Electrical Flows, Laplacian Systems, and Faster Approximation of Maximum Flow in Undirected Graphs Cristiano, Kelner , et al

Karthik Abinav

1 Introduction

This paper gives a laplacian solver based algorithm to the max flow problem. Using electrical flows and fast laplacian solvers, the running time to the (1-ϵ)-approximate s-t flow problem in $\widetilde{O}(mn^{\frac{1}{3}}\epsilon^{\frac{-11}{3}})$. The paper starts of by giving a simpler algorithm which runs in time $\widetilde{O}(m^{\frac{3}{2}}\epsilon^{\frac{5}{2}})$. Then some key observation in this algorithm is used to get the modified algorithm that runs in time $\widetilde{O}(m^{\frac{4}{3}}\epsilon^{-3})$. Then using the smoothing and sampling techniques given by [insert bibtex link to karger's paper]. The main drawback with this algorithm is that, it can get the approximate value of the flow. But, there is no technique to map the flow in the sparse graph to the original graph. The paper also discusses an algorithm to dual of this problem that is the minimum s-t cut. It gives a $(1+\epsilon)$- algorithm to the minimum s-t cut problem running in time $\widetilde{O}(m + n^{\frac{4}{3}}\epsilon^{\frac{-8}{3}})$. The advantage of this algorithm is that, it can get the approximate cut along with the value of the minimum cut. Hence, a gap exists between the primal problem and the dual problem. This issue is addressed in a further work by Kelner, Orecchia, et. al.

2 Notations

In this section, some notations that will be used throughout this summary will be discussed.

u_e - This defines the capacity associated with the edge e.

U - This defines the ratio between the highest capacity to the lowest capacity in the graph i.e. $\frac{\max_e u_e}{\min_e u_e}$

Every edge in the graph is arbitrarily oriented for notational purposes. An edge that comes into a vertex v is denoted by $E^+(v)$ and every edge that leaves the vertex v is defined as $E^-(v)$.

A vector f is a vector of flow. The i^{th} coordinate of the vector denotes the flow in the i^{th} edge for a given ordering of the edges

Every valid flow of value F has the following properties:

- Flow Conservation:
$$\sum_{e\in E^+(v)} f(e) - \sum_{e\in E^-(v)} f(e) = 0 \qquad \forall v \in V \setminus \{s,t\}$$
- Net flow value:
$$|f| = F = \sum_{e\in E^+(s)} f(e) - \sum_{e\in E^-(s)} f(e)$$
- Feasible flow :
$$f(e) \le u_e \qquad \forall e \in E$$

The maximum flow problem: The goal of this problem is to find a valid flow of value F^* such that the maximum flow that can be routed from s to t in the graph is F^*.
A $(1-\epsilon)$-approximation is an algorithm that returns a valid s-t flow, whose value is atleast $(1-\epsilon) * F^*$

2.1 Electrical flows

Every edge of a graph G, we visualize an edge as a resistor with resistance r_e. We can then define an energy associated with a flow f as follows:

$$\epsilon_r(f) = \sum_e r_e f_e^2$$

Lemma:
Among all valid flows of value F, the one that minimizes the energy is the one that follows kirchoff's laws of current and voltage

Proof:
Consider the case of two resistors in series and two resistors in parallel. If we prove the lemma for these two cases, it applies for any general circuit.

Series: In this case, there is just one possible flow, which is the electrical flow. Hence, statement trivially holds true.

Parallel: Let us start with an electrical flow. We will show that any change to the flows will only increase the energy. Let i_1 and i_2 be the electrical flow through resistors r_1 and r_2. From Kirchoff's law we have $i_1 * r_1 = i_2 * r_2$. The energy now is $\epsilon = r_1 * i_1^2 + r_2 * i_2^2$. Let us now redistribute the flow as $i_1 + \delta$ and $i_2 - \delta$. The value of the flow remains the same. The energy now becomes $\epsilon' = \epsilon + \delta^2(r_1 + r_2) \geq \epsilon$. Hence the lemma.

Incidence Matrix: The incidence matrix gives the details of the directed edges in a graph. The incidence matrix is denoted by B a n $*$ m matrix. The (i,j) entry is 1, if the edge j leaves vertex i. It is defined as -1, if the edge j is incident vertex i.

Hence, the max flow problem can now be stated as calculating a vector f, such that.

$$Bf = \chi_{s,t}$$

Here, $\chi_{s,t}$ is a vector which is 1 at the s^{th} coordinate, -1 at the t^{th} coordinate and 0 elsewhere.

Resistance matrix: This is a diagonal matrix i.e. the off-diagonal entries are all 0. The (i,i) entry of the matrix contains the value r_i, that is the resistance of the $i_t h$ edge.

