2.3.1 Using the minimax lower bound method

Let's apply this lower bound method to the query complexity of binary NAND trees.

It is very natural to try simply picking iid input bits. It turns out that if we do that, it's quite easy to analyze the efficiency of deterministic algorithms, because of a helpful theorem of Tarsi:

Theorem 14 (Tarsi [47])**.** *Let T be a finite NAND tree that is "spherically symmetric"—that is, all nodes at a common level have the same degree (in particular, all leaves are at the same level). Let the input bits be chosen iid. Then right-to-left DFS is an optimal algorithm in terms of the expected number of leaf queries.* (Proof omitted.)

Necessity of the spherical symmetry assumption: Consider the tree in Fig. 2.5, in which the triangle represents a large complete subtree of degree-2 NANDs. Select input bits iid in such a manner that every node in the tree retains probability bounded away from 0 and 1 of equaling 1. This can be accomplished by setting each input to 0 with probability p satisfying $p = (1-p)^2$, which happens to solve to $p = \frac{3-\sqrt{5}}{2}$. With this distribution, all vertices in a binary NAND tree, regardless of its structure, have probability p of equaling 0.

Comment: It is important to use this value of p. If we use a different value, then DFS will be much more efficient, having query complexity $O(N^{1/2})$ [39].

Now, in the tree in the Figure, if x_n happens to be a 0, the formula value is set to 1. So the expected number of queries is minimized by checking this vertex first, before if necessary tackling the large complete subtree, which will certainly require a large number of queries (exercise: any witness for the subtree is of size $\geq \sqrt{\frac{n-1}{2}}$).

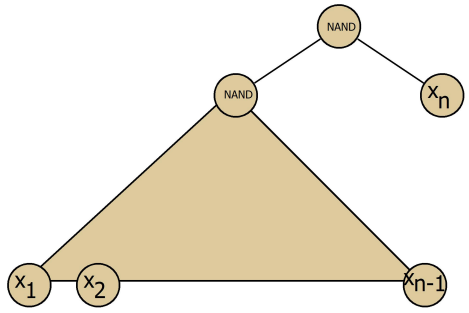

Figure 2.5: A non-spherically-symmetric tree

Applying Tarsi's theorem: We may now obtain a lower bound on the randomized query complexity of evaluating the complete binary NAND tree by evaluating the left-to-right DFS algorithm. Letting $\dot{T}_n^b$ be the complexity of this algorithm on this distribution, conditioned on the output value being b, we see that

$$\dot{T}_n^0 = 2\dot{T}_{n-1}^1$$

The more interesting case is next, when we examine the three different ways, totalling to $2p - p^2$, that one or both of the children can evaluate to 0. The first term here accounts for the children being $(0,1)$, the second for $(1,0)$, and the third term accounts for $(0,0)$:

$$\begin{aligned}\dot{T}_n^1 &= \frac{p(1-p)}{2p-p^2}\dot{T}_{n-1}^0 + \frac{p(1-p)}{2p-p^2}(\dot{T}_{n-1}^0 + \dot{T}_{n-1}^1) + \frac{p^2}{2p-p^2}\dot{T}_{n-1}^0 \\ &= \dot{T}_{n-1}^0 + \frac{1-p}{2-p}\dot{T}_{n-1}^1\end{aligned}$$

Now we have the vector recursion

$$\begin{pmatrix} \dot{T}^0_n \\ \dot{T}^1_n \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 1 & \frac{1-p}{2-p} \end{pmatrix} \begin{pmatrix} \dot{T}^0_{n-1} \\ \dot{T}^1_{n-1} \end{pmatrix}$$

When we substitute the above value of p, we find eigenvalues of $\frac{1+\sqrt{5}}{2}$ and $\frac{-4}{1+\sqrt{5}}$; the former is larger in absolute value, so we find that $\dot{T}^0_n, \dot{T}^1_n \in \Theta((\frac{1+\sqrt{5}}{2})^n) \cong 1.618^n$, which is to say $N^{\lg \frac{1+\sqrt{5}}{2}} \cong N^{0.694}$ where $N = 2^n$ is the number of leaves.

This leaves a gap to the upper bound of $N^{\lg \frac{1+\sqrt{33}}{4}} \cong N^{0.7537}$. The weakness is with the lower bound. In the NAND tree if both inputs to a vertex are 0, no reasonable algorithm will read leaves of both sub-trees of that vertex. So, in order to prove the best lower bound, we have to choose a distribution on inputs that precludes the possibility that both inputs to a node will be 0. Thus, a "hard" distribution will allow only the possibilities 01, 10 or 11 at the inputs to any node.

