Random Walks, Mixing Time, and Algorithmic Applications

CS 292F

June 9, 2022

1 Introduction

The term "random walk" dates back to 1905, when Karl Pearson issued a plea for help with the problem of analyzing the displacement of a man walking in random directions. The term has since become more precise, and has been gainfully studied as a problem in graph theory. Random walks find applications in physics, theoretical computer science, economics, and basically any field looking to stochastically model or approximate difficult environments.

In physics, a random walk may be used to model fluid-flows that no one could hope to model deterministically. Analogously, in algorithms, random walks are often used to approximate the solutions to hard problems. Because a random walk can be simulated using only local knowledge of the topology of a graph, we can compute walk-trajectories quite easily. Thus, if we can derive theoretical guarantees about the long-term behavior of the walk we are simulating, we can arrive at solutions to problems just by performing many of these easy, local updates.

Classically, random walks have been used to approximate the solutions to counting problems (say, counting the number of colorings of a graph). The problem of counting and the problem of uniform sampling are intimately related, so a good idea would be to uniformly sample a vertex of the graph out of a set of desirable vertices. Thus, the long-term behavior we would want out of a random walk designed for such a counting problem would be convergence to the uniform distribution over desirable vertices. The thorny issue of bounding the number of steps that need to be taken in order to approach this distribution – the "mixing time" – has been grappled with for several decades now. We will survey some results that relate the spectrum of the graph to the mixing time.

On the other hand, we will also consider problems where we want just the opposite. In the graph-clustering algorithm that we will survey, the goal is to find a set of vertices that is isolated from the rest of the graph. The algorithm accomplishes this by performing a short random walk, which will hopefully stay within the isolated cluster of vertices.

2 Definitions and Convergence

This section will follow the discussion of Spielman in Chapter 10 of SAGT.

Consider a random walk on G, a weighted, undirected, connected graph. If the walk is at vertex $v \in G$ at time $t \in \mathbb{N}$, it will move to $v' \in N_v$ with probability proportional to $w(v, v')$.

We can associate a probability simplex $\mathbf{p_t} \in \mathbb{R}^V$ with each time-step, where $p_t(i)$ is the probability that the walk is at vertex i at time t. Making the typical definition of $d(v)$ as the sum of the weights incident on vertex v, we can write each transition as follows:

$$p_t(b) = \sum_{(a,b) \in E_G} \frac{w(a,b)}{d(b)} p_{t-1}(a)$$

Let's try to define these transitions in terms of matrices, so that we can do spectral analysis on the walk. Letting M be the adjacency matrix of G and $D = \text{diag}(d(v))$, it can be verified that

$$MD^{-1}\mathbf{p_t} = \mathbf{p_{t+1}}$$

Spielman dubs MD^{-1} the "walk matrix," and gives it the letter W. Our goal will be to analyze the behavior of $W^t\mathbf{p}_0$. Specifically, we will show that $W^t\mathbf{p_0} \overset{t\to\infty}{\longrightarrow} \boldsymbol{\pi} := \boldsymbol{d}/\mathbf{1}^T\boldsymbol{d}$. Spectral theory is well-suited to analyzing the repeated application of symmetric operators. Alas, W is a change-of-basis away from symmetric. This means that while we can find n eigenvalues of W, the matrix will not necessarily admit a spectral decomposition. Scaling each entry $W(i,j)$ by $d(i)/d(j)$ will make it symmetric, so we define

$$A := D^{1/2}WD^{-1/2} = D^{-1/2}MD^{-1/2}$$

Finally, we note that we have basically arrived at the normalized Laplacian of G, $L_G = I - A$.

We will refer to the eigenvalues of the walk matrix thus:

$$\omega_1 > \omega_2 \geq \ \ldots \ \geq \omega_n$$

Large eigenvalues correspond to large subscripts in the Laplacian, so its the other way around for the walk matrix.

