Review article **Open Access**

Hari Krovi*

Models of optical quantum computing

DOI 10.1515/nanoph-2016-0136
Received August 2, 2016; accepted November 9, 2016

Abstract: I review some work on models of quantum computing, optical implementations of these models, as well as the associated computational power. In particular, we discuss the circuit model and cluster state implementations using quantum optics with various encodings such as dual rail encoding, Gottesman-Kitaev-Preskill encoding, and coherent state encoding. Then we discuss intermediate models of optical computing such as boson sampling and its variants. Finally, we review some recent work in optical implementations of adiabatic quantum computing and analog optical computing. We also provide a brief description of the relevant aspects from complexity theory needed to understand the results surveyed.

Keywords: optical quantum computing.

1 Introduction

Quantum computing is a model that holds the promise of exponential speed-up for several relevant problems such as factoring, finding discrete logarithms [1], solving Pell's equation [2], solving hidden subgroup problems [3–5], and simulating physical systems [6–8]. These problems have applications in public key cryptosystems and modeling of quantum systems such as high energy physics and condensed matter systems. Another important application of quantum algorithms is in the understanding of models of quantum chemistry, which would lead to efficient design of drugs and the analysis of their effects. All of these applications rely on efficient quantum algorithms and, in fact, represent problems for which no efficient classical algorithms exist. There are several other problems for which quantum systems provide a more modest advantage than an exponential speed-up. These problems include collision finding, which is applicable in secure hashing, solving differential equations using the finite element method [9], and search on graphs for marked vertices [10]. All these examples provide evidence that this model of computing is potentially more powerful than classical computing. However, it should also be pointed out that there is currently no evidence that quantum computers can solve NP-hard problems in polynomial time. The class NP stands for nondeterministic polynomial time (defined more explicitly in the next section) and consists of problems whose solution can be checked polynomial time by deterministic classical computers. The importance of this class stems from the fact that several practical problems lie in this class. Despite the lack of evidence of an exponential quantum speed-up for problems in this class, there are a lot of examples where one has a polynomial speed-up (such as a square-root speed-up) for several problems in this class (see, for example, [11]).

This speed-up over conventional digital computers has led to numerous proposals to implement quantum computers. Some of the proposed implementations are on superconducting qubits [12], ion traps [13], quantum dots [14], and quantum optics [15]. Each implementation scheme has its relative advantages as well as drawbacks. Most of these implementations focus on a model of quantum computing known as the circuit model. This model is closest to classical digital computers in the sense that the information is encoded in bits (or qubits in the quantum case). Quantum logic gates corresponding to a quantum circuit are applied to the qubits to get the desired output. Many of the quantum algorithms for which there is an exponential speed-up over classical algorithms are designed in this model. There are, however, several other quantum computing models. Some of the well-known ones include the cluster state model (which is relevant to optical quantum computing), the adiabatic model, and the quantum walk model. It can be shown that these models are equivalent to each other, which implies that any algorithm in one model can be converted to an algorithm in the other with a polynomial overhead.

Optical implementation of quantum gates is important for several reasons. First, communicating over quantum channels, such as the bosonic channel, requires one to implement quantum optical gates. In fact, optimal decoding of the communication codes usually involves quite general quantum circuits [16]. Second, quantum

***Corresponding author: Hari Krovi,** 10 Moulton Street, Mail Stop 19/1A, Quantum Information Processing Group, Raytheon BBN Technologies, Cambridge, MA 02138, USA, e-mail: hari.krovi@raytheon.com

key distribution (QKD) over long distances needs optics. QKD [17] is a scheme in which two parties can exchange secret keys that can be used in several cryptographic primitives. Long-distance QKD requires the use of repeaters, which are devices with quantum memory that can implement quantum gates, especially the so-called Bell measurement (which is a measurement that can be used to generate entanglement between two parties). However, the deployment of such schemes has been hampered by low rates and the high cost of infrastructure in the form of quantum memories, reliable single photon, and entangled sources. Recently, in [18], schemes that relax the requirements on the sources have been proposed. The tradeoff is that in this scheme, the detectors need to be more sophisticated and need to resolve multiple photons. However, this is considered a more plausible requirement by experimentalists. Several schemes for such repeaters exist using rare earth atoms [19, 20], ion traps [21], and optical repeaters [22]. All of these would require implementing optical gates for quantum information processing (even though they may not be universal quantum computers). In all of these schemes, one needs a reliable on-demand quantum memory and the ability to produce high-quality entangled states. In fact, these capabilities also allow one to do arbitrary quantum computation as well. Third, even for computing applications, optics becomes important when other schemes run out of resources. For example, since superconducting qubits are required to be stored at milli-Kelvin temperatures, this way of implementing quantum computers is fundamentally limited by the number of qubits that can be stored in a dilution refrigerator or a server of qubits. When one refrigerator runs of space, one can use several servers; however, they must be connected to each other in order to transport quantum information without measuring. This can be done using quantum optics, and it leads to the challenging problem of transferring quantum information from the superconducting device to an optical state. It also points to the need to develop high-fidelity quantum optical gates. These quantum optical interconnects can help increase the number of qubits used in computation by combining several servers. Indeed, there has been tremendous experimental progress in demonstrating various quantum computing models over a small number of modes and photons, but a detailed discussion of this progress and challenges in scaling up is outside the scope of this review. However, we provide a short review of experimental progress in Section 6.

In this review article, we focus on the optical implementation of some quantum computing models. Specifically, we focus on schemes to implement the circuit model, the adiabatic model, as well as some models believed to be intermediate between classical computing and quantum computing. In Section 2, we briefly review the necessary complexity theory. In Section 3, we discuss optical implementations of digital quantum computation. In Section 4, we discuss other models that are easier to implement using optics and are not as powerful as universal quantum computers but that have quantum computational power greater than classical computers do (assuming standard conjectures in complexity theory). Then in Section 5, we discuss analog models of quantum computation and some recent work in implementing them in optics. Section 6 discusses some of the experimental progress in building these models. Finally, in Section 7, we present some conclusions and outlook.

