CS 761 : Randomized Algorithms , Spring 2018 , Waterloo

Lecture 15 : Random walks on undirected graphs

We first analyze random walks on a line , and use it to design algorithms for 2SAT and 3SAT.

Then, we introduce the concepts in electrical networks and use it to analyze hitting times and cover time of random walks on undirected graphs.

2-SAT

Given a formula with two variables in each clause , there is a deterministic linear time algorithm to determine if there is a satisfying assignment or not.

We consider an algorithm for 2-SAT based on random walks (known as WalkSAT) :

Random walk 2SAT

1. start from an arbitrary assignment
2. Repeat up to $2on^2$ times, terminate if all clauses are satisfied
   - (a) Choose an arbitrary clause that is not satisfied
   - (b) Choose a random variable in the clause and switch its value
3. Return a satisfying assignment if it is found ; otherwise return "unsatisfiable".

Clearly the algorithm is correct when it returns a satisfying assignment .

We thus focus on the case that there is a satisfying assignment but the algorithm didn't find it.

How to analyze the algorithm as analyzing random walks? What random variables do we consider?

A natural choice is to consider the number of clauses satisfied by the current assignment , but it is not easy to analyze as it can vary considerably from step to step.

Instead, we consider how "close" is the current assignment to a satisfying assignment (assuming it exists).

Let S be a satisfying assignment .

Let A_i be the assignment in the i-th step of the algorithm.

Let X_i be the number of variables that have the same value in A_i and S.

If $X_i = n$, then $A_i = S$ and we win.

We will keep track of X_i when a satisfying assignment is not found yet.

First, if $X_i = 0$, then $\Pr(X_{i+1} = 1 \mid X_i = 0) = 1$.

Suppose $1 \le X_i \le n-1$. In an unsatisfied clause, since S is a satisfying assignment, there must be a variable in the clause that has different values in A_i and S.

Since we pick a random variable in that clause, with prob $\ge \frac{1}{2}$ we pick such a variable and "correct" its value.

Therefore, $\Pr(X_{i+1} = j+1 \mid X_i = j) \ge \frac{1}{2}$ and $\Pr(X_{i+1} = j-1 \mid X_i = j) \le \frac{1}{2}$.

To model it as a random walk problem, we consider a pessimistic version of the stochastic process.

$\Pr(Y_{i+1} = 1 \mid Y_i = 0) = 1$

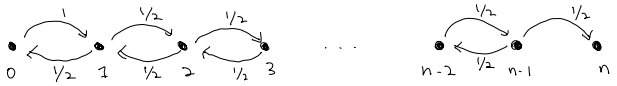
$\Pr(Y_{i+1} = j+1 \mid Y_i = j) = 1/2$

$\Pr(Y_{i+1} = j-1 \mid Y_i = j) = 1/2$.

Clearly, the expected time for Y to reach n is not faster than X to reach n.

We will give an upper bound on the expected time for Y to reach n.

This can be thought of as a random walk on a line.



Let h_j be the expected number of steps to reach n when starting from j.

Then $h_j = \frac{1}{2}(h_{j-1}+1) + \frac{1}{2}(h_{j+1}+1) = \frac{h_{j-1}+h_{j+1}}{2} + 1 \iff h_j - h_{j+1} = h_{j-1} - h_j + 2$

To compute h_j, we just need to solve the following system of linear equations.

$h_n = 0$

$h_j - h_{j+1} = h_{j-1} - h_j + 2$

$h_0 - h_1 = 1$ (because $h_0 = h_1 + 1$)

By induction, we see that $h_j - h_{j+1} = 2j+1$.

We'd like to determine $h_0 = h_0 - h_n = \sum_{i=0}^{n-1} (h_i - h_{i+1}) = \sum_{i=0}^{n-1} (2i+1) = 2\left(\frac{(n-1)n}{2}\right) + n = n^2$. □

So, if we run the algorithm for $2n^2$ steps, by Markov's inequality, the probability of not finding a satisfying assignment (if there exists one) is at most $1/2$.

Therefore, by running for $200n^2$ steps, the failure probability is at most 2^{-100}.

Clearly, the algorithm can be implemented in polynomial time.

3-SAT

The same approach can be applied for 3-SAT, giving a faster exponential time algorithm for it.

The main difference is the probability of "moving up" is only $1/3$.

Using the same notation, $h_n = 0$

$$h_j = \frac{2h_{j-1}}{3} + \frac{h_{j+1}}{3} + 1 \quad \Leftrightarrow \quad h_j - h_{j+1} = 2(h_{j-1} - h_j) + 3$$

$$h_0 - h_1 = 1$$

By induction, $h_j - h_{j+1} = 2^{j+2} - 3$. Summing up as before, $h_j = 2^{n+2} - 2^{j+2} - 3(n-j)$.

