Combinatorial optimization via highly efficient quantum walks

S. Marsh ●* and J. B. Wang ●†

Department of Physics, The University of Western Australia, Perth, WA 6009, Australia

We present a highly efficient quantum circuit for performing continuous time quantum walks (CTQWs) over an exponentially large set of combinatorial objects, provided that the objects can be indexed efficiently. CTQWs form the core mixing operation of a generalized version of the quantum approximate optimization algorithm, which works by "steering" the quantum amplitude into high-quality solutions. The efficient quantum circuit holds the promise of finding high-quality solutions to certain classes of NP-hard combinatorial problems such as the Travelling Salesman Problem, maximum set splitting, graph partitioning, and lattice path optimization.

I. INTRODUCTION

Combinatorial optimization problems are known to be notoriously difficult to solve, even approximately in general [1]. Quantum algorithms are able to solve these problems more efficiently, with a brute force quantum search offering a guaranteed square root speedup over the classical approach [2,3]. Such a speedup is, unfortunately, insufficient to provide practically useful solutions, since these combinatorial optimisation problems scale up exponentially.

Farhi *et al.* [4] proposed the quantum approximate optimization algorithm (QAOA), derived from approximating the quantum adiabatic algorithm on a gate model quantum computer, to find high-quality solutions for general combinatorial optimization problems [4]. More recently, we extended the QAOA algorithm to solve constrained combinatorial optimization problems via alternating continuous-time quantum walks over efficiently identifiable feasible solutions and solution-quality-dependent phase shifts [5]. Throughout this paper, we refer to this quantum-walk-assisted generalization as QWOA.

The core component of QWOA is the continuous time quantum walk (CTQW) [6–9], which acts as a "mixing" operator for the algorithm, with probability amplitudes transferred between feasible solutions of the problem. A CTQW over the undirected graph with adjacency matrix A, which connects all feasible solutions, is defined by the propagator $\hat{U}(t) = e^{-itA}$. Quantum walks have markedly different behavior to classical random walks due to intrinsic quantum correlations and interference [10–13], and they played a central role in quantum simulation

and quantum information processing [14–25]. CTQWs are particularly well known for their applications to quantum spatial search [11,26,27], where the system is evolved for a sufficient length of time under the addition of the graph Hamiltonian A and an oracular Hamiltonian encoding the marked element(s). However, in QWOA we apply CTQWs independently, where a quantum circuit for $\hat{U}(t) = e^{-itA}$ is used to map some initial amplitude distribution over the vertices to the distribution obtained after "walking" for time t. The oracular Hamiltonian encoding solution qualities are then applied sequentially, interleaved with further CTQWs. Of significance is that, for the graph structures considered in this paper, the runtime of a QWOA circuit can be made independent of the walk times, leading to a distinct algorithmic advantage.

In this paper, we discuss a significant and innovative application of QWOA to a wide range of combinatorial domains, which is defined as the set of feasible solutions to some specified combinatorial optimization problem. In Sec. II we describe the QWOA procedure and its quantum circuit implementation. In Sec. III, we detail our method for quantum walking over a variety of combinatorial domains. Specifically, the domains applicable to this method are those with an associated *indexing function*, which efficiently identifies each object with a unique integer index. Indexing combinatorial objects is called ranking in the literature, however, we refer to it here as "indexing" to avoid confusion with ranking objects by their quality. We show that the domain of combinatorial objects can be connected by any circulant graph, barring some minor restrictions, which would ensure a highly efficient quantum circuit implementation. We also show how to design a unitary that efficiently performs the indexing on computational basis states representing objects.

In Sec. IV, we give a number of applicable combinatorial domains along with their associated NP optimization problems, including well-known problems such as the Travelling Salesman Problem and constrained portfolio optimisation. In Sec. V, we present specific quantum circuits to implement the

