Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

Exercise Sheet 4 – Probability Amplification

Probability and Computing

Exercise 1 – Probability Amplification with Two-Sided Error

Suppose a Monte Carlo algorithm A solves a decision problem correctly with probability $1/2 + \varepsilon$ and incorrectly otherwise (for some $\varepsilon > 0$). Let A' be the algorithm that executes A independently t times and decides according to the majority outcome. Show that the error probability of A' is at most $e^{-2t\varepsilon^2}$.

Hint: The following may be useful: $\sum_{i=0}^{t} \binom{t}{i} = 2^t$.

Solution 1

Let $I \in \{0, \dots, t\}$ denote the number of executions producing the correct result. For A' to return an incorrect answer, it must hold that $I \leq t/2$. The result follows from the following estimation:

$$\begin{aligned}
\Pr[I \leq t/2] &= \sum_{i=0}^{\lfloor t/2 \rfloor} \Pr[I = i] = \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} (\tfrac{1}{2} + \varepsilon)^i (\tfrac{1}{2} - \varepsilon)^{t-i} \leq \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} (\tfrac{1}{2} + \varepsilon)^{t/2} (\tfrac{1}{2} - \varepsilon)^{t/2} \\
&\leq (\tfrac{1}{4} - \varepsilon^2)^{t/2} \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} \leq (\tfrac{1}{4} - \varepsilon^2)^{t/2} \sum_{i=0}^{t} \binom{t}{i} = (\tfrac{1}{4} - \varepsilon^2)^{t/2} \cdot 2^t \\
&= (1 - 4\varepsilon^2)^{t/2} \leq (e^{-4\varepsilon^2})^{t/2} = e^{-2t\varepsilon^2}.
\end{aligned}$$

Exercise 2 – Pathfinder

Let G be an undirected graph with n vertices and m edges. We seek a path of length k that does not visit any vertex more than once. The naive brute-force approach runs in time $O(n^k)$. We now consider a simple randomized approach that works as follows. In the first step, each vertex v is assigned a label $L(v)$ uniformly at random from $\{1, \dots, k\}$. In the second step, for every vertex v with $L(v) = 1$, a modified breadth-first search is started, where a vertex w can be discovered from a vertex u only if $L(u) = L(w) + 1$ holds. If a vertex with label k is reached, a path of length k is constructed and output.

(a) Show that the algorithm finds a path of length k with probability $1/k^k$, if such a path exists.

(b) Improve the success probability to $1 - 1/n$ by probability amplification. What is the resulting total running time?

Remark: This idea is known as "Color Coding". There exist more refined variants.

Solution 2

(a) Let $P = (v_1, \dots, v_k)$ be a path of length k in G. With probability $1/k^k$, each v_i receives color i for all $i \in [k]$. In this case, the breadth-first search will reach all vertices $v_1, \dots, v_k$ (along P or another path) and thus find a path of length k.

(b) We repeat the algorithm $t = k^k \cdot \ln n$ times. The success probability then becomes:

$$1 - (1 - k^{-k})^t \geq 1 - (e^{-k^{-k}})^t = 1 - e^{-\ln n} = 1 - \tfrac{1}{n}.$$

Since n breadth-first searches can be performed in time $O(nm)$, the overall runtime is $O(nm \cdot k^k \cdot \ln n)$.

Exercise 3 – Bonus: Random-Walk Solver for 3-SAT

Let $\varphi(x_1, \dots, x_n)$ be a 3-SAT formula with n variables and m clauses. We seek a satisfying assignment using the following algorithm:

```
Algorithm randomWalkSolver(φ):
    sample x_1, ..., x_n ~ Ber(1/2)
    for k = 1 to n/2 do
        if φ(x_1, ..., x_n) = 1) then
            break
        let C be an arbitrary unsatisfied clause of φ
        sample j ~ U({1, 2, 3})
        let x_i be the jth variable in C
        x_i ← 1 − x_i
    if φ(x_1, ..., x_n) = 1) then
        return (x_1, ..., x_n)
    return ⊥
```

If φ is unsatisfiable, clearly randomWalkSolver(φ) $= \bot$. Otherwise, let x^* be a satisfying assignment. Show that:

(a) With probability at least $1/2$, the initial random assignment $(x_1, \dots, x_n)$ agrees with x^* on at least $n/2$ variables.

(b) If $\varphi(x_1, \dots, x_n) = 0$, then one iteration of the loop will, with probability $1/3$, flip a variable so that it matches x^* on one more position.

(c) Use probability amplification to obtain an algorithm with success probability $1 - 1/n$. What is the total running time?

Solution 3

(a) For every "bad" assignment that agrees with x^* on fewer than $n/2$ variables, the bitwise complement agrees with x^* on more than $n/2$ variables. Thus, "bad" assignments make up at most half of all possible assignments. (Assignments agreeing on exactly $n/2$ variables yield a slight bias in our favor when n is even).

(b) In the unsatisfied clause C considered during the loop, at least one variable must differ between $(x_1, \ldots, x_n)$ and x^*, since x^* satisfies C. With probability $1/3$, we select this variable.

(c) From (a) and (b), the success probability for one run is at least $\frac{1}{2} \cdot 3^{-n/2}$ (intuitively: we start near x^* and move toward it). Repeating the algorithm $t = 2 \cdot 3^{n/2} \cdot \ln n$ times yields a success probability of

$$1 - (1 - \tfrac{1}{2} \cdot 3^{-n/2})^t \geq 1 - (e^{-\frac{1}{2} \cdot 3^{-n/2}})^t = 1 - e^{-\ln n} = 1 - \tfrac{1}{n}.$$

If m is the number of clauses in φ, then randomWalkSolver runs in time $O(nm)$. Altogether, this gives a total runtime of $O(3^{n/2} \cdot nm)$, which can be asymptotically better than the naive $O(2^n)$ algorithm.