

Automated design of photonic two-qubit quantum gates and circuits via optimization and pruning

Javier Osca ^{*}

*Department of Physics, [University of the Balearic Islands](#), E-07122 Palma, Spain
and [Institute for Cross-Disciplinary Physics and Complex Systems \(IFISC\)](#)
([CSIC-UIB](#)), E-07122 Palma, Spain*

Jiri Vala [†]

*Department of Physics, [Maynooth University](#), Maynooth, Co. Kildare, Ireland
and [Tyndall National Institute, University College Cork](#), “Lee Maltings,” Dyke Parade,
Cork, Ireland*



(Received 16 December 2025; accepted 19 August 2026; published 31 August 2026)

We develop an algorithm to automatically design photonic gates and circuits that is based on a combination of optimization and pruning methods. It starts with a general ansatz for optical circuits with tunable parameters that are then subjected to optimization. The algorithm simplifies the final circuit by pruning its unnecessary elements following predefined criteria. We demonstrate the algorithm by producing both approximate and numerically exact implementations of the CNOT gate and Deutsch-Jozsa, Grover, and Fourier transform two-qubit quantum circuits. All implementations of the different gates and circuits can be integrated into larger quantum information protocols.

DOI: [10.1103/xjrz-csmx](https://doi.org/10.1103/xjrz-csmx)

I. INTRODUCTION

Quantum computation has become central to future scientific development [1–3], promising a speedup in the solution of certain computational problems, including simulations of other quantum systems. It is possible to attain universal quantum computation using only linear optical elements and postselection [4–6]. These linear elements usually include beam splitters and phase shifters, while postselection is based on the partial measurement of some of the photonic output modes.

Our work is relevant to discrete-variable photonic quantum computing paradigm that follows the foundational paper by Knill *et al.* [4]. This kind of photonic implementation consists of three parts [7–9]. First, photons in a particular quantum state are emitted into the photonic circuit. Then, the photons go through different optical elements and finally, they are measured at the detectors. If the result in the measured modes is equal to some predefined postselection condition, then the output photonic state is accepted; otherwise, it is rejected. Photonic implementations of a quantum algorithm are therefore probabilistic and are characterized by a success probability.

^{*}Contact author: javier.osca@uib.cat

[†]Contact author: jiri.vala@mu.ie

The initial and final photonic states must have an interpretation in terms of logical quantum bits or qubits. In path encoding, each pair of modes represents a qubit and the occupation of those modes determines whether the qubit is in a logical basis state $|0\rangle$ or $|1\rangle$ [10,11].

In particular, we are interested in the compilation of arbitrary algorithms into photonic circuits. This is a challenging problem that involves considerable subtleties. The implementation of universal quantum gates requires the use of nonlinearities [5,12] that, in the case of linear photonic circuits, are introduced through postselection. As a consequence, the simple substitution of each gate by a photonic counterpart will diminish the success probability of the overall circuit with each new gate. This also implies the introduction of extra ancillary photons, thus increasing the required resources.

Previous works have considered some of these issues. The works dealing with automated design of photonic circuits are mainly based on the matrix decomposition of the intended photonic circuit [13]. Optimization has also been used to approximate photonic circuits by other photonic circuits using various ansätze [14]. However, these approaches require to have already a target photonic circuit. This approach has also been used to determine the limits of the success probabilities of different gates [15]. The fact that an initial optical implementation of a circuit is required before searching for alternative implementations presents certain limitations. On the other hand, artificial intelligence and machine learning tools have been proposed for circuit design. These tools employ abstract representations of the underlying architecture [16] or they are based on the exploration of large sets of randomly generated circuits [17].

In this work, we propose a method for the automated search of optimal circuit implementations by means of a combination of an optimization and pruning algorithm. The resulting photonic circuit will use only linear optical elements (with postselection) and path encoding to relate the logical qubits with the photonic states. This method also must consider the resource expenditure, seeking designs with a reduced number of optical elements. The target is the logical matrix representing the functionality of the desired algorithm in the qubit basis. To perform a circuit optimization, we need to perform various circuit simulations with different parameters. To this end, we implement the algorithm using stochastic optical quantum circuit simulator (SOQCS) [18,19].

This method is inspired by algorithms for approximate quantum compiling [20,21]. However, instead of building the final circuit element by element, we will use a large ansatz and will remove the unnecessary elements. The structure of the initial ansatz follows the principles of those of silicon chips for general photonic quantum computation [22] that must adapt to different functionalities. As will be shown, this procedure is suitable for obtaining linear photonic circuits with a small number of optical elements. The photonic circuits generated by the present method do not require previously entangled photons or the preparation of specific states. Furthermore, the algorithm may be configured to avoid leakage, therefore making unnecessary the use of nondemolition measurements [23] to integrate the obtained circuits into larger ones. Our approach is different from previous algorithms for automated search of photonic circuits [17,23] that start from a random combination of components.

To summarize, the standard quantum compilation aims at representing an n -qubit unitary operation as a sequence of some elementary quantum gates, acting on a small number of quantum bits but within the same qubit system, as articulated, for example, in Ref. [20]. Our task of representing quantum logic gates and operations in (discrete-variable) photonic systems is much more challenging. In these systems, qubits are encoded in photonic states and thus embedded in a higher-dimensional Hilbert space. More importantly, quantum computing operations, which also take place in a higher-dimensional space, are performed by linear optical elements and need to be aided by ancillary photonic modes and a postselection based on quantum measurement to implement quantum logic operations. Our quantum compilation strategy, based on rather unique combination of optimization with appropriately defined cost function and pruning, allows us to tackle these challenges.

In Sec. II, we will present the cost function needed for the optimization. We will also address therein how to deal with the fact that a computational subspace, which encodes logical information, is embedded in the full Hilbert space of a photonic quantum circuit of a larger dimension. Next, in Sec. III, we will describe the pruning algorithm designed to minimize the number of optical elements used by the photonic circuit. In Sec. IV, we will present example implementations of two-qubit gates and circuits. Finally, in the last section, we will present our conclusions.

II. OPTIMIZATION

A linear photonic circuit is built from optical elements such as beam splitters and phase shifters. For example, in a beam splitter, an input photonic state is transformed as

$$|n_1, n_2\rangle = \frac{(\hat{a}_1^\dagger)^{n_1}}{\sqrt{n_1!}} \frac{(\hat{a}_2^\dagger)^{n_2}}{\sqrt{n_2!}} |0, 0\rangle \rightarrow \frac{1}{\sqrt{n_1!n_2!}} (\hat{a}_1^\dagger \cos \theta + \hat{a}_2^\dagger e^{-i\phi} \sin \theta)^{n_1} \times (-\hat{a}_1^\dagger e^{i\phi} \sin \theta + \hat{a}_2^\dagger \cos \theta)^{n_2} |0, 0\rangle, \quad (1)$$

where $\hat{a}_i^\dagger$ is the photon creation operator in mode i , and θ and ϕ are two angles that characterize the beam splitter transformation. In a similar manner, a phase shifter is parametrized by a single angle α because it only introduces a phase shift $e^{i\alpha}$ in a photon state. In general, a photonic circuit will have n_m modes that may be occupied by any integer number of photons.

In this work, we will restrict ourselves to path encoding where a pair of modes encode a qubit. A logical zero is encoded as 01, where the first mode is unoccupied and the second contains one photon, and a logical one is then equivalent to 10. A photonic circuit implementing a logical gate (or a combination of these) will consist of an even number of qubit modes plus some extra ancillary modes. The latter are required to define postselection conditions. However, note that the interpretation of modes as ancillas or qubits may be different at the input or the output of the circuit. In general, a photonic circuit representing a quantum gate is described by a set of interconnected optical elements, a list of their parameters, a definition of the initial conditions of the ancilla modes, and the postselection conditions.

