CHAPTER 5

Random walk

In this lecture, we introduce the random walk. We will see an application of random walk theory to the analysis of a probabilistic method for solving instances of 2-SAT. After that, we will consider an application of the same algorithm to finding 3-colourings of 3-colourable graphs.

1. Random walks

A (discrete-time) **stochastic process** $(\mathbf{X}_t : t = 0, 1, 2, \ldots)$ is simply a sequence of random variables, indexed by a parameter that we will often refer to as "time".

Consider now a stochatic process taking values in $\{0, 1, \ldots, n\}$ that models the movement of a particle along the integers $\{0, 1, \ldots, n\}$. The position of this particle at time $t + 1$ depends only on the position of the particle at time t. In particular, in every time step the particle flips a coin to decide whether it will move a step to the left or to the right (of course, at the end points, the particle does not have a choice but to go in the only available direction). To make this more precise, for all $i = 1, 2, \ldots, n-1$ and $t = 0, 1, \ldots$

- $\mathbb{P}(\mathbf{X}_{t+1} = i - 1 \mid \mathbf{X}_t = i) = p$;
- $\mathbb{P}(\mathbf{X}_{t+1} = i + 1 \mid \mathbf{X}_t = i) = 1 - p$; while
- $\mathbb{P}(\mathbf{X}_{t+1} = 1 \mid \mathbf{X}_t = 0) = 1$; and
- $\mathbb{P}(\mathbf{X}_{t+1} = n - 1 \mid \mathbf{X}_t = n) = 1$.

Note that the above equations only specify the possible jumps, and the probability that they happen. The initial position $\mathbf{X}_0$ is often deterministic, but may also be chosen according to some distribution.

The stochastic process outlined above is called the **random walk** on the integers $\{0, 1, \ldots, n\}$. If $p = 1/2$, it is called the **symmetric random walk** for obvious reasons.

Remark. *The random walk is often called the drunkard's walk, due to the following interpretation. Late at night, a drunk person walks home from the pub. If he would walk in a straight line, it would take n steps to get home. However, walking in a straight line is hard when you are drunk, so every step that the drunk person takes is either randomly towards home (with probability p), or towards the pub (with probability $1 - p$). After t steps, the drunk person is $\mathbf{X}_t$ steps away from home, and this is a random walk with initial position $\mathbf{X}_0 = n$.*

There are many questions that you can ask about random walks. An important question (that we will not discuss in this lecture) is: as the amount of steps tends to infinity, what is the fraction of time that the random walk spends in each state? (Basically, this is asking about the so-called stationary distribution of the random walk.)

In this lecture, we will answer a different question: starting from n, what is the expected number of steps before the random walk reaches 0?

Lemma 5.1. *Let $(\mathbf{X}_t : t = 0, 1, \ldots)$ be the symmetric random walk on the integers $\{0, 1, \ldots, n\}$ starting at $\mathbf{X}_0 = n$, and let $\mathbf{T}$ be the (random!) number of steps before the random walk reaches 0. Then $\mathbb{E}[\mathbf{T}] = n^2$.*

Proof. Let E_j be the expected number of steps before reaching 0 when the random walk is currently at j. Then $\mathbb{E}[\mathbf{T}] = E_n$, and $E_0 = 0$. (Note that E_j implicitly depends on n.) By conditioning on the first step, we can derive a system of linear equations that the quantities E_j must satisfy: starting from $j \neq 0, n$, we must take at least one step, and we end up at $j - 1$

or $j+1$, with equal probability. Therefore, the expected number of steps after the first step, is either E_{j-1} or E_{j+1}. We thus find

$$E_0 = 0 \tag{5.1}$$

$$E_j = 1 + \frac{1}{2}E_{j-1} + \frac{1}{2}E_{j+1} \qquad j = 1, 2, \ldots, n-1 \tag{5.2}$$

$$E_n = 1 + E_{n-1} \tag{5.3}$$

The solution of this system of equations is easily found to be $E_j = n^2 - (n-j)^2 = 2nj - n^2$. In particular, we find that $\mathbb{E}[\mathbf{T}] = E_n = n^2$. □

