CS 798 - Algorithmic Spectral Graph Theory, Fall 2015, Waterloo

Lecture 15 : Maximum flow

We see how to compute approximate maximum flow in undirected graphs faster by computing electric flow and using multiplicative update, and we also discuss how to quickly round fractional flow into integral.

Maximum Flow in Undirected Graphs

In this problem, we are given an undirected graph where each edge e has a capacity c_e, a source vertex s, and a sink vertex t, and the objective is to find a maximum flow subjected to the capacity constraints, where a flow has to satisfy the flow conservation constraints.

More formally, we have two variables f_{uv} and f_{vu} for each edge uv, and the problem is to:

$$\max \sum_{e \in \delta^{out}(s)} f_e \qquad (\text{where } \delta^{out}(s) \text{ is the set of directed edges going out from } s)$$

$$\sum_{e \in \delta^{out}(v)} f_e = \sum_{e \in \delta^{in}(v)} f_e \qquad \text{for all } v \in V - \{s,t\} \quad (\text{where } \delta^{in}(v) \text{ is the set of incoming edges to } v)$$

$$f_e \le c_e \qquad \text{for all } e$$

$$f_e \ge 0 \qquad \text{for all } e$$

For simplicity, we assume $c_e = 1$ for all edge. So the optimal value of the linear program is in $[0, m]$ where m is the number of edges. As mentioned before, we can reduce the optimization problem to $O(\log m)$ decision problems, by doing binary search on the objective value and replace the objective function by the constraint $\sum_{e \in \delta^{out}(s)} f_e = k$.

Multiplicative update method

We think of the objective constraint ($\sum_{e \in \delta^{out}(s)} f_e = k$), the non-negative constraints ($f_e \ge 0 \ \forall e \in E$), and the flow conservation constraint ($\sum_{e \in \delta^{in}(v)} f_e = \sum_{e \in \delta^{out}(v)} f_e \quad \forall v \in V - \{s,t\}$) as "easy" constraints, and we will make sure to implement an oracle that always returns a solution satisfying them.

The capacity constraints ($f_e \le 1 \ \forall e \in E$) are the "hard" constraints, and we will use the multiplicative weight update method to deal with them. The combined constraint is of the form $\sum_e w_e f_e \le \sum w_e$ where $0 \le w_e \le 1$ and $\sum_e w_e = 1$.

A standard way to implement the oracle is to find a shortest path and send k units of flow

on this path. So, using the multiplicative update method, computing flow is reduced to computing shortest paths which can be implemented in linear time. However, the width of this oracle is k, and it requires $O(\frac{k \ln n}{\varepsilon^2})$ iterations for the average solution to converge to a feasible solution, and this is too slow for large k.

The new idea is to use the electrical flow algorithm to implement the oracle. It is almost as fast as the shortest path algorithm, as it turns out we can bound the width parameter.

By the analysis of the multiplicative weight update method, we can find an almost feasible solution ($f_e \le 1+\varepsilon \quad \forall e \in E$) if we could solve $O(\frac{\rho \ln n}{\varepsilon^2})$ subproblems of the following form:

(0) f satisfies the objective constraint, the non-negative constraints, and the flow conservation constraints.

(1) $\sum_e w_e f_e \le (1+\varepsilon) \sum_e w_e$. In the standard version, this is $\sum_e w_e f_e \le \sum_e w_e$, but it is easy to check that it is okay to respond with an approximate feasible solution, as we do approximation anyway.

(2) $f_e \le \rho$, where ρ is the width parameter.

If these subproblems can be solved (for different w_e), then the average solution would be an almost feasible solution, and by scaling it would satisfy all capacity with objective value $k(1-\varepsilon)$.

Remark: In the multiplicative update method in L14, the convergence rate is $O(\frac{\rho^2 \ln n}{\varepsilon^2})$.
That analysis is for the general case where the outcome is in $[-1,+1]$.
It is not difficult to show that if the outcome is in $[0,1]$, then the convergence rate is $O(\frac{\rho \ln n}{\varepsilon^2})$.