2 Brief introduction to complexity theory

In this subsection, we explain some core concepts of computational complexity that are useful to understand the power of various models of optical computing. The main reason for introducing complexity theory is that it is the right framework to discuss the computational power of various models. In addition to the classes $\mathcal{P}$ and $\mathcal{NP}$, which many may already be familiar with, we define the quantum class $\mathcal{BQP}$ as well as the polynomial hierarchy $\mathcal{PH}$, which is a generalization of $\mathcal{P}$ and $\mathcal{NP}$. Rather than use the formal definitions from computer science literature, we will present somewhat informal but, hopefully, more intuitive definitions. The reason for this choice is to make contact with the kinds of problems one hopes to solve using these models. For detailed definitions, see [23].

2.1 $\mathcal{P}$, $\mathcal{NP}$, and other classes

The class $\mathcal{P}$ is the set of decision problems (i.e. problems with a "yes" or "no" answer) that can be solved in time polynomial in the number of bits (n) used to describe the problem (this is typically called the input size). $\mathcal{P}$ is also the class of problems whose solution can be found in deterministic polynomial time. One usually refers to a polynomial time algorithm as being in $\mathcal{P}$ or equivalently as being *efficient*. In all of the definitions, we will refer to n as the input size and all scalings are in terms of this quantity. The class $\mathcal{NP}$ is the set of decision problems whose solutions are easy to check in deterministic

polynomial time. In fact, rather than using the solution itself in the definition, one can define this class with respect to a more general witness (which is a function of the problem instance) such that it is easy to check from the witness that a given string is a solution. While it may seem at first sight that being able to check might give enough power to find the solution, there is overwhelming evidence to suggest that this is not true. In fact, this dichotomy between efficiently checking solutions and efficiently finding solutions lies at the heart of the problem of whether two classes $\mathcal{P}$ and $\mathcal{NP}$ are equal or not (a major open question in computer science).

The quantum version of $\mathcal{P}$ is denoted $\mathcal{BQP}$ and stands for bounded quantum polynomial time. It is the class of decision problems that can be solved in polynomial time on a *quantum* computer. While $\mathcal{BQP}$ contains $\mathcal{P}$, it is unknown if there are problems outside of $\mathcal{P}$ that are in $\mathcal{BQP}$. Similarly, the relationship between $\mathcal{BQP}$ and $\mathcal{NP}$ is unknown. Currently, there is no evidence to suggest that $\mathcal{BQP}$ contains $\mathcal{NP}$. The relationship between these classes is depicted in Figure 1.

The concept of $\mathcal{NP}$ completeness, in a sense, distills the hardest problems in the class $\mathcal{NP}$. When a problem is $\mathcal{NP}$ complete, it means first that the problem is in the class $\mathcal{NP}$ (i.e. it is a decision problem with an efficient verifier to check for membership) and that every other problem in the class can be *reduced* to it. A reduction here is essentially a polynomial time classical algorithm that takes the solution of the $\mathcal{NP}$ complete problem and converts it to the solution of an instance of an arbitrary problem in $\mathcal{NP}$. This would mean that if we can solve an $\mathcal{NP}$ complete problem, then we can solve every other problem in $\mathcal{NP}$ with only a polynomial overhead. This overhead is needed to convert the solution of one problem to the solution of the other. In this sense, $\mathcal{NP}$ complete problems are the hardest ones in the class $\mathcal{NP}$. However, there are several problems that are as hard as any problem in $\mathcal{NP}$; i.e. a solution to them would imply a solution to every $\mathcal{NP}$ problem, but they are not decision problems. For instance, finding a solution to a constraint satisfaction problem (CSP) rather than deciding if there is a solution or not is one such problem that is not in $\mathcal{NP}$. However, its solution can be used to solve all the problems in $\mathcal{NP}$. Such problems are called $\mathcal{NP}$ hard problems. Most problems that we encounter in practice are not decision problems since we would like to know the solutions or at least some properties of the solutions.

A generalization of these classes is the so-called polynomial hierarchy $\mathcal{PH}$. The description here is adapted from [24]. The $\mathcal{PH}$ is a collection of (infinitely many) classes that are built up on $\mathcal{P}$ and $\mathcal{NP}$. In fact, $\mathcal{P}$ is the class at level 1 of the hierarchy and $\mathcal{NP}$ is the class at level 2. In order to understand the other levels, let us first revisit the class $\mathcal{NP}$. It can be defined from the point of view of CSPs. Suppose we have a CSP with constraints C_i (denoted together as C) that act on bit strings. A solution z is a bit string that satisfies all the constraints, and this problem is in $\mathcal{NP}$ if there is an efficient verifier V. This means that when V is given, the problem instance (i.e. the constraints) and a bit string z can check if z satisfies the constraints; i.e. $V(C, z)=1$. The decision version of this problem and one that (strictly speaking) lies in $\mathcal{NP}$ is "does there exist a z such that $V(C, z)=1$?". The generalization of this to verifiers that accept two bit strings is the third level of $\mathcal{PH}$. In other words, the problems in the third level of $\mathcal{PH}$ are of the type "for all z_1, does there exist a z_2 such that $V(C, z_1, z_2)=1$?". For reasons similar to the ones that lead us to believe $\mathcal{P}\neq\mathcal{NP}$, we do not expect the third level to equal the second (i.e. the third level does not *collapse* on to the second). Building in this way, we can construct infinitely many levels, which do not collapse onto the first or, indeed, we do not expect them to collapse at any level. The reason this hierarchy is considered is that if classical simulation of certain models of quantum computing is possible, then $\mathcal{PH}$ collapses to the third level. Since we do not expect the hierarchy to collapse at any level, this gives evidence of the power of these quantum models. Put another way, this is evidence that classical computers cannot simulate these quantum models in polynomial time.

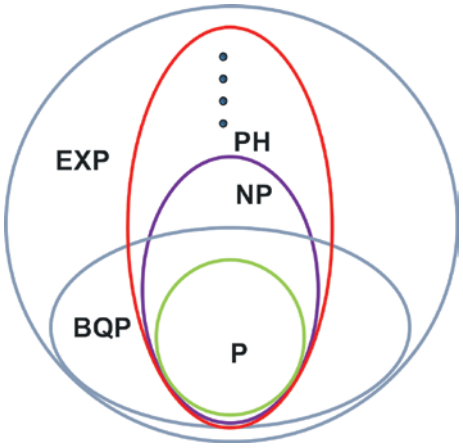

Figure 1: Conjectured relationship between complexity classes in the polynomial hierarchy.