Remark. *The random walk on the integers that we defined in this section, is only one specific example of a random walk. In general, one can define random walks on the vertices of a (finite or infinite) graph, so that in each step the random walk randomly moves to a neighbour of the current vertex. If you want to learn more about random walks, there are many good books available. For the application of random walks to randomised algorithms, see for example [MR95, Chapter 6].*

2. 2-SAT

2.1. Satisfiability problems. A **boolean variable** is a variable that can take two values, TRUE and FALSE. If x is a boolean variable, then its negation $\bar{x}$ is TRUE if x is FALSE, and it is FALSE if x is TRUE. Two (or more) boolean variables (or their negations) can be combined using conjunctions ($\vee$, interpreted as "and") and disjunctions ($\wedge$, interpreted as "or") as follows:

- $x_1 \vee x_2$ is TRUE precisely when both x_1 and x_2 are TRUE;
- $x_1 \wedge x_2$ is TRUE precisely when at least one of x_1 and x_2 is true.

Using negation, conjunction, and disjunction, boolean variable can be combined to form **formulas**. A formula is said to be **satisfiable** if there exists an assignment of values to its variables, such that the formula evaluates to true.

Example. *$\phi_1(x_1, x_2, x_3) = (x_1 \wedge \bar{x}_2) \vee x_3$ is a formula on three variables that is satisfiable (for example $x_1 =$ TRUE, $x_2 =$ FALSE, $x_3 =$ FALSE satisfies ϕ_1). On the other hand $\phi_2(x) = x \wedge \bar{x}$ is a simple example of an unsatisfiable formula.*

We now introduce the **satisfiability problem** SAT.

Problem:	SAT
Input:	A Boolean formula ϕ
Output:	Yes, if ϕ is satisfiable, no otherwise.

SAT is an NP-complete problem. In this lecture, we will talk about a special case of the general satisfiabiliy problem that is much easier. We need a little more terminology to define this problem.

A **clause** is a disjunction of boolean variables (or their negations). A formula is said to be in **conjunctive normal form** if is a conjunction of clauses. It can be shown that every formula can be written in conjunctive normal form. Therefore, we shall assume that all the formulas that we work with are in conjunctive normal form.

Example. *The formula ϕ_1 from the previous example can be written in conjunctive normal form as $\phi_1(x_1, x_2, x_3) = (x_1 \vee x_3) \wedge (\bar{x}_2 \vee x_3)$.*

The problem k-SAT is the satisfiability problem restricted to formulas in which every clause contains at most k variables.

Problem:	k-SAT
Input:	A boolean formula ϕ in conjunctive normal form, in which every clause uses at most k variables.
Output:	Yes, if ϕ is satisfiable; no otherwise.

Note that in the input for k-SAT, the number of variables per clause is restricted, but the number of clauses is not. While k-SAT is NP-complete for all $k \geq 3$, the cases $k = 1$ and $k = 2$ are in P. In this lecture, we will consider a probabilistic algorithm for solving k-SAT instances.

2.2. A probabilistic algorithm for solving 2-SAT. In what follows, we assume that $\phi \equiv \phi(x_1, x_2, \dots, x_n)$ is a formula in conjunctive normal form, in which every clause contains at most two variables (or negations of variables). The following is a probabilistic algorithm for finding a satisfying assignment for ϕ.

(i) Pick an arbitrary assignment of TRUE and FALSE to the x_i. If our initial choice satisifes ϕ, then we are done.
(ii) Otherwise, pick an arbitrary unsatisfied clause. Choose one of the (at most) two variables occurring in it, and flip its value.
(iii) Repeat Step (ii) for at most $2n^2$ times, or until ϕ is satisfied.

Note that if ϕ is not satisfiable, the algorithm will never return an assignment. If, on the other hand, ϕ is satisfiable, the algorithm may return a satisying assignment, but does not necessarily do so. The curious appearance of $2n^2$ in the last line of the algorithm is to ensure that the probability of success is at least $1/2$ if ϕ is satisfiable.