It is no longer possible to apply Tarsi's theorem to such distributions, since the inputs are not iid.

However, a careful analysis by Saks and Wigderson [44] shows that the upper bound of $N^{\lg \frac{1+\sqrt{33}}{4}}$ is tight. They use a different input distribution q, with the property that two siblings in the tree never both evaluate to 0; and show that DFS is still optimal for that distribution.

Comment: For MAJ3, Tarsi's theorem does not apply, and in fact, DFS (with pruning) is *not* optimal; it can be helpful to query some inputs without fully evaluating subtrees. See [44].

2.4 Randomized and distributional complexity with error ("Monte Carlo")

We consider now randomized algorithms which are permitted to err but still have a worst-case guarantee: for all inputs x, the probability of error must be at most λ. A randomized algorithm can be described by a probability distribution p on seeds (strings of coin tosses) r. Each such seed may be correct on some x's and incorrect on other x's; so the guarantee that we have error probability at most λ on any input x is a constraint on the allowable distributions p. It is understood that our expressions are functions of the size parameter n, and that for any n, there are only finitely many inputs x. However there may be infinitely many seeds r.

The randomized complexity of problem L, with error parameter λ, is

$$\mathrm{Rand}_\lambda(L) = \inf_{\substack{p: \\ \text{err. prob.} \le \lambda \\ \text{for all } x}} \max_x E_{r \in p} T(r, x)$$

where $T(r, x)$ is the complexity (run-time, query complexity, communication complexity, or whatever we care about) of the algorithm using seed r on input x.

Observe that the set of allowable distributions p is closed and convex, call it $R_{L,\lambda}$.

The distributional complexity of L is stated in terms of a distribution q on inputs x. Since there are only finitely many x of given size n, the space of distributions q is compact, so we write max rather than sup.

The error constraint in distributional complexity is that every seed r that is allowed, must err with probability at most λ on the distribution q.

$$\mathrm{Dist}_\lambda(L) = \max_q \inf_{\substack{r: \\ \text{err. prob.} \le \lambda \\ \text{on } q}} E_{x \in q} T(r, x)$$

We have the following extension of weak LP duality to the Monte Carlo case.

Theorem 15 (Yao)**.** *Let* $a, b > 1,\ 1/a + 1/b = 1$. *Then* $\text{Rand}_\lambda \geq \frac{1}{a}\text{Dist}_{b\lambda}$. *In particular* $\text{Rand}_\lambda \geq \frac{1}{2}\text{Dist}_{2\lambda}$.

We'll prove this twice, once "directly" and once by looking at the formal dual LP.

Proof. Fix any $\varepsilon > 0$. Let p_ε be a distribution on seeds r s.t.

$$\forall x: \quad E_{p_\varepsilon}(T(r,x)) \leq \text{Rand}_\lambda + \varepsilon/a \tag{2.3}$$

$$\forall x: \quad \Pr_{p_\varepsilon}(\text{Error on } x) \leq \lambda \tag{2.4}$$

Let q be an arbitrary distribution on x. Set

$$T_q = E_{r\in p_\varepsilon, x\in q}(T(r,x)). \qquad \text{Note } T_q \leq \text{Rand}_\lambda + \varepsilon/a \quad \text{by 2.3}$$
$$\lambda_q = \Pr_{r\in p_\varepsilon, x\in q}(r \text{ errs on } x). \qquad \text{Note } \lambda_q \leq \lambda \quad \text{by 2.4}$$

Let

$$\text{Slow}(q) = \{r : E_{x\in q}(T(r,x)) > aT_q\}$$
$$\text{Faulty}(q) = \{r : \Pr_{x\in q}(r \text{ errs on } x) > b\lambda_q\}$$

Let $\mathcal{E}(r,x)$ be the indicator for algorithm r erring on input x. Picture two tables:

$$\begin{array}{c} \\ r \\ \\ \end{array}\!\!\begin{array}{c} x \\ \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & T(r,x) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ r \\ \\ \end{array}\!\!\begin{array}{c} x \\ \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \mathcal{E}(r,x) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \end{array}$$

$\text{Slow}(q)$ is a set of "bad" (high expectation over q) rows in the first table, $\text{Faulty}(q)$ is a set of "bad" (high expectation over q) rows in the second table. Now by the Markov ineq.,

$$\Pr_{r\in p_\varepsilon}(\text{Slow}(q)) < 1/a$$
$$\Pr_{r\in p_\varepsilon}(\text{Faulty}(q)) < 1/b.$$

