Advanced Algorithms: Midterm Exam

Ivani Patel

October 28, 2022

Let C be a finite set of colors and G be a directed graph where v_0) is a designated start node. Each edge of G is labeled with a color in C and multiple edges can share the same color. From v_0), one may have infinitely many walks αsince there is no upper bound on the length of walks and there could be cycles in G. For each such walk α (that must start with v_0), we may collect the sequence c(α) of colors on the edges on the walk. We use P to denote the set of all walks α starting from v_0 and use C to denote all the resulting color sequences:

$$C = c(\alpha) : \alpha \in P.$$

Clearly, C may be an infinite set (of color sequences).

1. (10pts) Let u be a node in G. From this u, one may have multiple outgoing edges, say $\langle$u,$v_1\rangle$, $\langle$u,$v_2\rangle$, $\cdots$, $\langle$u,$v_k\rangle$, for some $k \geq 2$, whose colors are all the same. One can understand the same color as a triggering event that leads from current node u to a "next" node chosen nodeterministically from v_1,$\cdots$,v_k. Clearly, if I show you a walk αin P, then there could be many edges on the walk that are the result of many nondeterministic choices. Please develop and justify a metric M_1 which is a function of G that measures the nondeterminism on all walks in P. (Your M_1 is high when nondeterminism is high.) You need also show me an algorithm in computing such M_1. In case when efficient algorithm is hard to obtain, please also provide an approximation algorithm.

   (a) Given that there is a graph G in that graph each node has a color. We need to find a metric function of G that measures nondeterminism on all walks in P.

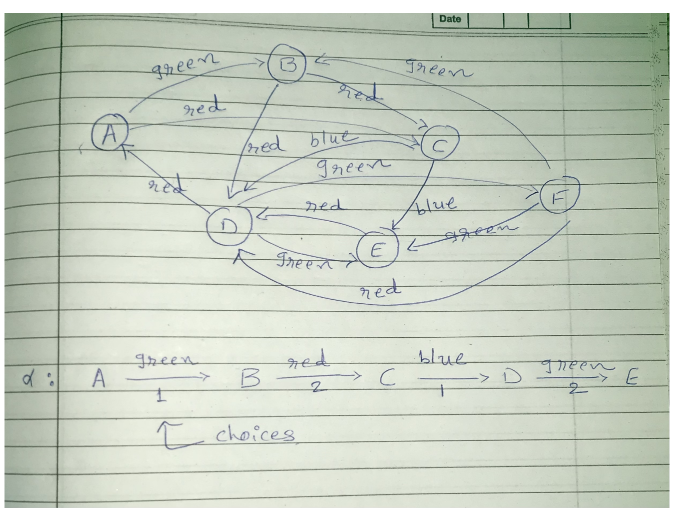

Figure 1: Graph G

Step 1: We take αas walk of graph with n nodes beginning from v_0 and characterize the number of nondeterministic choices on this walk $\alpha$is (α).

So we can say the nondeterminism N in walk a is

$$N = \frac{\log_2(\#\alpha)}{n}$$

Step 2: The set of all walks in graph G with length n is G_n, and total number of walks around G_n is $\#G_n$. We take A as adjacency matrix of G so we get

$$\#G_n = det(A^n)$$

Step 3:We separate the walks in graph G as nondeterministic walks and deterministic walks and modify graph G to G'. As every triggering event on each node holds different weight, we assume the weight on each node is equivalent to the number of walks on that node. The adjacency matrix of graph G' is A' and we get total number of walks on G' as

$$\#G'_n = det(A'^n)$$

Step 4: We get the nondeterminism M_1 on G as

$$M_1 = \lim_{n \to +\infty} (\frac{\log_2 \#G'_n}{n} - \frac{\log_2 \#G_n}{n})$$

Rewriting this

$$M_1 = \lim_{n \to +\infty} (\frac{\log_2 det(A'^n)}{n} - \frac{\log_2 det(A^n)}{n})$$

Step 5: We know that matrices A and A' are positive. So we use Perron-Frobenuis theorem to get the largest eigen value of A_n and A'_n when $n \to \infty$. So we get

$$\lambda = \lim_{n \to +\infty} (\frac{\log_2 det(A^n)}{n})$$

and

$$\lambda' = \lim_{n \to +\infty} (\frac{\log_2 det(A'^n)}{n})$$

Step 6: We get M_1 as

$$M_1 = \lambda' - \lambda$$

2. (10pts) Multiple walks can share the same color sequence. Hence, if we measure nondeterminism from the angle of color sequences in C, the metric would be very different. Color sequences in C may also have "nondeterminism" which can be understood in the following way. Consider two color sequences in C, say

$$red, yellow, green, green, red, ...$$

and

$$red, yellow, blue, green, yellow....$$

The first two colors in the sequences are the same: red followed by yellow. However, the third colors are different (green in the first sequence and blue in the second). One may say that there is a nondeterministic choice of the next color right after the first two colors. Please develop and justify a metric M_2 which is a function of G that measures the nondeterminism on all color sequences in C. You need also show me an algorithm in computing such M_2. In case when efficient algorithm is hard to obtain, please also provide an approximation algorithm.

(a) Given two color sequences in C. We need to find metric M_2 which is a function of G that measures the nondeterminism on all color sequences in C.