In particular, $h_0 = 2^{n+2} - 3n - 4$, which is more than the brute-force algorithm of trying all 2^n assignments.

To improve it there are two observations:

① We can find a better initial assignment. In particular, if we pick a random initial assignment, then the expected number of matches with S is $n/2$.

(This alone is not enough, $h_{n/2}$ is not much smaller than h_0.)

② Since the random walk drifts toward 0, if the algorithm runs longer, then it is more likely that it is well below $n/2$.

In this case, we should restart from a random initial assignment instead of keep running it.

These ideas lead to the fastest algorithm for 3-SAT known some time ago.

Schöning's 3SAT algorithm

① Repeat up to N times, terminating if all clauses are satisfied

ⓐ Start with a random initial assignment

ⓑ Repeat up to $3n$ steps, terminating if all the clauses are satisfied

- switch the value of a random variable in an unsatisfied clause

② Return a satisfying assignment if it is found; otherwise return "unsatisfiable".

Suppose we start with an (unsatisfied) assignment with j mismatches with S.

In $3j$ steps, it reaches n if there are j steps "down" and $2j$ steps "up", which happens with probability $\binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j}$

This is a lower bound on the probability of reaching n in $3n$ steps.

To have a good estimate of it we use Stirling's formula: $\sqrt{2\pi m}\left(\frac{m}{e}\right)^m \le m! \le 2\sqrt{2\pi m}\left(\frac{m}{e}\right)^m$

Using it, $\binom{3j}{j} = \frac{(3j)!}{(2j)!\,(j)!} \ge \frac{\sqrt{2\pi(3j)}}{4\sqrt{2\pi(2j)}\sqrt{2\pi(j)}}\left(\frac{3j}{e}\right)^{3j}\left(\frac{e}{2j}\right)^{2j}\left(\frac{e}{j}\right)^{j}$

$$\binom{2j}{j} \frac{}{(2j)!\,(j)!} \quad \frac{}{4\sqrt{2\pi(2j)}\sqrt{2\pi(j)}} \quad \binom{}{} \quad \binom{}{2j} \binom{}{j}$$

$$= \frac{\sqrt{3}}{8\sqrt{\pi}\sqrt{j}} \left(\frac{27}{4}\right)^j = \frac{c}{\sqrt{j}} \left(\frac{27}{4}\right)^j \quad \text{for } c = \frac{\sqrt{3}}{8\sqrt{\pi}} \quad (\text{a constant})$$

Therefore, starting with an assignment with j mismatches with S, the probability of reaching n is at least $\binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j} \geq \frac{c}{\sqrt{j}}\left(\frac{27}{4}\right)^j\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j} = \frac{c}{\sqrt{j}}\left(\frac{1}{2}\right)^j$.

So, the probability of the algorithm to find a satisfying assignment in $3n$ steps is

$$\geq \sum_{j=0}^{n} \frac{c}{\sqrt{j}}\left(\frac{1}{2}\right)^j \cdot \Pr(\text{a random assignment has } j \text{ mismatches with } S)$$

$$= \sum_{j=0}^{n} \frac{c}{\sqrt{j}}\left(\frac{1}{2}\right)^j \cdot \binom{n}{j}\left(\frac{1}{2}\right)^n$$

$$\geq \frac{c}{\sqrt{n}}\left(\frac{1}{2}\right)^n \sum_{j=0}^{n} \binom{n}{j}\left(\frac{1}{2}\right)^j$$

$$= \frac{c}{\sqrt{n}}\left(\frac{1}{2}\right)^n\left(1+\frac{1}{2}\right)^n \quad (\text{by the binomial theorem})$$

$$= \frac{c}{\sqrt{n}}\left(\frac{3}{4}\right)^n$$

In $3n$ steps, the success probability p is at least $\frac{c}{\sqrt{n}}\left(\frac{3}{4}\right)^n$.

Thus, the expected number of times until a success is at most $\frac{\sqrt{n}}{c}\left(\frac{4}{3}\right)^n$.

Since each time takes $3n$ steps, the expected running time is $O\left(n^{1.5}\left(\frac{4}{3}\right)^n\right)$. □

This is not much slower than the current fastest algorithm, with randomized running time $O(1.308^n)$.

Undirected graphs

In an undirected graph, the transition probability $P_{uv} = 1/d(u) \quad \forall\, u,v \in V$.

The Markov chain is irreducible if the graph is connected.

Also, it can be checked that the Markov chain is aperiodic if and only if the graph is non-bipartite, as an odd cycle can be used to show that $(P_{uv})^t > 0$ for a large enough t for any $u,v \in V$.

Using the same proof as in the Eulerian directed graphs, it is easy to check that $\pi_v = d(v)/2m$ is a stationary distribution, so we have the following result from the fundamental theorem of Markov chains.