It should be mentioned that there is substantial evidence that $\mathcal{P}\neq\mathcal{NP}$. If, however, it turns out that $\mathcal{P}=\mathcal{NP}$, then it is known that the polynomial hierarchy collapses. In this scenario, the figure depicted above would contain only the class $\mathcal{P}$. Similar to the polynomial hierarchy, there is another hierarchy called the exponential hierarchy. This builds on classes $\mathcal{EXP}$ and $\mathcal{NEXP}$, which stand

for deterministic exponential time and nondeterministic exponential time computing, respectively. These classes are exponential time analogues of $\mathcal{P}$ and $\mathcal{NP}$.

2.2 Approximate solutions and random instances

In this subsection, we explain variations in the definition of $\mathcal{NP}$, types of algorithms that address the problems with these variations, and what one can realistically hope in terms of speed-ups. The characterization of a problem in $\mathcal{NP}$ as a CSP supposes that a solution satisfies all the constraints. In most practical situations where such problems might arise, we may not care about a string that satisfies all the constraints but rather a large fraction of them, say 90%. One might hope that this change in the definition of the problem might make it easier to find a polynomial time algorithm (or make it possible for it to exist). Unfortunately, this is not true for a lot of $\mathcal{NP}$ complete problems. For several problems such as traveling salesman, 3-SAT, and others, determining if a string satisfies more than a certain fraction of constraints is $\mathcal{NP}$ complete. This threshold fraction seems to be different for different problems in $\mathcal{NP}$. This is essentially the content of a famous theorem in computer science called probabilistically checkable proofs; i.e. for constraint satisfaction and several other $\mathcal{NP}$ complete problems, there exists a threshold of approximation such that no polynomial time algorithm exists that can achieve an approximation ratio better than the threshold unless $\mathcal{P}=\mathcal{NP}$. Despite these seemingly negative results, approximation algorithms can provide the best possible solutions to NP-hard problems for suitable approximation ratios. In fact, if $\mathcal{P}\neq\mathcal{NP}$ as most evidence suggests, then approximation algorithms are the only way to "solve" NP-hard problems in polynomial time. For approximation ratios where the problem is NP-hard, polynomial or constant factor speed-up is still possible with quantum algorithms.

A second direction that we could tweak the definition of $\mathcal{NP}$ completeness in hopes of finding efficient algorithms is to impose a distribution over the problem instances. The definition of $\mathcal{NP}$ completeness or $\mathcal{NP}$ hardness is a statement when all instances of the problem are considered. In other words, the hardest instances of the problem make it $\mathcal{NP}$ hard or complete. One might wonder how often these occur in practical situations or, more precisely, if one picks instances at random from some reasonable distribution, then it may turn out that with high probability, a given instance can be solved efficiently. However, this turns out to be false as well. It can be shown that for every $\mathcal{NP}$ complete problem, there exists a nontrivial distribution that makes it $\mathcal{NP}$ complete. These two examples show that the property of $\mathcal{NP}$ completeness is robust to variations in its definition [23].

2.3 Types of speed-up

In designing computing models or novel processors, one need not aim to solve $\mathcal{NP}$ complete problems efficiently. There are several types of speed-up over the state-of-the-art (SOA) that one can aim for and that will still make it worthwhile to develop the model. These are explained below. Although these are important types of speed-up, they are by no means exhaustive. Indeed, there are infinitely many types of speed-up as one can obtain any function of the run-time of the SOA as the run-time of the algorithm of interest.

1. *Exponential.* This is the speed-up obtained when the algorithm of interest is exponentially better than the run-time scaling of the SOA algorithm. More precisely, if the run-time of the algorithm of interest is $T(n)$ [where $T(n)$ is some function of the input size n], then the run-time of the SOA is $T=O(2^{T(n)})$, for example.
2. *Subexponential.* This kind of speed-up can be thought of as the next best one, although in practice, it can be just as good. It is defined as follows. If the run-time of the algorithm of interest is $T(n)$, then the run-time of the SOA is $T'=O(2^{\sqrt{T(n)}})$.
3. *Polynomial.* The next one we would like to describe is a polynomial speed-up, which means that the run-times of the SOA and the new algorithm are related by a polynomial, i.e. $T'=O(T^{c_1}(n))$, where c_1 is some constant independent of n.
4. *Constant factor.* The last one we would like to mention is, in a sense, the least interesting but which nevertheless can be substantial in some cases. This is called a constant factor speed-up where $T=c_2T(n)$, where c_2 is a constant. This kind of speed-up can be substantial if the constant c_2 is large. In complexity theory, these are usually ignored as one is only interested in functions growing as n.

There are several more that one can construct that are in between exponential and polynomial. These different classes of computational advantages point to the fact that models can be useful even if they do not lead to an exponential speed-up. An exponential cost solution in a particular model may not improve in *scaling* over the SOA but could lead to a substantial improvement via a large constant factor advantage.

3 Computing with optical circuits

In this section, we survey various schemes to perform quantum computation digitally, i.e. by encoding information into qubits and using optical gates to perform the computation. For any computation classical or quantum, there is a notion of a universal gate set. For classical computation, the NAND gate suffices, and for reversible classical computation, the TOFFOLI gate suffices [25]. For quantum computation, one needs to append this with other gates. One standard gate set used in quantum computing consists of the CNOT gate; Hadamard gate denoted as H; phase gate, sometimes denoted as S; and its square-root, denoted T. The TOFFOLI gate is a three-bit logic gate, where if the first two bits (the control bits) are set to 1, then it toggles the last bit (the target bit). The CNOT gate is similar except that it has a single control bit; i.e. it acts on two bits (control and target), and if the control bit is set to 1, then it toggles the target bit. All the gates mentioned above act on either a single qubit or a pair of qubits (except the TOFFOLI, which acts on three qubits), and hence, they are either 2×2 or 4×4 matrices. They are defined as follows.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1}$$