Electrical flow

Here we show that the electric flow algorithm can be used to solve the subproblem with the following properties:

(0) It returns f that satisfies flow conservation constraints, non-negative constraints, and objective value constraint.

(1) When the maximum flow problem is feasible, the oracle can always return f satisfying $\sum_e w_e f_e \le (1+\varepsilon) \sum_e w_e$.

(2) When the maximum flow problem is feasible, the oracle can always return f with $f_e \le O(\sqrt{m})$.

(3) The oracle can be implemented in $\tilde{O}(m)$ time.

Assuming this can be done, we can find an $(1-\varepsilon)$-approximate solution in $O(\frac{\sqrt{m} \ln n}{\varepsilon^2})$ iterations, and the total running time is $\tilde{O}(m^{3/2}/\varepsilon^2)$.

It remains to construct the oracle with the above guarantees. As we said earlier we use electric flow.

Oracle The oracle can be obtained by setting $r_e = w_e + \frac{\varepsilon W}{m}$ where $W = \sum_e w_e$, and compute the electric flow that sends k units of electric flow from s to t with r_e as the resistance of edge e.

This is the oracle. So the whole algorithm is very simple. In each iteration, w_e is updated by $w_e^{t+1} = w_e^t (1 + \frac{\varepsilon}{\rho} f_e^t)$ and update r_e accordingly, then compute the electric flow. So, if the flow on an edge is over its capacity, we will increase its resistance based on its violation to decrease the future flow on this edge. Taking the average over all the electric flow is a good approximation.

Analysis

We check the four properties one by one.

(0) It is clear by construction that the flow conservation constraints, the non-negativity constraints, and the objective value constraint are all satisfied.

(1) If the maximum flow problem is feasible, then there is a flow of k units from s to t, without violating the capacity constraints (so $f_e \le 1 \ \forall e \in E$).

The total energy of this flow is at most $\sum_e \tilde{f}_e^2 r_e \le \sum_e r_e = \sum_e (w_e + \frac{\varepsilon W}{m}) = (1+\varepsilon)W$.

Since electric flow f minimizes the energy, we must have $\sum_e f_e^2 r_e \le (1+\varepsilon)W$, or otherwise we can conclude that the maximum flow problem is infeasible

We would like to bound $\sum_e w_e f_e$, and we use Cauchy-Schwarz.

$\left(\sum_e w_e f_e\right)^2 \le \left(\sum_e w_e f_e^2\right)\left(\sum_e w_e\right) \le \left(\sum_e r_e f_e^2\right)\left(\sum_e w_e\right) \le (1+\varepsilon)W^2$, and so we have $\sum_e w_e f_e \le (1+\varepsilon)\sum_e w_e$, as required.

(2) Clearly the energy on one edge cannot be more than the total energy,

so $\frac{f_e^2 \varepsilon W}{m} \le f_e^2 r_e \le (1+\varepsilon)W \Rightarrow f_e \le \sqrt{\frac{(1+\varepsilon)}{\varepsilon} m} = O(\sqrt{m})$, as required.

(3) As we saw in L12, computing electric flow is the same as solving Laplacian systems, which we saw in L13 that can be done in $\tilde{O}(m)$ time.

This completes the $\tilde{O}(m^{1.5})$ time algorithm to give an $(1-\varepsilon)$-approximation of the maximum flow problem in undirected graphs, matching the best known combinatorial method.

Next we present an improvement to $\tilde{O}(m^{4/3})$ time.

Faster Algorithm

One bottleneck of the algorithm is that it takes $O(\sqrt{m})$ iterations, because the flow of an edge may be as high as $\Omega(\sqrt{m})$. Consider the example where there are k disjoint paths of length k between s and t, and there is one edge between s and t. Then there will be $(k+1)/2$ units of flow on the edge between s and t. As there are k^2+1 edges in the graph, the flow on that edge is $\Theta(\sqrt{m})$.