Theorem For any connected non-bipartite undirected graph, a random walk will converge to the distribution $\pi_v = d(v)/2m$ regardless of the initial distribution, where $m = |E(G)|$.

Next, we are interested in studying the following quantities:

① Hitting time $h_{u,v}$; ② Commute time $C_{u,v} = h_{u,v} + h_{v,u}$;

③ Cover time : the first time when all vertices are visited at least once.

④ Mixing time : how fast the random walk converges to the unique limiting distribution.

Interestingly, there are close connections between the quantities ①-③ and the concepts in electrical networks. So, we will introduce some basics of electrical networks in the following, and we will analyze ④ next time.

Electrical networks

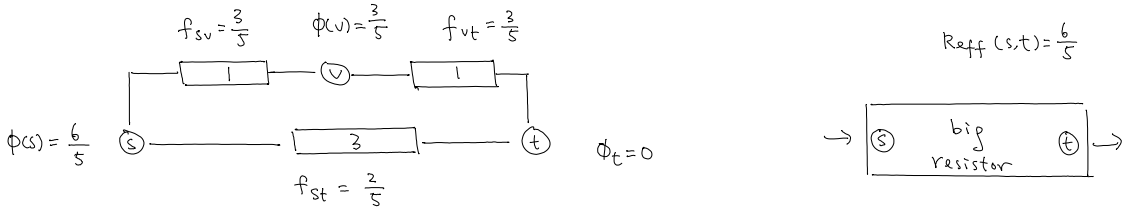
An electrical network is an undirected graph, where each edge e is a resistor with resistance r_e.

The electrical flows in these networks are governed by two rules:

Kirchhoff's law : the sum of incoming currents is equal to the sum of outgoing currents.

Ohm's law : there exists a voltage vector $\phi: V \to \mathbb{R}$ such that $\phi_u - \phi_v = f_{uv} r_{uv}$, where f_{uv} is the electrical flow across the edge uv, which is positive in the forward direction and negative in the backward direction.

For example, if one unit of current is injected into node s and removed from node t, then



The effective resistance between s and t is defined as the voltage difference between s and t when one unit of electrical flow is injected into s and removed from t.

The effective resistance can be understood as the resistance of the whole graph as a big resistor.

We use R_{uv} to denote the effective resistance between u and v.

Theorem For any $u, v \in G$, the commute time $C_{uv} = 2m R_{uv}$, where every edge has resistance one.

Proof Roughly speaking, the proof goes by showing that the two quantities satisfy the same set of equations.

For $v \in V$, let $N(v)$ denote the set of neighbors of v in G, and $d(v) = |N(v)|$ be the degree of v.

Let $\phi_{xv} := \phi_x - \phi_v$ be the voltage difference between x and v, when $d(x)$ amperes are injected into each $x \in V$, and $2m$ amperes are removed from v.

For this electrical flow, by Kirchhoff's law, $d(x) = \sum_{w \sim x} f_{xw}$ (recall $f_{wx} = -f_{xw}$ for any w, x)

$$= \sum_{w \in N(x)} (\phi_x - \phi_w) \qquad \text{// by Ohm's law}$$

$$= \sum_{w \in N(x)} (\phi_{xv} - \phi_{wv}) \text{, for each } x \in V-v.$$

On the other hand, the hitting times also satisfy the same set of equations.

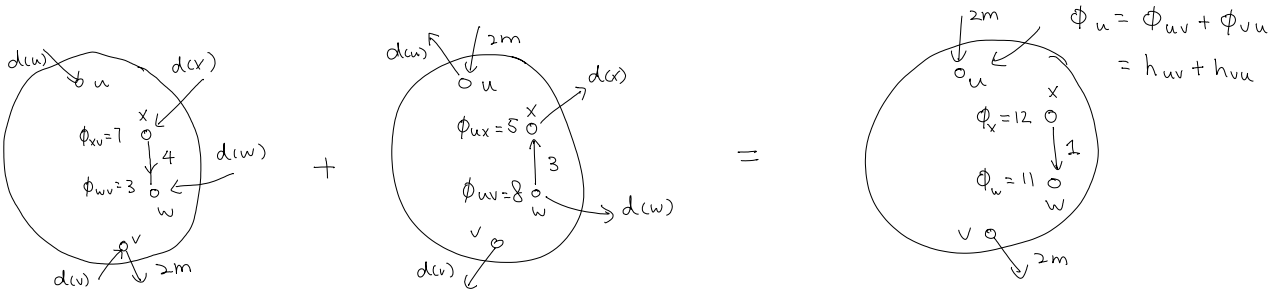
To see this, note that $h_{xv} = 1 + \sum_{w \in N(x)} h_{wv} / d(x)$, which is equivalent to $d(x) = \sum_{w \in N(x)} (h_{xv} - h_{wv})$, for each $x \in V-v$.

