CS5314
Randomized Algorithms

Lecture 22: Markov Chains
(Solving 3SAT)

Objectives

- The 3SAT problem:

  Given a formula F, each clause with exactly 3 literals;

  Decide if F is satisfiable

- This lecture will discuss a randomized algorithm for 3SAT and make use of Markov Chains to analyze its performance

Application: Solving 3SAT

- Unlike the case with 2 literals (2SAT), 3SAT problem is NP-Complete!
- Let n = # variables in F
- We can solve this in $O(2^n)$ steps (of scanning the clauses) by brute force method

➔ Later, we show a faster randomized algo …

- Before that, let's see what happens if we re-use the previous 2SAT algorithm:

1. Start with an arbitrary assignment
2. Repeat m times, terminating with all clauses satisfied
   (a) Choose a clause that is currently not satisfied
   (b) Choose uniformly at random one of the literals in the clause and switch its value
3. If valid assignment found, return it
4. Else, conclude that F is unsatisfiable

Application: Solving 3SAT (3)

- Let us follow the same approach as before to investigate the performance of the algorithm ➔ Start by bounding the expected time to get a satisfying assignment when F is indeed satisfiable
- Let A^* = this particular assignment
- Also, let A_t = the assignment of variables after the t^{th} iteration of Step 2
- Let X_t = the number of variables that are assigned the same value in A^* and A_t

Application: Solving 3SAT (4)

- First, when $X_t = 0$, any change in the current assignment A_t must increase the # of matching assignment with A^* by 1

➔ $$\Pr(X_{t+1} = 1 \mid X_t = 0) = 1$$

- When $X_t = j$, with $1 \leq j \leq n-1$, we will choose a clause that is false with the current assignment A_t, and change the assignment of one of its variable next … the value of X_{t+1} can either be $j-1$ or $j+1$

Application: Solving 3SAT (5)

So, for j, with $1 \leq j \leq n-1$ we have

$$Pr(X_{t+1} = j+1 \mid X_t = j) \geq 1/3$$
$$Pr(X_{t+1} = j-1 \mid X_t = j) \leq 2/3$$

- The above equations follow since at least one variable from the selected clause are assigned differently in A^* and A_t
- Again, note that the stochastic process $X_0, X_1, X_2, \ldots$ is not a Markov chain

Application: Solving 3SAT (6)

- To simplify the analysis, we invent a true Markov chain $Y_0, Y_1, Y_2, \ldots$ as follows:

$$Y_0 = X_0$$
$$\Pr(Y_{t+1} = 1 \mid Y_t = 0) = 1$$
$$\Pr(Y_{t+1} = j+1 \mid Y_t = j) = 1/3$$
$$\Pr(Y_{t+1} = j-1 \mid Y_t = j) = 2/3$$

- When compared with the stochastic process $X_0, X_1, X_2, \ldots$, it takes more time for Y_t to increase to n … (why??)

Application: Solving 3SAT (7)

- Thus, the expected time to reach n from any point is larger for Markov chain Y than for the stochastic process X

So, we have

E[time for X to reach n starting at X_0]

$\leq$ E[time for Y to reach n starting at Y_0]

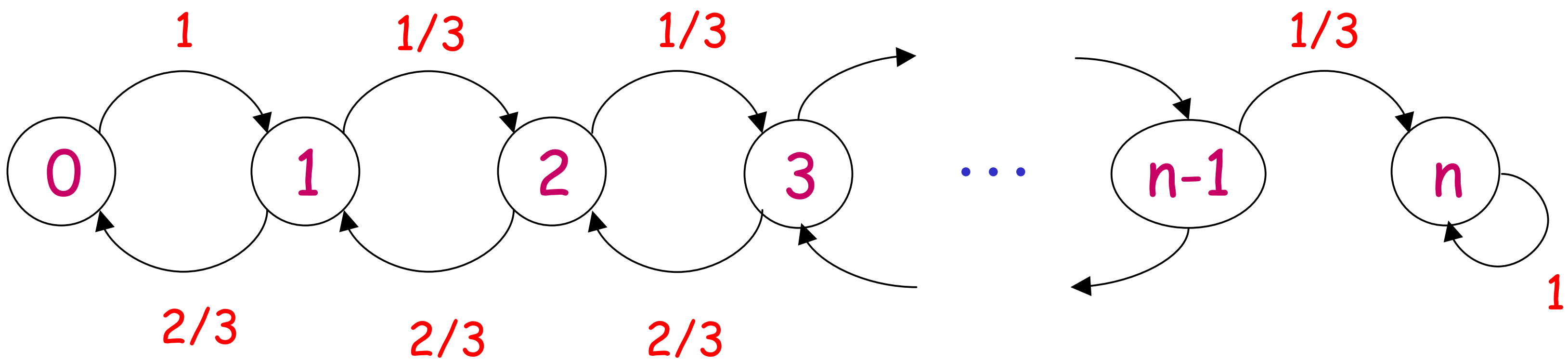
Question: Can we upper bound the term

E[time for Y to reach n starting at Y_0]?