If we consider an ansatz where the set of optical elements $\mathcal{C}$, the mapping between modes and qubits $\mathcal{M}$, and the initialization $\mathcal{I}$ and postselection conditions $\mathcal{P}$ are set, then the circuit transformation matrix will depend only on a list of parameters that can be optimized. That is,

$$f(\theta_1, \theta_2, \dots, \theta_n, \mathcal{C}, \mathcal{M}, \mathcal{I}, \mathcal{P}) \rightarrow (M, p_1, p_2, \dots, p_{2^{n_q}}), \quad (2)$$

where θ_i is an angle parametrizing an optical element in the circuit and M is the unitary matrix representing a quantum state transformation in the logical Hilbert space. M is a $2^{n_q} \times 2^{n_q}$ matrix, where n_q is the number of qubits. Additionally, each p_i represents the success probability of the circuit for one of the 2^{n_q} possible input basis states, with the corresponding output state encoded in the associated column of M . In general, all the individual probabilities p_i can be different but they must be equal, $p_1 = p_2 = \dots = p_{2^{n_q}} = p$, for the photonic circuit to properly represent a quantum algorithm. This will be further discussed below.

We perform a multiobjective optimization procedure to implement the functionality of a given logical gate in a photonic circuit. The multiple objectives are combined using a collection of weighted criteria. We define a cost function C as a combination of five criteria,

$$C(\vec{\mathcal{P}}) = [c \cdot C_{\text{AT}}(\vec{\mathcal{P}})^g + b \cdot C_{\text{BT}}(\vec{\mathcal{P}})^g + l \cdot C_{\text{LT}}(\vec{\mathcal{P}})^g + r \cdot C_{\text{ST}}(\vec{\mathcal{P}})^g + o \cdot C_{\text{OT}}(\vec{\mathcal{P}})^g]^{1/g}, \quad (3)$$

where C_{AT} is an accuracy test or how close is our circuit to the target, C_{BT} is a “balance” test, C_{LT} is a leakage test, and C_{ST} is the success test. Additionally, we have the C_{OT} or “orientation test” that will give a cost advantage to those configurations that can be easily simplified. We will expand on this in the pruning section. $\vec{\mathcal{P}} = (\theta_1, \theta_2, \dots, \theta_n)$ are the parameters to be optimized while c, b, l, r , and o are the weights of the different tests. We have observed that in our case the search seems to converge faster if $g = 3$. The hyperparameter values must be selected to allow the different cost functions to

have commensurable magnitudes along the search and avoid the dominance of one criterion over the others. This requires some coarse tuning, but no fine-tuning is needed in general. Note that all cost functions must take non-negative values throughout the parameter space in order to avoid mutual cancellations when they are not at their local minima.

C_{AT} may refer to C_{HS} , a Hilbert-Schmidt-like distance, or alternatively to the combination $d \cdot C_D + h \cdot C_H$ that represents the sum of an alternative definition of distance and a heuristic function. We can switch between the two definitions upon configuration of the search method. In the same way as before, d and h are real numbers working as weights.

Given an input in the Hilbert space of qubits, it will be decoded into a photonic input state for the circuit. Once a simulation is performed, the output state will be normalized and encoded back into a logic state. For example, $|01\rangle \rightarrow |0\rangle$ or $|10\rangle \rightarrow |1\rangle$. Output states are normalized and the matrix M is built. M represents the relation between input and outputs in the basis subspace that encodes a logical qubit value. A Hilbert-Schmidt-type distance is defined,

$$C_{\text{HS}}(\vec{\mathcal{P}}) = \left| 1 - \frac{1}{2^{n_q}} \text{Tr}(M^\dagger(\vec{\mathcal{P}}) \cdot U) \right|, \quad (4)$$

where $M(\vec{\mathcal{P}})$ is the qubit-encoded matrix for the ansatz circuit given the parameters $\vec{\mathcal{P}}$ and the unitary matrix to be approximated is U . This is a useful measure to quantify the proximity of the current ansatz to the target circuits. However, it may happen that small values of the Hilbert-Schmidt test are attained by means of the attribution of the largest part of the error to the output related to a particular input (i.e., to a single column of the matrix). Therefore, overall the circuits perform well in relation to the Hilbert-Schmidt distance with small errors unless a particular input ket is used that introduces a large error in the output.

For this reason, we prefer to use a distance test,

$$C_D(\vec{\mathcal{P}}) = \frac{1}{2^{n_q}} \left(\sum_{i,j} |(M^\dagger(\vec{\mathcal{P}}) \cdot U)_{i,j} - I_{i,j}|^f \right)^{1/f}, \quad (5)$$

which represents the difference between the target and ansatz matrices evaluated element by element. In contrast with the Hilbert-Schmidt test, this cost function presents two limitations. First, it fixes the global phase of the output transformation. Although the global phase is physically irrelevant in a quantum operation, imposing a specific value introduces an additional constraint on the optimization. Second, this cost function is less smooth and can lead to a noisier optimization landscape than the Hilbert-Schmidt test, which may make the optimization process more challenging. If used by itself, it will be more difficult for the optimization algorithm to find a solution. However, the linear optical elements available for photonic circuits (beam splitter and phase shifters) make it very easy to adjust the global phase of a circuit. This has the unexpected effect of restricting the search space, therefore allowing to find out solutions easily. On the other hand, we have found this cost function to work well when paired with a heuristic cost function,

$$C_H(\vec{\mathcal{P}}) = \left(\sum_i |(M^\dagger(\vec{\mathcal{P}}) \cdot U)_{i,i} - I_{i,i}|^f \right)^{1/f}, \quad (6)$$

where only the adequacy of the diagonal is roughly checked. In both cost functions, f is an integer; we are using $f = 3$ in C_D and $f = 2$ in C_H . A proper circuit that correctly approaches the target solution must score zero. This cost function happens to be simpler and smoother than C_D , leading the optimization algorithms to the proper regions of the parameter space where the best solutions can be found. Note, however, that there are also many configurations of the parameters that do not properly approximate the target, and that will result in a minimum of this cost function if considered alone. Combining both the distance test and the heuristic test, we have found a good behavior of the search algorithm both in accuracy and in the number of steps for convergence.

Furthermore, a multiobjective optimization is needed because, in general, the success probability of the circuit depends on each of the possible (2^{n_q}) elements of the input qubit basis. An output state is calculated for each element of this input basis. Each output state is then postselected, normalized, and mapped onto the logical basis to obtain the transformation matrix of the ansatz circuit. However, the normalization removes the information about the success probability of obtaining each output state, which must be considered separately. In general, a photonic circuit may have a different success probability depending on the input basis state.

For a circuit to implement the functionality of a logical gate (or a combination of gates), all input basis states must have the same success probability. Otherwise, an input state given by a linear superposition of basis states would acquire different amplitude factors for its components after postselection, resulting in a distorted output state and preventing the circuit from correctly implementing the desired quantum operation.

In general, a randomly generated circuit does not necessarily satisfy this condition, since postselection success probabilities are not guaranteed to be equal for all input states. Achieving a constant success probability is therefore a feature that must be obtained through proper circuit design. To quantify this requirement, we define a “balance test,” which is zero when all (p_i) probabilities are equal and provides a measure of the deviation from this ideal balanced condition.

$$C_{\text{BT}}(\vec{P}) = \frac{\sqrt{\sum_{i < j}^{2^{n_q}} |p_i^2 - p_j^2|}}{\bar{p}}, \quad (7)$$

where $\bar{p}$ is the average of all p_i .