---

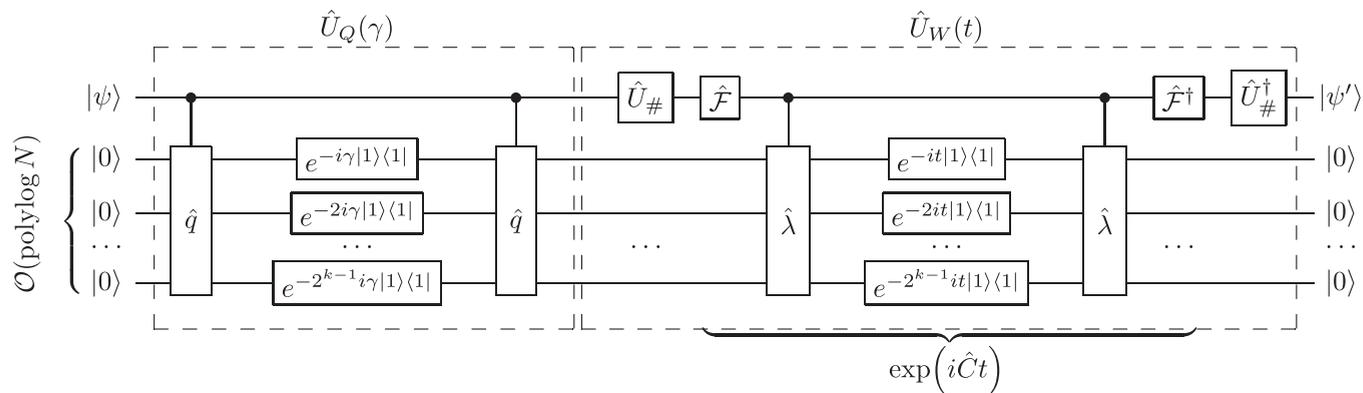*samuel.marsh@research.uwa.edu.au

†jingbo.wang@uwa.edu.au

FIG. 1. An illustrative QWOA circuit for optimization over combinatorial domains having efficient indexing and un-indexing functions. The circuit performs $|\psi'\rangle = \hat{U}_W(t)\hat{U}_Q(\gamma)|\psi\rangle$ with $\mathcal{O}(\text{polylog } N)$ depth. Here the operator $\hat{q}$ evaluates the quality of a solution to k bits of precision, and $\hat{\lambda}$ computes the eigenvalue of the circulant matrix to the same precision. The rotation angles γ and t can be varied by a classical optimizer.

indexing functions. Finally, we describe the relevance of the indexing unitary to quantum search in Sec. VI and then make some concluding remarks.

II. QUANTUM WALK-ASSISTED OPTIMIZATION ALGORITHM

QWOA uses the continuous time quantum walk as an ansatz, distinguished from the original derivation of the QAOA as a discretised adiabatic evolution. Specifically, in the QWOA framework the quantum system evolves as [5]

$$|\vec{\gamma}, \vec{t}\rangle = \hat{U}_W(t_p)\hat{U}_Q(\gamma_p)\dots\hat{U}_W(t_1)\hat{U}_Q(\gamma_1)|s\rangle. \quad (1)$$

In a combinatorial optimization context, the QWOA procedure can be interpreted as follows.

(1) $|s\rangle$ is the initial state, which can be taken to be an equal superposition over all of the feasible combinatorial solutions.

(2) $\hat{U}_Q(\gamma) = e^{i\gamma\hat{Q}}$, where $\hat{Q}$ is a diagonal matrix with diagonal elements $q(x)$ corresponding to the solution quality of the combinatorial object x. As such it applies a phase shift to each combinatorial object, proportional to γ and its *quality*.

(3) $\hat{U}_W(t)$ performs a continuous-time quantum *walk* for time t over the combinatorial domain; its details will be discussed in the next section.

(4) The set of $2p$ parameters $\vec{\gamma} = (\gamma_1, \dots, \gamma_p)$ and $\vec{t} = (t_1, \dots, t_p)$ are chosen to maximize the expectation value $\langle\vec{\gamma}, \vec{t}|\hat{Q}|\vec{\gamma}, \vec{t}\rangle$, representing the average measured solution quality.

The QWOA state evolution consists of an interleaved series of phase shifts, which introduce a bias to solutions dependent on their quality, and quantum walks to mix amplitude between solutions. A higher choice of p leads to better solutions at the cost of a longer quantum computation with more variational parameters. In the general case, a classical optimizer can be used to vary the parameters $\vec{\gamma}$ and $\vec{t}$ such that the expectation value of the solution quality is maximized. A circuit diagram for the QWOA is illustrated in Fig. 1, consisting of a solution quality dependent phase shift $\hat{U}_Q(\gamma)$ followed by a quantum walk over the feasible solutions $\hat{U}_W(t)$.

The $\hat{U}_Q(\gamma)$ component of the QWOA circuit is straightforward to implement for any combinatorial optimization

problem in the NPO complexity class. Put another way, given a combinatorial object x, we require that the solution quality $q(x)$ can be efficiently computed. If so, there is an efficient quantum circuit that implements the desired solution quality-dependent phase shift [28]. We show this implementation in the left dashed box of Fig. 1. With this in mind, the remainder of this paper proposes a generic approach to efficiently implementing $\hat{U}_W(t)$ over a wide range of combinatorial domains.

III. QUANTUM WALKS OVER COMBINATORIAL DOMAINS

Consider a set of M combinatorial objects, each encoded as an n-bit string. An example is k-combinations of a set $S = \{s_0, s_1, \dots, s_{n-1}\}$, where each combinatorial object corresponds to a unique k-combination of S. A straightforward encoding of a specific k-combination as a length-n binary string is $x = x_0 x_1 \dots x_{n-1}$, where $x_j = 1$ iff s_j is selected as part of the combination. In this case, the aim is to perform a quantum walk over a graph connecting each of the $M = \binom{n}{k}$ possible k-combinations of S. Current approaches resort to sparse or approximate Hamiltonian simulation of the XY model Hamiltonian [29,30], which is not ideal as its corresponding
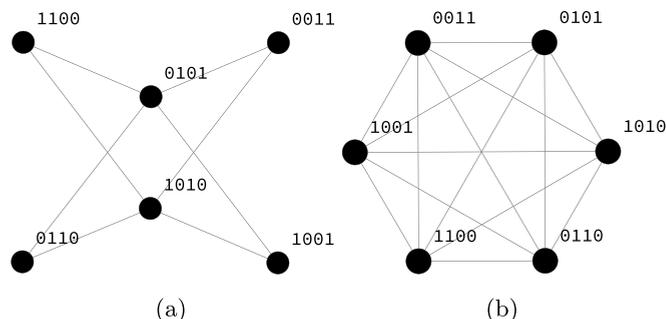

FIG. 2. (a) XY model connectivity of length-4 bitstrings with exactly 2 bits on. The two central solutions have double the degree of the outer ones, introducing bias. (b) Using our proposed approach with the complete graph maintains symmetry among solutions.
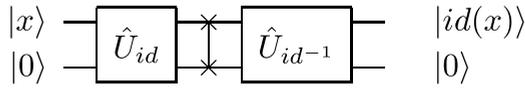
FIG. 3. Circuit for $\hat{U}_\#$, using an ancilla register of the same size as the input register.

quantum circuit changes with the value of the walk time t (requiring a longer circuit to approximate the quantum walk dynamics for a longer time). Furthermore the XY model does not maintain symmetry among the relevant vertices, introducing bias into the optimization algorithm, as illustrated in Fig. 2. Our method provides the ability to resolve this issue with (for example) the complete graph or the cycle graph, preserving symmetry.