Let's now skip liberally through pages of careful argument to prove that $W^t\mathbf{p_0}$ converges to $\boldsymbol{\pi}$, and instead give the gist of the argument. It is clear that $\mathbf{d}$ is an eigenvector of W with eigenvalue 1. It is a simple argument by contradiction to show that all other eigenvalues are no larger than 1. The Perron-Frobenius theorem tells us that:

1. All other eigenvalues are *stricly* smaller than one
2. There is an elementwise positive vector with eigenvalue 1 (we already knew this, its $\mathbf{d}$)

Recalling our discussion of the power method from class, we can see that if we expand $\mathbf{p_0}$ into an eigenbasis of W, all components of $\mathbf{p_0}$ not along $\boldsymbol{\pi}$ will disappear as we repeatedly apply W to it.

So we see that a random walk on a graph converges uniquely to a stationary distribution, $\boldsymbol{\pi}$, for any initial probability simplex $\boldsymbol{p_0}$. Spielman goes on to derive asymptotics for a walk's

pointwise convergence to $\boldsymbol{\pi}$, which is done precisely by bounding the term corresponding to ω_2 in the power method. This isn't quite enough for our purposes, so we'll return to this topic in Section 4.

An important technical note – any random walk can be modified so that it stays at each vertex with probability 1/2 at each time-step without modifying its stationary distribution or its asymptotic convergence time. Thus, many papers *assume* that walks are lazy, because it simplifies many computations.

3 On the equivalence of walks and Markov Chains

A Markov Chain is a random process characterized by the memorylessness property. That is, it is a sequence of random variables $(X_i : \Omega \to \mathbb{R})_{i \in \mathcal{A}}$ satisfying

$$\Pr(X_n = x_n | X_{n-1} = x_{n-1} ... X_1 = x_1) = \Pr(X_n = x_n | X_{n-1} = x_{n-1})$$

where $\mathcal{A}$ is either $\mathbb{N}$ or $\mathbb{R}^+$.

We will notate the right hand side of the above expression $P(x_{n-1}, x_n)$ and call it the transition kernel.[1]

This model of stochastic processes is evidently quite a bit more general than random walks as we've described them. Slightly surprisingly, however, an important class of Markov Chains can be shown to be equivalent to random walks on undirected graphs. These are the Markov Chains $\mathcal{M} = (X_i)_{i \in \mathcal{A}}$ where

1. $\mathcal{A} = \mathbb{N}$
2. $|\Omega| < \infty$
3. $\mathcal{M}$ is time-reversible. That is, (X_n) converges in distribution to $\boldsymbol{\pi} : \Omega \to \mathbb{R}$ such that $P(x, y)\pi(x) = P(y, x)\pi(y)$ [2]

How do we construct a graph so that a random walk on the graph simulates $\mathcal{M}$? Define $G = (V_G, E_G, w)$ as follows:

1. $V_G = \Omega$
2. Add (x, y) to E_G if $P(x, y) > 0$ (by reversibility, $P(x, y) > 0 \iff P(y, x) > 0$) [3]
3. Let $w(x, y) := P(x, y)\pi(x) = P(y, x)\pi(y)$

From our discussion in section 2, we can see that the walk matrix of G is exactly the transpose of the transition kernel of $\mathcal{M}$. This means they will share the same eigenvalues, among other things.

[1] P doesn't depend on the time-step because, in this paper, we make the additional assumption that the dynamics of the Markov Chain do not change with time.

[2] To gain intuition about this definition, imagine two walks on a ring graph. One always moves clockwise, the other moves clockwise with probability 0.5 and counterclockwise with probability 0.5. Imagine playing a video of the chain, and trying to distinguish whether the video is reversed or not.

[3] This is how reversibility is necessary for such a construction to be well-defined

This is really good news. Reversible chains are nice to work with, and are well suited to problems in Markov Chain Monte Carlo. We are now justified in considering the underlying graph of a Markov Chain when reasoning about its convergence properties.

Fun fact: one can also see that directed graphs are enough to capture arbitrary stationary Markov Chains. We didn't learn any spectral techniques for directed graphs in class, so I ignore such processes.

