Randomised Algorithms

Lecture 5: Random Walks, Hitting Times and Application to 2-SAT

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025

UNIVERSITY OF CAMBRIDGE

Outline

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

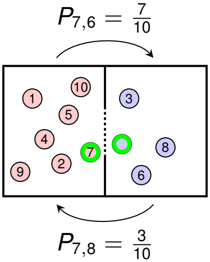
SAT and a Randomised Algorithm for 2-SAT

The Ehrenfest Markov Chain

Ehrenfest Model

- A simple model for the exchange of molecules between two boxes
- We have d particles labelled $1, 2, \ldots, d$
- At each step a particle is selected uniformly at random and switches to the other box
- If $\Omega = \{0, 1, \ldots, d\}$ denotes the number of particles in the red box, then:

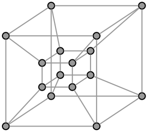
$$P_{x,x-1} = \frac{x}{d} \quad \text{and} \quad P_{x,x+1} = \frac{d-x}{d}.$$



Let us now enlarge the state space by looking at each particle individually!

Random Walk on the Hypercube

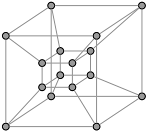
- For each particle an indicator variable $\Rightarrow \Omega = \{0, 1\}^d$
- At each step: pick a random coordinate in $[d]$ and flip it



Analysis of the Mixing Time

(Non-Lazy) Random Walk on the Hypercube

- For each particle an indicator variable $\Rightarrow \Omega = \{0,1\}^d$
- At each step: pick a random coordinate in $[d]$ and flip it



Problem: This Markov Chain is periodic, as the number of ones always switches between odd to even!

Solution: Add self-loops to break periodic behaviour!

Lazy Random Walk (1st Version)

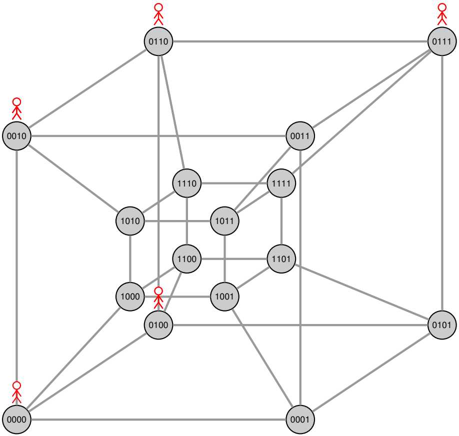
- At each step $t = 0, 1, 2 \ldots$
  - Pick a random coordinate in $[d]$
  - With prob. $1/2$ flip coordinate.

Lazy Random Walk (2nd Version)

- At each step $t = 0, 1, 2 \ldots$
  - Pick a random coordinate in $[d]$
  - Set coordinate to $\{0,1\}$ uniformly.

These two chains are equivalent!

Example of a Random Walk on a 4-Dimensional Hypercube



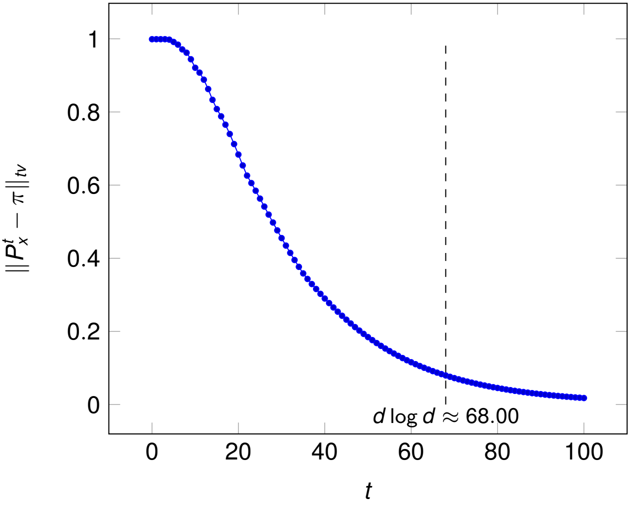
t	Coord.	X_t			
0	2	0	0	0	0
1	3	0	1	0	0
2	3	0	1	0	0
3	4	0	1	1	0
4	2	0	1	1	1
5	4	0	1	1	1
6	2	0	1	1	0
7	4	0	0	1	0
8	3	0	0	1	0
9	1	0	0	1	0
10	done!	0	0	1	0

Once all coordinates have been picked at least once, the state is uniformly at random in $\{0,1\}^d$.

