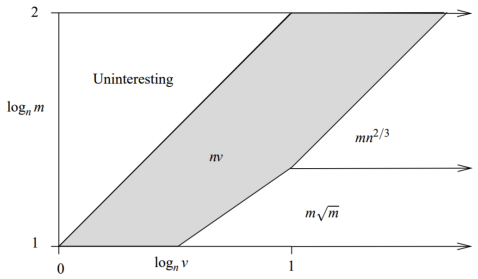

Figure 9.1: Bounds for exact undirected uncapacitated maximum flow problem. v is the maximum flow value.

The maximum flow problem has lots of applications such as airline scheduling, image segmentation, recommendation systems, etc. To make the problem as simple as possible, we consider the uncapacitated and undirected version. In this problem, we are given an unweighted undirected graph $G = (V, E)$ with m edges and n vertices. There is two special vertex s, t and our goal is to find a flow $f \in \mathbb{R}^E$ such that

$$\max_{B^\top f = c \cdot 1_{st}, \|f\|_\infty \leq 1} c$$

where the incidence matrix $B \in \mathbb{R}^{E \to V}$ is defined by $B_{(i,j),i} = 1$ and $B_{(i,j),j} = -1$ and $1_{st} \in \mathbb{R}^V$ is a vector that is -1 on s, 1 on t and 0 otherwise. The constraint $B^\top f = 1_{st}$ indicates that f is a unit flow starts from the vertex s and ends at the vertex t. The dual problem, called minimum $s - t$ cut, is given by

$$\min_{x_s = 1, x_t = 0} \sum_{i \sim j} |x_i - x_j|$$

where x is a potential on V.

This problem has been studied for many decades. Before the use of first-order methods, the fastest algorithms take $O^*(m^{3/2})$, $O^*(mn^{2/3})$ and $O^*(m + nv)$ time where v is the maximum flow value. To get a flow with value $(1 - \varepsilon)v$, the previous best was $m\sqrt{n}/\sqrt{\varepsilon}$.

Problem	Algorithm	Cost per step	# steps
$\min_x f(x)$	Mirror Descent	∇f	$(\frac{GR}{\varepsilon})^2$
	Accelerated Gradient Descent	∇f	$\sqrt{\frac{\beta R^2}{\varepsilon}}$
	Accelerated Gradient Descent	∇f	$O^*(\sqrt{\frac{\beta}{\alpha}})$
	Accelerated Coordinate Descent	$\sum_i \frac{\sqrt{\beta_i}}{\sum_j \sqrt{d_j}} \frac{\partial f}{\partial x_i}$	$O^*(\sum_i \sqrt{\frac{\beta_i}{\alpha}})$
$\min_x \sum_{i=1}^n \ell_i(x)$	Accelerated Stochastic Descent	$\sum_i \frac{\gamma_i}{\sum \gamma_i} \nabla f_i$	$O^*(n + \sqrt{n \sum_i \frac{\gamma_i}{\alpha}})$

Figure 9.1: For the f problem, we assume that $R = \|x_0 - x^*\|_2$, f is G-Lipschitz, $\alpha \cdot I \preceq \nabla^2 f \preceq \beta \cdot I$ and $\frac{\partial^2 f}{\partial x_i^2} \leq \beta_i$. For the $\sum \ell_i$ problem, we assume that $\sum \nabla^2 \ell_i \succeq \alpha$ and that $\nabla^2 \ell_i \preceq \gamma_i$.

9.2 Direct Applications of First Order Methods

Let us recap the first-order methods we have covered first.

For simplicity, we consider the minimum $s-t$ cut problem first. The constraint $x_s = 1$ and $x_t = 0$ is very simple and we can ignore them. Now, we use the algorithm in Table 9.1 one by one. To fix the notation, we assume the following:

Definition 9.2.1. We are given an undirected uncapacitated graph with m edges, n vertices. Let d_i be the degree of vertex i and Δ be the maximum degree.