Any time Markov Chain is written from this point forward, the reader should understand it to be a reversible, finite-state, stationary, Markov Chain

4 Mixing

In Section 2, we saw that a random walk on a graph will converge to a stationary distribution that is proportional to the degree of each vertex. From our discussions in Section 3, we know that we can use this result to show that reversible Markov Chains will enjoy this same property. In this section, we seek to characterize the rate at which Graphs/Markov Chains converge to their stationary distribution. Though we could of course do this just as well in the setting of graphs, we will prefer to use the language of Markov Chains, as this is better suited to our goal of analyzing MCMC algorithms.

Recalling our discussion of the power method from Section 2, it is clear that the distance to the stationary distribution will be governed by the largest non-unitary eigenvalue of the walk matrix (which is the same as that of the Markov Kernel).

We make the following definition in order to capture the distance of a Markov Chain to its stationary distribution, π, at time t.

$$\Delta(t) = \max_{i,j} \frac{|P(i,j)^t - \pi_j|}{\pi_j}$$

Every row of $P(i,j)^t$ should go to π with t, but $\Delta(t)$ will pick the $P(i,j)$ that fractionally deviates most from π_j. So $\Delta(t)$ captures convergence in a "uniform" sense. This, together with the bound's agnosticism with respect to the initial distribution make it more compelling than the one presented in Chapter 10 of Spielman.

Unsurprisingly, the following is true:

$$\Delta(t) \leq \frac{\lambda_{max}^t}{\min \pi_j} \tag{1}$$

where $\lambda_{max}^t := \omega_2$ is the second-largest eigenvalue of the walk matrix/transition matrix. Proving this bound is not difficult, but would require a foray into weedy, algebraic territory. Instead, we appeal again to the intuition supplied by the power method. The minimum in the denominator comes from distributing the maximum over the fraction, but can intuitively be thought of as the penalty paid in reaching remote nooks and crannies of the chain.

We have many equivalent ways of analyzing the spectrum of a walk. All characteristic operators (A, L, W, and P) are related by such simple transformations that we could use any one of their eigenvalues to understand mixing. However, the main point to bear in mind is that the long-term behavior of a walk is determined by its extreme eigenvalue. Because it is often difficult to understand the spectrum of a walk, we will choose to exploit the link

between the spectrum of the Laplacian and the combinatorial structure of a graph to get more workable bounds.

5 Applications

5.1 Counting

In this section, we have a brief look at the techniques involved with using random walks on counting problems.

We have now seen that the Fiedler value of the underlying graph of a Markov Chain bounds mixing time. We recall that there is a relationship between the Fiedler value of a graph and its conductance. Given all this, it is reasonable to expect that the mixing time of Markov Chains can be analyzed by analyzing the conductance of their underlying graph.

Recall Cheeger's inequality for a graph, G

$$\Phi_G \leq \sqrt{2\nu_2} \tag{2}$$

where ν_2 is the second smallest eigenvalue of G's Laplacian. Now note that the eigenvalues of the walk matrix (and so the Markov-kernel) are related to those of the Laplacian by

$$\omega_i = 1 - \nu_i$$

Equation (1) then becomes

$$\omega_2 \leq 1 - \Phi_G^2/2 \tag{3}$$

So, the mixing time bound from an earlier section becomes:

$$\Delta_U(t) \leq \frac{(1 - \frac{1}{2}\Phi_G^2)^t}{\min_{j \in U} \pi(j)} \tag{4}$$

Which (using $1 + x \approx e^x$) leads to

$$\tau(\epsilon) \leq \frac{2}{\Phi_H^2}(\ln \pi_{\min}^{-1} + \ln \epsilon^{-1}) \tag{5}$$

Almost all early work on rapidly mixing Markov Chains uses (5), because it is "easier" to tightly bound the conductance of a graph [4]

So how do we use such bounds in practice? Consider the problem of approximating the permanent of a matrix $A \in \{0,1\}^{n \times n}$:

$$\text{per}A = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)}$$

The permanent looks a whole lot like the determinant, except that pesky sum prevents the use of efficient divide and conquer methods for its computation.