Furthermore, not every photonic state can be identified with an equivalent qubit state. For example, an output state with more than one photon in the same mode does not have a logical interpretation. The leakage test C_{LT} provides information on the amount of probability that the output state “leaked” from the computational subspace to its complement in the overall Hilbert space. It can be written as

$$C_{\text{LT}}(\vec{P}) = \sum_{i=0_j}^{2^{n_q}} \frac{p_i^o}{p_i}, \quad (8)$$

where p_i^o is the probability to leak out of the logical Hilbert space for a given input i . This quantity can be calculated by summing the amplitude squared of those kets of the output that cannot be encoded into the qubit basis. Usually, circuits in which this restriction is not considered (like the one in Refs. [24,25]) have larger success probabilities; however, they cannot be used to build larger circuits without additional components for nondemolition measurements [23].

Finally, $C_{\text{ST}} = 1 - \bar{p}$ or success test is related with the overall success probability of the circuit. A success probability of 100% corresponds to $C_{\text{ST}} = 0$. Some coarse tuning of the cost function weight is needed to allow a proper optimization process. However, a detailed fine-tuning of those parameters does not lead to a different set of solutions because of the existence of photonic circuits that fulfill simultaneously all the given criteria. The only exception is the weight of the overall success probability of the circuit. Identity circuits, for example, are solutions of success probability $p = 1$ that are not compatible with small C_{AT} if we are searching for photonic implementations of the CNOT. In this particular case, it is enough to give a small value to r to avoid for the appearance of pathological (close to $p = 0$) solutions.

Using this cost function, we have tested various optimization algorithms. We have obtained the best results using L-BFGS-B [26] that has the advantage that it does not require an explicit definition of the cost function gradient; the algorithm can generate a numerical approximation instead. The optimization requires navigating through a potentially complex optimization landscape. For a fixed cost function, the practical performance of the optimization depends on the choice of its initial point in a parameter space. To maximize this performance, we first generate 100 000 random points and

evaluate their cost function. We then preselect 50 points with the smallest value. Using this set, we start the optimization initially with 2000 steps. Afterward, we evaluate the outcome and select the ten best results, which are then subjected to the full optimization with the practical limit set at 15 000 steps. Each of the first two phases of this process require 100 000 cost function evaluations, and the final optimization 150 000 evaluations, leading to approximately comparable computational costs for each phase of the algorithm. We observed in the optimization runs that faster convergence usually implies smaller final values of the cost function.

It is known from the work by Knill [27] that generic unitary operations are hard to even approximate; specifically, the relevant scaling with the number of qubits is exponential. The problem, studied by Knill, is equivalent to a standard quantum compilation when a given unitary operation over n qubits is approximated by a quantum circuit of universal elements over the same number of qubits. Our problem is much more complicated as it requires to map quantum information processing into a (discrete-variable) photonic system. Quantum computing operations are acting on a higher-dimensional Hilbert space, which may lead to leakage to states outside of those encoding qubits. They are performed by linear optical elements and thus need to be aided by ancillary photonic channels/modes, whose number is *a priori* unknown, and a postselection based on single-photon detection.

In particular, each evaluation of the cost function implies the simulation of a photonic circuit. These computations were carried out using the library SOQCS [18]. A simulation consists in the calculation of the amplitudes of all possible outcomes of the circuit. Those complex amplitudes of probability are obtained by means of the calculation of a matrix permanent. SOQCS library allows for multiple calculation methods. In this work, we have chosen an implementation of the Ryser method [28] that allows for the use of parallelism in the calculation of the permanents. The computational cost of each permanent scales as $O(n2^n)$, where n is the number of photons. On the other hand, the number of amplitudes to be obtained (therefore, the number of permanents to be calculated) is $\binom{n+m-1}{n}$ for a single circuit evaluation. Here, m represents the number of degrees of freedom of a photonic circuit, which in double rail encoding is equivalent to the number of qubit encoding modes. For example, a two-qubit ansatz with two-photon ancilla will have $n = 4$ and $m = 4$. Note that postselected modes do not increase m because their output is determined if the gate is successful.

Both the number of photons n and the degrees of freedom m depend on the number of qubits. Additionally, the depth of the photonic circuit will increase the number of parameters to be optimized, but we consider this effect to be of second order in comparison with the exponential growth of the cost of the circuit evaluation with an increasing number of qubits.

With an Intel I7-10750H@2.60GHz with 16 GB of RAM memory, a circuit like the one presented in Fig. 3 required around 1 h to be obtained using 10 processors working in parallel. Note, however, that the pruning phase of the algorithm (to be explained next) implies an undetermined number of iterations of the pruning process. Therefore, exact times may vary between executions. Detailed benchmarks of SOQCS for this particular machine are published in Ref. [19].

III. PRUNING

The conventional way of constructing a quantum circuit is to build it component by component as suggested in Ref. [20]. This strategy applied to photonic circuits would consist of adding an optical element and performing an optimization in the way explained in the previous section. If the result is not satisfactory, new components can be added to the circuit. The inconvenience of this strategy is that there are no bounds to the growth of the circuit and no guarantee that a solution will be found.

We propose here a different strategy. We start from a large ansatz that can be optimized to provide a good approximation of the intended quantum algorithm (built from a combination of logical gates) and remove the elements that are considered irrelevant. For example, a beam splitter with a θ parameter close to 0° or 180° can be removed, while a beam splitter with a θ parameter close

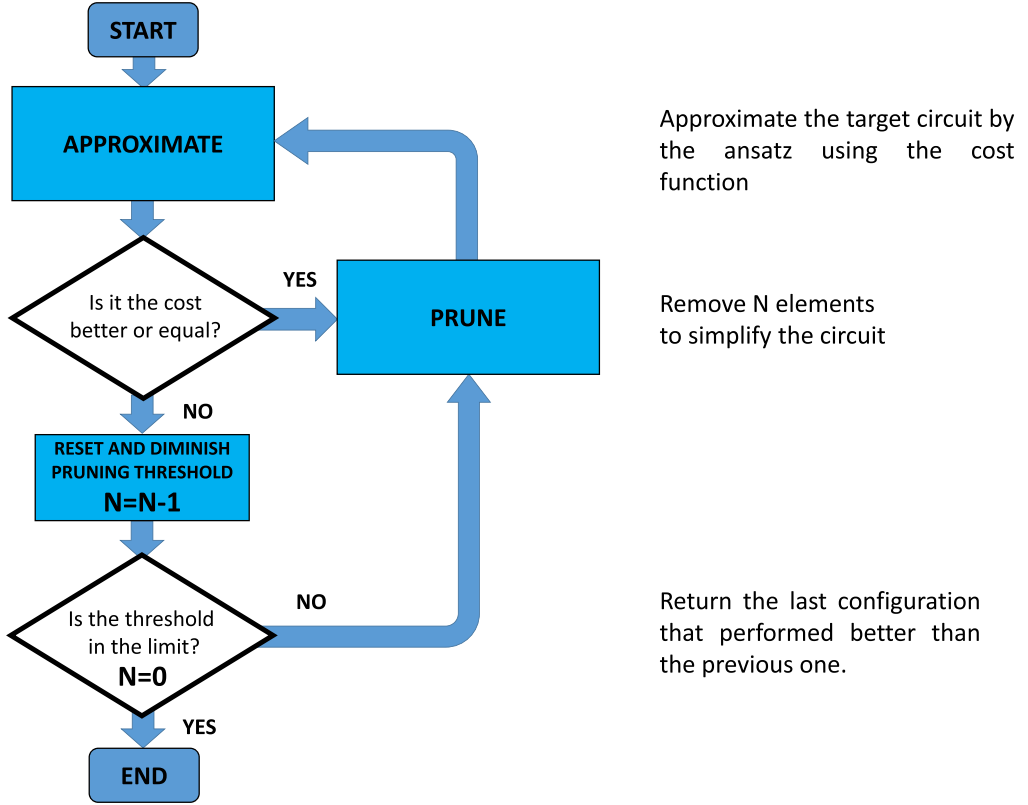


FIG. 1. Diagram of the pruning method.

