Lecture 7: Random Walks & SAT

Nicolás Rivera   John Sylvester   Luca Zanetti   Thomas Sauerwald

Lent 2019

UNIVERSITY OF CAMBRIDGE

Outline

Random Walks and Reversibility

k-Sat

A Polytime Algorithm for 2-Sat

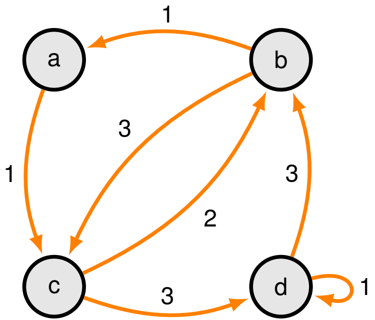
Schöning's Algorithm for 3-Sat

Random Walks on Weighted Graphs

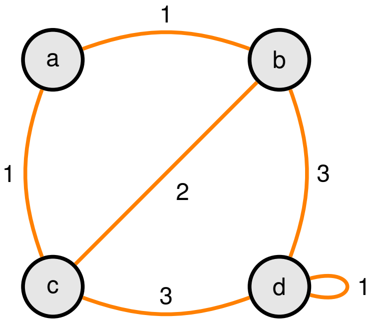
An *(edge) weighted* graph $G = (V, E, w)$ where $w : E \to \mathbb{R}_+$ on the edges.

A *Simple Random Walk (SRW)* on a weighted graph G is a MC on $V(G)$ with

$$P(i,j) = \begin{cases} \frac{w(ij)}{\sum_{xy \in E} w(xy)} & \text{if } ij \in E \\ 0 & \text{if } ij \notin E \end{cases}.$$



Directed



Undirected

Reversible Markov chains

- Any Markov chain can be described as random walk on a weighted directed graph.

Definition

A Markov chain on $\mathcal{I}$ with transition matrix P and stationary distribution π is called reversible if, for any $x, y \in \mathcal{I}$,

$$\pi(x)P(x,y) = \pi(y)P(y,x)$$

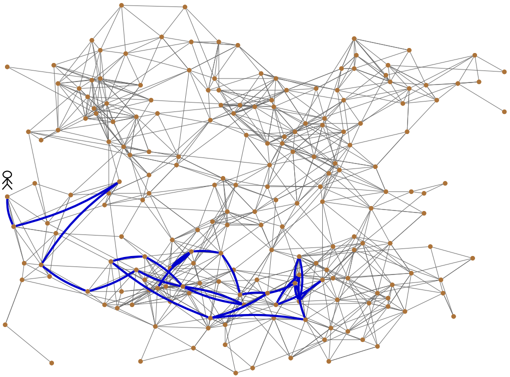
- Reversible Markov chains are equivalent to random walks on weighted undirected graphs.
- A reversible Markov Chain identified with the (undirected) weighted graph $G = (V, E, w)$ has stationary distribution given by

$$\pi(i) = \frac{\sum_{j:ij \in E} w(ij)}{2\sum_{xy \in E} w(xy)}$$

Random Walks on Graphs

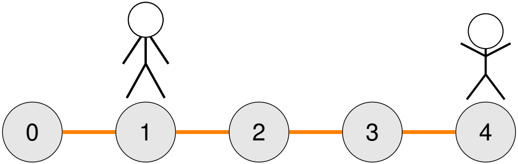
A *Simple Random Walk (SRW)* on a graph G is a Markov chain on $V(G)$ with

$$P(i,j) = \begin{cases} \frac{1}{d(i)} & \text{if } ij \in E \\ 0 & \text{if } ij \notin E \end{cases}, \qquad \text{and} \qquad \pi(i) = \frac{d(i)}{2|E|}$$



Random Walk on a path

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



Proposition

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

Random Walk on a path

Proposition

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

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

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

Proof: Let $f(k) = h(k, n)$ and observe that $f(n) = 0$. 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

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

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

Random Walks and Reversibility

k-Sat

A Polytime Algorithm for 2-Sat

Schöning's Algorithm for 3-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:

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

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

- If each clause has k literals we call the problem *k-SAT*.
- 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:
  - $\rightarrow$ Model Checking and hardware/software verification
  - $\rightarrow$ Design of experiments
  - $\rightarrow$ Classical planning
  - $\rightarrow$ ...

Outline

Random Walks and Reversibility

k-Sat

A Polytime Algorithm for 2-Sat

Schöning's Algorithm for 3-Sat

RAND 2-SAT Algorithm

(1) Start with an arbitrary truth assignment.