THEOREM 5.2. *If ϕ is satisfiable, then with probability at least $1/2$, the algorithm finds a satisfying assignment.*

PROOF. Since we assume that ϕ is satisfiable, there is at least one satisfying assignment $y = (y_1, y_2, \dots, y_n)$ such that $\phi(y)$ evaluates to TRUE. Let $\mathbf{X}_0$ be the initial arbitrary assignment, and let $\mathbf{X}_t$ be the assignment after t steps. We will track the random variable $\mathbf{D}_t$, which gives the "distance" between the (random) assignment $\mathbf{X}_t$ and the satisfying assignment y, i.e. the number of entries in which $\mathbf{X}_t$ differs from y, or

$$\mathbf{D}_t = \# \{ i : \mathbf{X}_{t,i} \neq y_i \} . \tag{5.4}$$

($\mathbf{D}_t$ is known as the Hamming distance between $\mathbf{X}_t$ and y.) Note that $\mathbf{D}_t$ takes its values in $\{0, 1, \dots, n\}$. Clearly, if $\mathbf{D}_t = 0$, then $\mathbf{X}_t = y$, and the algorithm stops.[1] We let $\mathbf{S}$ denote the first time that $\mathbf{D}_t = 0$. Note that the algorithm finds a satisfying assignment precisely when $\mathbf{S} \leq 2n^2$. Suppose that we finished the t-th update, and that $\mathbf{D}_t = j > 0$. In the next update, exactly one variable changes its value, so that $\mathbf{D}_{t+1}$ becomes either $j - 1$ or $j + 1$.

It is easily verified that the probability of decreasing is always at least $1/2$. Therefore, the expected value of $\mathbf{S}$ is at most the expected number of steps of a random walk on the integers $\{0, 1, \dots, n\}$, starting at n, before reaching 0. It follows from Lemma 5.1 that $\mathbb{E}[\mathbf{S}] \leq n^2$. By Markov's inequality, we find that

$$\mathbb{P}\left(\mathbf{S} \leq 2n^2\right) \geq 1 - \frac{\mathbb{E}[\mathbf{S}]}{2n^2 + 1} \geq 1 - \frac{n^2}{2n^2 + 1} > \frac{1}{2}, \tag{5.5}$$

which shows that indeed the algorithm finds a satisfying assignment within $2n^2$ steps with probability at least $1/2$. □

REMARK. *Note that we can decrease the failure probability of the algorithm, by executing sufficiently many independent runs. In particular, if ϕ is a satisfiable formula, and we run the algorithm k times, the probability that none of these runs is successfull is at most $(1/2)^k$.*

[1] Note that y is not necessarily the only satisfying assignment, so the algorithm may in fact stop at an earlier moment.

3. Colouring 3-colourable graphs

3.1. Introduction. Let G be graph. Recall that G is called k-colourable, if we can assign (at most) k colours to its vertices, so that no two neighbouring vertices get the same colour.

If we know that G is 2-colourable (i.e. bipartite), then it is easy to find a 2-colouring by breadth-first search.

It is natural to try to generalise this to higher chromatic numbers. For example, if we know that G is 3-colourable, does this help us to actually find a 3-colouring? It is tempting to answer this question affirmative, e.g. we know that the neighbourhood of a given vertex must be 2-colourable, so it is easy to colour any neighbourhood using only two colours. Unfortunately, to date, the best known polynomial time algorithm [Chl07] still uses $O(n^{0.2072})$ colours! It is also known that finding a 4-colouring of G is NP-hard [KLS93].

We will show that colouring 3-colourable graphs using $O(\sqrt{n})$ colours is easy. For this, we need the following easy lemma, the proof of which is left as an exercise to the reader.