to 90° or 270° degrees can be replaced by a swap operation. The overall idea consists of removing or substituting all these irrelevant beam splitters and reoptimizing the resulting circuit. With this in mind, we prune the N optical elements closest to a pruning condition. This condition is quantified as the minimum angular distance of the θ parameter to one of the trivial configurations. Note that 0° and 360° are equivalent pruning configurations but they need to be evaluated separately. θ angles are confined to the interval $0^\circ < \theta < 360^\circ$. The substitution of a beam splitter by an identity or a swap operation is valid up to a phase. In general, there is π phase between a beam splitter of $\theta = 180^\circ$ and an identity operator. Furthermore, the angle ϕ of the beam splitter may not be zero. After pruning, we may include this phase by adding two phase shifters to the optical circuit or we may assume that it will be absorbed by the remaining optical elements during the optimization process. These additional phase shifters may be further pruned if their parameters α are close to 0° . The parameters obtained by optimization before pruning can be used as a starting point of the new optimization process. If the new optimization is satisfactory, pruning can be carried out again. In the opposite case, the pruning threshold should be reduced by one before continuing. The algorithm stops when the pruning threshold arrives to zero. We can see a sketch of the pruning algorithm in Fig. 1.

As stated above, we have the option to add phase shifters in place of beam splitters when those are pruned. This is to maintain the information of phase changes previously associated with the beam splitter. This is more useful for the search algorithm when few beams plitters remain in the circuit but makes the pruning slower in large circuits. In any case, if pruning leads to an empty mode (i.e., with no optical elements acting on it), a phase shifter is added instead to preserve the mode phase freedom. This phase shifter may be further removed if close to 0° .

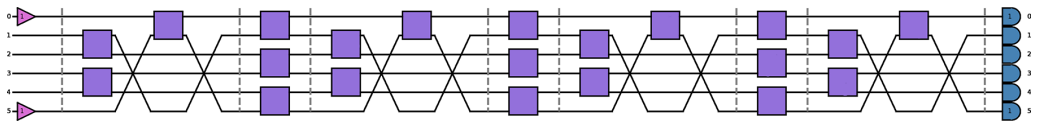


FIG. 2. Ansatz diagram. We use an ansatz made of layers of beam splitters, each connected to a pair of modes. A column of beam splitters defines a layer. The pair of modes connected by a beam splitter alternates between each layer.

Since a given target functionality can generally be achieved by different sets of parameters corresponding to different optical implementations, the cost function can be slightly modified to favor equivalent solutions that are closer to the pruning conditions. Specifically, we can favor those solutions with beam splitter angles close to 0° , 180° , 90° , or 270° , as these angles correspond to configurations in which optical elements can be removed or simplified. This guides the optimization toward implementations that can be simplified after the optimization process with minimal change to the target functionality. This is achieved by adding an additional cost term to the multiobjective cost function,

$$C_{OT}(\vec{\mathcal{P}}) = \frac{1}{n_\theta} \sum_i^{n_\theta} \sin(2\theta_i)^2, \quad (9)$$

where n_θ is the number of θ angles in the beam splitters. We can further add an extra term to reward $\alpha = 0$ conditions in the phase shifters. These extra terms are the minimum distance of the current angle α of each phase shifter to 0° (and 360°).

Finally, we have included an extra optimization step when the number of elements to be pruned reaches zero. Two phase shifters or two beam splitters one after another on the same modes may be substituted by a single equivalent element that summarizes the functionality of the two previous ones. For example, two phase shifters may be substituted by a single one whose phase is the sum of the original ones. In other words, some of the known simplification rules may also be included as part of the pruning process.

IV. RESULTS

A. CNOT

We have applied this method to search for photonic implementations of the CNOT gate. We use an ansatz of the kind depicted in Fig. 2 with seven layers of three beam splitters in each layer. The ansatz will have four modes encoding two qubits plus two additional ancillary modes. These extra ancillary modes will be initialized with one photon and each ancilla mode will also be postselected with one photon. We have determined with repeated optimizations that one ancilla mode is not sufficient to reach a good approximation of the target with values of the Hilbert-Schmidt cost function in the range 0.1–0.2. On the other hand, the more ancilla modes are used, the lower is the success probability with small or no improvement on the cost function, so our objective is to minimize the number of ancillas. If ansatz circuits with more than two ancilla modes are used, it is not uncommon to obtain solutions where the extra ancillas are not used but redirected directly to the output instead.

Different executions of the algorithm will lead to different solutions because the initial ensemble of configurations in parameter space is selected randomly. We show in Figs. 3 and 4 two different outputs obtained with our approach. Please note that for clarity some labels of the modes have been interchanged manually to remove unnecessary swap operations. This last operation is purely aesthetic, the circuit matrix, the success probability, and the interconnections between optical elements are not altered.

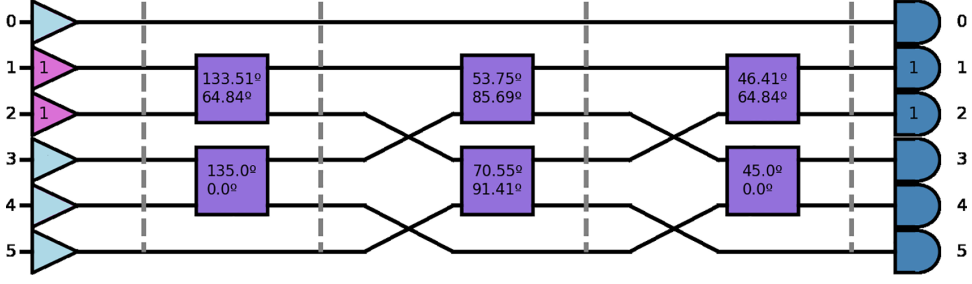


FIG. 3. An approximate CNOT implementation obtained automatically from a combination of an optimization and a pruning algorithm. The weights used in the optimization are $c = 1.0$, $b = 0.1$, $l = 0.1$, $r = 0.01$, and $o = 0.1$. The accuracy test weights are $d = 1.0$ and $h = 0.5$. The mapping between qubits and modes at the input is $q_0 \rightarrow (5, 0)$ and $q_1 \rightarrow (3, 4)$, while the mapping at the output is $q_0 \rightarrow (5, 0)$ and $q_1 \rightarrow (4, 3)$. The rest of the modes are ancillas initialized to one photon each. This gate success probability is $p = 5.7\%$.

Figure 3 shows an approximate implementation of the CNOT gate ($C_{HS} = 0.003$) using six beam splitters. It has a maximum error of 0.6% if inputs $|10\rangle$ or $|11\rangle$ are used. For a $|10\rangle$ input, the output should be $|11\rangle$ with probability 1 but it is 0.994 instead. The remaining probability 0.006 (0.6%) increases the probability of the output $|11\rangle$ as an error.

This CNOT implementation has a success probability of around 5.7%. It represents a typical solution that will be found by the algorithm. Nevertheless, better implementations can be found after repeated calls. Furthermore, for this particular gate, there are solutions with the phase parameter of the beam splitters ϕ fixed to zero, thus restricting the search space. For example, we can see an exact solution in Fig. 4. This uses seven beam splitters, one more than in the previous approximate implementation, but this solution is numerically exact ($C_{HS} : 6 \times 10^{-6}$) within machine precision and has an improved success probability of around 7.4%. These are similar values to the implementations of the CZ gate based on conditional phase shift operations in Refs. [4,29] using the same number of ancilla modes.