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2}$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix} \tag{3}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \tag{4}$$

The implementation of these gates in optics depends on the type of encoding used. The difficulty of performing the computation depends on encoding the states as well as implementing the gates. In the schemes presented below, one can see this tradeoff as some schemes shift the difficulty toward encoding, such as Gottesman-Kitaev-Preskill (GKP) and coherent state quantum computing (CSQC) (explained below), so that gate implementation is easier whereas some, like Knill-Laflamme-Milburn (KLM), have complicated gate operations but relatively easier encoding. Before we embark on the descriptions of the models and encodings, we describe here the types of optical gates at our disposal and the relative difficulty in implementing them. First, we briefly explain creation and annihilation operators that are used to describe the optical gates. The space in which the quantum optical computation takes place is called Fock space. It is spanned by states that have a specific number of photons in each mode. On this space, one can define a creation operator (for each mode) that creates a photon in that mode, i.e. $a^\dagger|n\rangle = \sqrt{n+1}\,|n+1\rangle$. An annihilation operator is its Hermitian conjugate and essentially removes a photon from that mode, i.e. $a|n\rangle = \sqrt{n}\,|n-1\rangle$. Any operation in this space (the Hilbert space) can be written as a polynomial in the creation and annihilation operators. The passive mode transformations are beamsplitters and phaseshifters, defined as follows (acting on the mode creation and annihilation operators). In the expressions below, a and b are the annihilation operators of two modes:

$$U_B = \exp(-i\eta(ab^\dagger + a^\dagger b)) \tag{5}$$

and

$$U_P = \exp(-i\theta a a^\dagger). \tag{6}$$

when this set is appended with the squeezing operator defined below, it generates the set of Gaussian operations. The squeezing operator is

$$U_S = \exp\left(\frac{1}{2}(za^2 - z^* a^{\dagger 2})\right). \tag{7}$$

Decomposing an arbitrary Gaussian operator into a sequence of the above primitives is described in [26]. However, Gaussian operations alone cannot produce all unitary operations. One needs at least one non-Gaussian operator in the toolkit to generate arbitrary unitary operators. Moreover, if a quantum computer was designed using only the set of Gaussian operations, then it can be simulated efficiently using classical computers. It turns out that only one (nontrivial) non-Gaussian operator suffices to complete the universal gate set [27]. Some non-Gaussian operations include nonlinear interactions such as self-phase modulation, cross-phase modulation, sum (or difference) frequency generation, and four wave mixing. Out of these, perhaps cross-phase modulation is most common. However, implementing these interactions with high enough strength is very challenging. The cross-phase modulation (or cross-Kerr gate) can be described as the following unitary:

$$U_{XPM} = \exp(-i\chi a^\dagger a b^\dagger b), \tag{8}$$

where $\mathcal{X}$ can be thought of as the strength of the interaction. These operations form progressively larger sets containing the previous ones as described above. This summarizes the possible optical gates that can be applied.

We can now describe the common detection or measurement operations. Since these may be familiar to most readers, we will not discuss them in detail, but rather list them and mention the class they belong to. The common types of detectors are homodyne, heterodyne, single photon detectors, and photon number resolving detectors. The first two fall into the category of Gaussian operations, where as the last two involve non-linear processes. These detectors make it possible to move the complexity away from state preparation or gate implementation if needed.

3.1 Types of encodings into optical states

1. *Unary* [28]: In this type of encoding shown in Figure 2, there is a single photon in one out of N modes, where N is the total number of orthogonal states in the Hilbert space. If the number of qubits is n, then $N=2^n$; i.e. it is usually exponential in the input size. This encoding is very inefficient since it uses an exponential number of modes. However, the advantage is in the implementation of quantum gates. In this encoding, any unitary operation can be viewed as a unitary operation on the modes. Any unitary operation on the modes, in turn, can be decomposed into a network of beamsplitters and phaseshifters. In other words, one only needs passive mode operations to implement any unitary. The decomposition algorithm of any passive mode unitary into a network of beamsplitters and phaseshifters is described in [29].
2. *Dual-rail* [30, 31]: This encoding is a standard encoding that uses a single photon in one out of two modes to represent a qubit shown in Figure 3. If the photon is in the left mode, then it is the qubit zero, and if it is in the right one, it is a one. With this encoding, it was shown in [30] that the universal set of quantum gates can be implemented using beamsplitters, phaseshifters, and cross-Kerrs of strength π. However, implementing cross-Kerr gates of strength π is extremely hard. As described above, this encoding takes two optical modes. To encode the bit zero, we put a single photon in the left mode and to encode a one, there is a single photon in the right mode. This encoding is efficient in the number of modes and is the standard encoding for optical quantum computation. However, in [32], it was shown that inline implementation of the cross-phase modulation needed to implement quantum gates is extremely challenging.

   In order to circumvent this problem, a scheme was proposed in [31] to perform arbitrary quantum computations using only passive mode operations or linear optics. However, in order to be able to implement the full set of quantum gates, the scheme needs several (albeit polynomial in the input size) ancillary photons, single photon measurements with fast feedforward of measurement outcomes. The ancillary photon requirement of this scheme is experimentally forbidding given that effective single photon sources are difficult to make. Subsequent schemes (see [15] for a review) have improved this requirement considerably.
3. *GKP* [33]: This encoding makes use of states that form a comb in phase space. More precisely, $|0\rangle$ is encoded

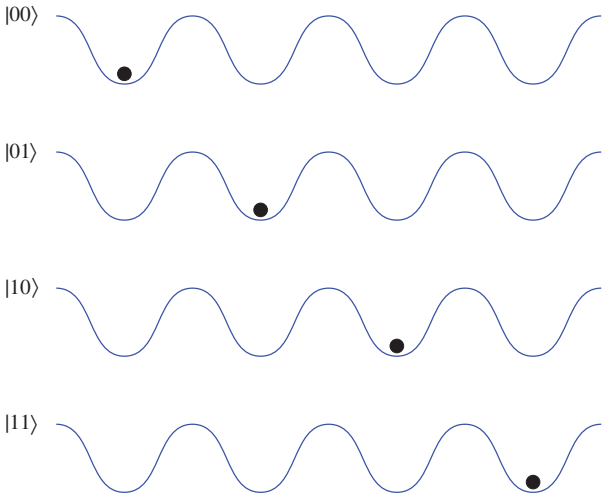

Figure 2: Unary encoding of a four-dimensional Hilbert space. Here, the wells represent optical modes and the black dot represents a single photon in that optical mode. The four states that span the Hilbert space correspond to a single photon in one of the four wells. This encoding is inefficient for a high dimensional space since it requires a lot of optical modes.