This implies the energy can now be re-written as $R^T f R = ||R^{\frac{1}{2}} f||^2$

Effective Resistance: For an electrical flow f of value 1 in a resistive electrical network, let ϕ be the vertex potentials. The effective resistance of the network with respect to the resistances r is given by :

$$R_{eff}(r) = \phi(s) - \phi(t)$$

Graph Laplacian: The weighted Laplacian of a graph G with respect to edge resistances r is defined as:

$$BR^{-1}B^T$$

Congestion of an edge: Congestion of an edge e with respect to flow f is defined as $cong_f(e) = \frac{|f_e|}{u_e}$

Lemma: $R_{eff}(r) = \epsilon_r(f)$

Proof: $R_{eff}(r) = \phi(s) - \phi(t) = \phi^T \chi_{s,t}$

Since f is an electrical flow, f(u,v) $= \frac{\phi_u - \phi_v}{r_{u,v}}$. Hence, $f = R^{-1}B^T\phi$.

$\epsilon_r(f) = f^T R f = (R^{-1}B^T\phi)^T R(R^{-1}B^T\phi) = \phi^T L\phi$

$L\phi = \chi_{s,t}$

$\epsilon_r(f) = (L^{-1}\chi_{s,t})^T L(L^{-1}\chi_{s,t}) = \chi_{s,t}^T L^{-1}\chi_{s,t}$

$R_{eff}(r) = (L^{-1}\chi_{s,t})^T \chi_{s,t}$

3 δ-approximate flow

Given the following as input:
$\delta > 0$
$F > 0$
vector r of resistances in which the ratio of the largest to smallest resistance is atmost R

an algorithm returning the following output can be computed in time $\widetilde{O}(m \log(\frac{R}{\delta}))$

Vector potentials $\widetilde{\phi}$, An s-t flow $\widetilde{f}$ of value F. Additionally the output satisfies the following :

- $\epsilon_r(\widetilde{f}) \leq \epsilon_r(f) * [1+\delta]$
- For every edge e,

  $|r_e f_e^2 - r_e \widetilde{f}^2| \leq \frac{\delta}{2mR}\epsilon_r(f)$
- $\widetilde{\phi}(s) - \widetilde{\phi}(t) \geq (1 - \frac{\delta}{12nmR}) * F * R_{eff}(r)$

The algorithms that follow will use the above algorithm as a sub-routine to obtain a δ-approximate electrical flow, for appropriate values of δ.

4 A $\widetilde{O}(m^{\frac{3}{2}}\epsilon^{\frac{-5}{2}})$ time flow algorithm

In this section, first we will have a look at a simpler version of the algorithm. After taking key insights from this algorithm, we can make appropriate observations and modifications to improve the running time.

4.1 (ϵ, ρ)- oracle

This is the second sub-routine that will be used by the algorithm. The construction of such an algorithm uses the δ-approximate electrical flow algorithm.

Input
Real number $F > 0$
w - vector of edge weights, where $w_e \geq 1 \ \ \forall e \in E$

Output
A s-t flow f satisfying the following criteria

- If $F \leq F^*$, output a flow f s.t.
  - $|f| = F$
  - $\frac{\sum\limits_e w_e * cong_f(e)}{\sum\limits_e w_e} \leq (1+\epsilon)$
  - $\max\limits_e cong_f(e) \leq \rho$
- If $F > F^*$, output a flow f sastisfying the above or "fail".

4.2 Main Algorithm

The main algorithm calls the (ϵ, ρ) subroutine many times with different weights and then takes an average of all the values returned by the oracle as the required value of flow. Note, the flow returned by this algorithm should maintain the congestion criteria for each of the edges. On the other hand, the oracle may return a flow whose congestion at each edge is as bad as ρ. But, since the oracle gives a upper bound on the average congestion, the algorithm uses that critically to, in some sense "correct" the value of flows.

Input
A graph G
a vector u of edge capacities
A target flow value F
(ϵ, ρ) - oracle O

Output
Either a flor f or "fail" indicating $F > F^*$

Note, this algorithm requires a target flow value F as an input. Hence, to get a value of maximum flow, we can perform a recursive doubling and binary search on the value of F to get to the value F^*.

The algorithm goes as follows:

Note: The superscript on the weight vector impies the weights in the i^{th} iteration of the algorithm.

- Initialize the weights w^0 as $w_e = 1$ for all the edges e. Initialize N as $\frac{2\rho \ln(m)}{\epsilon^2}$
- for i from 1 to N:
  - Query the oracle O with w^{i-1} and target flow as F
  - if O returns a "fail", return "fail"
  - else:
    Let f^i be the value of the returned flow. Update weights as
    $$w^i = w^{i-1} * \left(1 + \frac{\epsilon * cong_{f^i}(e)}{\rho}\right)$$
- Return f as $\frac{(1-\epsilon)^2}{(1+\epsilon)N} \left(\sum_i f^i\right)$