Coupon Collector $\rightsquigarrow$ mixing time should be $O(d \log d)$

We won't formalise this argument here (see *[Ex. 4/5.11]*)

Total Variation Distance of Random Walk on Hypercube ($d = 22$)



Theoretical Results (by Diaconis, Graham and Morrison)

RANDOM WALK ON A HYPERCUBE 53

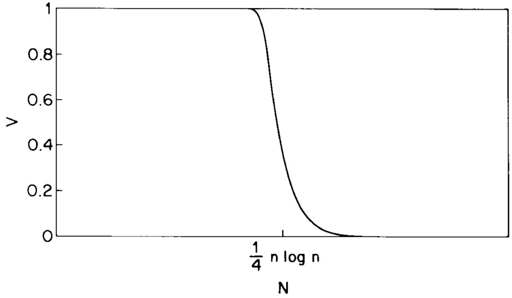


Fig. 1. The variation distance V as a function of N, for $n = 10^{12}$.

Source: "Asymptotic analysis of a random walk on a hypercube with many dimensions", P. Diaconis, R.L. Graham, J.A. Morrison; Random Structures & Algorithms, 1990.

- This is a numerical plot of a theoretical bound, where $d = 10^{12}$
  (Minor Remark: This random walk is with a loop probability of $1/(d+1)$)
- The variation distance exhibits a so-called cut-off phenomena:
  - Distance remains close to its maximum value 1 until step $\frac{1}{4} n \log n - \Theta(n)$
  - Then distance moves close to 0 before step $\frac{1}{4} n \log n + \Theta(n)$

Outline

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT

Stopping and Hitting Times

A non-negative integer random variable τ is a stopping time for $(X_t)_{t \geq 0}$ if for every $s \geq 0$ the event $\{\tau = s\}$ depends only on $X_0, \ldots, X_s$.

Example - College Carbs Stopping times:

✓ "We had rice yesterday" $\rightsquigarrow$ $\tau := \min\{t \geq 1 \colon X_{t-1} = \text{"rice"}\}$

× "We are having pasta next Thursday"

For two states $x, y \in \Omega$ we call $h(x, y)$ the hitting time of y from x:

$$h(x,y) := \mathbf{E}_x[\tau_y] = \mathbf{E}\left[\tau_y \mid X_0 = x\right] \quad \text{where } \tau_y = \min\{t \geq 1 : X_t = y\}.$$

Some distinguish between $\tau_y^+ = \min\{t \geq 1 \colon X_t = y\}$ and $\tau_y = \min\{t \geq 0 \colon X_t = y\}$

A Useful Identity

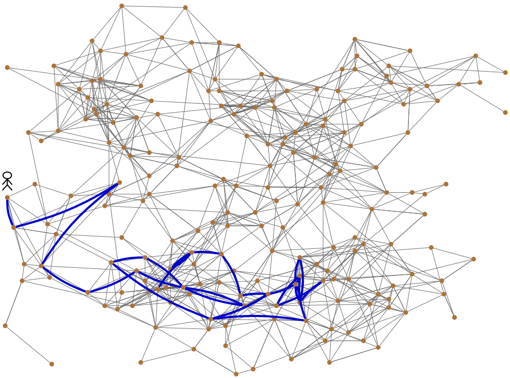
Hitting times are the solution to a set of linear equations:

$$h(x,y) \overset{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x,z) \cdot h(z,y) \qquad \forall x, y \in \Omega.$$

Random Walks on Graphs

A Simple Random Walk (SRW) on a graph G is a Markov chain on $V(G)$ with

$$P(u,v) = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u,v\} \in E, \\ 0 & \text{if } \{u,v\} \notin E. \end{cases}, \qquad \text{and} \qquad \pi(u) = \frac{\deg(u)}{2|E|}$$



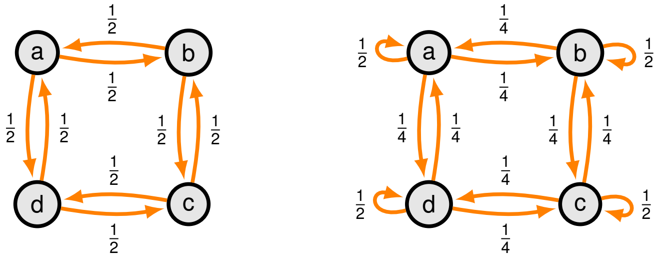
Lazy Random Walks and Periodicity

The Lazy Random Walk (LRW) on G given by $\widetilde{P} = (P + I)/2$,

$$\widetilde{P}_{u,v} = \begin{cases} \frac{1}{2\deg(u)} & \text{if } \{u,v\} \in E, \\ \frac{1}{2} & \text{if } u = v, \\ 0 & \text{otherwise.} \end{cases}$$

P - SRW matrix
I - Identity matrix.

Fact: For any graph G the LRW on G is aperiodic.