Lemma 5.3. *Let G be a graph on n vertices with maximum degree Δ, then G can be coloured using $\Delta + 1$ colours in polynomial (in n) time.*

Theorem 5.4 ([Wig83]). *If G is a 3-colourable graph, then it can be coloured using $O(\sqrt{n})$ colours in polynomial time.*

Proof. As long as G contains a vertex of degree at least $\sqrt{n}$, we colour its neighbourhood using two colours, and remove the vertex and its neighbourhood from G. We will never use the two colours that we used again. The number of vertices of degree at least $\sqrt{n}$ is at most $\sqrt{n}$, so this step uses at most $2\sqrt{n}+1$ colours. After this step, we are left with a graph in which each vertex has degree at most $\sqrt{n}$. By Lemma 5.3, the remaining vertices can be coloured using at most $\sqrt{n}+1$ colours in polynomial time, thus resulting in a vertex-colouring of G using at most $3\sqrt{n}+2$ colours. $\square$

3.2. Colouring dense 3-colourable graphs. A graph G on n vertices is called δ-dense if it has minimum degree at least δn. In this section, we show that a probabilistic polynomial time algorithm for 3-colouring a 3-colourable graph exists, provided that the graph is dense.

Theorem 5.5. *Let $\delta > 0$. There exists a probabilistic polynomial time algorithm for finding a 3-colouring of δ-dense 3-colourable graphs, that returns a valid colouring with probability at least $1/2$.*

In the proof of this result, we will use the following lemma, which can be proved using the probabilistic method with alterations. Recall that a dominating set is a subset of the vertices $S \subseteq V$, such that $S \cup N(S) = V$.

Lemma 5.6. *If G is a graph on n vertices and minimum degree d, then it contains a dominating set of size at most $\frac{\ln(d+1)+1}{d+1}n$.*

Proof of Theorem 5.5. Let us outline the procedure for finding a 3-colouring of G.

(i) Find a dominating set S of size $O(\log n)$.
(ii) Make a list of all 3-colourings of S.
(iii) For every 3-colouring of S, try to extend it to a 3-colouring of G.

If G is 3-colourable, then some 3-colouring of S can be extended to a 3-colouring of G.

<u>Finding a dominating set of logarithmic size:</u> Pick a random set $\mathbf{S}$ by adding each vertex to $\mathbf{S}$ with probability $p = \frac{2\ln n}{\delta n}$, independently of all other vertices. We will show that with probability at least $1 - 2n^{-0.2}$ the resulting set $\mathbf{S}$ is a dominating set of size at most $\frac{3\ln n}{\delta}$ (in which case we will call $\mathbf{S}$ good). This fails if either $\mathbf{S}$ is not a dominating set, or if it is too large. To start with the size of $\mathbf{S}$: note that $|\mathbf{S}|$ follows a binomial distribution, so is has to be concentrated around its mean, and we can use for example Chernoff's bound to bound the

probability that $\mathbf{S}$ is too large. In particular, we find that

$$\mathbb{P}\left(|\mathbf{S}| > \frac{3\ln n}{\delta}\right) \leq \left(\frac{\mathrm{e}^{0.5}}{1.5^{1.5}}\right)^{\frac{2\ln n}{\delta}} \leq n^{-0.2}. \tag{5.6}$$

We also need to show that $\mathbf{S}$ is dominating with high probability. In order to do this, note that if some vertex v is not dominated by $\mathbf{S}$, in particular it has no neighbour in the set $\mathbf{S}$. As v has at least δn neighbours, the latter event happens with probability at most

$$(1-p)^{\delta n} = \left(1 - \frac{2\ln n}{\delta n}\right)^{\delta n} \leq \mathrm{e}^{-2\ln n} = \frac{1}{n^2}. \tag{5.7}$$

Hence, by the union bound, the probability that at least one of the vertices is not covered by $\mathbf{S}$, it at most $n\frac{1}{n^2} = \frac{1}{n}$.

We conclude that the probability that $\mathbf{S}$ is not good with probability at most $n^{-0.2} + n^{-1} \leq 2n^{-0.2}$. In the remainder of the proof, we shall assume that we selected a good $\mathbf{S}$.

Enumerate the 3-colourings of $\mathbf{S}$: We will colour the vertices in $\mathbf{S}$ using the three colours in $\mathbb{Z}_3 = \mathbb{Z}/3\mathbb{Z}$ (the reason that we choose the colours in the additive group $\mathbb{Z}_3$ is that later it will be convenient to be able to sum colours). Note that the number of such 3-colourings is at most

$$3^{|\mathbf{S}|} \leq 3^{\frac{3\ln n}{\delta}} = n^{\frac{3\ln 3}{\delta}}, \tag{5.8}$$

which is polynomial in n.

