On "The Computational Complexity of Linear Optics" by Scott Aaronson and Alex Arkhipov

Maris Ozols

April 22, 2011

Contents

1 Introduction

The great promise of the field of quantum computing is that one day we will be able to build devices that operate according to the laws of quantum mechanics and lets us solve computational problems that are not tractable on a classical computer. However, as of today, there is still a long way to go before this goal can be reached, since physically implementing quantum computers is very challenging and their computational power is not completely understood either.

The assertion that quantum computers are more powerful that classical ones is in sharp contrast to the Church–Turing thesis—another reasonable assumption, motivated by several decades of work that begun in the early days of computer science. At that time mathematicians, such as Kleene, Church, and Turing, were trying to formalize the notion of computation by proposing a wide range of models, such as *recursive functions*, λ-*calculus*, and *Turing machines.* However, these models turned out to be polynomially equivalent, thus indicating that the notion of efficient computability is of a fundamental significance in mathematics. Nowadays Turing machine has become the "golden standard" of a model for efficient computation, and the intuition that has resulted from this work is summarized as

The extended Church–Turing thesis. *A computational problem that is efficiently solvable by any realistic physical device, is also efficiently solvable by a probabilistic Turing machine.*

Resolving the tension between these two opposing views is a fundamental problem of modern science, and solving it would be a major scientific breakthrough. Thus said, even a partial answer (such as the one given by the paper [1] being discussed) gives a non-trivial insight into the problem. From this point of view it is also important to perform experiments to gather evidence on the either side of the argument.

Clearly, it is not possible in principle to prove the validity of the Church–Turing thesis, since one can never be sure that all laws of physics have been discovered. However, it is possible to disprove it or at least seriously challenge its validity, e.g., by showing that quantum algorithms can solve some computational problems significantly faster than the best known classical algorithm. In this way we can base our beliefs about the validity of the Church–Turing thesis on our intuition from complexity theory about hardness of certain computational problems.

A well known example of such instance is Shor's algorithm for the problem of factoring integers. His result can be rephrased as follows:

> *It is not possible to efficiently simulate a quantum computer on a classical one, unless it is possible to efficiently factor integers.*

However, it is not clear how hard it is to factor on a classical computer. I do not have a good intuition about this, but apparently a polynomial time algorithm for factoring would be less surprising for computer scientists than some other structural changes in complexity theory, such as collapse of the polynomial hierarchy. To explain this better, let me give a brief introduction to complexity theory.

1.1 Complexity theory

Typically, the type of problems studied in complexity theory are the so-called *decision problems*, i.e., problems for which the algorithm has to output either "yes" or "no" (e.g., the problem of checking if a given number is a prime). The complexity of an algorithm is typically characterized by its running time as a function of the input length (e.g., the number of bits necessary to write down the number to be checked for primality).

The simplest complexity class P corresponds to problems that can be solved on a classical deterministic computer in polynomial time. The class NP of problems that can be solved in non-deterministic polynomial time can be characterized as follows:

- for a "yes"-instance there *exists* an advice string that lets the algorithm to verify in polynomial time that this is indeed a "yes"-instance;
- for a "no"-instance there is *no* "advice" that would fool the algorithm in believing that this might be a "yes"-instance.

In other words, the problems in class NP can be formally described using a single existential quantifier ($\exists$).

Complexity classes P and NP lie at the 0th and 1st level of what is known as the *polynomial hierarchy* PH: problems that can be described using the second order logic, i.e., any finite number of existential ($\exists$) and universal quantifiers ($\forall$). It is a major open problem in computer science to decide if the complexity classes P and NP are actually different. The evidence obtained so far seems to suggest that $\mathsf{P} \neq \mathsf{NP}$. If it turns out that $\mathsf{P} = \mathsf{NP}$, this would be very surprising, since that would imply efficient algorithms for many very hard problems. In particular, this would imply the collapse of the polynomial hierarchy. For similar reasons it seems unlikely that the polynomial hierarchy could collapse to a different level. However, there is no such strong evidence that factoring is hard classically—this is the sense in which [1] try to provide a stronger argument against Church–Turing thesis than Shor's algorithm.

Class	Description
P	polynomial time
NP	non-deterministic polynomial time
PH	polynomial hierarchy (second order logic)
BPP	bounded-error probabilistic polynomial time
BQP	bounded-error quantum polynomial time

Table 1: The most important complexity classes.

