Quantum Complexity Theory | Sevag Gharibian
Summer 2019, University of Paderborn

Lecture 9: Boson sampling

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."
— Richard P. Feynmann

Contents

Introduction. We began in Lecture 1 by stating that the field of quantum computation is at a critical crossroads, with one of the following statements being necessarily false: The Extended Church-Turing Thesis is true, integer factorization does not have a polynomial time classical algorithm, or large-scale universal quantum computers can be built. Since this crossroads arose due to the discovery of Shor's quantum polynomial-time factoring algorithm, a natural goal is to try and show the Extended Church-Turing Thesis is false by running Shor's algorithm on a "large enough" quantum computer.

Unfortunately, there are caveats to this. First, even if Shor's algorithm could be implemented experimentally, this does not rule out the second statement — that perhaps there is an efficient *classical* algorithm for factoring. More worrisome is the fact that we are arguably not close to a functioning universal quantum computer capable of breaking today's RSA keys. For example, to a theoretician, a quantum random walk on the line is a rather basic construct; yet, implementing such a walk *efficiently* (i.e. resources scaling polynomially with the length of the walk) in an actual photonic system is highly non-trivial, requiring ideas such as *time multiplexing.*

Luckily, if our goal is to disprove the Extended Church-Turing Thesis, we do not necessarily need a *universal* quantum computer. Rather, a sufficiently restricted quantum model may still be able to solve "hard" problems, and yet be implementable on a large scale via near-term "noisy intermediate scale quantum devices" (NISQ). This quest for an experimental demonstration of quantum computational speedup has fallen under the moniker of "quantum supremacy", with multiple candidate approaches to date: The Instantaneous Quantum Polynomial-Time (IQP) model, random circuit sampling, and the deterministic quantum computation with one quantum bit (DQC1) model. Here, however, we shall focus on a framework which has elicited a particularly beautiful collaboration between the computer science and physics communities: *Boson sampling.*

Organization. We begin in Section 1 with an introduction to non-interacting bosonic systems. Section 2 describes the connection between such systems and computation of the matrix permanent. Using this background, Section 3 defines the Boson Sampling problem. Finally, Sections 3.1 and 3.2 discuss intractability of exact and approximate Boson Sampling for classical computers.

1 Of hedgehogs and photons

The basic premise of Boson sampling is to use *non-interacting Bosonic systems* to implement a computational task which is "easy" quantumly, yet provably "hard" classically. For our purposes, "bosons" will be "photons", and to begin with, we will equate "photons" with "hedgehogs".

The hedgehog model of computing. Suppose we have n identical hedgehogs, and $m \geq n$ burrows (numbered 1 to m). The hedgehog model is as follows:

1. Before nightfall, the first n burrows contain precisely 1 hedgehog each[1].
2. During the night, each hedgehog can move from its current burrow to any other. Some rules for this:
   - Parties are allowed, i.e. a burrow can host multiple hedgehogs.
   - No hedgehogs are created or destroyed in this process, i.e. we have conservation of hedgehogs.
3. When the night ends, we measure: How many hedgehogs are in each burrow?

To formalize this model, we can work in the *hedgehog number basis*, which is different from the usual standard basis for qubit systems. Namely, to specify the positions of all n hedgehogs, we use basis state

$$|S\rangle = |s_1 s_2 \cdots s_m\rangle$$

where $s_i \in \{0, \ldots, n\}$ denotes the number of hedgehogs in burrow i. The s_i are called "occupation numbers", and this basis the "occupation number basis".

Exercise. Why are we only concerned with the *number* of hedgehogs per burrow? (Hint: Which keyword used above essentially says this is the only defining characteristic of the hedgehogs?)

The set of all such valid basis states is denoted

$$\Phi_{m,n} := \left\{ (s_1, \ldots, s_m) \mid s_i \in \{0, \ldots, n\} \text{ and } \sum_{i=1}^{m} s_i = n \right\}. \tag{1}$$

Exercise. Why is the summation condition required in the definition of $\Phi_{m,n}$?

Of course, we're not just dealing with any old hedgehogs, but quantum hedgehogs; thus, we allow superpositions over hedgehog states:

$$|\psi\rangle = \sum_{S \in \Phi_{m,n}} \alpha_S |S\rangle,$$

where as usual $\sum_S |\alpha_S|^2 = 1$. A crucial point to note here is that unlike with m qubit systems, the vector space we are working in is *not* a tensor product of m systems of dimension n; we revisit this shortly.