The next step is to check the method for actual quantum circuits instead of a single gate. We have considered a Deutsch-Jozsa (DJ) algorithm with one input qubit and one work qubit [30], a Grover algorithm with two input qubits [31], and a two-qubit quantum Fourier transform [32]. The implementation of our method allows to define circuits in Qiskit [33] as input to the compilation algorithm.

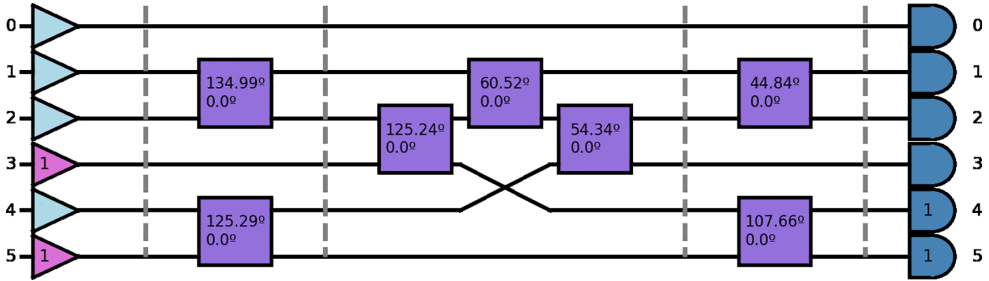


FIG. 4. An exact CNOT implementation obtained automatically from a combination of an optimization and a pruning algorithm. The weights used in the optimization are similar, as shown in Fig. 3, but the beam splitter phases are fixed to zero. In this particular case, we used the Hilbert-Schmidt test with $c = 1$ instead of the combination of a distance and a heuristic. The mapping between qubits and modes at the input is $q_0 \rightarrow (4, 0)$ and $q_1 \rightarrow (1, 2)$, while the mapping at the output is $q_0 \rightarrow (1, 0)$ and $q_1 \rightarrow (2, 3)$. The rest of the modes are ancillas initialized to one photon each. This gate' success probability is $p = 7.4\%$ (similar to the controlled phase-flip (CZ) operations in Refs. [4,29]).

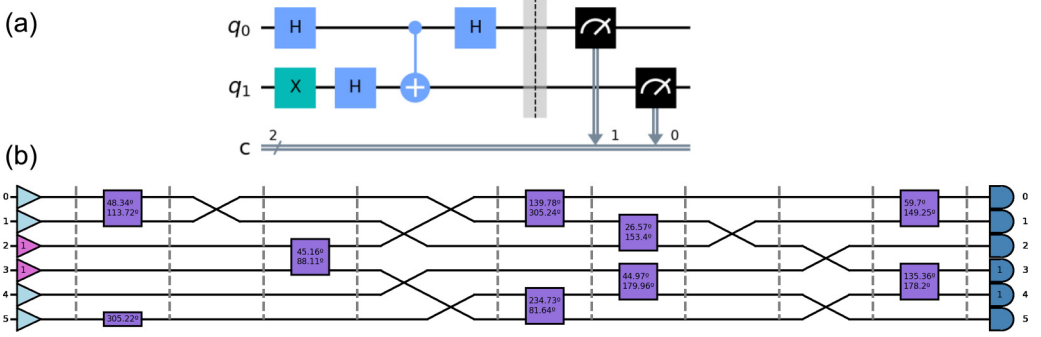


FIG. 5. (a) A quantum circuit of the two-qubit Deutsch-Jozsa algorithm defined in Qiskit [33] for an oracle $f(x) = x$. This circuit is used as an input to the compiler. (b) An exact photonic implementation for the circuit in panel (a) was obtained automatically from the iterative combination of optimization and pruning. The weights used in the optimization are $c = 1.0$, $b = 0.1$, $l = 0.1$, $r = 0.01$, and $o = 0.1$. The accuracy test weights are $d = 1.0$ and $h = 0.5$. The mapping between qubits and modes at the input is $q_0 \rightarrow (0, 1)$ and $q_1 \rightarrow (5, 4)$, while the mapping at the output is $q_0 \rightarrow (0, 1)$ and $q_1 \rightarrow (5, 2)$. The rest of the modes are ancillas initialized to one photon each.

B. Deutsch-Jozsa circuit

In Figs. 5(a), 6(a), 7(a), and 8(a), DJ [30] target circuits are shown for oracles $f(x) = x$, $f(x) = \bar{x}$, $f(x) = 0$, and $f(x) = 1$, respectively. In Fig. 5(b), the corresponding optical implementation for the DJ circuit with the $f(x) = x$ oracle is plotted. It contains nine optical elements, eight beam splitters, and one phase shifter. The same photonic circuit with different parameters for its optical elements can be used to implement the DJ circuit with the $f(x) = \bar{x}$, $f(x) = 0$, and $f(x) = 1$ oracle. All of these implementations are exact and, furthermore, all of them improve on the solution obtained by direct substitution of each gate by their optical counterparts by using two less optical elements. The CZ gate presented by Knill [29] already contains six optical elements (four beams plitters and two phase shifters). Furthermore, two additional Hadamard gates are needed to transform the CZ into a CNOT. Note that a Hadamard gate can be made with a single beam splitter, while an X gate of the original circuit can be obtained by a simple swap operation; therefore, it does not add any optical elements to the count. Therefore, these photonic circuits already represent a small improvement in the quantity of resources and more importantly have been automatically generated by a computer. The success probability of this circuit is $p = 7.4\%$, which is also similar to the implementations of

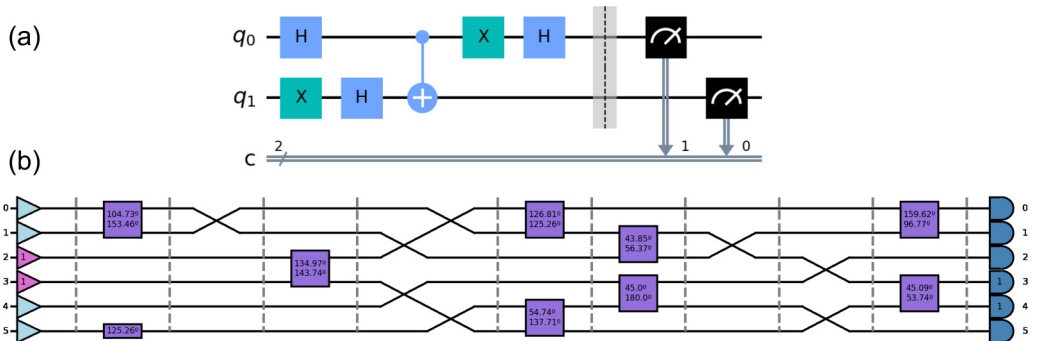


FIG. 6. (a) A quantum circuit of the two-qubit Deutsch-Jozsa algorithm defined in Qiskit [33] for an oracle $f(x) = \bar{x}$. (b) An exact photonic implementation for the circuit in panel (a) with the same structure as the one obtained in Fig. 5.