Lemma 9.2.2. *Mirror descent gives a $O(mn \sum_i d_i^2)$ time algorithm for the exact minimum $s-t$ cut problem.*

Proof. Take $x_0 = 0$. Note that $R = \|x_0 - x^*\|_2 \leq \sqrt{n} \|x^*\|_\infty \leq \sqrt{n}$. The Lipschitz constant of $\sum_{i \sim j} |x_i - x_j|$ is upper bounded by $2\sqrt{\sum_i d_i^2}$ where Δ is the maximum degree. If $\varepsilon < \frac{1}{2}$, one can round off the solution by thresholding. More precisely, we let $x(t)_i = 1_{x_i \geq t}$. Note that

$$\sum_{i \sim j} |x_i - x_j| = \mathbb{E}_{t \in [0,1]} \sum_{i \sim j} |x(t)_i - x(t)_j|.$$

Therefore, we can find $t \in [0,1]$ such that $\sum_{i \sim j} |x(t)_i - x(t)_j| \leq \sum_{i \sim j} |x_i - x_j|$. Since $x(t)$ is a 0-1 vector, we have that

$$\sum_{i \sim j} |x(t)_i - x(t)_j| \leq \left\lfloor \sum_{i \sim j} |x_i - x_j| \right\rfloor.$$

If $\varepsilon < \frac{1}{2}$, the value of the right-hand side is the optimum and so is the left-hand side.

To analyze the running time, note that it takes $O(m)$ time to compute ∇f. Therefore, the total running time is

$$O(m) \times O((\sqrt{\sum_i d_i^2} \sqrt{n})^2) = O(mn \sum_i d_i^2).$$

□

Lemma 9.2.3. *Accelerated gradient descent gives a $O(m\sqrt{mn\Delta})$ time algorithm for the exact minimum $s-t$ cut problem.*

Proof. Since the absolute function is not differentiable, we consider the problem

$$f_\delta(x) = \sum_{i\sim j} \sqrt{|x_i - x_j|^2 + \delta^2}.$$

Suppose that we can find a x such that $f_\delta(x) < \min_x f_\delta(x) + \frac{1}{4}$. Then, we have that

$$f(x) \leq f_\delta(x) < \min_x f_\delta(x) + \frac{1}{4} \leq \min_x f(x) + m\delta + \frac{1}{4}.$$

Hence, setting $\delta = \frac{1}{4m}$, we have the total error less than $\frac{1}{2}$. Again, we can then threshold the solution.

To analyze the runtime, note that

$$\frac{d^2}{dx^2}\sqrt{x^2+\delta^2} = \frac{\delta^2}{(x^2+\delta^2)^{3/2}} \leq \frac{1}{\delta}.$$

Using this, we have that

$$\nabla^2 f_\delta(x) \preceq \frac{1}{\delta} B^\top B \preceq \frac{2\Delta}{\delta}\cdot I.$$

Hence, the total running time is

$$O(m) \times O(\sqrt{\frac{\Delta}{\delta}\cdot\sqrt{n}^2}) = O(m\sqrt{mn}\sqrt{\Delta}).$$

□

Exercise 9.2.4. Convince yourself that the accelerated gradient descent with guarantee $O^*(\sqrt{\frac{\beta}{\alpha}})$ gives the same result as before up to polylog.

Lemma 9.2.5. *Accelerated coordinate descent gives a $O^*(\sqrt{mn}\sum_i d_i^{3/2})$ time algorithm for the exact minimum $s-t$ cut problem.*

Remark. This is better than accelerated gradient descent because $\sqrt{mn}\sum_i d_i^{3/2} \leq m\sqrt{mn\Delta}$.

Proof. Since the accelerated coordinate descent we wrote in the table requires strong convexity, we consider the problem

$$f_\delta(x) = \sum_{i\sim j} \sqrt{|x_i - x_j|^2 + \delta^2} + \frac{\alpha}{2}\|x\|^2.$$

with $\delta = \frac{1}{8m}$ and $\alpha = \frac{1}{8n}$. It is easy to see that this formulation only gives constant error.

