CMPUT 498/501: Advanced Algorithms **Fall 2025**

Lecture 10 (Oct 6, 2025): Random Walks on Graphs

Lecturer: Mohammad R. Salavatipour *Scribe: Priyanshu Thakkar, Bhavya Thakkar*

10.1 Random Walks

A random walk is a process that models a path of successive random steps. Let there be a sequence of independent and identically distributed random variables $X_1, X_2, \dots$ taking values in $\mathbb{R}^d$, representing the *steps* of the walk. Starting from an initial position $z \in \mathbb{R}^d$, the random walk is given by the sequence $(S_n)_{n\geq 0}$ where

$$S_0 = z, \quad \text{and} \quad S_n = S_{n-1} + X_n \text{ for } n \geq 1.$$

The position at time n can be expressed as

$$S_n = z + \sum_{i=1}^{n} X_i.$$

The key property is that each step X_i is chosen independently from the same distribution, independent of previous steps or the current position. At each discrete time step, the walk moves from its current location by adding a randomly chosen displacement.

Although the individual steps are independent random variables, the positions $S_0, S_1, S_2, \dots$ themselves form a dependent sequence, since each position depends on all previous steps.

Definition 1 (2-SAT Problem) *Given a 2-CNF formula Φ, i.e., a formula consisting of m clauses on n variables, each of the form*

$$(x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_4) \wedge (x_3 \vee \neg x_8) \wedge \dots$$

we ask whether the formula Φ is satisfiable.

Algorithm 1 Randomized 2-SAT Algorithm

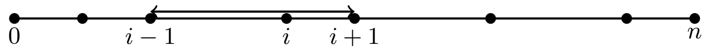
```
x ← arbitrary assignment in $\{0,1\}^n$
while there exists an unsatisfied clause do
  Pick an unsatisfied clause $C$
  Pick uniformly at random one of the two variables in $C$
  Flip its value
end while
return x
```

Analysis: Assume that Φ is satisfiable, and let A denote a satisfying assignment. We refer to the values of variables in A as correct values. Let n denote the total number of variables, and let X_i represent the number of variables in the current iteration that have the correct value with respect to A. The algorithm terminates when $X_i = n$, and note that all $X_i \geq 0$.

The algorithm stops when $X_i = n$. In each iteration, the value of X_i changes by exactly one, i.e.,

$$|X_i - X_{i+1}| = 1.$$

This behavior can be viewed as a random walk on a line with a barrier at 0. At every step, we move one position either to the left or to the right.



Our goal is to determine the expected time to reach n. Note that

$$\mathbb{P}[X_{i+1} = 1 \mid X_i = 0] = 1.$$

Claim 1 *Let $1 \le X_i \le n-1$ and consider a random unsatisfied clause C. Then*

$$\mathbb{P}[X_{i+1} = j+1 \mid X_i = j] \ge \frac{1}{2}, \quad \textit{and} \quad \mathbb{P}[X_{i+1} = j-1 \mid X_i = j] \le \frac{1}{2}.$$

Proof. Consider an unsatisfied clause C that disagrees with A in at least one variable. We increase the number of agreements with probability at least $\frac{1}{2}$:

$$\begin{cases} \text{If both variables disagree with } A\text{, the count increases by 1 with probability 1,} \\ \text{If one variable disagrees, the count increases by 1 with probability } \frac{1}{2}. \end{cases}$$

Thus the claim follows. ■

However, this process is not a Markov chain.