From hedgehogs to photons. To move from the world of hedgehogs to photons, we make two simple substitutions: Replace the word "hedgehog" with "photon", and "burrow" with "mode" (for this lecture, a "mode" can be thought of as a spatial mode, meaning a "location" of a photon). We can now rephrase our discussion above formally in the setting of photons:

1. At the start of the experiment, our apparatus has n photons in the first n modes, and the remaining $m - n$ modes are empty, i.e. our start state is

$$|1_n\rangle := |1^n 0^{m-n}\rangle \in \Phi_{m,n}.$$

[1]Most hedgehog species are nocturnal. They are also very cute.

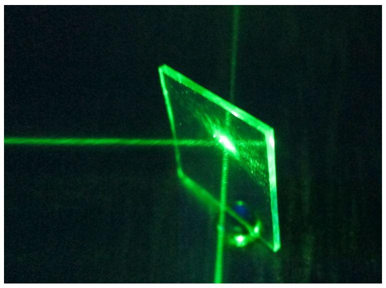

Figure 1: As this is the closest we will get to experiment in this course, it is worth seeing an actual piece of hardware: Depicted above is a beamsplitter designed to reflect 80% of incoming light, and transmit the remaining 20%. Intuitively, a beamsplitter implements a Hadamard gate. (Source: `https://commons.wikimedia.org/wiki/File:Flat_metal-coated_beamsplitter.png`.)

2. Formalizing the set of allowed operations (i.e. how the hedgehogs choose to switch burrows) is trickier, as we are working in a Hilbert space without a tensor product structure (in contrast to qubit systems). To see the full picture takes two steps: (1) We first look at the "idealized" case in which we have 1 photon and m modes; this will be analogous to modeling an m-dimensional qudit. The unitaries U in this case will hence be $m \times m$. (2) We then show how to map any $m \times m$ unitary U up to the full space $\Phi_{m,n}$ spans, which requires an understanding of how U acts on *multi*-photon configurations.

   Single photon configurations. Denote the subset of single photon configurations as $\Phi_{m,1} =: \{|i\rangle\} \subset \Phi_{m,n}$, i.e. $|i\rangle$ has $s_i = 1$ for some $i \in [m]$ and $s_i = 0$ otherwise. Restricted to this space, one can think of the entire system as a single m-dimensional qudit, with the ith "standard basis state" given by $|i\rangle = |0^{i-1}10^{m-i}\rangle$ (i.e. imagine encoding the basis states in unary, not binary). The set of allowed operations on this space, as expected, is the set of all $m \times m$ unitary matrices U.

   What makes optical setups appealing is that any such U can be written $U = U_T \cdots U_1$ for $T \in O(m^2)$, where each U_k is an $m \times m$ unitary or *optical element* falling into one of two classes: *Phase shifters* and *beam splitters*. These optical elements are relatively easy to implement in a lab; see Figure 1. Restricted to the single photon basis $\Phi_{m,1}$, each U_k acts non-trivially only on some pair of modes i and j, i.e. on unary basis states $|i\rangle$ and $|j\rangle$, and hence can be represented as a 2×2 unitary

$$U_k = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

   where the rows are labelled by $|i\rangle$ and $|j\rangle$, respectively. On $\Phi_{m,1}$, U_k acts as expected:

$$|i\rangle \mapsto a|i\rangle + c|j\rangle, \qquad |j\rangle \mapsto b|i\rangle + d|j\rangle, \qquad |k\rangle \mapsto |k\rangle \text{ for any } k \neq i, j. \tag{2}$$

   Exercise. Consider the Pauli X_{12} gate applied to modes 1 and 2. Then, the 2×2 optical element has matrix representation (restricted to Span($|1\rangle, |2\rangle$))

$$X_{12} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

   What is $X_{12}|1\rangle$, $X_{12}|2\rangle$, and $X_{12}|k\rangle$ for $k > 2$?

   Exercise. Write down the full 3×3 matrix representation for X_{12} with respect to the $\Phi_{m,1}$ basis when $m = 3$.