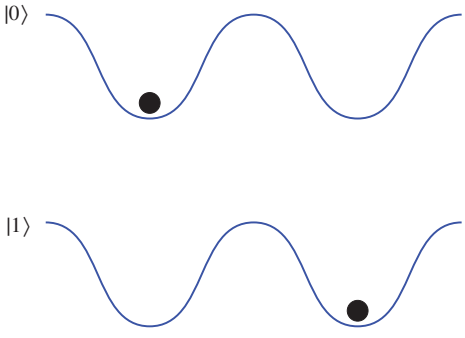

Figure 3: Dual rail encoding where each well is an optical mode and the particle is a single photon.

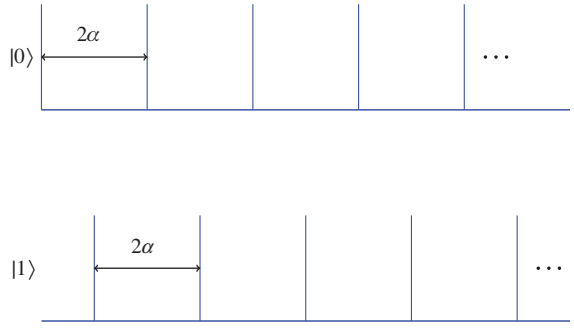

Figure 4: GKP encoding of qubits into a superposition of coherent states.
These correspond to a coherent superposition of infinitely squeezed states with a spacing of 2α. The zero state and the one state have a relative shift in order to make them orthogonal in the Hilbert space. In practice, a finite amount of squeezing suffices to make them almost orthogonal.

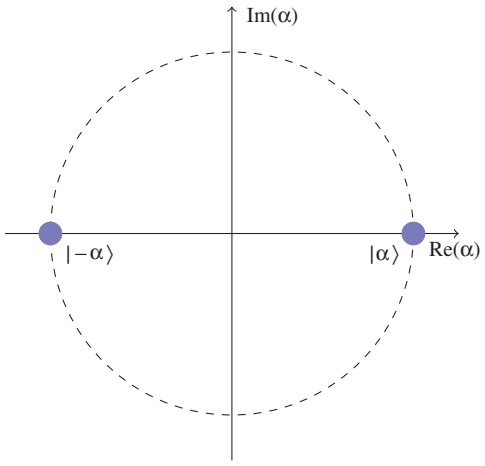

Figure 5: Coherent state encoding as a superposition of two coherent states.

into a coherent superposition of infinitely squeezed states in phase space with a spacing of 2α between two states in the superposition and a $|1\rangle$ in a superposition with the same spacing but shifted by π/α. This is shown in Figure 4, where each of the lines is an infinitely squeezed state in phase space. The combination or superposition of such states forms the comb shown. While this encoding is hard to implement, the difficulty can be boiled down to a single optical state. If one had access to the so-called cubic phase state, then the optical gates needed to perform the computation can be done using only Gaussian operations. The cubic phase state in turn can be produced using a two mode squeezed vacuum with one mode mixed with a coherent state on a beamsplitter and measured with a photon number resolving (PNR) detector. The quality of the cubic phase state depends on the photon number measured. This is investigated further in [34].

4. *Coherent states (CSQC)* [35]: This encoding makes use of superpositions of coherent states. This is shown in Figure 5. Choosing a pair of coherent states $|\alpha\rangle$ and $|-\alpha\rangle$, with $|\alpha|^2>2$, zero and one are encoded as follows.

$$|0\rangle \propto |\alpha\rangle + |-\alpha\rangle \tag{9}$$

$$|1\rangle \propto |\alpha\rangle - |-\alpha\rangle. \tag{10}$$

The two states are not orthogonal, but for coherent states with large amplitude, they are almost orthogonal. Since they are not strictly orthogonal, the normalization for each is not the same although it is close to $1/\sqrt{2}$. Once this encoding is achieved, all of the quantum gates can be implemented using only Gaussian operations without the need for nonlinearities.

4 Intermediate models of optical computation

There are several intermediate models of quantum computation. The computational power of these models seems to lie in between that of classical and universal quantum computation. The most important one in this class is boson sampling. The boson sampling model of computing uses only linear optics and single photons and PNR detectors. There are other models that use various other types of resources such as squeezing, homodyne, or heterodyne detection. In each of these models, there exists strong evidence that classical digital computers cannot simulate them efficiently. This evidence comes from the use computational complexity. For boson sampling, it can be shown that if classical computers can simulate them efficiently, then the polynomial hierarchy $\mathcal{PH}$ collapses to the third level. As mentioned earlier, there is strong evidence that the $\mathcal{PH}$ does not collapse at any level. While this does not constitute a proof of computational power, it provides complexity theoretic evidence that there is a scaling improvement for these models over classical digital computers. We describe some of these models below.

The boson sampling model was introduced in [36] and was shown to be a model that is likely to be intermediate between classical digital computing and quantum computing. This model consists of single photons in a number of modes (roughly the square of the number of photons) with the other modes containing the vacuum. The boson sampling circuit essentially consists of a network of beamsplitters and phaseshifters implementing gates and then detected on PNR detectors. For large system sizes, one can

expect only a single photon in each output mode if the network of beamsplitters and phaseshifters is random. An interesting aspect of this model is that there is no encoding into qubits even though there is a circuit. While it can be shown that this model is computationally powerful, this lack of a simple encoding makes it harder to find useful applications.

This model has been extended to scattershot boson sampling, which uses spontaneous parametric downconverter (SPDC) sources. SPDC sources produce a two mode state, which can herald a single photon after measuring one of the modes with a PNR detector. Scattershot boson sampling works by using several SPDC sources, which herald single photons, which are then multiplexed into the first few modes for computation. It has been shown that this model of boson sampling also has the same computational complexity.