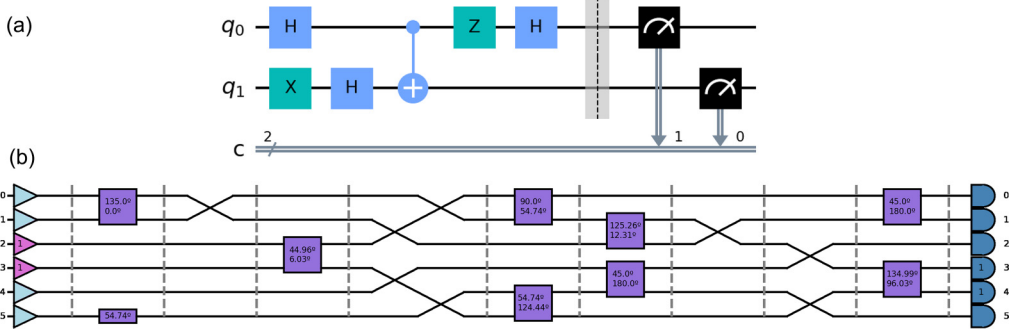


FIG. 7. (a) A quantum circuit of the two-qubit Deutsch-Jozsa algorithm defined in Qiskit [33] for an oracle $f(x) = 0$. (b) An exact photonic implementation for the circuit in panel (a) with the same structure as the one obtained in Fig. 5.

the CZ gate in Refs. [4,29]. Note that the optical circuit is matching the output of the target circuit for any input state of the basis, even if only the state $|00\rangle$ is relevant as the input of the algorithm.

We observe greater improvements in the number of optical elements obtained by this method (in comparison to gate-by-gate substitution) as the target circuits become larger and more involved. We point out that the result of the compilation into photonic circuits does not respect the usual separation of the logical circuit into the oracle and the rest of operations. The purpose of this example is to demonstrate that the proposed method can automatically generate photonic implementations of different quantum functionalities from their logical circuit descriptions, independently of how those functionalities are conventionally decomposed into circuit blocks.

C. Grover circuit

We show in Fig. 9(a) a two-qubit circuit implementing a particular case of a Grover algorithm. An exact photonic implementation was obtained automatically and is shown in Fig. 9(b). The algorithm implements this circuit with a success probability $p = 7.4\%$. This is equivalent to that of a single CZ [29]. This improves over the naive approach of direct substitution of each gate that will give a much lower $p = (2/27)^2$. Furthermore, with 8 beam splitters, this solution is smaller than the 18 optical elements (14 beam splitters and 4 phase shifters) required by the gate substitution approach.

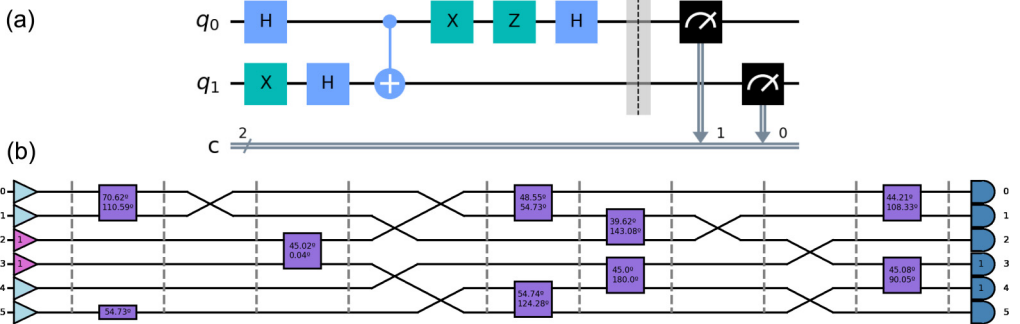


FIG. 8. (a) A quantum circuit of the two-qubit Deutsch-Jozsa algorithm defined in Qiskit [33] for an oracle $f(x) = 1$. (b) An exact photonic implementation for the circuit in panel (a) with the same structure as the one obtained in Fig. 5.

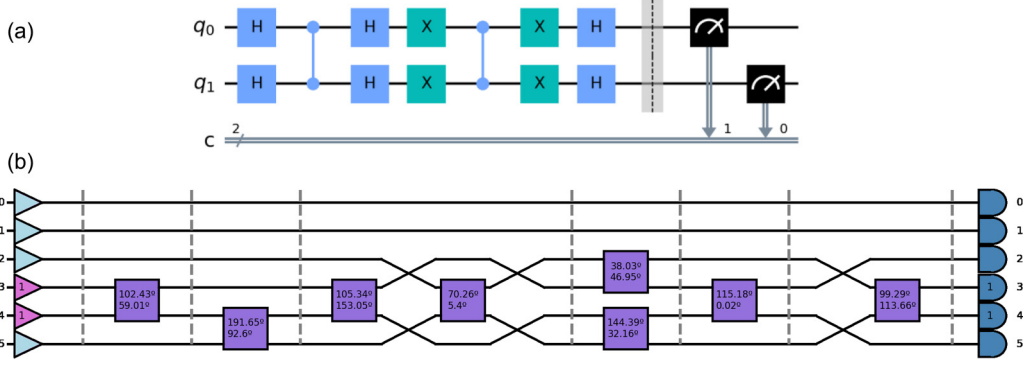


FIG. 9. (a) A quantum circuit of the two-qubit Grover algorithm defined in Qiskit [33]. This circuit is the input of the compiler. (b) An exact implementation for the circuit in panel (a) was obtained automatically from the iterative combination of optimization and pruning. The weights used in the optimization are the same as in previous figures. The mapping between qubits and modes at the input is $q_0 \rightarrow (2, 0)$ and $q_1 \rightarrow (1, 5)$, while the mapping at the output is $q_0 \rightarrow (2, 1)$ and $q_1 \rightarrow (0, 5)$. The rest of the modes are ancillas initialized to one photon each.

D. Fourier transform

We present both a quantum circuit implementing a Fourier transform and its corresponding approximate optical implementation in Fig. 10. The result is a photonic circuit with three ancilla modes, a success probability $p = 0.0148$, and a $C_{HS} = 0.00075$. Additional testing shows a noticeable decrease in accuracy when fewer than three ancillas are employed and no significant improvement when a larger number is considered. Success probability also decreases with large numbers of ancilla modes. Both the output and the errors are more complicated to describe than in previous sections; therefore, we show the output for all input states of the basis in Fig. 11. The obtained results are close to the intended output. The largest error corresponds to a deviation of 0.02 from the expected probability of 0.25 for one outcome in the output probability distribution of a particular input state. The use of three ancilla modes is a good compromise between resource expenditure, accuracy, and success probability.

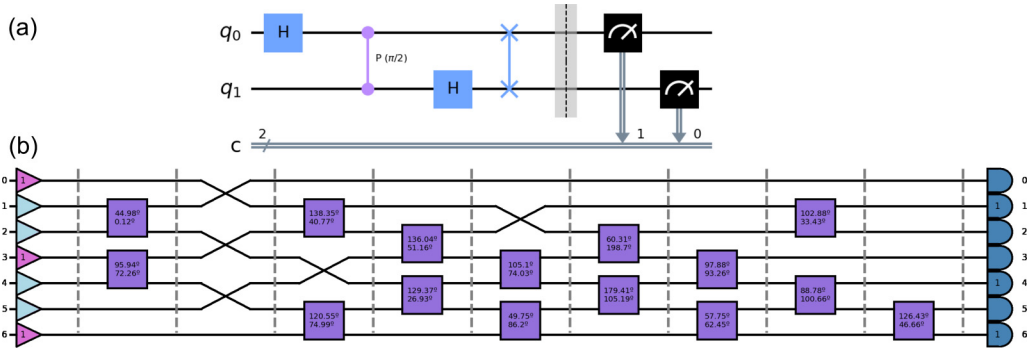


FIG. 10. (a) A quantum circuit of the two-qubit Fourier transform algorithm defined in Qiskit [33]. (b) An approximate implementation for the circuit in panel (a) with a $C_{HS} = 0.00075$. The weights used in the optimization are the same as in previous figures. The mapping between qubits and modes at the input is $q_0 \rightarrow (1, 2)$ and $q_1 \rightarrow (4, 5)$, while the mapping at the output is $q_0 \rightarrow (3, 2)$ and $q_1 \rightarrow (0, 5)$. The rest of the modes are ancillas initialized to one photon each.