Restricted to this $\Phi_{m,1}$ basis, phase shifters and beamsplitters have intuitively simple representations (for $\theta \in \mathbb{R}$):

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

Thus, phase shifters are essentially phase gates, and beamsplitters are analogous to Hadamard gates.

Exercise. How does a phase shifter applied to modes i and j act on basis states $|i\rangle, |j\rangle \in \Phi_{m,1}$? How about a beam splitter?

Multi-photon configurations. Focusing on single-photon configurations gave an intuitive sense of what phase shifters and beamsplitters do, but in reality our full system of n photons in m modes is not m-dimensional, but $M = |\Phi_{m,n}|$-dimensional.

Exercise. Show that $M = \binom{m+n-1}{n}$.

Given any $m \times m$ unitary U (which recall can be implemented with phase shifters and beamsplitters), we hence need a way of mapping U to the larger M-dimensional space to understand its action on *all* basis states in $\Phi_{m,n}$, as opposed to just $\Phi_{m,1}$. (In other words, how does U_k act on modes which contain multiple photons, such as $|20\rangle$?) This mapping $\varphi : \mathcal{U}(\mathbb{C}^m) \mapsto \mathcal{U}(\mathbb{C}^M)$ turns out to be a homomorphism, meaning for us that it obeys $\varphi(U) = \varphi(U_T) \cdots \varphi(U_1)$. Thus, it suffices understand its action on the 2×2 optical elements U_k, which is:

$$\langle st|\varphi(U_k)|uv\rangle = 0 \qquad \text{if} \quad s + t \neq u + v \tag{3}$$

$$\langle st|\varphi(U_k)|uv\rangle = \sqrt{\frac{u!v!}{s!t!}} \sum_{\substack{p+q=u \\ p \leq s \\ q \leq t}} \binom{s}{p}\binom{t}{q} a^p b^{s-p} c^q d^{t-q} \qquad \text{if} \quad s + t = u + v. \tag{4}$$

Exercise. Why is Equation (3) equal to 0? (Hint: Which property of the hedgehogs must we preserve?)

Exercise. Setting $a = d = 0$ and $b = c = 1$, confirm that Equations (3) and (4) correctly recover the action of Pauli X when restricted to $\Phi_{m,1}$.

Exercise. How does a phase shifter act on basis state $|20\rangle \in \Phi_{2,2}$?

Exercise. What is the overlap onto $|11\rangle \in \Phi_{2,2}$ if we start with $|20\rangle \in \Phi_{2,0}$ and apply $U_k = X$? How about the overlap onto $|02\rangle$? What does this suggest intuitively about how X acts on multiple photons?

Putting it all together. In sum, given any desired $m \times m$ unitary U (including ones which will later encode hard problems), in an optical setup one can implement U by a sequence of $O(m^2)$ phase shifters and beamsplitters, and the effective action of this U on the larger M-dimensional Hilbert space is prescribed by $\varphi(U) = \varphi(U_T) \cdots \varphi(U_1)$.

3. At the end of the experiment, we measure with respect to basis $\Phi_{m,n}$ to see which modes the photons are in. Let D_U denote the distribution obtained, assuming the experiment implemented $m \times m$ unitary U. Then, the probability of observing configuration S is

$$\Pr[S \in \Phi_{m,n}] = |\langle S|\varphi(U)|1^n\rangle|^2 .$$

2 Connection to the matrix permanent

To now connect our optics setup to hard computational problems, we return to our hedgehog model of computing, and study a related thought experiment. In this experiment, we have n indistinguishable hedgehogs and n burrows. The rules are as follows:

1. Before nightfall, burrow $i \in [n]$ contains precisely one hedgehog, the hedgehog labelled i.
2. During the night, hedgehog i moves to burrow j with probability a_{ij}.
3. When the night ends, we ask: What is the probability that each burrow contains precisely one hedgehog?

Let us derive a formula for this probability, for which we simply have to *count* all configurations the hedgehogs could end up in. For starters, observe that the probability that the ith hedgehog remains in the ith burrow is just $a_{11} \cdots a_{nn}$.

Exercise. What is the probability that for all i, hedgehog i moves to burrow $(i \mod n) + 1$?