Now, we discuss the runtime for minimizing f_δ. Note that $\beta_i = \frac{2d_i}{\delta}$ where d_i is the degree of vertex i. Note that we will sample

Hence, the total running time is

$$\begin{aligned} O(\sum \frac{\sqrt{d_i}}{\sum_j \sqrt{d_j}} d_i) \times O^*(\sum_i \sqrt{\frac{d_i}{\delta\alpha}}) &= O^*(\sum_i \frac{d_i^{3/2}}{\sum_j \sqrt{d_j}} \sum_i \sqrt{mnd_i}) \\ &= O^*(\sqrt{mn}\sum_i d_i^{3/2}). \end{aligned}$$

□

Lemma 9.2.6. *Accelerated stochastic descent gives a $O^*(m\sqrt{mn})$ time algorithm for the exact minimum $s-t$ cut problem.*

Remark. This is better than accelerated coordinate descent because $\sqrt{mn}\sum_i d_i^{3/2} \geq \sqrt{mn}\sum_i d_i = m\sqrt{mn}$.

Proof. Similar to accelerated coordinate descent, we consider the problem

$$f_\delta(x) = \sum_{i\sim j} \sqrt{|x_i - x_j|^2 + \delta^2} + \frac{\alpha}{2} \|x\|^2 .$$

with $\delta = \frac{1}{8m}$ and $\alpha = \frac{1}{8n}$.

Note that $\gamma_i = \frac{2}{\delta}$. Hence, the total running time is

$$\begin{aligned} O(1) \times O^*(m + \sqrt{m \sum_{i\sim j} \frac{1}{\delta\alpha}}) &= O^*(\sqrt{m \cdot m \cdot m \cdot n}) \\ &= O^*(m\sqrt{mn}). \end{aligned}$$

□

Consider the common case the graph is sparse and $\Delta = O(1)$, then all accelerated algorithms has the same running time $O^*(n^2)$! To see why this is the case, we can simply consider the line graph with the starting point $x_0 = 1$. It takes exactly n steps for the information moves from s to t. In general, mn is also the lower bound for this type of first-order methods.

9.3 Precondition by Laplacian

One may think this is a sort of an impossible result, namely, any first-order methods need to take $\Omega(mn)$ time. However, this is false. In general, first-order methods depends on how you parameterize the space, or equivalently, how you measure the distance. One key observation is that we do not need to use the standard Euclidean space. Recall that the gradient step is of the form

$$x^{(k+1)} = \arg\min_x f(x^{(k)}) + \nabla f(x^{(k)})^\top (x - x^{(k)}) + \left\|x - x^{(k)}\right\|_2^2 .$$

In general, we can replace $\left\|x - x^{(k)}\right\|_2^2$ by any other inner product $(x - x^{(k)})^\top H (x - x^{(k)})$. For simplicity, we write

$$\|v\|_H = \sqrt{v^\top H v}.$$

In this case, we simply need to redefine everything parameter using this inner product. Namely,

$$R = \left\|x^{(0)} - x^*\right\|_H , \quad \alpha H \preceq \nabla^2 f \preceq \beta H \quad \text{and} \quad f(x) - f(y) \le G \|x - y\|_H .$$

When we use other inner product in first-order methods, we call the matrix H we use is the preconditioner. When we apply first-order methods, we need to aware that we can choose the matrix H. Ideally, we want to choose a matrix H capture how the problem looks like. However, we also need to note that each iteration of first-order method need to solve a problem of the form

$$\min \nabla f(x^{(k)})^\top (x^{(k)} - x) + \frac{\beta}{2} \left\|x^{(k)} - x\right\|_H^2 .$$

Therefore, we need to choose H that is simple enough. For the maximum flow problem, we note that $\sum_{i\sim j} |x_i - x_j|$ looks like $\sum_{i\sim j} |x_i - x_j|^2 = x^\top B^\top B x$. Hence, it is natural to use $H = B^\top B$.

When we pick other inner product, it may not make sense to use coordinate descent or stochastic descent because one step of these algorithms may not be cheap anymore. Since accelerated gradient descent is better than mirror descent, we analyze the accelerated gradient descent only.