We choose to focus on circulant connectivity because all circulant graphs are diagonalized by the Fourier transform. Thus, the highly efficient quantum Fourier transform (QFT) can be used to diagonalize any choice of circulant graph, while the circuit structure remains largely unchanged [31–34]. The first step of our approach is to unitarily map the M binary strings corresponding to valid combinatorial objects, which are "scattered" in a larger space of 2^n binary strings, to a canonical subspace of the first M binary strings (i.e., 0 through to $M - 1$). This is done using a so-called "indexing unitary" $U_\#$. The second step is to perform a continuous time quantum walk over the first M computational basis states connected as a circulant graph, where M is not necessarily a power of 2. After "un-indexing" using $U_\#^\dagger$, a CTQW over the circulantly connected combinatorial objects is performed. Our approach is summarized with

$$\hat{U}_W(t) = \hat{U}_\#^\dagger \hat{\mathcal{F}}_M^\dagger \exp(i\hat{\lambda}t) \hat{\mathcal{F}}_M \hat{U}_\# \,, \qquad (2)$$

where $\hat{U}_\#$ is the indexing unitary, $\hat{\mathcal{F}}_M$ is the quantum Fourier transform modulo M [35], and $\hat{\lambda}$ is a diagonal matrix containing the M eigenvalues of the circulant adjacency matrix, which describes the connectivity of the objects. In the following subsections we describe each component of the overall quantum walk, and the corresponding circuit implementation.

A. Indexing with $\hat{U}_\#$

Let $S = \{s_1, s_2, \ldots, s_M\}$ be a set of bitstrings encoding combinatorial objects. Assume the combinatorial objects can be encoded using n qubits. For example, k-combinations from $[n] = \{0, 1, \ldots, n - 1\}$ may be represented by an n-bit string with exactly k bits set. We wish to perform a continuous time quantum walk (CTQW) over S. Consider a bijective function $id : S \to [M]$ that identifies each element of S with its unique integer "index," where both id and id^{-1} are efficiently computable. Typical indexing functions usually order lexicographically, although for our purposes the manner of ordering is not relevant. Then with the application of a unitary that performs indexing on the subspace of valid combinatorial bitstrings, quantum walking over S is reduced to quantum walking over $[M]$.

We now briefly explain how to construct the indexing unitary $\hat{U}_\#$, given classical algorithms for indexing and un-indexing. Since we assume we have an efficient classical algorithm for indexing, we can construct a reversible circuit to perform $\hat{U}_{id}|x\rangle|z\rangle = |x\rangle|z \oplus id(x)\rangle$. Similarly, we can construct a unitary for un-indexing, $\hat{U}_{id^{-1}}|id(x)\rangle|z\rangle = |id(x)\rangle|z \oplus x\rangle$. It is straightforward to check that

$$\hat{U}_\#|x\rangle|0\rangle \equiv U_{id^{-1}} \hat{S} \hat{U}_{id}|x\rangle|0\rangle = |id(x)\rangle|0\rangle \,, \qquad (3)$$

where $\hat{S}$ swaps the first and second registers. We give the circuit diagram in Fig. 3. Clearly, un-indexing is performed with $\hat{U}_\#^\dagger$.

The indexing algorithm depends on the specific combinatorial family. However, there is a general framework by Wilf [36] that applies when the combinatorial objects are constructed recursively. For example, an arbitrary n-choose-k combination is built from the $\binom{n-1}{k}$ subsets not including n, and the $\binom{n-1}{k-1}$ subsets that do include n. In this sense, the construction of a combinatorial object can be thought of as a certain path on a directed graph, with each step partially building the object. The initial point of the walk—for example, (n, k) in the lattice describing all set combinations—describes the so-called *order* of the object. In this framework, the indexing algorithm has complexity scaling as a polynomial in the object's order. As per [36], some illustrative combinatorial families for which this framework applies include k-subsets
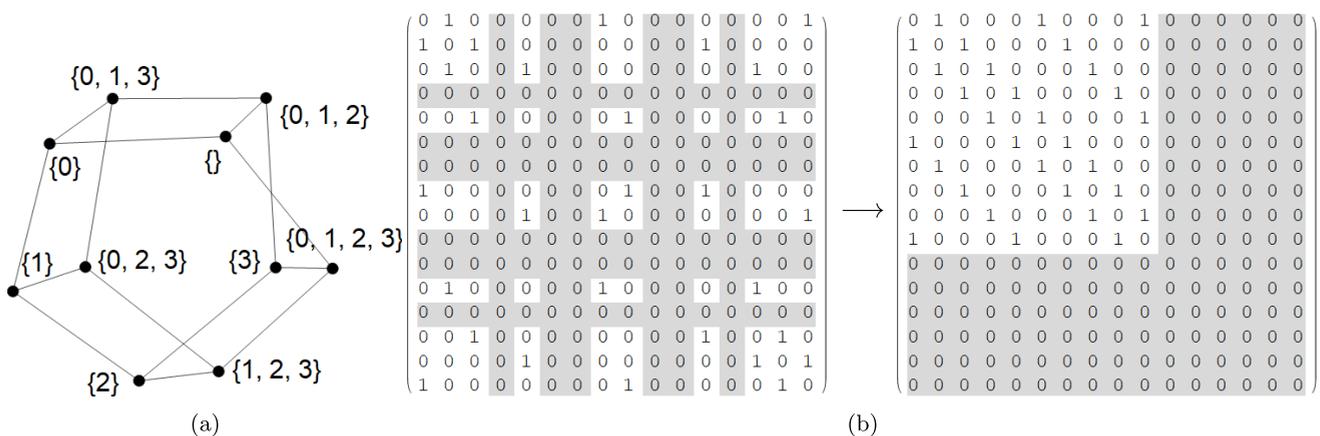

