Randomised Algorithms

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

Thomas Sauerwald (tms41@cam.ac.uk)
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 ⇒ Ω = \{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

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

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

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

<table>
<thead>
<tr>
<th>( t )</th>
<th>Coord.</th>
<th>( X_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>10</td>
<td>done!</td>
<td>0 0 1 0</td>
</tr>
</tbody>
</table>
Total Variation Distance of Random Walk on Hypercube \((d = 22)\)

\[ \|P_x^t - \pi\|_tv \]

\[ d \log d \approx 68.00 \]
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
A Simple Random Walk (SRW) on a graph $G$ is a Markov chain on $V(G)$ with

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

Recall: $h(u, v) = \mathbb{E}_u[\min\{t \geq 1 : X_t = v\}]$ is the hitting time of $v$ from $u$. 
The Lazy Random Walk (LRW) on $G$ given by $\tilde{P} = (P + I) / 2$, 

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

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

SRW on $C_4$, Periodic

LRW on $C_4$, Aperiodic
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
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 \( \frac{1}{\pi(u)} = n \)
For animation, see full slides.
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 $P_n$?
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:

$$h(x, y) = 1 + \sum_{z \in \Omega \setminus \{y\}} P(x, z) \cdot h(z, y) \quad \forall x \neq y \in V.$$  

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,

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

\[\square\]
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
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:

SAT: \((x_1 \lor \overline{x}_2 \lor \overline{x}_3) \land (x_1 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_4 \lor \overline{x}_3) \land (x_4 \lor \overline{x}_1)\)

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

- If each clause has \(k\) literals we call the problem \(k\)-SAT.
- In general, determining if a SAT formula has a solution is NP-hard.
- In practice solvers are fast and used to great effect.
- A huge amount of problems can be posed as a SAT:
  - Model checking and hardware/software verification
  - Design of experiments
  - Classical planning
  - …
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 $\alpha$ be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

**Example 1 : Solution Found**

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

\[
\begin{array}{ccccc}
  t & x_1 & x_2 & x_3 & x_4 \\
  0 & F & F & F & F \\
  1 & F & T & F & F \\
  2 & T & T & F & F \\
  3 & T & T & F & T \\
\end{array}
\]

$\alpha = (T, T, F, T)$. 

\[
\begin{array}{cccc}
  0 & 1 & 2 & 3 \\
\end{array}
\]

0 1 2 3 4
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 \(\alpha\) be any solution and \(X_i = |\text{variable values shared by } A_i \text{ and } \alpha|\).

**Example 2:** (Another) Solution Found

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

<table>
<thead>
<tr>
<th>(t)</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

\(\alpha = (T, F, F, T)\).
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 $\alpha$, then for any $i \geq 0$ and $1 \leq k \leq n - 1$,

(i) $\Pr[X_{i+1} = 1 | X_i = 0] = 1$

(ii) $\Pr[X_{i+1} = k + 1 | X_i = k] \geq 1/2$

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

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 4/5.16])

$E[\text{time to find sol}] \leq E_0[\min\{t : X_t = n\}] \leq E_0[\min\{t : Y_t = n\}] = h(0, n) = n^2$.

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

Proposition

Provided a solution exists, 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

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

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

Randomised-2-SAT

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