(2) Repeat up to $2n^2$ times, terminating if all clauses are satisfied:

  (a) Choose an arbitrary clause that is not satisfied

  (b) Choose one of it's literals UAR and switch the variables value.

(3) If a valid solution is found `return` it. O/W `return` unsatisfiable

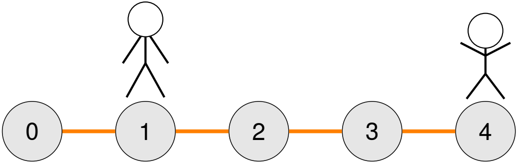
- Call each loop of (2) a *Step*. Let A_i be the variable assignment at step i.
- Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 1 : Solution Found

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

T F F T T T T T T F

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



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

RAND 2-SAT Algorithm

(1) Start with an arbitrary truth assignment.

(2) Repeat up to $2n^2$ times, terminating if all clauses are satisfied:

  (a) Choose an arbitrary clause that is not satisfied

  (b) Choose one of it's literals UAR and switch the variables value.

(3) If a valid solution is found `return` it. O/W `return` unsatisfiable

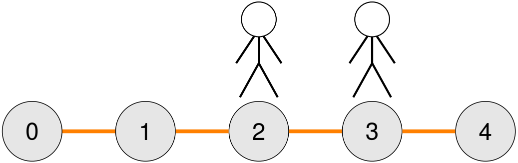
- Call each loop of (2) a *Step*. Let A_i be the variable assignment at step i.
- Let α be any solution and $X_i = |\text{variable values shared by } A_i \text{ and } \alpha|$.

Example 2 : Solution Found

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

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 RAND 2-SAT

If a valid solution exists then the expected number of iterations of loop (2) before RAND 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 (I may find another first).

Assume (pessimistically) that $X_0 = 0$ (we get non of our initial guesses right).

The stochastic process X_i is complicated to describe in full however by $(i) - (iii)$ we can bound it by Y_i- the SRW on the n-path from 0. This gives

$$\mathbf{E}[\,\text{time to find } \alpha\,] \leq \mathbf{E}_0[\inf\{t : X_t = n\}] \leq \mathbf{E}_0[\inf\{t : Y_t = n\}] = h_{0,n} = n^2. \quad \square$$

Proposition

Provided a solution exists the RAND 2-SAT Algorithm will return a valid solution in $O(n^2)$ time with probability at least $1/2$.

Boosting Success Probabilities

Boosting Lemma

Suppose a randomized algorithm succeeds with probability p and let $C \geq 1$ be any integer. Then $\frac{C}{p} \cdot \log n$ repetitions of the algorithm 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 $t = \frac{C}{p} \log n$ and observe that

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

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

RAND2-SAT

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

Outline

Random Walks and Reversibility

k-Sat

A Polytime Algorithm for 2-Sat

Schöning's Algorithm for 3-Sat

3-SAT

Schöning's Algorithm

(1) Start with a random truth assignment.

(2) Repeat up to n times, terminating if all clauses are satisfied:

  (a) Choose an arbitrary clause that is not satisfied

  (b) Choose one of it's literals UAR and switch the variables value.

(3) If a valid solution is found `return` it. O/W `return` unsatisfiable

Theorem

Schöning's Algorithm succeeds with probability at least $(1/3)^{n/2}/2$

Since each repetition runs in $O(n)$ time the Boosting lemma gives:

Corollary

3-SAT can be solved in time $O\left(n \cdot \sqrt{3}^n \cdot \log n\right) = O(1.733^n)$ w.h.p.

- In home work you will do a refined analysis giving $O(1.3334^n)$
- Best known algorithm is randomised and runs in time $O(1.3007^n)$ w.h.p.

Schöning's Algorithm: Basic Analysis

Theorem

Schöning's Algorithm succeeds with probability at least $(1/3)^{n/2}/2$

Proof: Consider some arbitrary correct satisfying assignment α.

Let A be the event that the initial truth assignment x agrees with α on at least $n/2$ variables. Note that $\mathbf{P}[\,A\,] \geq 1/2$ by symmetry.

Now, every iteration of Step (2) has at least a $1/3$ chance of increasing the agreement with α by 1. Why?

Recall each clause has three literals and α satisfies all clauses. Thus, if a clause is unsatisfied one of its literals is not in agreement with α. You then pick and flip one of these three literals uniformly.

Thus

$$\mathbf{P}[\,\text{Success}\,] \geq \mathbf{P}[\,\text{Success}|A\,]\,\mathbf{P}[\,A\,] \geq \left(\frac{1}{3}\right)^{n/2} \cdot \frac{1}{2}.$$