Exercise. Show that the probability that each burrow contains precisely one hedgehog is

$$\sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} =: \mathrm{Per}(A), \tag{5}$$

where S_n is the set of permutations acting on n elements, and A is the $n \times n$ matrix with entries a_{ij}.

Brief aside on the permanent. The quantity in Equation (5) is the *permanent* of matrix A, and has seen considerable attention for at least two centuries now (being mentioned in the 1812 memoirs of Binet and Cauchy). It looks remarkably like a related quantity — the *determinant* of A, whose formula is identical except each term in the sum is multiplied by the sign of the permutation σ. Yet, these two apples most certainly did not fall from the same tree — while the determinant is efficiently computable, the permanent is #P-hard to compute exactly, even if A consists only of entries from $\{0, 1\}$. The best known general algorithm for $\mathrm{Per}(A)$ is Ryser's algorithm, which requires $\Theta(n2^n)$ arithmetic operations. We do catch a break when A has only non-negative entries: In this case, there is a *fully polynomial-time randomized approximation scheme (FPRAS)* for approximating $\mathrm{Per}(A)$, i.e. for any inverse polynomial error ϵ, there is a polynomial-time randomized algorithm outputting $\mathrm{Per}(A)$ up to relative error $(1 \pm \epsilon)$. While this setting does apply to our hedgehog model above, it will crucially *not* apply for the type of matrices which arise through boson sampling.

Connection to optics. Recall that in our optics setup, any $m \times m$ unitary U can be performed (restricted to the single photon space) via a sequence of phaseshifters and beamsplitters. The key point is that if we run our optics experiment starting in configuration $|T\rangle \in \Phi_{m,n}$ and apply U, then one can show that the probability of observing end configuration $|S\rangle \in \Phi_{m,n}$ is given by

$$|\langle S|\varphi(U)|T\rangle|^2 = \frac{|\mathrm{Per}(U_{ST})|^2}{s_1! \cdots s_m! t_1! \cdots t_m!}, \tag{6}$$

for $|S\rangle = |s_1 \cdots s_m\rangle$, $|T\rangle = |t_1 \cdots t_m\rangle$, and U_{ST} defined via the following two-step process (this is necessary because we must account for the action of U on multi-photon configurations via φ):

1. Map U to U_T by listing t_i copies of column i of U.
2. Map U_T to U_{ST} by listing s_i copies of row i of U_T.

This process is best demonstrated with an example, for which we set $m = 3$, $n = 2$, $S = |200\rangle$ and $T = |110\rangle$:

$$U = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \qquad U_T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad U_{ST} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}.$$

Exercise. Show that U_{ST} is an $n \times n$ matrix, for n the number of photons.

Exercise. Show that if $|T\rangle = |1^n 0^{m-n}\rangle$, then U_T is U restricted to its first n columns.

Exercise. Show that if $|S\rangle = |T\rangle = |1^n 0^{m-n}\rangle$, then U_{ST} is the upper left $n \times n$ principal submatrix of U, i.e. the submatrix obtained by keeping the first n columns and rows of U.

In sum, if one could write down output probabilities of our photonic experiment, then one could compute permanents of matrices U_{ST}. In particular, as the last exercise above suggests, when $|S\rangle = |T\rangle = |1^n 0^{m-n}\rangle$, this boils down to the permanent of whichever $n \times n$ matrix A we are able to embed in the top-left block of U. Of course, there are some important questions to be answered: Which types of matrices can we embed into unitaries in such a fashion? How do we convert the ability to sample to the ability to estimate output probabilities? Can an experimental quantum optics device, which will inherently be subject to noise and imperfection, itself perform such estimation? These questions, along with the connection between the permanent and photonics setups, are the starting point of boson sampling.

3 Boson Sampling

We are now in position to semi-formally define the task of Boson Sampling.

Definition 1 (Boson Sampling)**.**

- *Input: An $m \times m$ unitary matrix U.*
- *Output: Define distribution D_U as follows. Starting in configuration $|1^n\rangle \in \Phi_{m,n}$, we imagine running the optics setup outlined in Section 1 with unitary U. For any configuration $S \in \Phi_{m,n}$, the probability of observing output configuration S is*

$$\Pr_{D_U}[S] = \frac{|\mathrm{Per}(U_{S,1^n})|^2}{s_1! \cdots s_m!}.$$

The output of Boson Sampling is to sample configurations according to D_U.