A closely related model is that of Gaussian state computing with PNR detectors [37]. This model consists of two-mode squeezed vacuum states in the input modes and a network of beamsplitters and phaseshifters as the optical quantum circuit with PNR detectors at the output. It can also be shown to be classically hard to simulate unless the $\mathcal{PH}$ collapses to the third level. If the PNR detectors are replaced with homodyne or heterodyne detectors, then this model can be efficiently simulated classically since all the operations are Gaussian. Using PNR detectors gives the model more computational power, although not enough to be able to do universal quantum computing. Such models might find applications in practical repeater technology [18].

5 Analog computing with optics

All the models discussed so far are digital quantum computers. In this section, we discuss analog implementations of quantum optical computing. There are two ways in which one can depart from digital computing. The first is by replacing a circuit consisting of gates with a continuous evolution according to some time-dependent Hamiltonian. The encoding of information, however, is digital, i.e. 0 and 1. This model is described in Section 5.1. In the second analog scheme, both the encoding and evolution are analog and there is no reference to digital bits of information. This model is often useful if one in interested in simulating the dynamics of a system by using another system whose dynamics is closely related to it. We discuss these two models in the next two subsections.

5.1 Adiabatic quantum computation

Adiabatic quantum computing is an algorithmic framework geared toward finding the solution to $\mathcal{NP}$ complete optimization problems. This was introduced in [38] building on quantum annealing introduced in [39]. In quantum annealing, one uses a quantum Ising spin glass to encode hard optimization problems. The anneal time depends on the low-energy spectrum of the Ising Hamiltonian. Quantum adiabatic optimization generalizes this framework to more general Hamiltonians than the Ising model does. In fact, in [40], this has been made into a computational model and it was shown that it is equivalent to the standard circuit model. In other words, there exists a polynomial time classical algorithm that maps a quantum circuit to the adiabatic model and vice versa. There is no evidence that adiabatic quantum computers can solve $\mathcal{NP}$ complete problems efficiently. In fact, in [41], physical mechanisms by which there would be exponentially small level-crossings preventing polynomial time quantum algorithms are described.

Quantum adiabatic optimization can be described as follows. One has two time-independent Hamiltonians: the initial Hamiltonian denoted H_0 and the final Hamiltonian denoted H_P. The initial Hamiltonian should have an easy-to-prepare ground state. Typically, one takes H_0 to be the sum over σ_x on each of the qubits, i.e.

$$H_0 = \sum_i \sigma_x^{(i)}, \tag{11}$$

whose ground state is the equal superposition over all basis states, i.e.

$$|\psi\rangle = \sum_x |x\rangle. \tag{12}$$

The final Hamiltonian is chosen so that its ground state encodes the solution to the combinatorial optimization problem. The protocol is to first prepare the ground state of the initial Hamiltonian and then apply the time-dependent Hamiltonian $H(t)$ given by

$$H(t) = (1 - t/T)H_0 + (t/T)H_P, \tag{13}$$

where T is the total time of evolution. The adiabatic theorem of quantum mechanics [42] states that if the system evolves slowly enough, then the quantum state remains in the ground state, leading to the solution at the end of the evolution. The time of evolution depends on the spectral gap of the Hamiltonian $H(t)$. More precisely, suppose that the difference between the energy of the first excited state and the ground state is $g(t)$ and its minimum value over the evolution is g, then the time of

evolution must scale as $T \sim 1/g^2$. One can pictorially think of adiabatic quantum computing as being a technique to slowly modify the energy landscape of the Hamiltonian from a simple one in the beginning to the one encoding an optimization problem toward the end of the evolution. This is in contrast to simulated annealing where one starts in a higher excited state and attempts to cool or anneal to the ground state while keeping the landscape constant. This difference is important and is the essential reason why, in adiabatic computation, the state does not get stuck in a local minimum. Since we start in the minimum energy state at the beginning, the state remains in the global minimum at all times if the evolution is slow enough. In simulated annealing, however, since the state does not start in the global minimum and the landscape has several local minima, it can often get stuck in a local minimum. This model can be used in conjunction with the circuit model to design new algorithms. In [43], quantum algorithms have been designed to find marked vertices on a graph faster than classical algorithms can. The intuition behind adiabatic evolution is illustrated in Figure 6.

5.2 Computation by analogy

Computation by analogy is a type of computing that far predates digital computing. The sextant, slide rule, and the Thompson brothers' wheel-and-disc integrators are a few examples of such early analog computers [44]. Analog optical computing (see [45] for a review) has been suggested as a means to leverage the speed-of-light advantage. In [46], where it was referred to as metaphoric computing, it has been developed and applied to fluid dynamics. In this model, in order to perform a computation, especially that of simulating a physical system, one constructs another physical system (easier to control and implement) that is as close as possible to the original one. There are two advantages with this approach. First, the advantage in run-time can potentially be far greater than in other models such as digital computers. Second, even in the absence of an exact match of parameters in the analogy, one can often observe important counterintuitive physical dynamical effects.

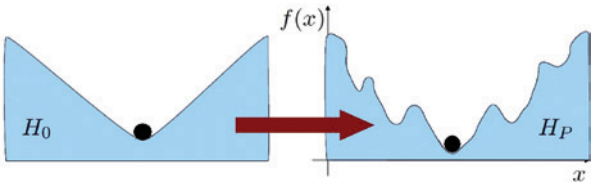

Figure 6: Adiabatic quantum computation.
The energy landscape is first a simple one with one global minimum. It is then slowly deformed to create the desired landscape. If this deformation is slow enough, then the system stays in the global minimum and does not get stuck in a local minimum.

Another major advantage of the optical system is that the need for error correction, which is the bane of almost all computing models, is less pronounced in this model. The main reason for this comes from the fact that a physical system simulating itself can be done in an error-free (or nearly error-free) fashion. Since we are trying to get as close as possible to a physical system by analogy, simulating it can also happen in essentially an error-free fashion (or at least with minimal error correction). However, there may be situations when this is not the case; i.e. there may be negative feedback or the simulation system has inherent noise. In such cases, error correction becomes necessary. The downside of this model is that it may lack programmability and the ability to simulate systems far from the physical system of interest.