The complexity classes just mentioned and some other important ones are summarized in Table 1, and the relations between them are illustrated in Figure 1. Here are some well-known inclusions:

$$\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{BQP}, \qquad \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PH}, \qquad \mathsf{BPP} \subseteq \mathsf{PH}.$$

The first one simply says that the models of *deterministic*, *randomized*, and *quantum* computation are increasingly more general. The second in essence says that finding a solution to a computational problem (class P) is at least as hard as guessing it and verifying its correctness (class NP), which in turn is a special case of what is allowed by the second order logic (class PH). The third inclusion is not obvious (it follows by Sipser–Lautemann theorem).

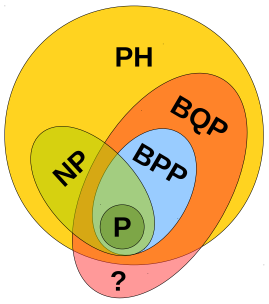

Figure 1: Relations between the complexity classes from Table 1.

Interestingly, the results in paper [1] suggest that quantum computers might have capabilities outside the polynomial hierarchy (in other words, BQP is not contained in PH). If this indeed would be the case, it would be very surprising, since that would mean that quantum computers can solve problems whose solutions cannot even be verified by a classical computer (since $\mathsf{NP} \subseteq \mathsf{PH}$). What is even more surprising, is that this would be achieved by a model that seems to be strictly weaker than the full power of BQP (in fact, it is not even known if one can perform universal classical computation in this model).

2 Computation with non-interacting bosons

2.1 Model of computation

The model of computation considered in [1] is based on identical non-interacting bosons. As a concrete example, one can consider a linear-optical network with single-photon inputs and non-adaptive photon-number measurements. Consider n photons in m modes where $n \leq m \leq \text{poly}(n)$. The state space of the system is spanned by computational basis states of the form $|s_1, \ldots, s_m\rangle$, where s_k is the number of photons in the kth mode and $\sum_{k=1}^{m} s_k = n$. The total dimension of this space is $M = \binom{m+n-1}{n-1}$. For the sake of definiteness, the initial state is taken to be $|1_n\rangle = |1, \ldots, 1, 0, \ldots, 0\rangle$, i.e., the first n modes being occupied by a single photon each. The only transformations that are allowed in this model are of the form $\varphi_n(U)$, where $U \in \mathrm{U}(m)$ is a single-photon unitary on m modes and $\varphi_n : \mathrm{M}_{m\times m} \to \mathrm{M}_{M\times M}$ extends the action of U from 1 to n photons. Here U can be implemented by decomposing it into $\Theta(m^2)$ two-level unitaries, known as Givens rotations, and implementing each of them using two phase shifters and a beam splitter. Finally, a measurement in the computational basis is performed, which gives the number of photons in each mode. The computational problem that is naturally solved by this model is the following:

Definition. *Given n and a description of $U \in \mathrm{U}(m)$, the* BosonSampling *problem is to produce samples from the probability distribution*

$$\Pr[S] := |\langle S|\varphi_n(U)|1_n\rangle|^2. \tag{1}$$

2.2 Transition matrix

Formally, the entries of the transition matrix $\varphi_n(U)$ in equation (1) are defined as follows:

Definition. *If $|S\rangle = |s_1, \ldots, s_m\rangle$ and $|T\rangle = |t_1, \ldots, t_m\rangle$ then for $A \in \mathrm{M}_{m\times m}$ the entries of $\varphi_n(A) \in \mathrm{M}_{M\times M}$ are defined as*

$$\langle S|\varphi_n(A)|T\rangle = \frac{\operatorname{perm}(A_{S,T})}{\sqrt{s_1! \cdots s_m! t_1! \cdots t_m!}}, \tag{2}$$

where $A_{S,T}$ is the $n \times n$ sub-matrix of A induced by rows $(1^{s_1}, 2^{s_2}, \ldots, m^{s_m})$ and columns $(1^{t_1}, 2^{t_2}, \ldots, m^{t_m})$, where k^{s_k} means that symbol k is repeated s_k times. Also, here

$$\operatorname{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)} \tag{3}$$

denotes the permanent *of matrix $A \in \mathrm{M}_{n\times n}$.*

Example. *Let $m = 2$ and $A = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$. If $|S\rangle = |1,1\rangle$ and $|T\rangle = |2,0\rangle$ then the corresponding tuples of indices are $S = (1,2)$ and $T = (1,1)$, thus*

$$A_{S,T} = \begin{pmatrix} a & a \\ c & c \end{pmatrix}, \quad \langle S|\varphi_2(A_{S,T})|T\rangle = \frac{\operatorname{perm}\left(\begin{smallmatrix} a & a \\ c & c \end{smallmatrix}\right)}{\sqrt{1! \cdot 1! \cdot 2! \cdot 0!}} = \frac{2ac}{\sqrt{2}} = \sqrt{2}ac. \tag{4}$$