Two remarks are in order: First, unlike all other computational problems we have seen in this course, Boson Sampling is *not* a decision or promise problem; rather, it is a *sampling* problem (i.e. the output is not a single bit, but a random sample over strings). Second, the "semi-formal" aspect of the definition above is that we have not specified the *precision* to which the sampling must be done (i.e. are we required to sample from D_U exactly? Approximately? Within what type of error?). These distinctions are crucial, and are discussed in the next two sections.

3.1 The exact case

The strongest variant of Boson Sampling would be to require the sampling to be *perfect* — i.e. one outputs samples from $\Phi_{m,n}$ exactly according to D_U. This is not experimentally realistic, as any physical setup is subject to noise. Nevertheless, in this case one can rigorously show the following result.

Theorem 2 (Exact classical Boson Sampling)**.** *Suppose there is a classical algorithm A which, given any $m \times m$ unitary U, solves the Boson Sampling problem exactly. We make no assumptions about A (e.g. it could be a* P, BPP, *or* PH *machine). Without loss of generality, we may view A as a deterministic machine which is fed a uniformly random string of bits r. By assumption, the sample produced by $A(U, r)$ is distributed exactly according to D_U. Then,*

$$\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{BPP}^{\mathrm{NP}^A}.$$

Deer in the headlights: Interpreting Theorem 2 . Theorem 2 is a bit stunning at first sight, so let us carefully unpack it.

- What it *does* say is that if A is a BPP machine (or even a PH machine!), then

  $$\mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{BPP}^{\mathrm{NP}^{\mathrm{BPP}}} \subseteq \mathrm{PH},$$

  which would collapse PH. Thus, it is highly unlikely that exact Boson Sampling is efficiently simulatable classically.

- What it *does not* say is anything about whether a quantum computer can perform exact Boson Sampling. And therein lies the magic of the theorem — Theorem 2 does *not* prove that exact Boson Sampling is #P-hard. Rather, it shows that if there is an efficient *classical* algorithm for Boson Sampling, then PH collapses.

  Exercise. Why would it be presumably "bad" if Theorem 2 actually showed Boson Sampling is #P-hard? (Hint: What would this say about the ability of quantum computers to solve Boson Sampling?)

  The way Theorem 2 accomplishes this feat is by exploiting the fact that the randomness r in any classical machine A can be "extracted" from it. In other words, a classical algorithm A is without loss of generality deterministic, up to an input string of uniformly random bits r.

  Exercise. Why is it not clear how to similarly "extract the randomness" out of a quantum computation?

- While theoretically interesting, Theorem 2 unfortunately does not suffice to rule out the Extended Church Turing thesis, as even an optical setup realistically cannot perform exact Boson Sampling due to experimental noise. Thus, we must consider *approximate* Boson Sampling.

3.2 The approximate case

While things work out neatly in the exact case, the approximate case (which is the relevant one) is much messier; in particular, we do not have a rigorous statement along the lines of Theorem 2. Rather, there is a two-step agenda in place, the second part of which currently relies on (arguably reasonable) conjectures:

1. **What is currently proven:** If one can classically simulate "approximate" Boson Sampling, then one could compute the permanent "approximately" and "on average" (i.e. for "most inputs").
2. **What relies on conjecture:** Computing the permanent "approximately" and "on average" is #P-hard.

Taken together, this agenda would yield that efficient classical simulation (i.e. in BPP) of "reasonable" Boson Sampling setups (i.e. allowing reasonable error, and taking into account "average-case" inputs as opposed to extremal worst-case inputs[2]) is likely impossible. Again, the agenda does *not* imply that simulating

[2]This distinction is crucial. A classic example is the canonical NP-complete problem 3-SAT; while intractable in the worst case, many (if not most) instances of 3-SAT can be solved efficiently in practice using heuristics.

Boson Sampling approximately and on average is #P-hard, but rather that any *classical* algorithm for such a simulation can be bootstrapped to solve #P-hard problems. Finally, let us stress that it is not clear whether even a *quantum* computer can solve approximate Boson Sampling on average — the biggest challenge is arguably the requirement for single photon sources (i.e. to prepare the initial state $|1^n\rangle$). Addressing this question is not the purpose of today's lecture.