6 Experimental progress

In this section, we briefly summarize some experimental progress in implementing these models of quantum computing. Probabilistic gates using linear optics have been demonstrated in [47–50]. Other gates such as XOR gate using linear optics has been demonstrated in [51]. More recently, in [52], controlled phase gates on four qubits have been demonstrated. This gate will flip the sign of a state if all the four qubits are set to 1. Development of photonic integrated circuits for quantum information processing is described in [53].

There have been many experiments demonstrating boson sampling. One of the first ones appeared in [54–57]. These experiments demonstrated boson sampling on six modes with three photons. This has been increased to experimentally distinguish between boson sampling distribution and the uniform distribution with three photons in up to 13 modes in [58] as well as three photons in nine modes in [59]. Scattershot boson sampling with six SPDC sources multiplexed into 13 modes has been demonstrated in [60].

Optical implementations of a variation of quantum annealing have been proposed in [61, 62] using coupled non-linear-optical systems. They also implemented one of those proposals in a small proof-of-concept experiment [61]. In [62], the Ising model is implemented using injection locked lasers, where the Ising spins are mapped onto one of two (right or left) circular polarizations of an

array of coupled "slave" laser oscillators that are driven by a strong "master" laser, whereas in [61], the Ising spins are mapped to a pair of orthogonal states of an array of coupled optical parametric oscillators. In both designs, the system is initialized in a random "low-temperature" state, and the Hamiltonian governing the Ising interactions – which is the Ising Hamiltonian *without* the transverse magnetic-field term – is slowly turned on.

7 Conclusions and outlook

In this survey, we described various models of computing that are potentially amenable to implementation via optical gates and optical evolution. We described the computational power of these models and the problems that can be solved within these models. We argued that optical implementations are important from the point of view of communications, cryptography, as well as quantum interconnects between servers of qubits.

There has been tremendous progress in optical quantum computing in the last few years. However, there still remain several experimental and theoretical challenges to be overcome. Some of these are the development of on-demand single photon sources, quantum memories, highly efficient detectors, good error-correcting codes specific to photon loss, and scalable fault-tolerant architectures. Progress in these areas would eventually lead to high-fidelity optical quantum processors.

Acknowledgments: We thank Saikat Guha for helpful discussions. We also thank the anonymous referees for several helpful comments that have substantially improved the manuscript.

References

[1] Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. In 35th Annual Symposium on Foundations of Computer Science, 1994 Proceedings. IEEE, 1994:124–134.
[2] Hallgren S. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. J ACM 2007;54:653–8.
[3] Grigni M, Schulman L, Vazirani M, Vazirani U. Quantum mechanical algorithms for the nonabelian hidden subgroup problem. In Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing. ACM, 2001:68–74.
[4] Bacon D, Childs AM, van Dam W. From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups. In 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05). IEEE, 2005:469–78.
[5] Krovi H, Rötteler M. An efficient quantum algorithm for the hidden subgroup problem over weyl-heisenberg groups. In Mathematical methods in computer science. Karlsruhe, Germany: Springer, 2008:70–88.
[6] Jordan SP, Lee KSM, Preskill J. Quantum algorithms for quantum field theories. Science 2012;336:1130–3.
[7] Berry DW, Childs AM, Cleve R, Kothari R, Somma RD. Exponential improvement in precision for simulating sparse Hamiltonians. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing. ACM, 2014:283–92.
[8] Krovi H, Russell A. Quantum Fourier transforms and the complexity of link invariants for quantum doubles of finite groups. Commun Math Phys 2015;334:743–77.
[9] Montanaro A, Pallister S. Quantum algorithms and the finite element method. Phys Rev A 2016;93:032324.
[10] Krovi H, Magniez F, Ozols M, Roland J. Quantum walks can find a marked element on any graph. Algorithmica 2016;74:851–907.
[11] Montanaro A. Quantum walk speedup of backtracking algorithms. arXiv preprint:arXiv:1509.02374.
[12] Blais A, Huang R, Wallraff A, Girvin SM, Schoelkopf RJ. Cavity quantum electrodynamics for superconducting electrical circuits: an architecture for quantum computation. Phys Rev A 2004;69:062320.
[13] Cirac JI, Zoller P. Quantum computations with cold trapped ions. Phys Rev Lett 1995;74:4091.
[14] Loss D, DiVincenzo DP. Quantum computation with quantum dots. Phys Rev A 1998;57:120.
[15] Kok P, Munro WJ, Nemoto K, Ralph TC, Dowling JP, Milburn GJ. Linear optical quantum computing with photonic qubits. Rev Mod Phys 2007;79:135.
[16] Krovi H, Guha S, Dutton Z, da Silva MP. Optimal measurements for symmetric quantum states with applications to optical communication. Phys Rev A 2015;92:062333.
[17] Gisin N, Ribordy G, Tittel W, Zbinden H. Quantum cryptography. Rev Mod Phys 2002;74:145.
[18] Krovi H, Guha S, Dutton Z, Slater JA, Simon C, Tittel W. Practical quantum repeaters with parametric down-conversion sources. Appl Phys B 2016;122:1–8.
[19] Saglamyurek E, Sinclair N, Jin J, et al. Broadband waveguide quantum memory for entangled photons. Nature 2011;469:512–5.
[20] Usmani I. Mapping multiple photonic qubits into and out of one solid-state atomic ensemble. Nat Commun 2010;1:1–7.
[21] Kielpinski D, Meyer V, Rowe MA, et al. A decoherence-free quantum memory using trapped ions. Science 2001;291:1013–1015.
[22] Azuma K, Tamaki K, Lo H-K. All-photonic quantum repeaters. Nat Commun 2015;6:6787.
[23] Arora S, Barak B. Computational complexity: a modern approach. Cambridge: Cambridge University Press, 2009.
[24] Farhi E, Harrow AW. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674*, 2016.
[25] Nielsen MA, Chuang IL. Quantum computation and quantum information. Cambridge: Cambridge University Press, 2010.
[26] Braunstein SL. Squeezing as an irreducible resource. Phys Rev A 2005;71:055801.
[27] Lloyd S, Braunstein SL. Quantum computation over continuous variables. In: Quantum Information with Continuous Variables. Dordrecht: Springer, 1999, 9–17.

