Randomized Algorithms 2017A – Lecture 5
Effective Resistance and Algorithms for SAT*

Robert Krauthgamer

1 Reminder: Graphs as Electrical Networks

Recall that in an electrical network, we view a graph as a collection of (undirected) resistors. When we create a potential difference ϕ_{uv} between two vertices, it induces an electrical flow (current), which is (i) a feasible flow in the sense of flow preservation (KCL), and (ii) creates potentials (voltages) on all other vertices (KVL), and (iii) the flow is inverse proportional to the potential difference, and directed accordingly (Ohm's Law).

Observation: The amount of flow shipped from u to v grows linearly with ϕ_{uv}.

Example: Suppose G is a path on the vertices u, w, v, and we create potential difference ϕ_{uv}. Then (KCL)

$$\phi_{uw} = f_{uw} r_{uw} = f_{wv} r_{wv} = \phi_{wv}.$$

Since the LHS and RHS sum up to ϕ_{uv}, each of them is exactly $\frac{1}{2}\phi_{uv}$, and thus $f_{uw} = f_{wv} = \frac{1}{2}\phi_{uv}$ is the amount of flow.

Observation: In fact, we can also multiply the add two difference potential functions, and the flows will add up (and vice versa).

2 Effective Resistance

Effective Resistance: The *effective resistance* between vertices u, v in an electrical network, denoted $\mathrm{R}_{\mathrm{eff}}(u, v)$, is the potential difference ϕ_{uv} we need to create between u and v to induce exactly one unit of current flowing from u to v.

To understand it, suppose that when we create potential difference of a unit, some F^* units flow from u to v (recall it depends on G). Now if we scale the potential difference by some ϕ_{uv}, the amount of flow will scale to $\phi_{uv} F^*$, and for this (flow amount) to be equal to 1, we should of course

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

choose the scaling factor to be $\phi_{uv} = 1/F^*$, i.e., $\mathrm{R}_{\text{eff}}(u,v) = 1/F^*$. Perhaps a more appropriate name for F^* could be $F_{\text{eff}}(u,v)$, but actually it is called *effective conductance* $C_{\text{eff}}(u,v)$.

The name comes from the viewpoint that the entire network can be "simulated" by a single resistor between u, v, with resistance $r_{uv} = \mathrm{R}_{\text{eff}}(u,v)$, then the effective resistance between u, v (in it) would be the same. Indeed, if we create a unit potential difference, then $1 = \phi_{uv} = f_{uv}\,\mathrm{R}_{\text{eff}}(u,v)$, thus the amount of flow is $f_{uv} = 1/\,\mathrm{R}_{\text{eff}}(u,v) = F^*$, exactly as in G.

Observe that $\mathrm{R}_{\text{eff}}(u,v)$ is symmetric.

We can now show that the effective resistance is essentially the same as the commute time.

Theorem 7: Let $G = (V,E)$ be an undirected graph. Then

$$\forall u, v \in V, \qquad C_{uv} = 2|E|\,\mathrm{R}_{\text{eff}}(u,v).$$

Lemma 8: Let N_z be the electrical network corresponding to G, when we inject $\deg(u)$ units of flow at every vertex $u \in V$, and extract $\sum_{u \in V} \deg(u) = 2|E|$ units of flow at z. Then the potential differences ϕ^{N_z} satisfy

$$\forall u \in V, \qquad \phi^{N_z}_{uz} = H_{uz}.$$

Proof of Lemma 8 and Theorem 7: Was seen in class.

Theorem (Thomson's Principle revisited): Let f be a unit of flow that flow from u to v that minimizes the energy. Then

$$\mathrm{R}_{\text{eff}}(u,v) = E(f).$$

Proof: Was seen in class.

Theorem (Rayleigh's Monotonicity Law): If $\{r(e)\}$ and $\{r'(e)\}$ are sets of resistances on the edges of the same graph G, such that $r(e) \leq r'(e)$ for all $e \in E$,

$$\forall u, v \in V, \qquad \mathrm{R}_{\text{eff}}{}^{(r)}(u,v) \leq \mathrm{R}_{\text{eff}}{}^{(r')}(u,v).$$

The proof follows directly from Thomson's Principle above.

Corollary: For all $(u,v) \in E$, we have $\mathrm{R}_{\text{eff}}(u,v) \leq 1$ and thus $C_{uv} \leq 2|E|$.

The proof follows by observing that adding an edge is equivalent to reducing the resistance of an edge.

This proves our Theorem 3 (claimed earlier without a proof).

Vertex cut (series composition in disguised form): Suppose every $u - v$ path goes through a vertex w. Then

$$\mathrm{R}_{\text{eff}}(u,v) = \mathrm{R}_{\text{eff}}(u,w) + \mathrm{R}_{\text{eff}}(w,v).$$

Exer: Prove this using composition of the flows. (Another proof is by multiplying by $2|E|$ and using commute time.)