Lemma 9.3.1. *Accelerated gradient descent with preconditioner $B^\top B$ gives a $O^*(m\sqrt{mv})$ time algorithm for the exact minimum $s - t$ cut problem.*

Proof. Similar to before, we use the function

$$f_\delta(x) = \sum_{i\sim j} \sqrt{|x_i - x_j|^2 + \delta^2}$$

with $\delta = \frac{1}{4m}$. Note that

$$\nabla^2 f(x) \preceq \frac{1}{\delta} B^\top B = \frac{1}{\delta} H.$$

and

$$R^2 = \left\| x^{(0)} - x^* \right\|_H^2 = \sum_{i\sim j} |x_i - x_j|^2 \leq \sum_{i\sim j} |x_i - x_j| = v.$$

Hence, the total running time is

$$O^*(m) \times O(\sqrt{\frac{1}{\delta} \cdot v}) = O(m\sqrt{mv})$$

where we used that we can solve $\min_x c^\top x + \|x\|_H^2$ in $O^*(m)$ time. □

Going back to the line graph example, we note that $v = O(1)$ and hence this algorithm takes $m^{3/2}$ time, much faster than the naive way to use first order method.

9.4 Duality Trick

In the previous sections, we showed how to find the minimum $s-t$ cut. Now, we show that one can read off the maximum flow solution from the regularized minimum $s-t$ cut solution. Suppose x is the minimizer of the problem

$$\ell(x) \stackrel{\text{def}}{=} \sum_{i\sim j} \sqrt{|x_i - x_j|^2 + \delta^2} + \frac{\alpha}{2} \sum_{i\sim j} |x_i - x_j|^2.$$

Lemma 9.4.1. *Let x is the minimizer of the problem*

$$\ell(x) \stackrel{\text{def}}{=} \sum_{i\sim j} \sqrt{|x_i - x_j|^2 + \delta^2} + \frac{\alpha}{2} \sum_{i\sim j} |x_i - x_j|^2.$$

Let $\phi(y) = \sqrt{y^2 + \delta^2} + \frac{\alpha}{2} y^2$ and $f \in \mathbb{R}^E$ defined by $f_e = \phi'((Bx)_e)$. Then, $B^\top f = 0$, $\|f\|_\infty \leq 1 + \alpha$ and the flow value of f is at least $v - \delta m$.

Proof. We can write $\ell(x) = \sum_{e\in E} \phi((Bx)_e)$. By the optimality of x, the gradient is given by

$$0 = \nabla \ell(x) = B^\top \phi'(Bx) = B^\top f$$

Next, we note that

$$\phi'(y) = \frac{y}{\sqrt{y^2 + \delta^2}} + \alpha y.$$

Since $0 \leq x \leq 1$, we have that $\|f\|_\infty \leq 1 + \alpha$.

Finally, the flow value can be found by considering the amount of flow leaving a random level set $C_t = \{x \geq t\}$:

$$\begin{aligned}
&\text{Flow value of } f \\
=&\int_0^1 \sum_{i\in C_t, j\in C_t^c, i\sim j} f_{(i,j)} dt \\
=&\sum_{i\sim j} f_{(i,j)}(x_i - x_j) \\
=&\sum_{i\sim j} \frac{(x_i-x_j)^2}{\sqrt{(x_i-x_j)^2+\delta^2}} + \frac{\alpha}{2}\sum_{i\sim j}(x_i-x_j)^2 \\
\geq&\sum_{i\sim j}\sqrt{(x_i-x_j)^2+\delta^2} - \sum_{i\sim j}\frac{\delta^2}{\sqrt{(x_i-x_j)^2+\delta^2}} \\
\geq&\sum_{i\sim j}\sqrt{(x_i-x_j)^2+\delta^2} - \delta m \\
\geq& v - \delta m
\end{aligned}$$

where we used that

$$\sum_{i\sim j}\sqrt{(x_i-x_j)^2+\delta^2} = \min_x \sum_{i\sim j}\sqrt{(x_i-x_j)^2+\delta^2} \geq \min_x \sum_{i\sim j}|x_i-x_j| = v.$$