Formalizing the agenda. In the remainder of this lecture, we shall sketch how Step 1 of the agenda above is formalized and shown. The computational problem capturing approximate permanent computation on average is the following.

Definition 3 (Gaussian Permanent Estimation (GPE))**.**

- *Input: (1) A random matrix $X \in \mathcal{L}(\mathbb{C}^n)$, each entry of which is distributed independently according to the standard Gaussian distribution $\mathcal{N}(0,1)$. (2) Precision parameter $\epsilon > 0$, specified in unary. (3) Success probability parameter $\delta > 0$, specified in unary.*
- *Output: With probability at least $1-\delta$ (with respect to the randomness in choosing X), output a value $z \in \mathbb{R}$ satisfying*
$$|\mathrm{Per}(X)|^2 - \epsilon n! \leq z \leq |\mathrm{Per}(X)|^2 + \epsilon n!.$$

Exercise. Which parameter above captures the notion of solving for the permanent "approximately"? Which parameter captures "on average"?

The main theorem of this lecture is the following, which states that if efficient classical simulation of approximate Boson Sampling on average is possible, then GPE is in PH.

Theorem 4 (Main Theorem)**.** *Let D_U be the Boson Sampling distribution from Definition 1 for $m \times m$ input unitary U. Suppose there exists a classical algorithm A which, given precision parameter $\epsilon > 0$ in unary, outputs a sample from distribution D'_U such that $|D_U - D'_U| \leq \epsilon$ in time polynomial in m and $1/\epsilon$. Then,*
$$\mathrm{GPE} \in \mathrm{BPP}^{\mathrm{NP}^A}.$$
For clarity, $|D_U - D'_U|$ denotes the total variation distance between distributions D_U and D'_U.

Let us also formalize what we mean by a "classical algorithm A" in Theorem 4 above. Roughly, we shall think of A as an approximate Boson Sampling *oracle*, i.e. we will not care about its internal workings (other than it being deterministic), but just its input/output behavior.

Definition 5 (Approximate Boson Sampling oracle)**.** *A deterministic algorithm A, to be treated as an oracle, which takes in as input:*

- *an $m \times m$ unitary matrix U,*
- *a precision parameter $\epsilon > 0$ encoded in unary,*
- *and "random" string $r \in \{0,1\}^{\mathrm{poly}(n)}$.*

Let $D_A(U,\epsilon)$ denote, for any fixed U and ϵ, the distribution over outputs of A when r is uniformly random. Then, A outputs samples from $\Phi_{m,n}$ distributed according to distribution $D_A(U,\epsilon)$ such that, for all U, ϵ,
$$|D_A(U,\epsilon) - D_U| \leq \epsilon.$$

3.2.1 Proof of Theorem 4

We are now ready to move to the final stage of this lecture; giving a proof sketch of Theorem 4. Again, let us stress that this theorem only says that classical simulation of approximate Boson Sampling solves a problem related to computation of the permanent, GPE. It does *not* tell us whether GPE is hard to begin with (this would be the job of Step 2 of the agenda of Section 3.2, which currently relies on conjectures), nor does it say anything about whether quantum computers can simulate approximate Boson Sampling.

Proof sketch of Theorem 4. Let X, ϵ, δ be inputs to GPE, where recall X is $n \times n$. We wish to bootstrap a (classical) approximate Boson Sampling oracle A to approximate $|\mathrm{Per}(X)|^2$ within additive error $\pm\epsilon n!$, with success probability at least $1-\delta$ over the random choice of X, in class $\mathrm{FBPP}^{\mathrm{NP}^A}$. (To be technically correct, we use the base class FBPP here in place of BPP; the former is the function analogue of BPP which is allowed to output strings longer than length 1.) For this, we will need two technical ingredients:

1. The Hiding Lemma.
2. Stockmeyer's approximate counting algorithm.

We will introduce these when the time comes; for now, let us begin with the "naive" approach for solving GPE using A.

The "naive" approach. As suggested at the end of Section 2, we will attempt to embed $n \times n$ matrix X in the top left corner of an $m \times m$ unitary U, so that simulating Boson Sampling on U will output configuration $|1^n\rangle \in \Phi_{m,n}$ with probability precisely $|\mathrm{Per}(X)|^2$. This gives us the ability to sample according to $|\mathrm{Per}(X)|^2$ — to then convert this into the ability to estimate the scalar $|\mathrm{Per}(X)|^2$ itself, we apply Stockmeyer's algorithm.