Example 1: The path: $C_{1n} = H_{1n} + H_{n1} = 2H_{1n}$ by symmetry. By the vertex-cut (series composition), $\mathrm{R_{eff}}(1,n) = n-1$, and thus $C_{1n} = 2(n-1)\,\mathrm{R_{eff}}(1,n) = 2(n-1)^2$. We conclude that $H_{1n} = (n-1)^2$.

Notice this is also the cover time of that path.

Example 2: The lollipop: The "lollipop" graph is a path of $n/2$ edges from u to v, where this last vertex v forms a clique with $n/2-1$ new vertices. It can be easily seen $H_{uv} = (n/2)^2$ while $H_{vu} = \Theta(n^3)$ and also $\mathrm{cov}(G) = \Theta(n^3)$.

Exer: Prove these bounds (it's actually easy to get precise formulas).

Hint: Use the effective resistance formula and Theorem 5 (the spanning tree).

Series Composition: Consider two graphs, G_1 and G_2 on disjoint sets of vertices, and in each of them fix two vertices $s_i, t_i \in G_i$ for $i = 1, 2$. Their series composition is the graph $\bar{G}$ obtained by taking their disjoint union but identifying t_2 with s_1. Then

$$\mathrm{R_{eff}}^{\bar{G}}(s_1, t_2) = \mathrm{R_{eff}}^{G_1}(s_1, t_1) + \mathrm{R_{eff}}^{G_2}(s_2, t_2).$$

Notice this is exactly the same as the vertex cut above.

Parallel Composition: Let G_1 and G_2 be as above. Their parallel composition is the graph $\bar{G}$ obtained by taking their disjoint union but identifying s_1 with s_2 (call it $\bar{s}$), and identifying t_1 with t_2 (call it $\bar{t}$). Then

$$\frac{1}{\mathrm{R_{eff}}^{\bar{G}}(\bar{s}, \bar{t})} = \frac{1}{\mathrm{R_{eff}}^{G_1}(s_1, t_1)} + \frac{1}{\mathrm{R_{eff}}^{G_2}(s_2, t_2)}.$$

Exer: Prove this using Ohm's Law.

3 Algorithm for 2-SAT

Problem definition: In the 2-SAT problem, the input is a 2-CNF formula F with m clauses over n boolean variables, and the goal is the decide if F is satisfiable.

This problem is in P (notice it is not MAX-2SAT). In contrast, for every $k \geq 3$, the k-SAT problem is NP-hard.

Exer: Show that 2-SAT can be solved in polynomial time.

Algorithm A:

1. Start with an arbitrary assignment

2. While the assignment does not satisfy F

2.1 pick an arbitrary unsatisfied clause, pick one of its variables at random, and flip its value

3. output the satisfying assignment

Formally, if F is unsatisfiable, then it will never stop, hence we need to stop the algorithm at some point, and we can call that Algorithm A'. Anyway, our goal is to prove that if F is satisfiable, then the algorithm will (probably) find one quickly enough.

Theorem [Papadimitriou, 1991]: The expected number of iterations for the above algorithm to find a satisfying assignment, assuming one exists, is $O(n^2)$.

Proof: Was seen in class.

4 Algorithm for 3-SAT

Can we generalize it to 3-SAT? We of course don't expect a polynomial runtime.

Naive approach: Let a^* and $Y_t \in \{0, 1, \ldots, n\}$ be as before, then for every $0 < j < n$,

$$\Pr[Y_t = j+1 | Y_t = j] \geq 1/3$$
$$\Pr[Y_t = j-1 | Y_t = j] \leq 2/3,$$

and as before, we may consider (for sake of analysis) these are equalities. In other words, Y_t is a random walk but with a drift to the left. What is $H_{\max}$?

Each hitting time $H_{n-j,n}$ satisfies

$$H_{n-j,n} = 1 + \tfrac{1}{2}(H_{n-j-1,n} + H_{n-j+1,n}),$$

and of course $H_{n,n} = 0$ and $H_{0,n} = 1 + H_{1,n}$. It is easy to verify, by induction, that

$$H_{n-j,n} = 2^{n+2} - 2^{j+2} - 3(n-j),$$

which gives a bound of $O(2^n)$ on the expected number of steps to reach a satisfying assignment.

Key insights:

1. An initial assignment that is *random* starts around $n/2$.

2. If we did not reach a satisfying assignment after "enough" steps, we have probably drifted to the left, and it's better to start from a new random assignment.

Thus, the algorithm below performs a *short* random walk with a random start. Of course, it should be repeated many times (this would be Algorithm B').

Algorithm B:

1. Start with an assignment chosen uniformly at random

2. Repeat $3n$ times

2.1 if the assignment satisfies F, report it and exit

2.2 pick an arbitrary unsatisfied clause, pick one of its variables at random, and flip its value

Theorem [Schoening, 1999]: Algorithm B above finds a satisfying assingment, assuming one exists, with probability at least $(\frac{3}{4})^n / \operatorname{poly}(n)$. Thus a satisfying assignment can be found, with high probability, by $(4/3)^n \operatorname{poly}(n)$ repetitions of the algorithm.

Proof: Was seen in class.