If we compute this for all basis states with $n = 2$ photons, we get

$$\varphi_2 : \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{matrix} |10\rangle \\ |01\rangle \end{matrix} \mapsto \begin{pmatrix} a^2 & \sqrt{2}ab & b^2 \\ \sqrt{2}ac & ad+bc & \sqrt{2}bd \\ c^2 & \sqrt{2}cd & d^2 \end{pmatrix} \begin{matrix} |20\rangle \\ |11\rangle \\ |02\rangle \end{matrix} \tag{5}$$

Let us give some more intuitive explanation of the expression in equation (2). The denominator just takes care of the normalization and reflects the fact that bosons in the same mode are indistinguishable. To interpret the numerator, assume for simplicity that $|S\rangle = |T\rangle = |1, \ldots, 1\rangle$ so that $A_{S,T} = A$, and think of entries of A as probabilities. If we interpret A_{ij} as the probability for a photon to jump from mode i to j, then a term in equation (3) corresponding to permutation σ describes the probability of photons being permuted according to σ. The same interpretation still works if we have more than one photon in a mode.

The mapping φ_n defined in equation (2) has two very nice properties that are not obvious at all from its definition:

Lemma. *The mapping φ_n has the following properties:*

- *φ_n is a homomorphism, i.e., $\varphi_n(A \cdot B) = \varphi_n(A) \cdot \varphi_n(B)$;*
- *if U is unitary then so is $\varphi_n(U)$.*

These properties imply that φ_n is a unitary representation of the group $\mathrm{U}(m)$. In fact, it is an irreducible representation.

Interestingly, there are alternative ways of defining the same map φ_n. The one most commonly used in physics is in terms of polynomials of creation operators. It is based on the observation that there is a bijection between the quantum states and homogenous polynomials. On the computational basis states this bijection is defined as

$$\phi : |s_1, s_2, \ldots, s_m\rangle \mapsto x_1^{s_1} x_2^{s_2} \ldots x_m^{s_m} / \sqrt{s_1! s_2! \ldots s_m!}, \tag{6}$$

where $x_1, \ldots, x_m$ are commuting variables, with the inverse given by

$$\phi^{-1} : x_1^{t_1} x_2^{t_2} \ldots x_m^{t_m} \mapsto |t_1, t_2, \ldots, t_m\rangle \sqrt{t_1! t_2! \ldots t_m!}. \tag{7}$$

Both ϕ and ϕ^{-1} are extended by linearity to the whole space. In addition, we define a linear map ϕ_U corresponding to unitary $U \in \mathrm{U}(m)$ as

$$\phi_U : (x_1, x_2, \ldots, x_m) \mapsto (x_1, x_2, \ldots, x_m)U. \tag{8}$$

It replaces variable x_k by a linear combination $\sum_{i=1}^{m} x_i U_{ik}$. Then we can obtain the extension of $U \in \mathrm{U}(m)$ to n photons as $\varphi_n(U) = \phi^{-1} \circ \phi_U \circ \phi$.

Another interesting thing to note is that if the variables $x_1, \ldots, x_m$ are assumed to be anti-commutative (e.g., as in the case of fermions), then instead of the permanent one obtains the determinant of the matrix:

$$\det(A) = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}. \tag{9}$$

2.3 Connection to complexity theory

The connection that relates the BosonSampling problem to complexity theory is the observation that the measurement outcome probabilities of a boson computer according to equation (2) are expressed in terms of a permanent. A seminal result in complexity theory says that

Theorem (Valiant [2]). *Computing* $\mathrm{perm}(A)$ *is* #P*-hard.*

Here #P is the complexity class of counting problems[1] where the algorithm has to output the number of advices associated to a "yes" instance of a decision problem in NP, and "#P-hard" means that it is at least as hard as solving any other problem in the class #P.

This is in sharp contrast to the complexity of computing determinant of a matrix in case of fermions, which can be done in time that is polynomial in the size of the matrix (e.g., by using LU, Cholesky or QR decomposition). This striking difference has inspired a joke by computer scientist Avi Wigderson, who apparently once said that "bosons got the harder job".