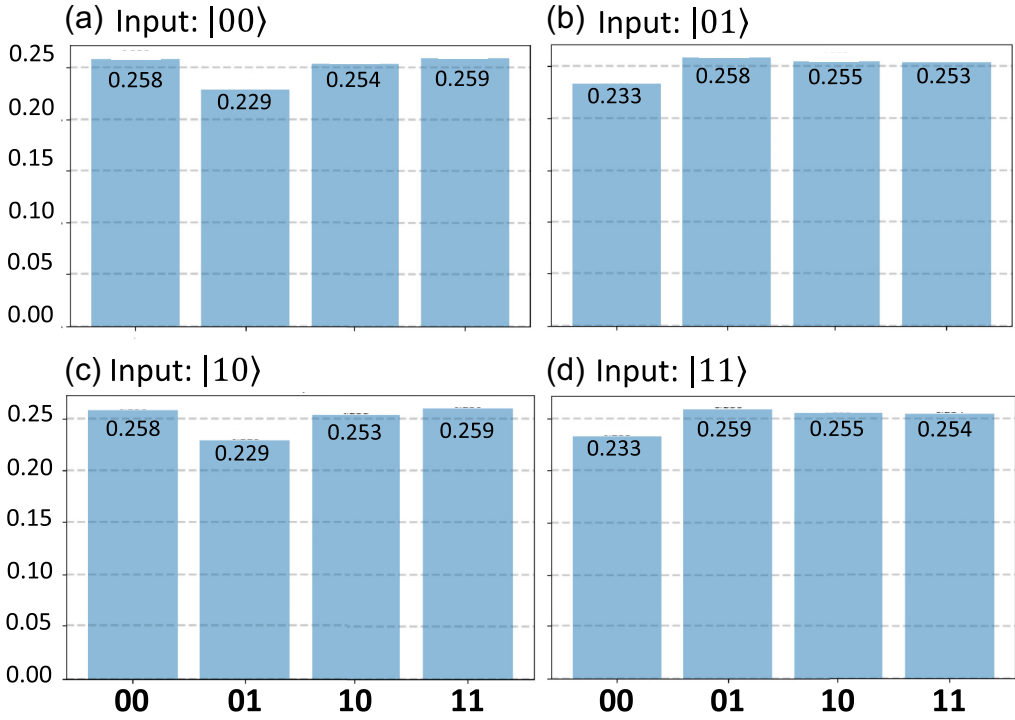


FIG. 11. Probability at the output obtained from SOQCS for the circuit in Fig. 10(b) in case of (a) a $|0, 0\rangle$ input, (b) a $|0, 1\rangle$ input, (c) a $|1, 0\rangle$ input, and (d) a $|1, 1\rangle$ input. All outputs are within a distance of 2% with respect to the intended outcome.

V. CONCLUSIONS

We have presented a combination of an optimization and pruning method that differs from usual quantum compiling approaches [17,20]. Instead of adding elements to find an optical implementation for a quantum algorithm, we start with optimization of general ansatz, and then we prune those unnecessary elements following an established criteria. While this approach is computationally intensive in general, it is quite appropriate for photonic implementations of quantum operations and protocols because of the limited number of optical elements and their clear pruning conditions. We use this approach to find in an automated manner new circuit designs of the CNOT gate and Deutsch-Jozsa, Grover, and Fourier transform two-qubit circuits.

ACKNOWLEDGMENTS

J.O. acknowledges funding by MCIN/AEI/10.13039/501100011033 under Project No. PID2023-151975NB-I00 and by the European Union NextGenerationEU/PRTR. J.V. acknowledges the funding from the IrelandQCI project, cofunded, as a part of the EuroQCI initiative, by the European Union under the Digital Europe Programme No. 101091520, and the Irish Government's Department of Environment, Climate, and Communication. We also thank Dr. Paul Watts for useful comments on the manuscript.

DATA AVAILABILITY

The data that support the findings of this article are not publicly available. The data are available from the authors upon reasonable request.

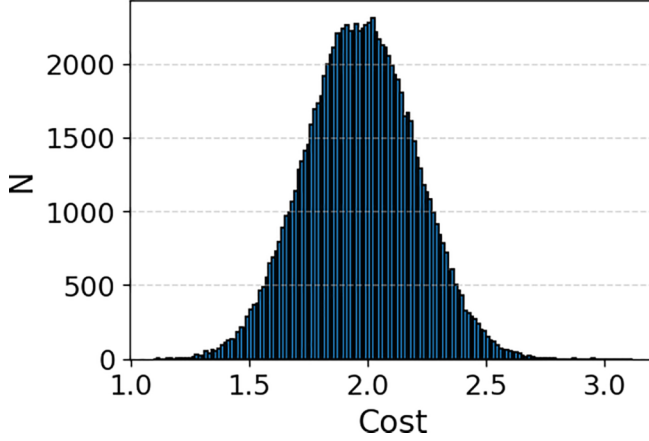


FIG. 12. Histogram with the total cost function value distribution at the end of the first phase.

APPENDIX A: STATISTICS

In Figs. 12 and 13 are printed the histograms of the cost function data at the optimization phases 1 and 2, respectively. The cost function distribution that arises from randomly choosing 100 000 points has a roughly Gaussian shape centered around 2. From this set of points, we choose the 50 closest to 0. It would require a much larger set of random samples to find out good outcomes of the cost function just by random search. In phase 2, a search is performed for the best 50 points with a limit of 2000 evaluations. The result is a roughly uniform distribution with values close or inferior to 0.2–0.3. A full optimization is carried out for the ten best points with a limit of 15 000 evaluation. The final result of these ten points is also a roughly uniform distribution between 0.14 and 0.3. Note that, depending on the solution, pruning may improve the total cost function of the final circuit by removing unnecessary elements that only add error.

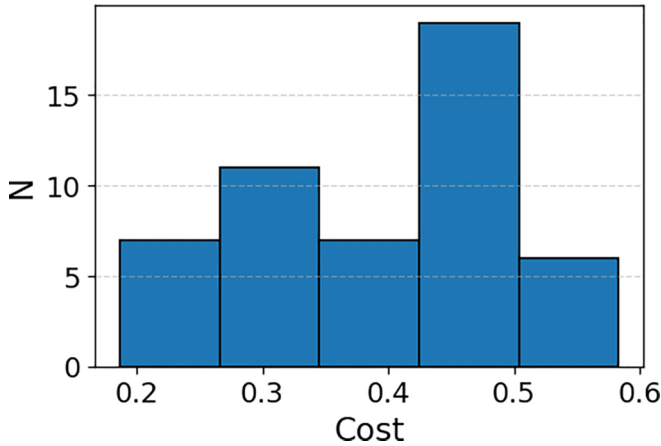


FIG. 13. Histogram with the total cost function value distribution at the end of the second phase.

APPENDIX B: TABLES

In this Appendix, we provide the parameters of the optical elements for the different circuits generated in the results section of this work for greater clarity. Table I presents the parameters for the approximate and exact CNOT implementations, Table II the parameters of the four Deutsch-Jozsa optical circuits, and Tables III and IV for the two-qubit Grover and Fourier transform implementations. These optical elements are numbered from left to right and top to bottom, as shown in the circuits above.

TABLE I. Approximate and exact CNOT implementations.

Approximate CNOT			Exact CNOT		
Beam splitter	θ	ϕ	Beam splitter	θ	ϕ
1	133.51	64.84	1	134.99	0.00
2	135.00	0.00	2	125.29	0.00
3	53.75	85.69	3	125.24	0.00
4	70.45	91.41	4	60.52	0.00
5	46.41	64.84	5	54.34	0.00
6	45.00	0.00	6	44.84	0.00
			7	107.66	0.00

TABLE II. Deutsch-Jozsa implementations for the four functions.