[28] Cerf NJ, Adami C, Kwiat PG. Optical simulation of quantum logic. Phys Rev A 1998;57:0R1477.
[29] Reck M, Zeilinger A, Bernstein HJ, Bertani P. Experimental realization of any discrete unitary operator. Phys Rev Lett 1994;73:58.
[30] Chuang IL, Yamamoto Y. Simple quantum computer. Phys Rev A 1995;52:3489.
[31] Knill E, Laflamme R, Milburn GJ. A scheme for efficient quantum computation with linear optics. Nature 2001;409:46–52.
[32] Shapiro JH. Single-photon kerr nonlinearities do not help quantum computation. Phys Rev A 2006;73:062305.
[33] Gottesman D, Kitaev A, Preskill J. Encoding a qubit in an oscillator. Phys Rev A 2001;64:012310.
[34] Ghose S, Sanders BC. Non-gaussian ancilla states for continuous variable quantum computation via gaussian maps. J Mod Opt 2007;54:855–69.
[35] Ralph TC, Gilchrist A, Milburn GJ, Munro WJ, Glancy S. Quantum computation with optical coherent states. Phys Rev A 2003;68:042319.
[36] Aaronson S, Arkhipov A. The computational complexity of linear optics. In: Proceedings of the forty-third annual ACM symposium on Theory of computing, ACM, 2011, 333–42.
[37] Lund AP, Laing A, Rahimi-Keshari S, Rudolph T, O'Brien JL, Ralph TC. Boson sampling from a gaussian state. Phys Rev Lett 2014;113:100502.
[38] Farhi E, Goldstone J, Gutmann S, Lapan J, Lundgren A, Preda D. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. Science 2001;292:472–5.
[39] Kadowaki T, Nishimori H. Quantum annealing in the transverse Ising model. Phys Rev E 1998;58:5355.
[40] Aharonov D, Van Dam W, Kempe J, Landau Z, Lloyd S, Regev O. Adiabatic quantum computation is equivalent to standard quantum computation. SIAM Rev 2008;500:755–87.
[41] Altshuler B, Krovi H, Roland J. Anderson localization makes adiabatic quantum optimization fail. Proc Natl Acad Sci USA 2010;107:12446–50.
[42] Messiah A. Quantum mechanics. vol. i. North Holland: Elsevier Science & Technology Books, 1961.
[43] Krovi H, Ozols M, Roland J. Adiabatic condition and the quantum hitting time of Markov chains. Phys Rev A 2010;82:022333.
[44] Jack Copeland B. The modern history of computing. In: Zalta EN, ed. The Stanford Encyclopedia of Philosophy, 2008. http://plato.stanford.edu/archives/fall2008/entries/computing-history/, fall 2008 edition.
[45] Solli DR, Jalali B. Analog optical computing. Nat Photonics 2015;9:704–6.
[46] Tsang M, Psaltis D. Metaphoric optical computing for fluid dynamics. In: Integrated Optoelectronic Devices 2005, International Society for Optics and Photonics. Baltimore, Maryland: Optical Society of America, 2005, 1–8.
[47] Pittman TB, Jacobs BC, Franson JD. Probabilistic quantum logic operations using polarizing beam splitters. Phys Rev A 2001;64:062311.
[48] Pittman TB, Fitch MJ, Jacobs BC, Franson JD. Experimental controlled-not logic gate for single photons in the coincidence basis. Phys Rev A 2003;68:032316.
[49] O'Brien JL, Pryde GJ, White AG, Ralph TC, Branning D. Demonstration of an all-optical quantum controlled-not gate. Nature 2003;426:264–7.
[50] Patel M, Altepeter JB, Hall MA, Medic M, Kumar P. Experimental characterization of a telecommunications-band quantum controlled-not gate. IEEE J Sel Topics Quantum Electron 2009;15:1685–93.
[51] Pittman TB, Jacobs BC, Franson JD. Demonstration of non-deterministic quantum logic operations using linear optical elements. Phys Rev Lett 2002;88:257902.
[52] Stárek R, Mičuda M, Miková M, et al. Experimental investigation of a four-qubit linear-optical quantum logic circuit. Sci Rep 2016;6:33475.
[53] Mower J, Harris NC, Steinbrecher GR, Lahini Y, Englund D. High-fidelity quantum state evolution in imperfect photonic integrated circuits. Phys Rev A 2015;92:032322.
[54] Tillmann M, Dakić B, Heilmann R, Nolte S, Szameit A, Walther P. Experimental boson sampling. Nat Photonics 2013;7:540–4.
[55] Spring JB, Kolthammer WS, Gates JC, et al. Experimental boson sampling. Science 2012;339:798–801, (arXiv: 1212.2622).
[56] Broome M, Fedrizzi A, Rahimi-Keshari S, et al. Experimental boson sampling. In Conference on Coherence and Quantum Optics, Optical Society of America, 2013, W5B–3.
[57] Crespi A, Osellame R, Ramponi R, et al. Integrated multimode interferometers with arbitrary designs for photonic boson sampling. Nat Photonics 2013;7:545–9.
[58] Spagnolo N, Vitelli C, Bentivegna M, et al. Experimental validation of photonic boson sampling. Nat Photonics 2014;8:615–20.
[59] Carolan J, Meinecke JDA, Shadbolt PJ, et al. On the experimental verification of quantum complexity in linear optics. Nat Photonics 2014;8:621–6.
[60] Bentivegna M, Spagnolo N, Vitelli C, et al. Experimental scattershot boson sampling. Sci Adv 2015;1:e1400255.
[61] Marandi A, Wang Z, Takata K, Byer RL, Yamamoto Y. Network of time-multiplexed optical parametric oscillators as a coherent Ising machine. Nat Photonics 2014;8:937–42.
[62] Utsunomiya S, Takata K, Yamamoto Y. Mapping of Ising models onto injection-locked laser systems. Opt Express 2011;19:18091–108.