FIG. 4. (a) A Möbius ladder graph connecting subsets of $\{0, 1, 2, 3\}$ that do not contain exactly two elements. (b) The adjacency matrix before and after applying the indexing unitary. Rows colored grey correspond to invalid states.

of $[n]$, permutations of $[n]$ with k cycles, and k-partitions of $[n]$. Associated with each of these example families is a formula to obtain the number of objects of a given order—the binomial coefficients, the Stirling numbers of the first kind, and the Bell numbers, respectively. This is a property induced by Wilf's framework, and applies to all the combinatorial families within it. Thus for our purposes, the number of objects M is always explicitly known (or can be determined efficiently).

Finally, given the ability to index objects of a given order, the indexing algorithm can be extended to handle a range of orders $a \leqslant k \leqslant b$. This makes our framework naturally applicable to many more relevant combinatorial domains. For example, to index a combination $\vec{c} = (c_0, \ldots, c_{j-1})$ of $[n]$ with respect to all combinations of $\leqslant K$ elements,

$$id_{\leqslant K}(\vec{c}) = id_{\text{len}(\vec{c})}(\vec{c}) + \sum_{k=0}^{\text{len}(\vec{c})-1} \binom{n}{k}, \qquad (4)$$

where $\text{len}(\vec{c})$ is the order of $\vec{c}$, i.e., the number of elements in the combination.

B. Preparation of the initial state

In the original QAOA, one prepares an equal superposition over all $N = 2^n$ computational basis states. However, in general, up to half of these bitstrings do not correspond to a valid combinatorial object contained in the set S, i.e., $M < N$.

Instead, to prepare the equal superposition $|s\rangle$ over the valid combinatorial objects, we first apply the quantum Fourier transform modulo M to the $|0\rangle^{\otimes n}$ state. There are highly depth-efficient quantum circuits that can be used to implement the Fourier transform with arbitrary modulus [37]. This creates the superposition $(1/\sqrt{M}) \sum_{x=0}^{M-1} |x\rangle$. Then the un-indexing unitary $U_\#^{-1}$ can be carried out, thus efficiently preparing the desired initial state

$$|s\rangle = \frac{1}{\sqrt{M}} \sum_{x \in S} |x\rangle. \qquad (5)$$

C. Quantum walk with $\exp(i\hat{C}t)$

We now connect the indices in a circulant manner, defining a circulant matrix $\hat{C}$ of size $M \times M$. This circulant matrix is diagonalized by the Fourier matrix $\hat{\mathcal{F}}_M$. Therefore the quantum walk over the graph with adjacency matrix C is

$$\exp(i\hat{C}t) = \hat{\mathcal{F}}_M \exp(i\hat{\lambda}t)\hat{\mathcal{F}}_M^\dagger. \qquad (6)$$

Again, the quantum Fourier transform modulo M can be used to efficiently perform the diagonalization.

The diagonal unitary $\exp(i\hat{\lambda}t)$ can be implemented efficiently using well-known methods from [28], similar to
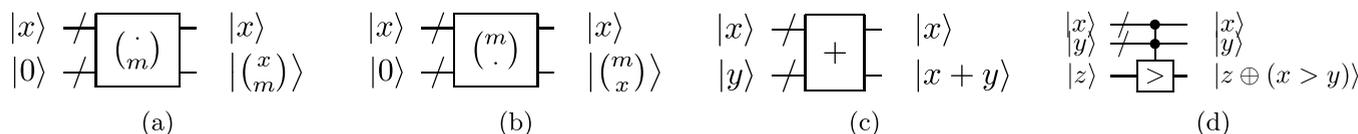
$\hat{U}_Q(\gamma)$. We require that the diagonal elements of $\hat{\lambda}$ be efficiently computable, which is the case for graphs having efficiently computable eigenvalues. This holds for circulant graphs when the maximum degree grows polynomially, or when the eigenvalues are known in closed-form (e.g., complete, cycle, and Möbius ladder graphs). Since each unitary can be implemented efficiently, the entire walk is efficient. It is anticipated that a low choice of p, leading to a relatively shallow circuit, will have a quantum advantage for near-term NISQ applications as discussed in [38].

IV. APPLICATIONS

In the following, we give an overview of a number of combinatorial structures and give their associated indexing functions. A continuous time quantum walk can be efficiently performed over each of following domains using the above-described scheme.