Exercise. Given the ability to run an experiment which accepts with probability $0 < p < 1$, what is the naive way to estimate the scalar p? Why does this approach not necessarily work to estimate $|\mathrm{Per}(X)|^2$ above (i.e. why do we seem to need Stockmeyer's algorithm)?

To begin, recall that X is $n \times n$. Let our Boson Sampling setup have n photons and $m \gg n$ modes (e.g. $m = O(\frac{1}{\delta} n^5 \log^2 \frac{n}{\delta})$ suffices). By rescaling our input as $X' := X/\sqrt{m}$, our task is to estimate

$$|\mathrm{Per}(X')|^2 = \frac{1}{m^n} |\mathrm{Per}(X)|^2$$

within additive error $\epsilon n!/m^n$. We proceed as follows:

1. Embed X' as the top-left $n \times n$ principal submatrix of a unitary U (we ignore how this is done for the moment).
2. Run the Boson Sampling oracle A with input (U, ϵ, r) for uniformly random r and inverse polynomial ϵ. This allows us to sample from distribution D'_U such that $|D_U - D'_U| \leq \epsilon$. Now, if it were true that $D_U = D'_U$, then we would be in luck, since the probability of observing precisely one photon in each of the first n modes is
$$\mathrm{Pr}_{D_U}[1^n] = \frac{|\mathrm{Per}(U_{1^n,1^n})|^2}{(1!)^n(0!)^{m-n}} = |\mathrm{Per}(X')|^2.$$
Given the ability to sample outcome 1^n with probability $|\mathrm{Per}(X')|^2$, we can now convert this to the ability to approximate the scalar $|\mathrm{Per}(X')|^2$ itself via the following theorem.

Theorem 6 (Stockmeyer's approximate counting). *Given Boolean function $f : \{0,1\}^n \mapsto \{0,1\}$, let p be the probability that a uniformly random input x causes f to accept, i.e.*
$$p = \mathrm{Pr}_{x \in \{0,1\}^n}[f(x) = 1] = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x).$$
For any error tolerance $g \geq 1 + \frac{1}{\mathrm{poly}(n)}$, p can be estimated within multiplicative error g in $\mathrm{FBPP}^{\mathrm{NP}^f}$.

Exercise. In an earlier exercise, we said the naive approach for exploiting the ability to run an experiment which accepts with probability $0 < p < 1$ into an estimation of p itself did not work. Why does Stockmeyer's algorithm get around this problem? (Hint: Does Stockmeyer's algorithm restrict p in any essential way?)

Reflection. It is now clear why Theorem 6 works only for *classical* algorithms; namely, any classical oracle implementation is just a Turing machine encoding a Boolean function $f : \{0,1\}^n \mapsto \{0,1\}$ (up to additional inputs to the machine, such as the random string r). This allows us to apply Stockmeyer's algorithm to count the number of satisfying assignments to f, and hence to estimate p. If A were instead to be *quantum*, an analogous statement is not known[3].

The problem with the naive approach. In our discussion of Stockmeyer's algorithm above, we assumed the Boson Sampling oracle is *perfect*, i.e. $D_U = D'_U$. However, recall that in our setup, A only satisfies $|D_U - D'_U| \leq \epsilon$. So we must ask: *How badly does this error affect our desired sampling outcome, $S = 1^n$?* Intuitively, since $\epsilon \in O(1/\operatorname{poly}(n))$, and since there are exponentially many (i.e. $\binom{m}{n}$ for $m \gg n$) possible experimental outcomes or configurations in $\Phi_{m,n}$, the *expected* error per configuration is exponentially small in n. However, we are not interested in *most* configurations — we are only interested in the output configuration $S = 1^n$. And it is entirely possible, since we make no assumptions about A, that *all* of the ϵ sampling error A makes is concentrated on this one outcome 1^n we care about (i.e. all other outcomes $S' \neq 1^n \in \Phi_{m,n}$ are sampled perfectly by A). This is a huge problem — $\Pr_{D_U}[1^n]$ could be exponentially small in n, whereas ϵ is as large as inverse polynomial in n, potentially wiping out our estimate of the former.