To address this, we define another random walk $Y_0, Y_1, \dots$ that **is** a Markov chain and serves as a pessimistic estimator for X_i. Let $Y_0 = X_0$, and define:

$$\mathbb{P}[Y_{i+1} = 1 \mid Y_i = 0] = 1, \quad \mathbb{P}[Y_{i+1} = j+1 \mid Y_i = j] = \frac{1}{2}, \quad \mathbb{P}[Y_{i+1} = j-1 \mid Y_i = j] = \frac{1}{2}.$$

Observation 1 *The expected time for Y to reach n is an upper bound on the expected time for X to reach n.*

Let Z_j denote the number of steps required to reach n from j, and define $h_j = \mathbb{E}[Z_j]$. We can write h_j as

$$h_j = \mathbb{E}[Z_j] = \mathbb{E}\left[\tfrac{1}{2}(1 + Z_{j-1}) + \tfrac{1}{2}(1 + Z_{j+1})\right] = \frac{h_{j+1} + h_{j-1}}{2} + 1.$$

The boundary conditions are

$$h_0 = h_1 + 1, \qquad h_n = 0.$$

Lemma 1 *For all $0 \le i \le n-1$, we have $h_i = h_{i+1} + 2i + 1$.*

Proof. We proceed by induction on i.

Base Case: For $i = 0$, we have $h_0 = h_1 + 1 = h_1 + 2(0) + 1$, so the relation holds.

Inductive Step: Assume the relation holds for some $i \geq 0$, i.e., $h_i = h_{i+1} + 2i + 1$. From the recurrence relation, we have

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

Solving for h_{j+1}:

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

By the inductive hypothesis, $h_{j-1} = h_j + 2(j-1) + 1$, so

$$h_{j+1} = 2h_j - (h_j + 2(j-1) + 1) - 2 = h_j - 2j - 1.$$

This completes the induction. ■

Theorem 1 *If Φ is satisfiable, the expected number of steps required to reach a satisfying assignment is at most n^2.*

Proof. By the lemma, summing from $i = 0$ to $i = n-1$:

$$h_0 = \sum_{i=0}^{n-1}(2i+1) = n^2.$$

■

Corollary 1 *If we run the algorithm for $2n^2$ steps, we find a solution (if one exists) with probability at least $\frac{1}{2}$. By repeating this process α times, we can find a solution in $2\alpha n^2$ steps with probability at least $1 - 2^{-\alpha}$.*

10.2 Random Walks on Graphs

In a graph, we begin at a vertex and arbitrarily select one of its neighbors and proceed to it. We again randomly select a neighbor from this vertex and proceed to it, and we continue this process indefinitely. This arbitrary sequence of vertices visited in this manner is called a random walk on the graph. Random walks are finite Markov chains with the property of time-reversibility. Mathematical models of random walks on graphs and finite Markov chains are quite similar to one another. Any Markov chain can be modeled as a random walk on a directed graph whose edges can be weighted. In addition, time-reversible Markov chains are essentially equivalent to random walks on undirected graphs, and symmetric Markov chains are equivalent to random walks on regular symmetric graphs.

Applications: Random walks on graphs have a wide range of important applications. They are used in approximation and estimation problems such as computing permanents and estimating volumes. Random walks also play a crucial role in randomized selection algorithms, derandomization of randomized algorithms, probability amplification, and pseudo-random generation.

Algorithm 2 Random Walk on Graph

```
Start from a vertex $s \in V$
for $t = 1$ to $T$ do
    Choose a uniformly random neighbor $v$ of current vertex $u$
    $u \leftarrow v$
end for
```

Quantities of Interest:

- **Hitting time:** For vertices $u, v \in V$, the hitting time H_{uv} is the expected number of steps required to reach v starting from u:
$$H_{uv} = \mathbb{E}[\#\text{ of steps to reach } v \mid \text{start at } u].$$
- **Commute time:** The expected number of steps to go from u to v and back:
$$C_{uv} = H_{uv} + H_{vu}.$$
- **Cover time:** The expected number of steps to visit all vertices starting from u:
$$C_u = \mathbb{E}[\#\text{ of steps to visit all vertices} \mid \text{start at } u], \quad C_G = \max_u C_u.$$

Application: Undirected s-t Connectivity (USTConn) - For an undirected graph $G(V, E)$ and two nodes $s, t \in V$, the problem is whether s and t are connected. This problem is easily solvable in $O(m+n)$ time by using DFS or BFS. But for extremley large graphs, a space-efficient solution is needed.