The idea of the improved algorithm is to delete the edges with too much flow going across them. This actually sounds counterintuitive, as those edges seem to be in optimal solutions, but it turns out that there will not be many of them, and so deleting them will not decrease the optimal value by much, while improving the convergence rate of the multiplicative update algorithm.

Modified Algorithm

The algorithm is the same except that whenever there is an edge with electric flow value greater than p, then we delete this edge from the graph.

Effective Conductance

To analyze the effect of deleting an edge, we need the concept of effective conductance.

We know by the Thompon's principle that the electric flow is the flow that minimizes energy, among all the flow that satisfies flow conservation constraints.

$\mathcal{E}_r(f) = \sum_e f_e^2 r_e = \sum_{uv \in E} \frac{(\phi_u - \phi_v)^2}{r_{uv}}$, where ϕ_v denotes the voltage on v.

If the flow f is an electric flow of 1 unit from s to t, then $R_{eff}(s,t) = \mathcal{E}_r(f)$.

Recall that $R_{eff}(s,t) = \phi_s - \phi_t$

By setting $\phi_t = 0$ and scaling $\phi_s = 1$, we have a flow of $1/R_{eff}(s,t)$ unit from s to t, and the energy of this flow is equal to $\mathcal{E}_r(f)/R_{eff}^2(s,t) = 1/R_{eff}(s,t)$, as the electric flow of c units from s to t has energy $\mathcal{E}_r(f)/c^2$ by scaling the flow on each edge by a factor of c.

Among all vectors on vertices, the voltage vector is the one that minimizes energy.

This is a "dual" version of Thompson's principle.

Theorem Let r be the vector of resistances. Let $C_{eff}(r) = \min_{\phi : \phi_s = 1, \phi_t = 0} \sum_{uv \in E} \frac{(\phi_u - \phi_v)^2}{r_{uv}}$.

Then $C_{eff}(r)$ is minimized by the voltage vector of the electric flow of $1/R_{eff}(s,t)$ units from s to t

proof We have seen that the value $1/R_{eff}(s,t)$ is attainable by that voltage vector.

So we just need to prove that any minimizer must be the voltage vector of an electric flow.

Consider the partial derivative on a variable ϕ_v, $\frac{\partial E(\phi)}{\partial \phi_v} = \sum_{u:uv\in E} \frac{2(\phi_u-\phi_v)}{r_{uv}} \quad \forall\, v\in V-\{s,t\}$.

A minimizer must have $\frac{\partial E(\phi)}{\partial \phi_v}=0$, and this defines a potential flow that satisfies conservation constraints. □

We use this theorem to calculate the change of effective resistance after deleting an edge.

Lemma Suppose f is an electric flow and e is an edge such that $f_e^2 r_e \geq \beta \mathcal{E}_r(f)$.

The effective resistance R' after removing an edge is at least $\frac{R}{1-\beta}$ where R is the original effective resistance.

proof Without loss of generality we assume that f is a flow from s to t of $1/R$ unit.

Using the previous theorem, $\frac{1}{R} = \min_{\phi:\,\phi_s=1,\,\phi_t=0} \frac{(\phi_u-\phi_v)^2}{r_{uv}} = \mathcal{E}_r(f)$

Let $e=xy$. Then $\frac{(\phi_x-\phi_y)^2}{r_{xy}} = f_{xy}^2 r_{xy} \geq \beta \mathcal{E}_r(f)$ by assumption.

By deleting the edge e, it is the same as increasing r_e to infinity.

So $\frac{1}{R'} = C_{eff}(r') \leq \sum_{\substack{uv\in E\\ uv\neq xy}} \frac{(\phi_u-\phi_v)^2}{r_{uv}} = \mathcal{E}_r(f) - \frac{(\phi_x-\phi_y)^2}{r_{xy}} \leq (1-\beta)\mathcal{E}_r(f) = \frac{1-\beta}{R}$.