[4]Recall the difficulty of bounding the eigenvalue of a planar graph. Now imagine doing the same with a graph on all q-colorings of a graph. As far as I can tell, however, spectral techniques are making a comeback in recent years.

Viewing A as an adjacency matrix of a graph, it has been shown that the problem of exactly computing the permanent is the same as counting maximal matchings, a $\#P$ problem. We now (loosely) present an approximation algorithm to this problem due to Jerrum and Sinclair, which was iterated upon for a more than a decade before it came to work on arbitrary, positive-entry matrices.

More formally, the goal of the construction is to create a Fully-Polynomial Randomized Approximation Scheme (FPRAS), which is a randomized algorithm polynomial in $n, \frac{1}{\epsilon}, \frac{1}{\delta}$, that produces an approximate count Z such that

$$\Pr((\exp -\epsilon)Z^* \leq Z \leq (\exp \epsilon)Z^*) \geq 1 - \delta$$

for any ϵ and δ, where Z^* is the true count. Typically δ is fixed at $1/4$.

It is a well-known result that an FPRAS is equivalent to an algorithm that samples "nearly-uniformly" from the graph that Z is counting. To see why, imagine that Z is trying to count the vertices of a graph H, and let $H_0 = \emptyset, H_1, \ ... \ H_{n-1}, H_n = H$ be a sequence of graphs such that $H_{i+1} = H_i + e_i$, and $\{e_1, ..., e_n\} = E_H$. Then, producing several uniform samplings of H_{i+1} and checking for membership in H_i allows one to form a Monte-Carlo estimate of $\frac{|H_i|}{|H_{i+1}|}$. Chaining these together for each i then gives an estimate of $1/H$. [5]

So we set about trying to sample uniformly from the set of perfect matchings of a graph givven its adjacency matrix. This is done by walking on a graph/Markov Chain! Here is a description of the chain C:

1. The statespace, Ω is the set of perfect matchings of G, $\mathcal{M}$ unioned with the set of "near-perfect matchings," $\bigcup_{(u,v)\in E_G} M(u,v)$, where $M(u,v)$ is the set of matchings with u and v unmatched.

2. Transitions are as follows:

   (a) From state $M \in \Omega$, choose an edge $(u,v) \in E_G$ uniformly.

   (b) if M is a perfect matching, transition to $M - (u,v)$

   (c) If M is an imperfect matching in $M(u,v)$ transition to $M + (u,v)$

   (d) If M is an imperfect matching containing (u,w) with v unmatched, transition to the matching $M(u,v) - (u,w)$

   (e) If M is an imperfect matching containing (v,w) with u unmatched, transition to the matching $M(u,v) - (v,w)$

   (f) Otherwise remain at M

   (g) Make the chain lazy with probability 1/2 (this can be done WLOG)

Why is this chain reversible and convergent to $\boldsymbol{\pi} = \frac{1}{|\Omega|}\mathbf{1}$? Because (u,v) is sampled randomly of the set of edges. Therefore, after each transition from matching $\mathcal{S}$ to $\mathcal{T}$, we make the reverse transition with equal probability. Thus the uniform vector will satisfy the detailed balance equations.

[5] In order to show that near-uniform sampling is good enough, the deltas and epsilons need to be carried through the argument with some care.

Intrepid readers with some leisure time should consult [1] for a lucid, combinatorial argument for the following theorem, which shows that this chain has a large conductance. This will imply that it converges "rapidly":

Theorem 1. *Jerrum and Sinclair (1989) If G is dense, then the conductance of the graph, H, underlying the Markov Chain C satisfies*

$$\Phi_H \geq \frac{1}{12n^6}$$

Corollary 2. *C gives a FPRAS.*

Proof. Consulting equation (5) and plugging in the Jerrum-Sinclair bound on Φ_H, as well as the bound that $\pi_{\min}^{-1} \leq |N|$ (which comes from the fact that $\boldsymbol{\pi}$ is the uniform), we see that the chain converges in polynomial time in n, $1/\epsilon$.