However, the fact that the probabilities of measurement outcomes can be expressed in terms of the permanent (see equation (2)), does not mean that bosons actually "compute" the permanent. All that the boson computer is doing is *sampling* from this probability distribution. Note that in general this is not sufficient to estimate any particular such probability, since different runs of the same experiment need not have the same outcome. In fact, it might be exponentially unlikely to obtain the desired measurement outcome, thus in principle it may not be feasible to estimate the associated probability. Nevertheless, one can still use results from complexity theory to show that a polynomial time classical algorithm for the BosonSampling problem would have significant consequences in complexity theory, such as collapse of the polynomial hierarchy.

[1] The output of an algorithm for a function problem is an integer (a value of the faction) rather than a single bit as in the case of decision problems.

As a side note, it is worth mentioning that Lesie Valiant is a computer scientist and his result seemingly has nothing to do with physics. However, the gadgets used in his proof [2] can be directly interpreted in terms of devices used in linear optics experiments, such as phase shifters and beam splitters. In some sense one can even say that he has abstractly rediscovered bosons. Moreover, his result can also be seen as a consequence of the KLM universality result [3]. Recently this connection was made explicit by Scott Aaronson who (on the occasion of Leslie Valiant receiving Turing's award) completely re-derived Valiant's proof in terms of linear optics [4].

3 Main results

This section contains a summary of the two main results of the paper.

Theorem (exact case)**.** *The exact* BosonSampling *problem is not efficiently solvable by a classical computer, unless polynomial hierarchy collapses.*

Even though this result seems to be very strong, it is not entirely satisfactory, since we require exact simulation of the boson computer. Clearly, any physical implementation of a boson computer would not be perfect, therefore it would not be able itself to solve the BosonSampling problem exactly. Thus it is unfair to require that the classical computer does so.

A significant part of the paper [1] is devoted to trying to extend the above theorem by showing the hardness even of approximately solving the BosonSampling problem. Authors succeed in showing that this is the case, under two conjectures about permanents of random Gaussian matrices. Here is a non-technical version of the result:

Theorem (approximate case)**.** *If the following two conjectures are true:*

1. *the permanent of a random Gaussian matrix is* #P*-hard to approximate and*

2. *it is not too concentrated around 0,*

then it is not possible to approximately solve the BosonSampling *problem, unless polynomial hierarchy collapses.*

Both conjectures are backed up by numerical evidence, however no rigorous proof is known.

The proof of the above theorem contains the following technical result that might be of independent interest, namely: any $m^{1/6} \times m^{1/6}$ submatrix of an $m \times m$ uniformly (according to the Haar measure) distributed random unitary matrix is close to a matrix of independent identically distributed Gaussians.

4 Experimental feasibility

Linear optics seems to be the obvious candidate platform in which one could attempt to experimentally implement a boson computer. However, there are alternative possibilities, such as bosonic excitations in solid-state systems.

The experiment would consist in preparing photons using single photon sources in the $|1_n\rangle$ state, using beam splitters and phase shifters to implement a transformation associated to a given m-mode unitary U, and using photodetectors to perform the readout.

The order of parameters where the experimentally obtained evidence could be regarded as significant is $n = 10$ and $m = 20$. However, at present the only experiment that has been performed is the Hong-Ou-Mandel dip which corresponds to $n = 2$. With the current technology it seems that the $n = 3$ and $n = 4$ cases should be possible. Probably a somewhat unexpected limitation is that for large values of n and m it would not anymore be possible to verify that the measurement outcomes are consistent with predictions, since the required amount of computation on a classical computer would not be feasible.

The main advantages of performing experiments on a boson computer is that such computer seems to be easier to build, compared to a full-scale quantum computer. The main reasons are that no interactions between pairs of bosons are needed, and that bosons are never used as qubits, so there is no need to store them (which might be a problem in the case of photons). Note that the absence of interactions in the boson computer does not imply that there is no entanglement in the system. In fact, there is a certain amount“free entanglement” due to the exchange symmetry of bosons.

The most difficult challenge that any implementation might change is making sure that all bosons are indistinguishable, i.e., the exchange symmetry is preserved. For example, in a linear optics implementation one has to make sure that all n photons arrive at the destination at the exact same time. In addition to this one also needs good photon sources and detectors.

References

[1] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. 2010. arXiv:1011.3245.

[2] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

[3] Emanuel Knill, Raymond Laflamme, and Gerard J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46–52, 2001. arXiv:quant-ph/0006088, doi:10.1038/35051009.

[4] Scott Aaronson. A linear-optical proof that the permanent is #P-hard. 2011. Available from: http://www.scottaaronson.com/papers/sharp.pdf.