With randomness and random walks, the problem can solved in $O(\log n)$ space, that is, in randomized log-space. Thus, USTConn $\in$ RL. In a breakthrough result, Reingold (2005) showed that actually this can in fact be done deterministically with the same space complexity, proving that USTConn $\in$ L.

10.2.1 Analysis of Random Walks Using Resistance Graphs

We can analyze random walks by thinking of the graph G as an electrical circuit. In this case, every node of the circuit is a vertex in G, and every edge has a resistance of 1 unit.

Using the fundamental concepts of electrical circuits. Kirchhoff's Current Law states that current coming into a node equals to current leaving out of the node. Ohm's Law relates voltage V, resistance R, and current I through the equation
$$V = RI$$

For resistors in series and parallel connections, the effective resistance is determined as follows:

$$\text{Series:} \quad R_{\text{total}} = R_1 + R_2,$$
$$\text{Parallel:} \quad \frac{1}{R_{\text{total}}} = \frac{1}{R_1} + \frac{1}{R_2}, \quad \text{which gives} \quad R_{\text{total}} = \frac{R_1 R_2}{R_1 + R_2}.$$

Definition: The effective resistance between two nodes u and v is the potential difference (i.e., voltage) between u and v when one unit of flow is sent from u to v.

Theorem 2 *Let G be a graph and let each edge be a unit resistor. Let R_{uv} be the effective resistance between vertices u and v. Then the commute time between u and v is*
$$C_{uv} = 2mR_{uv},$$
where m is the number of edges in G.

Proof. Suppose we connect a battery that injects $\deg(x)$ units of current into each node x, such that there is $2m$ units of flow out of v.

Let Φ_{uv} be the voltage difference between u and v.

We have:

$$\deg(u) = \sum_{ux \in E} (\text{current from } u \text{ to } x).$$

Each current term can be expressed as:

$$\sum_{ux \in E} \Phi_{ux} = \sum_{ux \in E} (\Phi_{uv} - \Phi_{xv}) = \deg(u)\Phi_{uv} - \sum_{ux \in E} \Phi_{xv}.$$

Thus,

$$\Phi_{uv} = 1 + \sum_{ux \in E} \frac{\Phi_{xv}}{\deg(u)}. \tag{1}$$

Let $\deg(u)$ denote the degree of node u, and consider the first step of a random walk. Then:

$$H_{uv} = 1 + \sum_{ux \in E} \frac{H_{xv}}{\deg(u)}. \tag{2}$$

Notice that equations (1) and (2) are the same linear equations for all nodes, which implies

$$H_{uv} = \Phi_{uv}.$$

Performing the same operation in reverse – injecting from every node and getting $2m$ from u gives

$$H_{vu} = \Phi'_{vu}.$$

In this second experiment, the flow is reversed. So, $2m$ units of current flow into v, and $\deg(x)$ units flow out of all nodes.

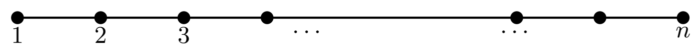
Combining the two:

- $2m$ units going into u
- $2m$ units going out of v
- In/out flow from every other node = 0

$$\Rightarrow \Phi_{uv} + \Phi'_{vu} = H_{uv} + H_{vu} = 2m \cdot R_{uv} = C_{uv}.$$

■

Example 1: Random walk on a path P_n Consider a path graph with n vertices labeled $1, 2, \ldots, n$.