□

The density requirement of the Jerrum-Sinclair theorem is needed in order to ensure the existence of certain paths used in the proof of the conductance bound. Roughly speaking, a "canonical path" is defined between each pair of vertices, and a "congestion" metric is defined on the graph, as the maximum ratio between the load of an edge in the stationary distribution and the edge's capacity. By showing congestion is small, one can show conductance is large.

As n gets large, the overwhelming majority of matrices are dense, so in some sense, there is hope that this algorithm can be improved. Indeed, in 2004, Jerrum et al. improved it to work on arbitrary matrices with non-negative elements.

The key insight that this discussion hoped to convey is the importance of the conductance of a graph in its rapid mixing. Cheeger's theorem, combined with the spectral bounds from Section 4, was enough to form a sufficient bound on mixing time in terms of conductance, though combinatorial techniques were needed in order to bound the conductance itself. Thus, we should think of conductance and the Fiedler value as a measure of the diffusive properties of a graph. This intuition will serve us well not only in counting problems, but also in the algorithm we discuss next.

5.2 Local Graph Clustering

In the previous section, the conductance was a measure of the global diffusive properties of a graph – if you can get a Markov Chain whose underlying graph has a high conductance, you can get close to a uniform sampling in fewer steps.

Recall that the conductance of a graph G was actually defined as the minimum over subsets $S \subset G$ of the conductance of the sets themselves:

$$\phi_S = \frac{w(\partial S)}{\min\{d(S), d(V-S)\}}$$

$$\Phi_G = \min_S \phi_S$$

How should we understand the conductance of a *set* as it pertains to random walks? It would be eminently reasonable to expect a set with low conductance to remain isolated under a random walk. Of course, this is indeed the case. Let $\tilde{W} = \frac{1}{2}(W + I)$ be the lazy-fication of a walk W on a graph G. Let $S \subset G$ with $|S| \leq |V_G|/2$. Then the following is true:

Lemma 3. *Let $\mathbf{p_0} \sim d(S)$. Then the probability that the walk leaves S in one step, $\mathbf{1}_{V-S}^T \mathbf{p_1} = \mathbf{1}_{V-S}^T \tilde{W} \mathbf{p_0}$ is $\phi_S/2$.*

Proof.

$$\mathbf{1}_{V-S}^T \tilde{W} \mathbf{p_0} = \sum_{a \in S} \frac{1}{2} p_0(a) \sum_{(a,b) \in E, b \notin S} \frac{w_{a,b}}{d(a)} = \frac{1}{2d(S)} \sum_{(a \in S, b \notin S) \in E} w_{a,b} = \frac{1}{2} \phi_S$$

□

So the conductance of a set is directly related to the probability of exiting that set through a random walk [6]

An induction argument can then be used to generalize this fact to the following:

Theorem 4. *Let variables be as in Lemma 3. Then $\mathbf{1}_{V-S}^T \tilde{W}^t \mathbf{p_0} \leq \frac{t\phi_S}{2}$*

□

In English, Theorem 4 says that the probability that the walk has left S is bounded above by ϕ and t. At this point, we might begin to see how a random walk starting in an isolated set might produce a cluster that resembles that set. If we start a walk (uniformly) in a set S is well-isolated, we can choose t as a function of ϕ in order to keep most of the probability mass in S.

Though this is one of the key insights needed for randomized local graph clustering, the actual algorithm, due to Spielman and Teng, needs a lot more. Let's make our goal a bit more formal. We hope to design a randomized algorithm that takes as input a target set size, s, a target conductance ϕ, and a seed vertex a, and returns a set T, containing a, with degree roughly s and conductance less than ϕ. The steps are the following.

1. Let $\boldsymbol{p_0} = \delta_{\mathbf{a}}$
2. While $t \leq 1/2\phi$, let $p_{t+1} = \tilde{W} \mathbf{p_0}$, replace all small entries of p_{t+1} with 0, and renormalize to be a simplex.
3. Try every $\mathbf{p_{t+1}}$-cut (there are n of these), and if any of them give a good enough cut, call it T and return it.