□

If we rescale down f by $1+\alpha$, we get a flow satisfies the constraints with flow value at least

$$\frac{v-\delta m}{1+\alpha} \geq v - \delta m - \alpha v.$$

Since accelerated gradient descent with precondition $B^\top B$ takes $O^*(\frac{m}{\sqrt{\alpha\delta}})$ time. Setting $\delta = \frac{\varepsilon v}{2m}$ and $\alpha = \frac{\varepsilon}{2}$, we get the following:

Lemma 9.4.2 ([52])**.** *Accelerated gradient descent with preconditioner H gives a $O^*(\frac{m}{\varepsilon}\sqrt{\frac{m}{v}})$ time algorithm to find a maximum flow with value at least $(1-\varepsilon)v$.*

Remark. To compare with the result for minimum $s-t$ cut, we put $\varepsilon = \frac{1}{v}$ and recovers $O^*(m\sqrt{mv})$.

Although accelerated gradient descent gives a fractional flow, there is algorithms to convert the flow to integral [50]. After that, we can fix the flow by augmenting path, which takes εmv. Therefore, the total running time is

$$\frac{m}{\varepsilon}\sqrt{\frac{m}{v}} + \varepsilon mv.$$

Choosing the best ε, we have the following:

Theorem 9.4.3. *Combining accelerated gradient descent with preconditioner H and augmenting path gives a $O^*(m^{5/4}v^{1/4})$ time algorithm to find exact maximum flow.*

Remark. For certain regime of v, this is better than the previous best algorithms $O^*(m^{3/2})$, $O^*(mn^{2/3})$ and $O^*(m+nv)$.

9.5 Precondition by Routing

The source of "$\sqrt{m}$" in the previous runtime is not from the diameter. The Laplacian preconditioner allows each iteration of the first-order method communicates information from one vertex to any another vertex.

Essentially, our previous algorithm is using ℓ_2 to approximate ℓ_1 and the discrepancy between that two norm is reflected in the runtime.

To avoid this discrepancy, we should not use ℓ_2 norm. One issue of other ℓ_p norm is that the runtime of first-order methods generally depends polynomially to ε. Therefore, it is more difficult to use the duality trick. Hence, we look at the maximum flow problem directly.

We can rewrite the maximum problem as

$$\min_{B^\top f = 1_{st}} \|f\|_\infty .$$

The optimum value for this problem is $\frac{1}{v}$. There are different ways [53] to find $f^{(0)}$, in $O^*(m)$ time, such that $B^\top f^{(0)} = 1_{st}$ and $\left\|f^{(0)}\right\|_\infty = O^*(\frac{1}{v})$. We can further simplify it to

$$\min_{B^\top f = 0} \left\|f + f^{(0)}\right\|_\infty .$$

In the last lecture, we represent any f with $B^\top f = 0$ by sum of cycles on tree. More general, let say we have a projection C such that the range of C is $\{f : \ B^\top f = 0\}$. Then, we can write the problem as

$$\min_f \left\|Cf + f^{(0)}\right\|_\infty .$$

To minimize this function, we use the following variant of gradient descent.

Lemma 9.5.1. *Given a norm $\|\cdot\|$ and a convex function f that is β smooth in $\|\cdot\|$, namely,*

$$f(y) \le f(x) + \nabla f(x)^\top (y - x) + \frac{\beta}{2} \|y - x\|^2 .$$

Consider the algorithm $x^{(k+1)} \leftarrow \mathrm{argmin}_y \left\{\nabla f(x^{(k)})^\top (y - x^{(k)}) + \frac{\beta}{2} \left\|y - x^{(k)}\right\|^2\right\}$, we have that

$$f(x_T) - f(x^*) \le \frac{2\beta R^2}{T} \quad \textit{with} \quad R = \max_{f(x) \le f(x_0)} \|x - x^*\|$$

for any x^*.

Using Lemma 9.5.1 and smoothen $\|\cdot\|_\infty$ appropriately, we have the following theorem.