In this simple graph, the effective resistance between vertices i and j (where $i < j$) is simply the number of edges between them:

$$R_{ij} = j - i.$$

Using the relationship $C_{ij} = 2mR_{ij}$ where $m = n - 1$ edges, we obtain the commute time:

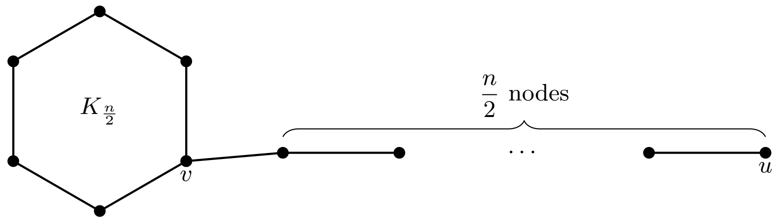
$$C_{ij} = H_{ij} + H_{ji} = 2(n-1)(j-i).$$

For the endpoints of the path:

$$C_{1n} = H_{1n} + H_{n1} = 2(n-1)^2.$$

This shows that the commute time grows quadratically with the length of the path.

Example 2: Lollipop graph L_n The lollipop graph consists of a complete graph $K_{n/2}$ (the "head") connected to a path of $n/2$ vertices (the "stick"). This graph demonstrates asymmetry in hitting times.



Let u be the endpoint of the path (away from the clique) and v be the vertex at which the path connects to the clique. The effective resistance between u and v is:

$$R_{uv} = \frac{n}{2}.$$

The total number of edges in the lollipop graph is the sum of edges in the complete graph and the path:

$$m = \binom{n/2}{2} + \frac{n}{2} \in \Theta(n^2).$$

The commute time is:

$$C_{uv} = H_{uv} + H_{vu} = 2mR_{uv} \in \Theta(n^3).$$

But, the hitting times are very asymmetric. Starting from the clique it takes:

$$H_{vu} = \left(\frac{n}{2}\right)^2 \in \Theta(n^2).$$

to walk to the end of the path takes: This means that the other direction takes a lot longer:

$$H_{uv} = C_{uv} - H_{vu} \in \Theta(n^3).$$

This asymmetry shows that $H_{uv} \neq H_{vu}$ in general. Walking from the narrow path into the highly connected clique is much harder than the other direction, as the random walk tends to get "lost" in the clique before finding the specific vertex v.

Theorem 3 *For any connected graph G with n vertices and m edges, the cover time satisfies*

$$C(G) \leq 2m(n-1).$$

Proof. Let T be a spanning tree of G and perform a depth-first search (DFS) traversal starting from vertex u. The DFS visits each edge of T exactly twice (once in each direction), giving a walk:

$$u = v_0, v_1, v_2, \ldots, v_{2n-2}.$$

The cover time from u is bounded by the total time to traverse this DFS walk:

$$C_u \leq H_{v_0 v_1} + H_{v_1 v_2} + \cdots + H_{v_{2n-3} v_{2n-2}}.$$

Each edge in the spanning tree is traversed twice, so we can rewrite this as:

$$C_u \leq \sum_{vw \in T} (H_{vw} + H_{wv}) = \sum_{vw \in T} C_{vw} = \sum_{vw \in T} 2mR_{vw}.$$

Since each edge in the tree has resistance $R_{vw} \leq 1$ (as it's a single edge), and the tree has $n-1$ edges:

$$C_u \leq 2m(n-1).$$

Since this holds for any starting vertex u, we have $C(G) = \max_u C_u \leq 2m(n-1)$. ∎

Examples: This general bound can be tightened for specific graph families:

- Path graph: For a path with n vertices and $m = n-1$ edges, the cover time is $C_G \in O(n^2)$.
- Complete graph K_n: For the complete graph with $m = \binom{n}{2} \in \Theta(n^2)$ edges, the cover time is $C_G \in O(n^3)$. This is the worst-case scenario among common graph families.

References

[1] Marek Biskup, *Random Walks and Random Graphs*, PCMI Lecture Notes. Available at: `https://www.math.ucla.edu/`

[2] László Lovász, *Random Walks on Graphs: A Survey*. Available at: `https://www.cs.cmu.edu/~15859n/RelatedWork/r`