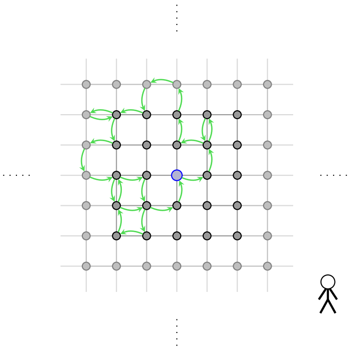
SRW on C_4, *Periodic* LRW on C_4, *Aperiodic*

Outline

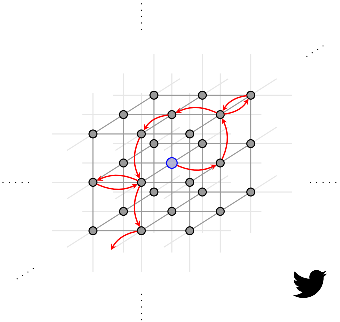
1921: The Birth of Random Walks on (Infinite) Graphs (Polyá)

Will a random walk always return to the origin?

Infinite 2D Grid



Infinite 3D Grid



“A drunk man will find his way home, but a drunk bird may get lost forever.”

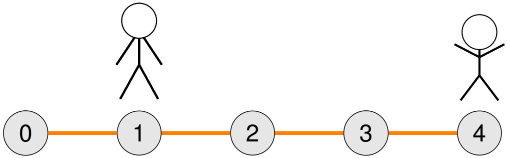
But for any regular (finite) graph, the expected return time to u is $1/\pi(u) = n$

SRW Random Walk on Two-Dimensional Grids: Animation

For animation, see full slides.

Random Walk on a Path (1/2)

The n-path P_n is the graph with $V(P_n) = [0, n]$, $E(P_n) = \{\{i, j\} : j = i + 1\}$.



Proposition

For the SRW on P_n we have $h(k, n) = n^2 - k^2$, for any $0 \leq k < n$.

Exercise: *[Exercise 4/5.15]* What happens for the LRW on P_n?

Random Walk on a Path (2/2)

Proposition

For the SRW on P_n we have $h(k,n) = n^2 - k^2$, for any $0 \leq k < n$.

Recall: Hitting times are the solution to the set of linear equations:

$$h(x,y) \overset{\text{Markov Prop.}}{=} 1 + \sum_{z \in \Omega \setminus \{y\}} P(x,z) \cdot h(z,y) \qquad \forall x,y \in V.$$

Proof: Let $f(k) = h(k,n)$ and set $f(n) := 0$. By the Markov property

$$f(0) = 1 + f(1) \quad \text{and} \quad f(k) = 1 + \frac{f(k-1)}{2} + \frac{f(k+1)}{2} \quad \text{for } 1 \leq k \leq n-1.$$

System of n independent equations in n unknowns, so has a unique solution.

Thus it suffices to check that $f(k) = n^2 - k^2$ satisfies the above. Indeed

$$f(0) = 1 + f(1) = 1 + n^2 - 1^2 = n^2,$$

and for any $1 \leq k \leq n-1$ we have,

$$f(k) = 1 + \frac{n^2 - (k-1)^2}{2} + \frac{n^2 - (k+1)^2}{2} = n^2 - k^2. \qquad \square$$

Outline

Application 3: Ehrenfest Chain and Hypercubes

Random Walks on Graphs, Hitting Times and Cover Times

Random Walks on Paths and Grids

SAT and a Randomised Algorithm for 2-SAT

SAT Problems

A Satisfiability (SAT) formula is a logical expression that's the conjunction (AND) of a set of Clauses, where a clause is the disjunction (OR) of Literals.

A Solution to a SAT formula is an assignment of the variables to the values `True` and `False` so that all the clauses are satisfied.

Example:

$$\text{SAT: } (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_4 \vee \overline{x_3}) \wedge (x_4 \vee \overline{x_1})$$

$$\text{Solution: } x_1 = \texttt{True}, \quad x_2 = \texttt{False}, \quad x_3 = \texttt{False} \quad \text{and} \quad x_4 = \texttt{True}.$$

- If each clause has k literals we call the problem k-SAT; n is the number of variables.
- In general, determining if a SAT formula has a solution is NP-hard
- A huge amount of problems can be posed as a SAT:
  - $\rightarrow$ Model checking and hardware/software verification
  - $\rightarrow$ Design of experiments
  - $\rightarrow$ Classical planning
  - $\rightarrow$ ...