When we talk about finding ($v \in \mathbb{R}^V$)-cuts, we mean find $\pi \in S_n$ such that

$$v(\pi(1)) \leq v(\pi(2)) \leq ... \leq v(\pi(n))$$

and look at all sets of the form:

$$S_j = \{\pi(i) \ : \ 1 \leq i \leq j\}$$

A "good enough" cut will be one that satisfies the input requirements on degree and conductance.

[6] Note that the factor of 1/2 in the above statement comes precisely from the fact that the walk is lazy.

Recall that in order to prove Cheeger's Inequality, we showed that for some j, ϕ_{Sj} was bounded above by the normalized Reyleigh Quotient:

$$\phi_{Sj} \leq \sqrt{2\frac{2\mathbf{v}^T L\mathbf{v}}{\mathbf{v}^T D\mathbf{v}}}$$

and then derived a vector that made the right-hand side small from the Fiedler vector.

We can't use the Fiedler vector, because we are only able to perform local computations on the graph. Why should we believe that $\mathbf{y} := \mathbf{p_T}$ will be a good cut vector? We have seen that the walk probably won't leave the set S, so the small probability mass outside of S in $\mathbf{p_T}$ will be killed in step 2. Thus, we are heuristically inclined to believe that $\mathbf{p_T}$ will have large entries inside some subset $T \subset S$, and small entries outside of S. This in turn means that one of the sets S_j should look quite a lot like S. This intuition helps us recognize that the truncation step is not actually necessary for the algorithm to work – the S_j will not be effected by this. Rather, truncation is used in Spielman and Teng's analysis of the algorithm.

This algorithm looks to strike a delicate balance: probability mass should diffuse throughout S, but not too much should leave it.

Unfortunately, this is as far as we will go in our discussion of Spielman and Teng's local clustering algorithm. Of all the simplifications that this paper has made, those of this section have certainly been the most glaring. For one thing, we do not get to start the random walk on a uniformly random vertex in S. Rather, we must find a seed-vertex a, that will yield a good cut. It is difficult to show that there's a high-probability way to find this vertex. For another thing, the convergence machinery we have presented needs to be moulded in order to be applicable here. A very intuitive perspective on convergence, due to Lovasz and Simonovits, is leveraged by Spielman and Teng. [7]

6 Conclusion

Perhaps, however, these examples have shown the scope of spectral insights in randomized algorithms. There is a bounty of ways of thinking about these kinds of problems. Writing this paper has been an exercise in translation – between Laplacian and walk matrix, between walk matrix and Markov Chain, and between spectrum and conductance. Personally, this exercise has shed light upon the workings of these algorithms.

Hopefully, the reader has gotten a taste for how random walks can be used to circumvent hardness in computation. The art of turning local computations into global results is theoretically intricate, computationally promising, and intuitively compelling. Though practical arguments in this field often require some sort of combinatorial machinery, spectral thinking as an organizing principle is empowering. Understanding the connections between a graph and its spectrum can give intuitive shortcuts through some hairy arguments.

[7] This perspective represents probability distributions as curves resembling CDFs, and establishes a link between the behaviors of these curves and the evolutions of random walks

References

[1] Mark Jerrum and Alistair Sinclair, *Approximating the permanent*, SIAM J. Comput. **18** (1989), no. 6, 1149–1178.

[2] László Lovász and Miklós Simonovits, *The mixing rate of markov chains, an isoperimetric inequality, and computing the volume*, 31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I, IEEE Computer Society, 1990, pp. 346–354.

[3] Alistair Sinclair and Mark Jerrum, *Approximate counting, uniform generation and rapidly mixing markov chains*, Inf. Comput. **82** (1989), no. 1, 93–133.

[4] Daniel A. Spielman and Shang-Hua Teng, *A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning*, SIAM J. Comput. **42** (2013), no. 1, 1–26.