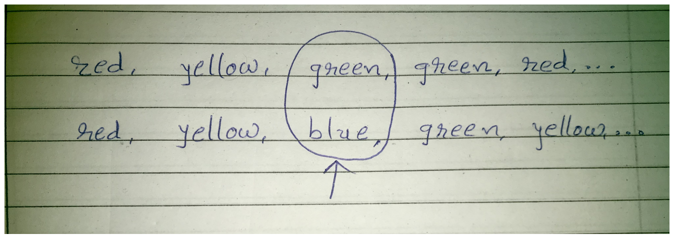

Figure 2: Color sequences in C

Step 1: As we have all color sequence in C and all the walks represented by color sequence starting from v_0, we see first two sequences are same while the next choice is nondeterministic.

Step 2: We build a new graph G' with starting node v'_0 which takes red as parent node. Each color sequence will be presented as a walk starting from v'_0 in G'. Same prefix color sequences shared the comparable walk in G'.

Up until this point the nondeterminism on all color sequences in C is indistinguishable to the average branches on each node in G'.

Step 3: The adjacency matrix of G' is A. The total number of branches in with the length n in G' is $\#G_n$.

$$\#G_n = det(A^n)$$

Step 4: So now the nondeterminism M_2 on graph G' will be

$$M_2 = \lim_{n \to +\infty} \left(\frac{\log_2 det(A^n)}{n}\right)$$

Step 5: As per Perron-Frobenuis theorem we can get the largest eigen value λ of A^n when $n \to \infty$ as

$$\lambda = \lim_{n \to +\infty} \left(\frac{\log_2 det(A^n)}{n}\right)$$

Step 6: The nondeterminism M_2 on all color sequences in C is

$$M_2 = \lambda$$

3. In good old days, a program is understood as a functional unit so that testing refers to figuring out its input/output relation. However, nowadays, a program (such as embedded systems code) is often reactive: it constantly interacts with its environment so that when you test it, it will not just give you one pair of input/output; instead, it will give you a test; i.e., a sequence of pairs of input/output:

$$(i_1, o_1), ..., (i_n, o_n)$$

for some n. Notice that the input sequence $i_1, ..., i_n$ is called a test case (provided by a test engineer) and the output sequence $o_1, ..., o_n$ is the test result. Now, we assume that each of i_j and o_j is a color in C. The program can be nondeterministic; i.e., one test case may have multiple test results. We now assume that the program is a blackbox (whose code is not available; not even assembly nor machine code nor design nor requirements. Do not use any code analysis techniques, design analysis techniques, requirements analysis techniques here since they are not applicable and you will get zero.). Sadly, researchers know little about testing a nondeterministic blackbox program! Good news is one can use the most stupid approach in testing such a program: try many many test cases and then collect many many test results. Our experiences are very intuitive: a more highly nondeterministic program is also harder to test.

(3.1, 10pts) Please develop and justify an algorithm to estimate, from those many many tests, how high the nondeterminism of the blackbox program under test is.

(3.2, 50pts) Please write a mini-paper (2-3 pages) to show an application and how your algorithm in 3.1 can be used, including but not limited to its strength and weakness.

(a) Given: Set of tests in form of sequence pairs of input/output $(i_1, o_1), ..., (i_n, o_n)$

Want: Matrix for nondeterministic of the blackbox program.

The nondeterminism of the blackbox program under test is the weakness of the program. So, lets assume that the nondeterminism of the program test is also the data of the program under one test. So we figure out how much data moved from inputs to outputs under one test.

(3.1)

Step 1: We take two dependent variable A, B as data sources yields are reliant

$$M =< i_1, i_2, ..., i_n >$$
$$N =< o_1, o_2, ..., o_n >$$

Step 2:Now we make a bipartite graph G with joint dispersion p(A,B) Consider that the number of finite colors in C is k.

Step 3: lets say shared data/entropy among M, N is I(M,N) and from p(M,N) we can compute I(M,N). So we characterize

$$\lambda_G = \max_{p(M,N)} I(M, N)$$

Step 4: We know the size of maxmatching M_1 so we can write

$$\lambda_G = \log_2 M_1$$

Moreover we comprehend that the size of k is amazingly enormous and n tends to propagation,

$$\lambda_G = \lim_{n \to +\infty} (\frac{1}{n} \log_2 \frac{No_{left}.No_{right}}{E})$$

From this we got the nondeterministic of the blackbox program under one test result.

$$\lambda_G = \lim_{n \to +\infty} (\frac{1}{n} \log_2 \frac{k^2}{n})$$

Step 5: As there are various test results, we have different y for each test results. So we figure out the variance S^2,

$$S^2 = \frac{\sum_{n=1}^{x} (\lambda_i - avg.\lambda)}{x} (x > 0, x \to \infty)$$

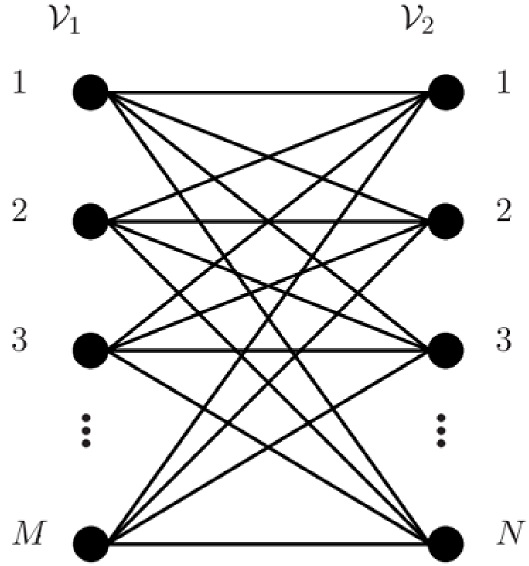

Figure 3: Bipartite graph joint distribution p(M,N)

If we get S^2, big we can say the nondeterministic of the blackbox program under test is high Otherwise the nondeterministic of the blackbox program under test is low.