There are a wide range of integer sequences with indexing and un-indexing algorithms (or equivalently an efficient closed-form expression for the nth element of the sequence a_n, where the inverse operation is also efficiently computable). A comprehensive list of such sequences can be found on the On-Line Encyclopedia of Integer Sequences (OEIS) [39], with some examples given below. For brevity we do not give details on the corresponding un-indexing functions; their implementations are similar to indexing with the same runtime complexity, and are available in the references provided in the below sections.

A. Set k-combinations

Let a binary string $x = x_1 \ldots x_n$ denote a combination, where $x_j = 1$ iff the jth element is selected. Then an efficient indexing algorithm, to index a given k-combination among all other k-combinations, is given in Algorithm 1.

Algorithm 1. INDEX_COMB(x, k).

$c \leftarrow$ list of j where $x_j = 1$

sort c in ascending order

return sum from $j = 1$ to k of $\binom{c[j]}{j}$

As discussed in Sec. III A, the k-combination indexing function can be "wrapped" to index combinations among others of chosen sizes. As a nontrivial example, in Fig. 4 we show a Möbius ladder graph connecting all subsets of $\{0, 1, 2, 3\}$ *except* those with exactly two elements.

FIG. 5. Notation for the unitary operations required to implement $\hat{U}_{id}$ for n-choose-k-combinations and n-permutations. Fundamental gate counts: (a) $\mathcal{O}(k \log n)$, (b) $\mathcal{O}(n \log n)$, (c) $\mathcal{O}(\log n)$, (d) $\mathcal{O}(\log n)$.

By choosing n to be even and $k = n/2$, this is the domain of the graph partitioning problem. Similarly, this indexing algorithm can be applied to the critical node detection problem [40]. Another real-world application is to constrained financial portfolio optimization in finance, where up to k assets are selected to create a financial portfolio with minimum risk and/or maximum return.

In Sec. V, we give two quantum circuits to perform efficient k-combination ranking, depending on how the combinations are represented in the quantum register.

B. Permutations

A permutation can be indexed in linear time using Algorithm 2 [41], where π is a permutation of $[n]$ and π^{-1} is the inverse permutation (which can be computed in linear time).

Algorithm 2. INDEX_PERM(n, π, π^{-1}).

---

if $n = 1$ **then**

    return 0

end if

$s \leftarrow \pi[n-1]$

swap $\pi[n-1]$ and $\pi[\pi^{-1}[n-1]]$

swap $\pi^{-1}[s]$ and $\pi^{-1}[n-1]$

return $s + n \times$ INDEX_PERM($n-1, \pi, \pi^{-1}$)

As an example, the well-known NP-hard Travelling Salesman Problem has domain corresponding to the $k!$ possible permutations of visiting each of k cities exactly once and then returning to the starting city. A straightforward encoding is to use $k\lceil \log k \rceil$ qubits, where each block of $\lceil \log k \rceil$ qubits represents the next city to visit. The indexing unitary $U_\#$ can then be used to map this encoding of a tour to an integer from 0 to $(k! - 1)$, and vice versa.

In Sec. V, we provide a simple quantum circuit to perform efficient permutation ranking.

C. Lattice paths

A Dyck path is a lattice path from $(0, 0)$ to (n, n) which never goes above the diagonal $y = x$. They are an example of a combinatorial structure characterized by the Catalan numbers, a sequence appearing in a number of combinatorial applications. Catalan numbers have an efficient indexing function provided in Algorithm 3, where the function

NUM_DYCK(i, j) counts the total number of Dyck paths between $(0, 0)$ and (i, j) [42].

Algorithm 3. INDEX_CATALAN(x).

$n \leftarrow$ number of bits of x

$c_1 \leftarrow 2$

for $j = 2$ to n **do**

    $c_j \leftarrow \max(x_{j-1} + 1, 2j)$

end for

$r \leftarrow 1$

for $j = 1$ to $n - 1$ **do**

    for $k = c_j$ to $b_j - 1$ **do**

        $r \leftarrow r +$ NUM_DYCK($n - j, n + j - k$)

    end for

end for

return r

There are other lattice path applications. For example, an arbitrary length-n path in three-dimensional (3D) space can be characterized by an n-letter word where each letter is chosen from an alphabet of six directions. Any "word" composed of a sequence of letters from some specified alphabet can be indexed, and constraints on some/all of the letters can also be incorporated [43].

V. QUANTUM CIRCUITS FOR INDEXING

In this section, we give some quantum circuits for U_{id}. We use the notation shown in Fig. 5 for the required unitary operations.

A. Combinations

Here we present quantum circuits for indexing the two main representations of k-combinations. Consider a selection of k integers from the set $[n]$. The first representation is to use a binary string x of length n, where $x_j = 1$ if j is selected. This clearly requires n bits for any k. The second representation is to directly represent the k combination as a sequence of k integers, requiring $O(k \log n)$ space. Both circuits follow the indexing method provided in Algorithm 1. To un-index, a circuit to prepare uniform superpositions of Dicke states can be used, such as [44].
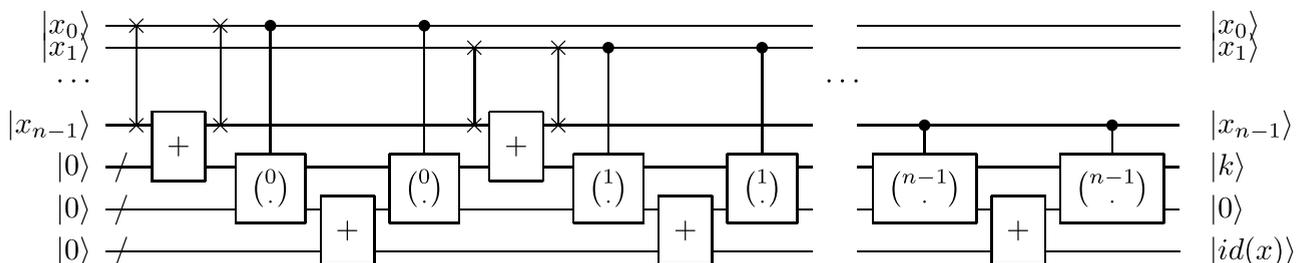