We conclude by a simple union bound that for every distribution q, there is an r s.t.

$$\Pr_{x\in q}(r \text{ errs on } x) \leq b\lambda_q \leq b\lambda$$
$$E_{x\in q}(T(r,x)) \leq aT_q \leq a\,\text{Rand}_\lambda + \varepsilon$$

Consequently,

$$\text{Dist}_{b\lambda} \leq a\,\text{Rand}_\lambda + \varepsilon$$

and since this holds for all $\varepsilon > 0$,

$$\text{Dist}_{b\lambda} \leq a\,\text{Rand}_\lambda.$$

□

It is worth nothing that in some applications, we restrict ourselves to algorithms with $T(r,x)$ a constant τ (or else use an upper bound τ on $T(r,x)$ just to simplify matters). In that case, which we'll call the uniform-complexity case, we don't need to balance between the demands of two different matrices, and Theorem 15 simplifies to

Theorem 16. *For uniform-complexity algorithms,* $\text{Rand}_\lambda \geq \text{Dist}_\lambda$.

We omit the proof.

Here is the second proof of Theorem 15. We restrict to $a = b = 2$ for simplicity.

Proof. The primal LP (call it LP_1) which defines $\text{Rand}_\lambda(L)$ is this: (Subscripts $_{..}$ are because we don't care the dimensions of the matrix):

$$\begin{array}{c} \begin{pmatrix} p_{r_1} & \dots & p_{r_{..}} & t \end{pmatrix} \end{array} \left(\begin{array}{c|c||ccc|ccc} \text{min} & & \multicolumn{3}{c|}{\text{subject to}} & & & \\ & 1 & -\lambda & \dots & -\lambda & 0 & \dots & 0 \\ & \wedge| & \wedge| & \wedge| & \wedge| & \wedge| & \wedge| & \wedge| \\ \hline 0 & 1 & -\mathcal{E}(r_1,x_1) & \dots & -\mathcal{E}(r_1,x_{..}) & -T(r_1,x_1) & \dots & -T(r_1,x_{..}) \\ \dots & 1 & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & -\mathcal{E}(r_{..},x_1) & \dots & -\mathcal{E}(r_{..},x_{..}) & -T(r_{..},x_1) & \dots & -T(r_{..},x_{..}) \\ 1 & 0 & 0 & \dots & 0 & 1 & \dots & 1 \end{array}\right)$$

In words, optimize (over p, a prob. dist., and t) to minimize t subject to the error probability on every x being $\leq \lambda$, and t being an upper bound on the expected runtime on every x.

The dual of LP_1 is

$$\left(\begin{array}{cc|c|ccccc} \text{max} & & 1 & -\lambda & \dots & -\lambda & 0 & \dots & 0 \\ \hline \text{subject to} & & & & & & & & \\ 0 & \geq & 1 & -\mathcal{E}(r_1,x_1) & \dots & -\mathcal{E}(r_1,x_{..}) & -T(r_1,x_1) & \dots & -T(r_1,x_{..}) \\ \dots & \geq & 1 & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \geq & 1 & -\mathcal{E}(r_{..},x_1) & \dots & -\mathcal{E}(r_{..},x_{..}) & -T(r_{..},x_1) & \dots & -T(r_{..},x_{..}) \\ 1 & \geq & 0 & 0 & \dots & 0 & 1 & \dots & 1 \end{array}\right) \begin{pmatrix} v \\ u_{x_1} \\ \dots \\ u_{x_{..}} \\ q_{x_1} \\ \dots \\ q_{x_{..}} \end{pmatrix}$$

This is a little harder to describe in words, but notice the dual ranges over probability distributions q on the inputs (as one would expect), and nonnegative functions u (the role of u is less obvious, but roughly it exists to balance the time and error constraints). By strong duality, the opt of the Dual LP equals $\text{Rand}_\lambda(L)$.

Let's write down the optimization problem for $\text{Dist}_{2\lambda}$: optimize (over m and distribution q on inputs) to maximize m, subject to the requirement that for every r, either $\sum q_x \mathcal{E}(r,x) > 2\lambda$ or $\sum q_x T(r,x) \geq m$. (Note, this is not an LP.) The maximum value we can achieve for m in this optimization is, by definition, $\text{Dist}_{2\lambda}$; let us denote $\mathcal{D} = \text{Dist}_{2\lambda}$ for brevity in the next lines, and q be a distribution achieving it in this optimization problem.