Beam splitter	$f(x) = x$		$f(x) = \bar{x}$		$f(x) = 0$		$f(x) = 1$	
	θ	ϕ	θ	ϕ	θ	ϕ	θ	ϕ
1	48.34	113.72	104.73	153.46	135.00	0.00	70.62	110.59
2	45.16	88.11	134.97	143.74	44.96	6.03	45.02	0.04
3	139.78	305.24	126.81	125.26	90.00	54.74	48.55	54.73
4	234.73	81.64	54.74	137.71	54.74	124.44	54.74	124.28
5	26.57	153.40	43.85	56.37	125.26	12.31	39.62	143.08
6	44.97	179.96	45.00	180.00	45.00	180.00	45.00	180.00
7	59.70	149.25	159.62	96.77	45.00	180.00	44.21	108.33
8	135.36	178.20	45.09	53.74	134.99	96.03	45.08	90.05
Phase shifter	305.22		125.26		54.74		54.73	

TABLE III. Grover implementation.

Beam splitter	θ	ϕ
1	102.43	59.01
2	191.65	92.60
3	105.34	153.05
4	70.26	5.40
5	38.03	46.95
6	144.39	32.16
7	115.18	0.02
8	99.29	113.66

TABLE IV. Fourier implementation.

Beam splitter	θ	ϕ
1	44.98	0.12
2	95.94	72.26
3	138.35	40.77
4	120.55	74.99
5	136.04	51.16
6	129.37	26.93
7	105.10	74.03
8	49.75	86.20
9	60.31	198.70
10	179.41	105.19
11	97.88	93.26
12	57.75	62.45
13	102.88	33.43
14	88.78	100.66
15	126.43	46.67

- [1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, UK, 2010).
- [2] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O'Brien, Quantum computers, *Nature (London)* **464**, 45 (2010).
- [3] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
- [4] E. Knill, R. Laflamme, and G. J. Milburn, A scheme for efficient quantum computation with linear optics, *Nature (London)* **409**, 46 (2001).
- [5] C. R. Myers and R. Laflamme, Linear optics quantum computation: An overview, in *Proceedings of the International School of Physics "Enrico Fermi"*, edited by G. Casati, D. L. Shepelyansky, P. Zoller, and G. Benenti, Volume 162: Quantum Computers, Algorithms and Chaos (IOS Press / Sage Publishing, Amsterdam, The Netherlands, 2006), pp. 45–93.
- [6] P. Kok, W. J. Munro, K. Nemoto, T. C. Ralph, J. P. Dowling, and G. J. Milburn, Linear optical quantum computing with photonic qubits, *Rev. Mod. Phys.* **79**, 135 (2007).
- [7] J. L. O'Brien, A. Furusawa, and J. Vučković, Photonic quantum technologies, *Nat. Photon.* **3**, 687 (2009).
- [8] T. C. Ralph and G. J. Pryde, Optical quantum computation, in *Progress in Optics* (Elsevier, 2010), Vol. 54, Chap. 4, pp. 209–269.
- [9] J. L. O'Brien, Optical quantum computing, *Science* **318**, 1567 (2007).
- [10] Q. Zhang, M. Li, Y. Chen, X. Ren, R. Osellame, Q. Gong, and Y. Li, Femtosecond laser direct writing of an integrated path-encoded CNOT quantum gate, *Opt. Mater. Express* **9**, 2318 (2019).
- [11] J. Wang, D. Bonneau, M. Villa, J. W. Silverstone, R. Santagati, S. Miki, T. Yamashita, M. Fujiwara, M. Sasaki, H. Terai, M. G. Tanner, C. M. Natarajan, R. H. Hadfield, J. L. O'Brien, and M. G. Thompson, Chip-to-chip quantum photonic interconnect by path-polarization interconversion, *Optica* **3**, 407 (2016).
- [12] G. J. Milburn, Quantum optical Fredkin gate, *Phys. Rev. Lett.* **62**, 2124 (1989).
- [13] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, Optimal design for universal multiport interferometers, *Optica* **3**, 1460 (2016).
- [14] S. A. Fldzhyan, M. Y. Saygin, and S. P. Kulik, Optimal design of error-tolerant reprogrammable multiport interferometers, *Opt. Lett.* **45**, 2632 (2020).
- [15] J. Eisert, Optimizing linear optics quantum gates, *Phys. Rev. Lett.* **95**, 040502 (2005).
- [16] M. Krenn, J. S. Kottmann, N. Tischler, and A. Aspuru-Guzik, Conceptual understanding through efficient automated design of quantum optical experiments, *Phys. Rev. X* **11**, 031044 (2021).

- [17] M. Krenn, M. Malik, R. Fickler, R. Lapkiewicz, and A. Zeilinger, Automated search for new quantum experiments, *Phys. Rev. Lett.* **116**, 090405 (2016).
- [18] Stochastic optical quantum circuit simulator (SOQCS), GitHub, 2023, <https://github.com/SOQCSAdmin/SOQCS>, accessed: 15 August 2013.
- [19] J. Osca and J. Vala, SOQCS: A stochastic optical quantum circuit simulator, *SoftwareX* **25**, 101603 (2024).
- [20] S. Khatiri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborge, and P. J. Coles, Quantum-assisted quantum compiling, *Quantum* **3**, 140 (2019).
- [21] L. Madden and A. Simonetto, Best approximate quantum compiling problems, *ACM Trans. Quantum Comput.* **3**, 1 (2022).
- [22] X. Qiang, X. Zhou, J. Wang, C. M. Wilkes, T. Loke, S. O’Gara, L. Kling, G. D. Marshall, R. Santagati, T. C. Ralph, J. B. Wang, J. L. O’Brien, M. G. Thompson, and J. C. F. Matthews, Large-scale silicon quantum photonics implementing arbitrary two-qubit processing, *Nat. Photon.* **12**, 534 (2018).
- [23] X. Gao, M. Erhard, A. Zeilinger, and M. Krenn, Computer-inspired concept for high-dimensional multipartite quantum gates, *Phys. Rev. Lett.* **125**, 050501 (2020).
- [24] T. C. Ralph, N. K. Langford, T. B. Bell, and A. G. White, Linear optical controlled-NOT gate in the coincidence basis, *Phys. Rev. A* **65**, 062324 (2002).
- [25] J. L. O’Brien, G. J. Pryde, A. G. White, T. C. Ralph, and D. Branning, Demonstration of an all-optical quantum controlled-NOT gate, *Nature (London)* **426**, 264 (2003).
- [26] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization, *ACM Trans. Math. Software* **23**, 550 (1997).
- [27] E. Knill, Approximation by quantum circuits, [arXiv:quant-ph/9508006](https://arxiv.org/abs/quant-ph/9508006).
- [28] P. Lundow and K. Markström, Efficient computation of permanents, with applications to boson sampling and random matrices, *J. Comput. Phys.* **455**, 110990 (2022).
- [29] E. Knill, Quantum gates using linear optics and postselection, *Phys. Rev. A* **66**, 052306 (2002).
- [30] D. David and J. Richard, Rapid solution of problems by quantum computation, *Proc. R. Soc. London A* **439**, 553 (1992).
- [31] L. K. Grover, in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, STOC ’96 (Association for Computing Machinery, New York, 1996), pp. 212–219.
- [32] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26**, 1484 (1997).
- [33] M. Treinish, J. Lishman, L. Bello, J. Gambetta, D. M. Rodríguez, M. Marques, J. Gacon, P. Nation, C. J. Wood, Ewinston, *et al.*, Qiskit/qiskit: Qiskit 2.3.0, <https://www.ibm.com/quantum/qiskit>, accessed: 1 December 2025.