This system of linear equations has a unique solution when the graph is connected (the reason is that the system is a Laplacian of a connected graph with one row eliminated, more explanations in lecture), and so $\phi_{xv} = h_{xv}$ for all $x \in V$ with $\phi_{vv} = h_{vv} = 0$.

By the same argument, h_{xu} is the voltage difference between x and u, when $d(x)$ amperes are injected into each $x \in V$, and $2m$ amperes are removed from u.

If we reverse this flow (by changing sign on each edge), h_{xu} is the voltage difference between u and x (note the order changed) when $2m$ amperes are injected into u and $d(x)$ amperes are removed from each $x \in V$.

Adding these two flows (the flow where amperes are injected into every node and removed from v, and the flow where amperes are injected into u and removed from every node) together edge by edge and the voltages vertex by vertex, we obtain a flow with $2m$ amperes injected into u and $2m$ amperes removed from v (check that Kirchhoff's law and Ohm's law are still satisfied) where the incoming amperes are equal to the outgoing currents at every node $x \in V-u-v$.



The voltage difference between u and v in this new flow is equal to $\phi_{uv} + \phi_{vu} = h_{uv} + h_{vu} = C_{uv}$.

If we scale down everything (the voltages and the flows) by a factor of $2m$, then this is a one unit of electrical flow from u to v.

(...

one unit of electrical flow from u to v.

By definition, R_{uv} = voltage difference of u and v in this scaled flow $= \frac{C_{uv}}{2m}$, hence $C_{uv} = 2m R_{uv}$. □

Once we have established the connection to electrical networks, we can use intuitions from physics to derive bounds.

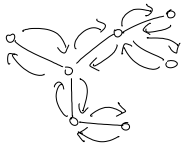
Corollary For any edge $uv \in E$, $C_{uv} \le 2m$.

Proof The voltage difference between u and v is at most one in a one-unit flow (by Ohm's law). □

Theorem The cover time of a connected graph is at most $2m(n-1)$.

Proof Let T be a spanning tree of G.

Consider a walk that goes through T where each edge in T is traversed once in each direction.



Then this is a walk that visits every vertex at least once.

So, the cover time is bounded by the expected length of this walk,

which is at most $\sum_{uv \in T} (h_{uv} + h_{vu}) = \sum_{uv \in T} C_{uv} \le (n-1) \cdot 2m$,

where the last inequality follows from the corollary. □

For the complete graph with n vertices, the cover time is $O(n \log n)$ (why?), but the above gives $O(n^3)$.

The following is a much better estimate of the cover time.

Theorem Let $R(G) = \max_{u,v} R_{uv}$ be the resistance diameter and $C(G)$ be the cover time of G.

Then, $m \cdot R(G) \le C(G) \le 2e^3 m R(G) \ln n + n$.

Proof Let $R(G) = R_{uv}$. Then we know that $2m R_{uv} = C_{uv} = h_{uv} + h_{vu}$.

Therefore, $C(G) \ge \max\{h_{uv}, h_{vu}\} \ge C_{uv}/2 = m R_{uv}$, hence the lower bound.

For the upper bound, note that the maximum hitting time is at most $2m R(G)$, regardless of which vertex.

So, if the random walk runs for $2e^3 m R(G)$ steps, by Markov's inequality, a vertex is not covered with probability at most $1/e^3$.

If the random walk runs for $2e^3 m R(G) \ln n$ steps, then this probability is at most $1/n^3$.

By union bound, some vertex is not visited in $2e^3 m R(G) \ln n$ steps is at most $1/n^2$.

When this happens, we just use the pessimistic bound that $C(G) \le n^3$.

Combining, we have $C(G) \le (1 - \frac{1}{n^2}) \cdot 2e^3 m R(G) \ln n + \frac{1}{n^2}(n^3) \le 2e^3 m R(G) \cdot \ln n + n$. □

Finally, from electrical networks, we have the Rayleigh's principle, which says that the effective resistance never increases when we decrease the resistance of some edge, and the effective resistance never decreases when we increase the resistance of some edge.

Applications of Rayleigh's principle (exercise): Suppose that G has k edge-disjoint paths from s to t, each of length at most l, then $R_{st} \leq l/k$.

Graph connectivity

If we want to test whether there is a path from s to t in an undirected graph, we can simply run a random with $2n^3$ steps.

Since the cover time is at most n^3, it succeeds with probability at least $1/2$ by Markov's inequality.

This algorithm only needs to use $O(\log n)$ space, and still runs in polynomial time.

References - 2SAT and 3SAT are from chapter 7 of Mitzenmacher and Upfal.

- Electrical networks and random walks in undirected graphs are from chapter 6 of Motwani-Raghavan.