FIG. 6. Quantum circuit for indexing a k-combination represented by $x = x_0 \ldots x_{n-1}$, where $x_i = 1$ if j is selected.
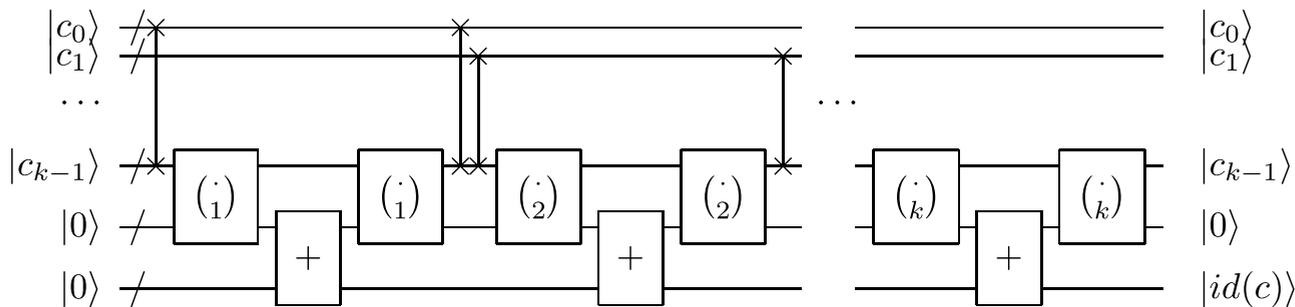
FIG. 7. Quantum circuit for indexing a k-combination represented by $(c_0, \ldots, c_{k-1})$ with elements in ascending order.

The first quantum circuit for indexing k-combinations of $[n]$ is given in Fig. 6. It consists of an input register $|x\rangle$ of n qubits and an output register of the same size, as well as two ancilla registers of $\mathcal{O}(\log k)$ and n qubits, respectively. The fundamental gate count is $\mathcal{O}(nk \log n)$, using the gate complexities for fundamental operations given in Fig. 5.

The quantum circuit for the second representation is shown in Fig. 7. The input, output, and ancilla register are of size $\mathcal{O}(k \log n)$. The fundamental gate count is $\mathcal{O}(k^2 \log n)$.

It depends upon the application as to which representation is more appropriate. If k scales proportionally to n, then the binary representation is more space and time-efficient. Otherwise, if k is small (or constant) then the second quantum circuit is appropriate.

B. Permutations

To design a quantum circuit for indexing permutations, we use the expression

$$id(\pi) = \{\ldots [d_0(n-1) + d_1] \cdot (n-2) + \cdots + d_{n-2}\} \cdot 1 \tag{7}$$

where d_i is the ith digit of the Lehmer code corresponding to π. Explicitly, $d_i = |\{j > i : \pi_j < \pi_i\}|$. Note that to un-index permutations a circuit to prepare superpositions of permutations can be used, such as [45].

Consider a permutation on $[n]$, $\pi = \pi_0 \ldots \pi_{n-1}$. The permutation indexing circuit requires an input register $|\pi\rangle = |\pi_0\rangle \ldots |\pi_{n-1}\rangle$ of size $\mathcal{O}(n \log n)$, to hold each of the n elements $|\pi_i\rangle$ having size $\mathcal{O}(\log n)$. An output register of the same size to hold $id(\pi)$, and an additional single qubit ancilla, are also required.
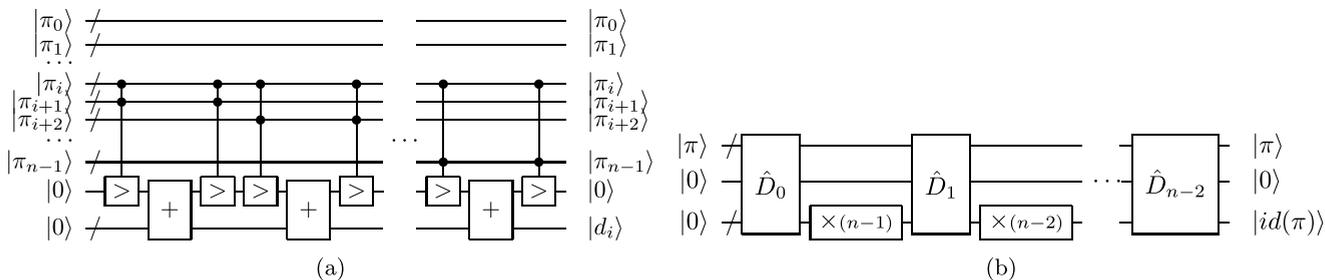
We first give a quantum circuit to implement a Lehmer operator D_i in Fig. 8(a). This circuit performs $|\pi\rangle|0\rangle \mapsto |\pi\rangle|d_i\rangle$, using $\mathcal{O}(n \log n)$ gates.