Application: Solving 3SAT (8)

Let us take a look of how the Markov chain Y looks like in the graph representation

- Recall that vertices represent the state space, which are the values that any Y_t can take on:



Application: Solving 3SAT (9)

Let h_j = E[time to reach n starting at state j]

Clearly,

$$h_n = 0 \quad \text{and} \quad h_0 = h_1 + 1$$

Also, for other values of j, we have

$$h_j = (2/3)(h_{j-1} + 1) + (1/3)(h_{j+1} + 1)$$

Solving the above equations give:

$$h_j = 2^{n+2} - 2^{j+2} - 3(n-j)$$

Application: Solving 3SAT (10)

- On average, it takes $\Theta(2^n)$ steps to find a satisfying assignment [no good…]
- To improve the performance further, we have a key observation:

> Once the algorithm starts, it is more likely to move toward 0 than toward n. The longer we run the process, the more likely that it will move to 0

➔ Better if we restart the process after a small number of steps !

Modified Algorithm

1. Repeat m times, stop if all clauses satisfied
   (a) Choose an assignment uniformly at random
   (b) Repeat $3n$ times, stop if all clauses satisfied
      i. Choose a clause that is not satisfied
      ii. Choose one of the variables in the clause uniformly at random and switch its assigned value
3. If valid assignment found, return it
4. Else, conclude that F is unsatisfiable

Analysis of Modified Algorithm

- Let q = the probability that the process reaches A^* in $3n$ steps (Step 1(b)) when starting with a random assignment
- Let q_{-j} = the probability that the process reaches A^* in $3n$ steps when starting with a random assignment that has j variables assigned differently with A^*

  (I.e., still needs j changes to be A^*)
- In the following, we shall obtain a lower bound for q_{-j}, then for q

Bounding q_{-j}

Question: When we start at an assignment with j variables assigned differently with A^*, how can we obtain a satisfying assignment in $3n$ steps (or fewer)?

Ans. Let E_k be the event that we move $j+2k$ steps, in which exactly k steps are "moving down"

Then, we obtain a satisfying assignment if either one of the events, E_1, E_2, …, E_j, happen

Bounding q_{-j} (2)

Thus, we have

$$q_{-j} \geq \Pr(E_1 \cup E_2 \cup \ldots \cup E_j)$$
$$\geq \Pr(E_j)$$
$$= C(3j, j)\ (2/3)^j\ (1/3)^{2j}$$
$$\geq \Big((c/j^{0.5})\ (27/4)^j\Big)\ (2/3)^j\ (1/3)^{2j}$$

… [from Stirling, with c = some constant]

$$= (c/j^{0.5})\ (1/2)^j$$

Also, $q_0 = 1$

Bounding q

- Let event H_j = the starting assignment differs from A^* in exactly j variables
- Then, q (which is the probability that the process can reach A^* in 3n steps) can be bounded by:

$$q = \sum_{j=0 \text{ to } n} \Pr(H_j)\, q_{-j}$$

$$\geq (1/2)^n + \sum_{j=1 \text{ to } n} C(n,j)\,(1/2)^n\,(c/j^{0.5})(1/2)^j$$

$$\geq (c/n^{0.5})(1/2)^n \sum_{j=0 \text{ to } n} C(n,j)\,(1/2)^j$$

$$= (c/n^{0.5})(1/2)^n (3/2)^n = (c/n^{0.5})(3/4)^n$$

… Back to the Algorithm

- Now, we know that if F is satisfiable, then with probability $\geq (c/n^{0.5})(3/4)^n$, we obtain a satisfying assignment

- Thus, the expected number of restarts so that we obtain a satisfying assignment is at most $(n^{0.5}/c)(4/3)^n$, so that the expected number of steps to obtain a satisfying assignment is $O(n^{1.5}(4/3)^n)$
  ➔ much better than brute force $O(2^n)$!!

Application: Solving 3SAT (11)

- Based on the previous discussion, if we set $m = 2r\,(n^{0.5}/c)(4/3)^n$, then we can easily show the following theorem:

Theorem: The modified 3SAT algorithm answers correctly if the formula is unsatisfiable.

Otherwise, with probability $\geq 1 - 1/2^r$, it returns a satisfying assignment