Try to extend the 3-colourings to G: For each of the valid 3-colourings of the set $\mathbf{S}$, we will try to extend it to a valid colouring of G. In fact, we will reduce the problem of extending a valid 3-colouring of $\mathbf{S}$ to an instance of 2-SAT, which we will then solve using the probabilistic algorithm that we considered in the previous section.

So suppose that we have a valid 3-colouring of the vertices in $\mathbf{S}$ (using the elements of $\mathbb{Z}_3$). Note that for each vertex v outside $\mathbf{S}$ there are at most two colours available, as v has at least one neighbour in $\mathbf{S}$.

We can try to extend the colouring of $\mathbf{S}$ as much as possible: as long as there is a vertex v for which at most one colour is available it is clear what to do. If there is no colour available (i.e. v has three neighbours that use different colours), then the current colouring of $\mathbf{S}$ cannot be valid, so we move to the next colouring of $\mathbf{S}$. If v has only one colour available, we can extend the partial colouring to v.

After a while, we are left with a set U of vertices, each of which has two available colours.[2] For this part, we need to work a little harder. If v is currently uncoloured (i.e. $v \in U$), all its neighbours necessarily have the same colour, $b(v)$ say, that we cannot use for v. For each such vertex we introduce a boolean variable $x(v)$, that will be interpreted as follows:

$$\begin{aligned} x(v) = \mathsf{TRUE} \quad &\Rightarrow \quad \text{colour } v \text{ using } b(v)+1 \\ x(v) = \mathsf{FALSE} \quad &\Rightarrow \quad \text{colour } v \text{ using } b(v)+2 \end{aligned} \tag{5.9}$$

Any assignment of values to the $x(v)$ extends the partial colouring to a colouring of G, but we have to make sure that no two adjacent vertices in U get the same colour. Therefore, for each edge (v, v') that has both end points in U, we need to ensure that its end points do not get the same colour. It is easily verified that this can be modelled by at most two disjunctions involving $x(v)$ (or its negation) and $x(v')$ (or its negation). All these clauses should be satisfied simultaneously, so this gives a formula ϕ in conjunctive normal form in which every clause contains exactly two variables (or negations). The number of clauses is at most $2n$.

Now suppose that the current colouring of $\mathbf{S}$ can indeed by extended to a colouring of G. Then ϕ is satisfiable, and hence by Theorem 5.2 (and the remark following it), we can find a satisfying assignment for ϕ with success probability at least $3/4$. By (5.9), this assignment immediately translates to a valid colouring of G.

[2]It may be the case that $U = \emptyset$, in which case we are lucky.

Analysis of the success probability: We will show that the probability of failure is at most 1/2. There are only two ways that this algorithm can fail: either we fail to select a good dominating set **S** (which happens with probability at most $2n^{-0.2}$), or it fails to find a good extension of the colouring. Note that there is at least one colouring of **S** that can be extended to a colouring of G, so the only way that the latter failure can occur is if the probabilistic algorithm for solving the satisfiability problem fails (which happens with probability at most 1/4).

We conclude that the probability of failure is at most $2n^{-0.2} + 1/4 \leq 1/2$, for sufficiently large n. $\square$

Bibliography

[Chl07] Eden Chlamtac. "Approximation algorithms using hierarchies of semidefinite programming relaxations". In: 48th Annual IEEE Symposium on the Foundations of Computer Science IEEE. 2007, pp. 691–701.

[KLS93] Sanjeev Khanna, Nathan Linial, and Shmuel Safra. "On the hardness of approximating the chromatic number". In: Proceedings of the 2nd Israel Symposium on the Theory of Computi IEEE. 1993, pp. 250–260.

[MR95] Rajeev Motwani and Prabhakar Raghavan. Randomized Algorithms. Cambridge International Series on Parallel Computation. Cambridge University Press, 1995.

[Wig83] Avi Wigderson. "Improving the performance guarantee for approximate graph coloring". In: Journal of the ACM 30.4 (1983), pp. 729–735.