Therefore, $R' \geq \frac{R}{1-\varepsilon}$. □

Theorem For $p=\tilde{O}(m^{1/3})$, the number of edges deleted is $\tilde{O}(m^{1/3})$.

(optional) This implies that the total complexity is $\tilde{O}(m^{4/3})$.

Let $R(i)$ be the effective resistance at step i.

Initial effective resistance

Let k^* be an optimal maximum flow between s and t.

This implies that there is an s-t cut with k^* edges.

So one of the edges in the cut would have flow value at least $1/k^*$, when we send one unit of flow from s and t.

Thus, $R(0) \geq \frac{1}{(k^*)^2}$.

Lower Bound on R(T)

When we delete an edge, it has a flow value at least p, and thus the energy on that edge is at least $p^2 r_e = p^2(w_e + \varepsilon w) \geq p^2 \varepsilon w \geq p^2\varepsilon$ (total energy), as total energy $\leq (1+\varepsilon)W$.

When we delete an edge, it has a flow value at least p, and thus the energy on that edge is at least $p^2 r_e = p^2\left(w_e + \frac{\varepsilon W}{m}\right) \geq \frac{p^2 \varepsilon W}{m} \geq \frac{p^2 \varepsilon}{(1+\varepsilon)\cdot m}$ (total energy), as total energy $\leq (1+\varepsilon)W$.

By setting $\beta = \frac{p^2\varepsilon}{(1+\varepsilon)m}$ and apply the lemma, the effective resistance increases by a factor of $\left(1 - \frac{p^2\varepsilon}{(1+\varepsilon)m}\right)^{-1}$ after we delete an edge.

So, if we have deleted h edges throughout the algorithm, then $R(T) \geq R(0)\cdot\left(1 - \frac{p^2\varepsilon}{(1+\varepsilon)m}\right)^{-h} \geq \frac{1}{(k^*)^2}\left(1 - \frac{p^2\varepsilon}{(1+\varepsilon)m}\right)^{-h}$

Upper Bound on R(T)

The energy of the flow of k units is at most $(1+\varepsilon)\cdot W(T)$.

(This is not precise because some edges have been removed. But we can ensure that the number of edges removed is at most a small fraction of k^*. Thus the energy of the flow is only slightly increased, by scaling up the flow on the remaining edges slightly. So the total energy would not be much higher than that.)

Thus, $R(T) \leq \frac{(1+\varepsilon)\cdot W(T)}{k^2}$, by scaling down the flow value by a factor of k.

Note that $W(t+1) = \sum_e w_e(t)\left(1 + \frac{\varepsilon}{p} f_e^t\right)$ by the multiplicative update rule

$$= \sum_e w_e(t) + \frac{\varepsilon}{p}\sum_e w_e(t) f_e^t \leq W(t)\left(1 + \frac{\varepsilon(1+\varepsilon)}{p}\right).$$

So, $W(T) \leq m\cdot\left(1 + \frac{\varepsilon(1+\varepsilon)}{p}\right)^T \leq m\cdot e^{\frac{\varepsilon(1+\varepsilon)T}{p}} \leq m\cdot e^{\frac{2\ln n}{\varepsilon}}$ as $T = \frac{p\ln n}{\varepsilon}$.

Therefore, $R(T) \leq \frac{(1+\varepsilon)W(T)}{k^2} \leq \frac{(1+\varepsilon)\,m\cdot e^{\frac{2\ln n}{\varepsilon}}}{k^2}$.

Putting together

Combining the upper and lower bound on $R(T)$, we have $\frac{1}{(k^*)^2}\left(1 - \frac{\varepsilon p^2}{m(1+\varepsilon)}\right)^{-h} \leq \frac{(1+\varepsilon)\,m\cdot e^{\frac{2\ln n}{\varepsilon}}}{k^2}$.