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

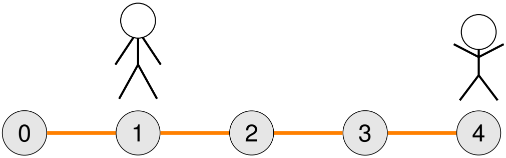
1: Start with an arbitrary truth assignment
2: **Repeat up to** $2n^2$ **times**
3: Pick an arbitrary unsatisfied clause
4: Choose a random literal and switch its value
5: **If** formula is satisfied **then return** "Satisfiable"
6: **return** "Unsatisfiable"

- Call each loop of (2) a step. Let A_i be the variable assignment at step i.
- Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 : Solution Found

$$\begin{array}{cccccccccccccc} (x_1 & \vee & \overline{x_2}) & \wedge & (\overline{x_1} & \vee & \overline{x_3}) & \wedge & (x_1 & \vee & x_2) & \wedge & (x_4 & \vee & \overline{x_3}) & \wedge & (x_4 & \vee & \overline{x_1}) \\ \mathtt{T} & & \mathtt{F} & & \mathtt{F} & & \mathtt{T} & & \mathtt{T} & & \mathtt{T} & & \mathtt{T} & & \mathtt{T} & & \mathtt{T} & & \mathtt{F} \end{array}$$

$$\alpha = (\mathtt{T}, \mathtt{T}, \mathtt{F}, \mathtt{T}).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	T	F	F
2	T	T	F	F
3	T	T	F	T

2-SAT

RANDOMISED-2-SAT (Input: a 2-SAT-Formula)

1: Start with an arbitrary truth assignment
2: **Repeat up to** $2n^2$ **times**
3: Pick an arbitrary unsatisfied clause
4: Choose a random literal and switch its value
5: **If** formula is satisfied **then return** "Satisfiable"
6: **return** "Unsatisfiable"

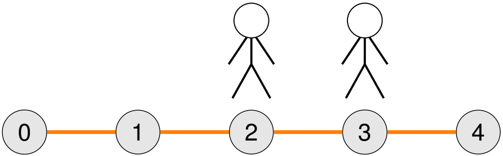
- Call each loop of (2) a step. Let A_i be the variable assignment at step i.
- Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 : (Another) Solution Found

$$(x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor \overline{x_3}) \land (x_1 \lor x_2) \land (x_4 \lor x_3) \land (x_4 \lor \overline{x_1})$$

T F F T T T T F T F

$$\alpha = (\mathtt{T}, \mathtt{F}, \mathtt{F}, \mathtt{T}).$$



t	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T
2	F	T	F	T
3	T	T	F	T

2-SAT and the SRW on the Path

Expected iterations of (2) in RANDOMISED-2-SAT

If the formula is satisfiable, then the expected number of steps before RANDOMISED-2-SAT outputs a valid solution is at most n^2.

Proof: Fix any solution α, then for any $i \geq 0$ and $1 \leq k \leq n-1$,

(i) $\mathbf{P}[X_{i+1} = 1 \mid X_i = 0] = 1$

(ii) $\mathbf{P}[X_{i+1} = k+1 \mid X_i = k] \geq 1/2$

(iii) $\mathbf{P}[X_{i+1} = k-1 \mid X_i = k] \leq 1/2$.

Notice that if $X_i = n$ then $A_i = \alpha$ thus solution found (may find another first).

Assume (pessimistically) that $X_0 = 0$ (none of our initial guesses is right).

The process X_i is complicated to describe in full; however by $(i) - (iii)$ we can **bound** it by Y_i (SRW on the n-path from 0). This gives (see also *[Ex 4/5.17]*)

$$\mathbf{E}[\text{time to find sol}] \leq \mathbf{E}_0[\min\{t : X_t = n\}] \leq \mathbf{E}_0[\min\{t : Y_t = n\}] = h(0, n) = n^2.$$

□

Running for $2n^2$ steps and using Markov's inequality yields:

Proposition

If the formula is satisfiable, RANDOMISED-2-SAT will return a valid solution in $O(n^2)$ steps with probability at least $1/2$.

Boosting Success Probabilities

Boosting Lemma

Suppose a randomised algorithm succeeds with probability (at least) p. Then for any $C \geq 1$, $\lceil \frac{C}{p} \cdot \log n \rceil$ repetitions are sufficient to succeed (in at least one repetition) with probability at least $1 - n^{-C}$.

Proof: Recall that $1 - p \leq e^{-p}$ for all real p. Let $t = \lceil \frac{C}{p} \log n \rceil$ and observe

$$\begin{aligned} \mathbf{P}[\, t \text{ runs all fail} \,] &\leq (1-p)^t \\ &\leq e^{-pt} \\ &\leq n^{-C}, \end{aligned}$$

thus the probability one of the runs succeeds is at least $1 - n^{-C}$. $\square$

Randomised-2-SAT

There is a $O(n^2 \log n)$-step algorithm for 2-SAT which succeeds w.h.p.