We want to use q in Dual LP_1. Do this by setting $u_x = \frac{\mathcal{D}}{2\lambda} q_x$ and $v = \mathcal{D}$. Now show feasibility; consider the two types of r. If $\sum q_x \mathcal{E}(r,x) > 2\lambda$ then

$$\begin{aligned} v - \sum_x \mathcal{E}(r,x) u_x - \sum_x T(r,x) q_x &= \mathcal{D} - \frac{\mathcal{D}}{2\lambda} \sum_x q_x \mathcal{E}(r,x) - \sum_x T(r,x) q_x \\ &\leq \mathcal{D} - \frac{\mathcal{D}}{2\lambda} \sum_x q_x \mathcal{E}(r,x) \\ &< 0 \end{aligned}$$

If $\sum q_x T(r,x) \geq \mathcal{D}$ then $v - \sum_x \mathcal{E}(r,x) u_x - \sum_x T(r,x) q_x \leq \mathcal{D} - 0 - \mathcal{D} = 0$.

Thus the assigned q, u, v are feasible for the Dual LP. The value of the objective function is $\mathcal{D} - \lambda \sum_x u_x = \mathcal{D} - \lambda \frac{\mathcal{D}}{2\lambda} \sum_x q_x = \mathcal{D} - \lambda \frac{\mathcal{D}}{2\lambda} = \mathcal{D}/2$. Consequently, the value of the dual LP is at least half of $\mathcal{D}$, which is to say, $\text{Rand}_\lambda(L) \geq \frac{1}{2} \text{Dist}_{2\lambda}$. □

Another way of thinking about this second argument, is that if we find any $\mathcal{D}$ for which the LP (call it $\mathrm{LP}_2(\mathcal{D})$), in variables q_x,

$$\forall r: \quad \sum_x q_x(\frac{\mathcal{D}}{2\lambda}\mathcal{E}(r,x) + T(r,x)) \geq \mathcal{D}$$

is feasible, then (by plugging $v = \mathcal{D}$ and $u_x = \frac{\mathcal{D}}{2\lambda}q_x$ into the dual of LP_1), we find that $\mathcal{D}/2$ is a lower bound on the value of LP_1.

2.4.1 An application of the Monte Carlo minimax lower bound

We consider two-party communication problems defined by boolean functions f. The setting is that Alice receives an input $x \in X$, Bob receives an input $y \in Y$, and they wish to jointly compute $f(x,y)$; the cost measure is the total number T of bits exchanged before Alice announces the output. There are various possible costs:

$$\text{deterministic } D(f) \geq \mathrm{Rand}^{\text{private}}(f) \text{ with private coins } \geq \mathrm{Rand}^{\text{public}}(f) \text{ with public coins.}$$

For the randomized complexities we may also consider their variants allowing λ probability of error,

$$\mathrm{Rand}^{\text{private}}_\lambda(f) \geq \mathrm{Rand}^{\text{public}}_\lambda(f).$$

Today we focus on $\mathrm{Rand}^{\text{public}}_\lambda(f)$. From the previous section we have

$$\mathrm{Rand}^{\text{public}}_\lambda(f) \geq \frac{1}{2}\,\mathrm{Dist}_{2\lambda}(f)$$

where

$$\begin{aligned}
\mathrm{Dist}_{2\lambda}(f) &= \max_q \min_{\substack{r: \\ \text{err. prob.} \leq 2\lambda \\ \text{on } q}} E_{(x,y)\in q} T(r,(x,y)), \quad \text{equivalently} \\
\mathrm{Dist}_{2\lambda}(f) &= \max_q \mathrm{Dist}^q_{2\lambda}(f) \quad \text{for} \quad \mathrm{Dist}^q_{2\lambda}(f) = \min_{\substack{r: \\ \text{err. prob.} \leq 2\lambda \\ \text{on } q}} E_{(x,y)\in q} T(r,(x,y))
\end{aligned}$$

Here q is a probability distribution on the set $X \times Y$, r is a deterministic communication protocol achieving error probability $\leq 2\lambda$ on distribution q, and $T(r,(x,y))$ is the communication complexity of r on input (x,y).

(NB, the usual inf over r has been replaced by a min here because there are only finitely many possible communication protocols, given that X and Y are finite sets.)

It is worth emphasizing that Dist is defined by maximization over all *joint distributions* on the pair x, y. If one maximizes only over product distributions, one generally gets a much lower value. (Although in the example we'll give today, it just so happens that the distribution we'll use is a product distribution.)