Using $k^*/k \leq m$, this implies $\ln\left(\frac{1}{m^3}\right) \leq \ln(1+\varepsilon) + \frac{2\ln n}{\varepsilon} + h\ln\left(1 - \frac{\varepsilon p^2}{m(1+\varepsilon)}\right)$

$$\Leftrightarrow \quad h \leq -\frac{\ln(1+\varepsilon) + \frac{2\ln n}{\varepsilon} + 3\ln m}{\ln\left(1 - \frac{\varepsilon p^2}{m(1+\varepsilon)}\right)}$$

$$\Rightarrow \quad h \leq \frac{\frac{6\ln n}{\varepsilon}}{\frac{\varepsilon p^2}{m(1+\varepsilon)}} \qquad \text{using } \ln(1-c) < -c \text{ for } c \in (0,1).$$

$$\Rightarrow \quad h \leq O\left(\frac{m\ln n}{p^2\varepsilon^2}\right)$$

Setting $p = \tilde{O}(m^{1/3})$, we have $h = \tilde{O}(m^{1/3})$, as required.

Discussions

The material is from the paper "Electrical flows, Laplacian systems, and faster approximation of maximum

flow in undirected graphs", by Christiano, Kelner, Madry, Spielman, Teng.

An open question is whether f_t converges to an optimal solution, instead of the average.

Since this paper, many faster algorithms for combinatorial problems are designed using Laplacian solvers.

For the maximum flow problem in undirected graphs, it is now known to get an $(1-\varepsilon)$-approximation in $\tilde{O}(m)$ time.

For the maximum flow problem in directed graphs, the Laplacian solver is combined with the interior point method to get an exact $O(m^{10/7})$ algorithm, and also the same approach gives an $\tilde{O}(m\sqrt{n})$ algorithm for minimum cost flow, both improving long-standing previous best bounds.

It is an exciting time when ideas from continuous optimization and combinatorial optimization interact to get faster algorithms in both areas.

Perfect Matching in Regular Bipartite Graphs

The bipartite matching problem is to find a maximum number of vertex disjoint edges in a bipartite graph. By Hall's theorem on bipartite matching, it is well-known that a regular bipartite graph always has a perfect matching (i.e. a regular bipartite graph with $2n$ vertices has a matching of size n).

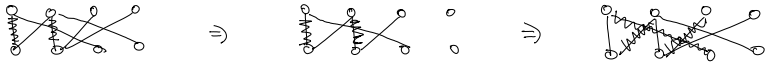
There is a recent result showing how to find a perfect matching in such graphs in $O(n \log n)$ time.

Note that this is sublinear time when the graph has much more than $n \log n$ edges (e.g. $\Omega(n^2)$ edges).

The algorithm and the analysis are very elegant, and the result can be extended to rounding fractional flow solution (i.e. $0 \le f_e \le 1$) into integral flow solution (i.e. $f_e \in \{0, 1\}$).

The traditional approach is to repeatedly find an augmenting path to enlarge the matching.

Pictorically,



current matching ⇒ augmenting path ⇒ larger matching

An augmenting path is a path $v_1 - v_2 - \cdots - v_{2\ell+1}$, where $v_{2i-1} - v_{2i}$ is an edge not in the current matching, $v_{2i} - v_{2i+1}$ is an edge in the current matching, and v_1 and $v_{2\ell+1}$ are unmatched vertices.

Theorem The current matching is maximum if and only if there is no augmenting path.

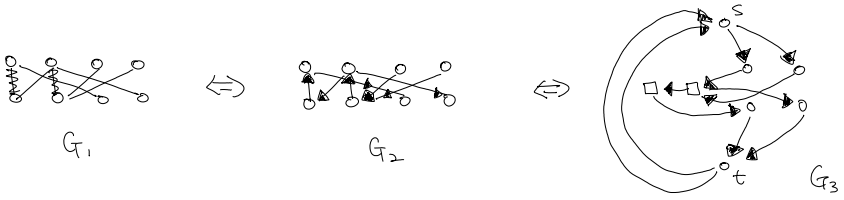
proof idea One direction is easy. If there is an augmenting path, then we can use it to enlarge the matching.