Theorem 9.5.2 ([55, 51])**.** *Given a projection matrix C on $\{f : \ B^\top f = 0\}$ and a vector $f^{(0)}$ such that $B^\top f^{(0)} = 1_{st}$ and $\left\|f^{(0)}\right\|_\infty = O^*(\frac{1}{v})$. Suppose it takes time $\mathcal{T}$ to compute matrix vector product Cv and matrix transpose product $C^\top v$. Then, we can find a f such that $B^\top f = 1_{st}$ and $\|f\|_\infty \le \frac{1+\varepsilon}{v}$ in time*

$$O^*\left(\frac{\|C\|^2_{\infty\to\infty}}{\varepsilon^2}(m + \mathcal{T})\right).$$

Remark. [56] designed a specified first-order method for $\|\cdot\|_\infty$ and obtained a running time $O^*\left(\frac{\|C\|_{\infty\to\infty}}{\varepsilon}(m + \mathcal{T})\right)$.

Proof. To smoothen $\|\cdot\|_\infty$, we consider the potential

$$\ell_\delta(f) = \mathrm{smax}_\delta(Cf + f^{(0)})$$

where

$$\mathrm{smax}_\delta(f) = \delta \log(\sum_i e^{f_i/\delta} + e^{-f_i/\delta}).$$

Suppose that $\ell_\delta(f) \leq \min_f \ell_\delta(f) + \frac{\varepsilon}{2v}$. Then, we have that

$$\left\|Cf + f^{(0)}\right\|_\infty \leq \ell_\delta(t) \leq \min_f \ell_\delta(f) + \frac{\varepsilon}{2v} \leq \min_f \ell(f) + \frac{\varepsilon}{2v} + \delta \log(2m).$$

Setting $\delta = \frac{\varepsilon}{2v \log(2m)}$, we have that $\left\|Cf + f^{(0)}\right\|_\infty \leq \min_f \ell(f) + \frac{\varepsilon}{v}$. Since $B^\top(Cf + f^{(0)}) = B^\top f^{(0)} = 1_{st}$, $Cf + f^{(0)}$ satisfies the requirements.

Now, we analyze how fast is the gradient descent in Theorem 9.5.1. It is easy to prove that ℓ_δ is

$$O(\frac{\|C\|^2_{\infty\to\infty}}{\delta}) = O^*(\frac{v\|C\|^2_{\infty\to\infty}}{\varepsilon}) \text{ smooth.}$$

Let f^* be the solution of $\min_{B^\top f = 1_{st}} \|f\|_\infty$. Then, we have

$$\|f^*\|_\infty = \|Cf^*\|_\infty \leq \left\|Cf^* + f^{(0)}\right\|_\infty + \left\|f^{(0)}\right\|_\infty = O^*(\frac{1}{v}).$$

Therefore, $R = O^*(\frac{1}{v})$ and $\beta = O^*(\frac{v\|C\|^2_{\infty\to\infty}}{\varepsilon})$. To get a solution with additive error $\frac{\varepsilon}{2v}$, the total runtime is

$$O(m + \mathcal{T}) \times O^*\left(\frac{\frac{v\|C\|^2_{\infty\to\infty}}{\varepsilon} \times \frac{1}{v^2}}{\frac{\varepsilon}{2v}}\right) = O^*\left(\frac{\|C\|^2_{\infty\to\infty}}{\varepsilon^2}(m + \mathcal{T})\right).$$

□

There is a C with $\|C\|_{\infty\to\infty} = O^*(1)$ [54] and hence it gives the following theorem.

Theorem 9.5.3. *We can find a flow with value at least* $(1-\varepsilon)v$ *in time*

$$O^*(\frac{m}{\varepsilon^2}).$$

References

[50] Donggu Kang and James Payor. Flow rounding. *arXiv preprint arXiv:1507.08139*, 2015.

[51] Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 217–226. SIAM, 2014.

[52] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. A new approach to computing maximum flows using electrical flows. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 755–764. ACM, 2013.

[53] Aleksander Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 245–254. IEEE, 2010.

[54] Richard Peng. Approximate undirected maximum flows in o (m polylog (n)) time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1862–1867. Society for Industrial and Applied Mathematics, 2016.

[55] Jonah Sherman. Nearly maximum flows in nearly linear time. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 263–269. IEEE, 2013.

[56] Jonah Sherman. Area-convexity, l &infty; regularization, and undirected multicommodity flow. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 452–460. ACM, 2017.