So now we need a technique for lower bounding the communication complexity of protocols which can tolerate error. The classic method for this is the *discrepancy method.*

A "rectangle" is a set $R = A \times B$ for $A \subseteq X, B \subseteq Y$. The significance of rectangles is that at the end of any communication protocol, Alice and Bob have identified that their input belongs to some rectangle. The whole protocol partitions $X \times Y$ into such rectangles. Given q, the best strategy for the players upon reaching a rectangle $A \times B$ is to announce the output 0 or 1 depending on which

of $q(R \cap f^{-1}(0))$ or $q(R \cap f^{-1}(1))$, respectively, is larger. The protocol is relatively likely to err on that rectangle if the two quantities are close. A protocol which achieves low error must, generally (w.r.t. distribution q), wind up in rectangles where the two quantities are not close.

This motivates the following definition. The discrepancy of rectangle R for boolean function f and distribution q is

$$\Delta(q, f, R) = |q(R \cap f^{-1}(0)) - q(R \cap f^{-1}(1))|$$

The overall discrepancy of f for distribution q is

$$\Delta(q, f) = \max_R |q(R \cap f^{-1}(0)) - q(R \cap f^{-1}(1))|$$

Theorem 17. *Let $\varepsilon > 0$.* $\mathrm{Dist}^q_{1/2-\varepsilon}(f) \geq \frac{\varepsilon}{2} \lg(\frac{\varepsilon}{\Delta(q,f)})$. *For uniform-complexity algorithms r,* $\mathrm{Dist}^q_{1/2-\varepsilon}(f) \geq \lg(\frac{2\varepsilon}{\Delta(q,f)})$.

Proof. We omit the proof of the non-uniform-complexity case.

For uniform complexity, let r be a deterministic algorithm achieving $\mathrm{Dist}^q_{1/2-\varepsilon}(f)$, and let τ be its complexity. Then the number of rectangles R it creates as the leaves of the protocol is 2^τ (or at most that). If we assume that Alice does the best thing at each leaf R (namely, announce "0" or "1" according to which outcome has greater probability mass in R) then

$$2\varepsilon = \sum_{\text{leaves} R} \Delta(q, f, R)$$

And so

$$2\varepsilon \leq 2^\tau \Delta(q, f).$$

□

Corollary 18. *For $\lambda < 1/2$,* $\mathrm{Rand}^{public}_\lambda(f) \geq \max_q \frac{1/4-\lambda/2}{1/4+\lambda/2} \mathrm{Dist}^q_{1/4+\lambda/2}(f) \geq \max_q \frac{(1/4-\lambda/2)^2}{1/2+\lambda} \lg \frac{1/4-\lambda/2}{\Delta(q,f)}$. *For uniform-complexity algorithms,* $\mathrm{Rand}^{public}_\lambda(f) \geq \max_q \mathrm{Dist}^q_\lambda(f) \geq \max_q \lg \frac{1-2\lambda}{\Delta(q,f)}$.

So ultimately to get a good lower bound on the Monte Carlo public coin complexity we'll want to choose a distribution q that achieves low discrepancy on large rectangles. That is of course problem-specific and can be interesting. Here we just cite one good example.

Application to the inner product function

A good example of the application of this method is to the communication complexity of the *inner product function:* $X = Y = (\mathbb{Z}/2)^n$ and $\mathrm{IP}(x, y) = \sum x_i y_i$ (note, addition is mod2). It can be shown (see [29] Ch. 3) that for $q =$ the uniform distribution, $\Delta(q, \mathrm{IP}) = 2^{-n/2}$. (Actually this isn't hard. Hint: viewed as a matrix, $\mathrm{IP} \cdot \mathrm{IP}^\dagger = 2^n I$. This tells you all eigenvalues of IP are $2^{n/2}$.) Applying Corollary 18 we now find: For $\lambda < 1/2$,

$$\mathrm{Rand}^{\text{public}}_\lambda(\mathrm{IP}) \geq \frac{(1/4 - \lambda/2)^2}{1 + 2\lambda}(n - 2 \lg \frac{4}{1 - 2\lambda})$$

and for uniform-complexity algorithms the bound is better,

$$\mathrm{Rand}^{\text{public}}_\lambda(\mathrm{IP}) \geq \frac{n}{2} - \lg \frac{1}{1 - 2\lambda}$$

In either formulation we have a linear communication complexity for any error rate strictly below $1/2$.

(Actually in the uniform-complexity case the $n/2$ can be improved to n.)