Another direction is to show that if the current matching M is not maximum, then there is

an augmenting path, by considering the union of a larger matching M^* and M. □

So, the maximum bipartite matching problem can be reduced to at most n subroutines of finding an augmenting path, which can be done in $O(m)$ time by a BFS (breadth first search).

The new idea is to replace BFS by random walk. Let me explain the algorithm by pictures.



G_1 is the original undirected bipartite graph with a matching M.

G_2 is the directed graph where each edge in the matching M is pointing upward, while every other edge (not in the matching) is pointing downward.

G_3 is the directed graph obtained from G_2, by contracting each edge in the matching M into a single (square) node. The source s has $d^{out}(v)$ edges to every unmatched vertex v on top, and every unmatched vertex u in bottom has $d^{in}(u)$ edges to the sink t. And t has $d^{in}(t)$ edges to s.

It is not difficult to show that ① G_1 has an augmenting path

$\Leftrightarrow$ ② G_2 has a directed path from a top unmatched vertex to a bottom unmatched vertex

$\Leftrightarrow$ ③ G_3 has a cycle from s to s.

Also, it is not difficult to verify that G_3 is an Eulerian directed graph (i.e. the indegree is equal to the outdegree for every vertex) if G_1 is a regular bipartite graph.

So, if we do a random walk in G_3, then the expected time to find an augmenting path is equal to the expected hitting time $H_{s,s}$ in G_3.

Recall that $H_{s,s} = 1/\pi_s$, where π_s is the probability of being in s in the stationary distribution, which is easy to compute in Eulerian directed graphs.

Claim In an Eulerian directed graph, the stationary distribution is $\pi_v = \frac{d^{out}(v)}{m}$ where

Claim In an Eulerian directed graph, the stationary distribution is $\pi_v = \frac{d^{out}(v)}{m}$ when edge uv is traversed with probability $\frac{1}{d^{out}(u)}$.

Proof $\pi_v = \sum_{u:uv \in E} \pi_u P_{u,v} = \sum_{u:uv \in E} \frac{d^{out}(u)}{m} \cdot \frac{1}{d^{out}(u)} = \frac{d^{in}(v)}{m} \overset{\text{Eulerian}}{=} \frac{d^{out}(v)}{m} = \pi_v$.

Since $\sum_v \pi_v = \sum_v \frac{d^{out}(v)}{m} = 1$, this is the unique stationary distribution. □

Therefore, $H_{s,s} = m / d^{out}(s)$.

In the i-th iteration when there are only i edges in the matching, $d^{out}(s) \geq (n-i) \cdot d$ assuming G_i is a d-regular graph.

So, in the i-th iteration, $H_{s,s} \leq dn/d(n-i) = n/(n-i)$.

Therefore, the total running time is $\sum_{i=0}^{n-1} n/(n-i) = O(n \log n)$.

With appropriate data structures, the total complexity of the algorithm is $O(n \log n)$.

Open problem: Can you extend this approach to non-regular bipartite graphs?

References

- Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs, by Christiano, Kelner, Madry, Spielman, Teng, 2012.
- Approximating undirected maximum flow in $O(m \text{polylog}(n))$ time, by Peng, 2016.
- Navigating central path with electrical flows: from flows to matchings and back, by Madry, 2013.
- Path finding method in linear programming: Solving linear programs in $\tilde{O}(\text{rank})$ iterations and faster algorithms for maximum flow, by Lee and Sidford.
- Perfect matching in $O(n \log n)$ time in regular bipartite graphs, by Goel, Kapralov, Khanna, 09.