Using the Lehmer subcircuit, indexing becomes straightforward as per Fig. 8(b). There are $\mathcal{O}(n)$ uses of $\hat{D}_i$, leading to a gate count of $\mathcal{O}(n^2 \log n)$ to index permutations in this way. Although there is a classically linear approach to indexing given in Algorithm 2, it appears to become quadratic when quantized due to requiring access to the input register in a superposition of locations, leading to an additional linear overhead [46].

VI. QUANTUM SEARCH

Finally, we briefly make explicit the connection with quantum search. The aim here is to search for one or more combinatorial objects having a desired property out of a set of M objects. Again, M is not necessarily a power of 2. Typically, one would encode each solution using n bits, for sufficiently large n, and perform a Grover search over the space of $N = 2^n$ binary strings (not marking the binary strings which do not correspond to any combinatorial object). Using the indexing unitary, however, the search space can be cut down from N to M. As per Brassard *et al.* [3], the Grover iteration over a set of size M takes the form

$$\hat{Q} = -\hat{\mathcal{F}}_M \hat{S}_0 \hat{\mathcal{F}}_M^{-1} \hat{S}_f, \tag{8}$$

where $\hat{S}_f$ conditionally negates the amplitude of objects meeting the search criteria and $\hat{S}_0$ conditionally negates the amplitude of the all-zero state. The only modification required is $\hat{S}_f \mapsto \hat{S}_\# \hat{S}_f \hat{U}_\#^\dagger$. This un-indexes the integers to their corresponding combinatorial object, so each object can be meaningfully interpreted and marked according to some

FIG. 8. (a) The quantum circuit for unitary operator $\hat{D}_i$, where $\hat{D}_i|\pi\rangle|0\rangle = |\pi\rangle|d_i\rangle$. Here, d_i is the ith digit of the Lehmer code corresponding to permutation π. (b) The quantum circuit for indexing a given permutation π.

combinatorial criteria and then be mapped back to an integer index. Thus, the modified Grover iteration is

$$Q' = -\hat{\mathcal{F}}_M \hat{S}_0 \hat{\mathcal{F}}_M^{-1} \hat{U}_\# \hat{S}_f \hat{U}_\#^\dagger . \qquad (9)$$

Quantum search over a set of M combinatorial objects with an associated indexing function can be performed in $\mathcal{O}(\sqrt{M/k})$ for k marked combinatorial objects. In general, this will lead to a constant-factor speedup over Grover search on the full Hilbert space.

VII. CONCLUSION

In this paper we established a general framework for combinatorial optimization via highly efficient continuous-time quantum-walk over finite but exponentially large sets of combinatorial objects. We focus on combinatorial families with an associated "indexing algorithm," which efficiently identifies the position of a given combinatorial objects among all objects of the same order. Examples of combinatorial families with associated indexing functions include combinations, permutations, partitions, and lattice walks under a variety of constraints. Using a quantum indexing unitary, the binary representation of the objects can be mapped to a smaller and simpler canonical subspace to allow straightforward implementation of the CTQW.

This approach is particularly beneficial for use as an quantum walk-based approximate optimization scheme, to optimize over nontrivial combinatorial domains. The proposed efficient quantum algorithm requires that the choice of graph connecting the combinatorial objects is circulant. The variational parameters in this case are the quantum walk times $\vec{t}$ and the quality phase factors $\vec{\gamma}$. A specific benefit of our approach is that the size and design of the quantum circuit is completely independent of these parameters, so the circuit does not need to be recompiled each time these parameters are updated. In addition, by choosing a symmetric graph, the optimization algorithm is not biased towards any solution over another. In this paper we also briefly discuss the relevance to Grover search over combinatorial domains.

Furthermore, each $\hat{U}_W$ in the QWOA state evolution does not need to remain the same operator. One can consider different connectivities among combinatorial objects for each quantum walk. For example, it may be beneficial to start with an initial quantum walk that is highly connected (cf. complete graph) and decrease the intersolution connectivity each time, ending with a quantum walk over the objects connected as a cycle graph. This approach may be beneficial to "hone in" on a high-quality solution through a systematically modifying $\hat{U}_W$; we leave this to future work.

Finally, it is worth noting that this approach does not work if the combinatorial domain cannot be efficiently indexed, or if its size cannot be efficiently determined. For example, independent sets do not have an efficient indexing function; even the problem of counting the number of independent sets of a graph belongs to the complexity class #P.

ACKNOWLEDGMENTS

[1] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 4th ed. (Springer-Verlag, Berlin, 2007).

[2] L. K. Grover, Chaos Solitons and Fractals **10**, 1695 (1999).

[3] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, in *Quantum Computation and Information (Washington, DC, 2000)*, AMS Contemporary Mathematics (American Mathematical Society, Providence, RI, 2002), Vol. 305, pp. 53–74.

[4] E. Farhi, J. Goldstone, and S. Gutmann, arXiv:1411.4028.

[5] S. Marsh and J. B. Wang, Quant. Info. Proc. **18**, 61 (2019).

[6] E. Farhi and S. Gutmann, Phys. Rev. A **58**, 915 (1998).

[7] J. Kempe, Contemp. Phys. **44**, 307 (2003).

[8] A. M. Childs, D. Gosset, and Z. Webb Science **339**, 791 (2013).

[9] K. Manouchehri and J. Wang, *Physical Implementation of Quantum Walks*, Quantum Science and Technology (Springer, Heidelberg, 2014).