The final ingredient: The Hiding Lemma. To resolve this problem, let us revisit why the sampling outcome $S = 1^n$ encoded the permanent of X' in the first place.

Exercise. Argue that $S = 1^n$ encodes $|\operatorname{Per} X'|^2$ only because we embedded X into the top-left $n \times n$ principal submatrix of unitary U. In other words, suppose we instead embed X (e.g.) into rows 2 to $n+1$ and columns 1 to n of U — which output configuration $S \in \Phi_{m,n}$ would now encode $|\operatorname{Per} X'|^2$?

As the exercise above suggests, the output configuration S we care about depends entirely on *where* in the matrix U we embed X'. Thus, if oracle A makes a large error on output configuration 1^n, no problem — we simply encode X' elsewhere in U. Of course, *a priori* we have no idea on which configurations A makes an error. So the obvious thing to do is to embed X' in a *random* $n \times n$ submatrix of U. How to implement this, however, is not obvious — in particular, we need to do the embedding cleverly so that U looks completely random to A (i.e. A should have no way of distinguishing where in U the submatrix X' was hidden). By doing so, we may argue that *even if A acts adversarially*, it cannot corrupt the particular output configuration we care about with non-negligible probability. Thus, the error incurred by A can be neglected with high probability, and we can then apply the "naive" approach using Stockmeyer's algorithm above.

This "hiding/embedding" trick is accomplished by the Hiding Lemma.

Lemma 7 (Hiding Lemma)**.** *Fix n, $\delta > 0$, and $m \gg n$ (e.g. $m \in \Theta(\frac{1}{\delta} n^5 \log^2 \frac{n}{\delta})$ suffices). There exists a $\mathrm{BPP}^{\mathrm{NP}}$ algorithm B which, given an $n \times n$ matrix X with entries independently and identically distributed as $\mathcal{N}(0,1)$, behaves as follows: B succeeds with probability $1 - O(\delta)$ over the choice of X, and conditioned on succeeding, outputs a random $m \times m$ unitary U according to a distribution D_X, such that both of the following hold:*

1. *Assuming B succeeds with non-zero probability on X, we have that $X' = X/\sqrt{m}$ occurs as a uniformly random $n \times n$ submatrix of U.*

[3]Intuitively, the problem quantumly is that "computational paths" can destructively interfere and cancel out, so that the acceptance probability p of a quantum circuit is no longer the sum of a set of non-negative numbers, but the sum of both positive and negative numbers.

2. *The matrix U looks "uniformly random". (Formally, the distribution over $m \times m$ matrices produced by first drawing X, and then conditioning on B succeeding on input X, is precisely the Haar measure over $m \times m$ unitaries.)*

In words, the Hiding Lemma does exactly what we need — property (1) states that the matrix for which we wish to compute the permanent, X', is embedded in a *random* location of unitary U. If we could assume the approximate Boson Sampling oracle A is "honest", this alone might suffice. However, since we cannot assume anything about A, we have our failsafe — property (2) says that not only is X' randomly embedded into U, but that there is information theoretically no way to tell, given U alone, *where* X was hidden. (Here we are implicitly using the fact that if two distributions, or more generally density operators, have trace distance 0, then there is no possible measurement which distinguishes these operators with probability better than random guessing.) Thus, it can be shown that the $\epsilon \in O(1/\operatorname{poly}(n))$ error incurred by our approximate Boson sampling oracle is highly unlikely to affect the particular output configuration $S \in \Phi_{m,n}$ which corresponds to where X' was hidden. Thus, we can apply Stockmeyer's algorithm to the choice of S output by the Hiding Lemma to estimate X', yielding the claim of Theorem 4.

We close by remarking that the proof of the Hiding Lemma is rather technical, and thus omitted for the purposes of this lecture. However, it is not entirely surprising that random submatrices of Haar-random unitaries look "Gaussian" — a standard approach for sampling Haar-random unitaries is roughly to begin by picking all entries as i.i.d. Gaussian entries, and then adjusting all columns to be orthonormal via the Gram-Schmidt procedure. Indeed, this is precisely one of the reasons that GPE is defined according to Gaussian instances to begin with. (Of course, the proof of the Hiding Lemma remains non-trivial.)

$\square$