[10] H. Tang, C. Di Franco, Z.-Y. Shi, T.-S. He, Z. Feng, J. Gao, K. Sun, Z.-M. Li, Z.-Q. Jiao, T.-Y. Wang, M. S. Kim, and X.-M. Jin, Nat. Photon. **12**, 754 (2018).

[11] A. M. Childs and J. Goldstone, Phys. Rev. A **70**, 022314 (2004).

[12] G. S. Engel, T. R. Calhoun, E. L. Read, T.-K. Ahn, T. Mancal, Y.-C. Cheng, R. E. Blankenship, and G. R. Fleming, Nature **446**, 782 (2007).

[13] S. D. Berry and J. B. Wang, Phys. Rev. A **82**, 042333 (2010).

[14] S. Lloyd, Science **273**, 1073 (1996).

[15] A. Ambainis, SIAM J. Comput. **37**, 210 (2007).

[16] B. L. Douglas and J. B. Wang, J. Phys. A: Math. Theor. **41**, 075303 (2008).

[17] S. D. Berry and J. B. Wang, Phys. Rev. A **83**, 042317 (2011).

[18] A. Schreiber, A. Gábris, P. P. Rohde, K. Laiho, M. Štefaňák, V. Potoček, C. Hamilton, I. Jex, and C. Silberhorn, Science **336**, 55 (2012).

[19] J. A. Izaac, J. B. Wang, P. C. Abbott, and X. S. Ma, Phys. Rev. A **96**, 032305 (2017).

[20] F. Levi, Nat. Phys. **13**, 926 (2017).

[21] Y. Ming, C.-T. Lin, S. D. Bartlett, and W.-W. Zhang, npj Comput. Mater. **5**, 88 (2019).

[22] M. E. Tai, A. Lukin, M. Rispoli, R. Schittko, T. Menke, D. Borgnia, P. M. Preiss, F. Grusdt, A. M. Kaufman, and M. Greiner, Nature **546**, 519 (2017).

[23] N. C. Harris, G. R. Steinbrecher, M. Prabhu, Y. Lahini, J. Mower, D. Bunandar, C. Chen, F. N. C. Wong, T. Baehr-Jones, M. Hochberg, S. Lloyd, and D. Englund, Nat. Photon. **11**, 447 (2017).

[24] Z. Yan, Y.-R. Zhang, M. Gong, Y. Wu, Y. Zheng, S. Li, C. Wang, F. Liang, J. Lin, Y. Xu, C. Guo, L. Sun, C.-Z. Peng, K. Xia, H. Deng, H. Rong, J. Q. You, F. Nori, H. Fan, X. Zhu, and J.-W. Pan, Science **364**, 753 (2019).

[25] M.-A. Miri and A. Alù, Science **363**, eaar7709 (2019).

[26] J. G. Morley, N. Chancellor, S. Bose, and V. Kendon, Phys. Rev. A **99**, 022339 (2019).

[27] A. Callison, N. Chancellor, F. Mintert, and V. Kendon, New J. Phys. **21**, 123022 (2019).

[28] A. M. Childs, Quantum information processing in continuous time, Ph.D. thesis, Massachusetts Institute of Technology, 2004.

[29] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, Algorithms **12**, 34 (2019).

[30] Z. Wang, N. C. Rubin, J. M. Dominy, and E. G. Rieffel, Phys. Rev. A **101**, 012320 (2020).

[31] X. Qiang, T. Loke, A. Montanaro, K. Aungskunsiri, X. Zhou, J. L. O'Brien, J. B. Wang, and J. C. F. Matthews, Nat. Commun. **7**, 11511 (2016).

[32] S. S. Zhou and J. B. Wang, Roy. Soc. Open Sci. **4**, 160906 (2017).

[33] S. S. Zhou, T. Loke, J. A. Izaac, and J. B. Wang, Quant. Info. Proc. **16**, 82 (2017).

[34] X. Qiang, X. Q. Zhou, J. W. 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, Nat. Photon. **12**, 534 (2018).

[35] A. Y. Kitaev, arXiv:quant-ph/9511026.

[36] H. S. Wilf, Adv. Math. **24**, 281 (1977).

[37] R. Cleve and J. Watrous, in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science* (IEEE, Washington, DC, 2000), pp. 526–536.

[38] S. Bravyi, D. Gosset, and R. König, Science **362**, 308 (2018).

[39] OEIS Foundation Inc., The on-line encyclopedia of integer sequences, http://oeis.org (2019).

[40] M. Lalou, M. A. Tahraoui, and H. Kheddouci, Comput. Sci. Rev. **28**, 92 (2018).

[41] W. Myrvold and F. Ruskey, Inf. Proc. Lett. **79**, 281 (2001).

[42] Z. Kása, Acta Univ. Sapientiae, Inform. **1**, 109 (2009).

[43] N. A. Loehr, in *Bijective Combinatorics, Discrete Mathematics and its Applications* (CRC Press, Boca Raton, FL, 2011), pp. 187–188.

[44] A. Bärtschi and S. Eidenbenz, in *Fundamentals of Computation Theory* (Springer International Publishing, Cham, Germany, 2019), pp. 126–139.

[45] M. Chiew, K. de Lacy, C. H. Yu, S. Marsh, and J. B. Wang, Quant. Info. Proc. **18**, 302 (2019).

[46] D. W. Berry, M. Kieferová, A. Scherer, Y. R. Sanders, G. H. Low, N. Wiebe, C. Gidney, and R. Babbush, npj Quantum Inf. **4**, 22 (2018).