A Realization Scheme for Quantum Multi-Object Search

Zijian Diao[1] Goong Chen[2] Peter Shiue[3]

Abstract

We study the quantum circuit design using 1-bit and 2-bit unitary gates for the iterations of the multi-object quantum search algorithm. The oracle block is designed in order to efficiently implement any sign-flipping operations. A chief ingredient in the design is the permutation operator which maps a set of search targets to another set on which the sign-flipping operation can be easily done. Such a proposed algorithmic approach implicates a minimal symmetric group generation problem: how to generate elements of a symmetric group using the smallest number of concatenations with a set of given generators. For the general case, this is an open problem. We indicate how the complexity issues depend on the solution of this problem through simple examples.

[1]Department of Mathematics, Ohio University-Eastern, St. Clairsville, OH 43950; diao@ohio.edu. Supported in part by an Ohio University Research Challenge Award.
[2]Department of Mathematics, Texas A&M University, College Station, TX 77743-3368; gchen@math.tamu.edu. Supported in part by a DARPA QuIST grant and a TAMU TITF initiative.

[3]Department of Mathematical Science, University of Nevada-Las Vegas, Las Vegas, NV 89154; shiue@unlv.nevada.edu.

1 Introduction

The quantum search algorithm due to L.K. Grover [12] has the advantage of a quadratic speedup over the classical serial search on an unsorted database. Grover's algorithm deals with single-object search. Its quantum circuit design is given in [11]. When there is more than one search target, as what is prevalent in most search problems, algorithms for multi-object search have been studied in [2]–[8].

For multi-object search problems, the number of *items satisfying the search criterion* (i..e, *search targets*) is not known a priori in general. This results in a *quantum counting* problem for which eigenvalue estimates must be made in order to determine the cardinality (see k in (1.1) below) of the search target set; see [5, 6]. No quantum circuit design for the general multi-object search algorithm is yet available, even though some *block diagram* has been suggested in [6].

Let $D = \{w_i \mid i = 1, 2, \ldots, N\}$, where $N = 2^n$, be an unsorted database which is encoded as basis quantum states $\widehat{D} = \{|w_i\rangle \mid i = 1, 2, \ldots, N\}$. Without loss of generality, we assume that the set of search targets is

$$W = \{|w_1\rangle, |w_2\rangle, \ldots, |w_k\rangle\}. \tag{1.1}$$

Elements in W are identified through queries with the (block box) oracle function f:

$$f(w_i) = \begin{cases} 1 & \text{if} \quad 1 \le i \le k, \\ 0 & \text{if} \quad k+1 \le i \le N. \end{cases} \tag{1.2}$$

Recall from [7] that the unitary operator corresponding to the generalized Grover search engine is given by

$$U = -\boldsymbol{I}_s \boldsymbol{I}_f, \tag{1.3}$$

where

$$\boldsymbol{I}_s = \mathbf{1} - 2|s\rangle\langle s|, \quad |s\rangle \equiv \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |w_i\rangle, \tag{1.4}$$

is the "inversion about the average" operator, while

$$\boldsymbol{I}_f = \mathbf{1} - 2 \sum_{i=1}^{k} |w_i\rangle\langle w_i| \tag{1.5}$$

is the "selective sign-flipping" operator, since

$$\boldsymbol{I}_f |w_i\rangle = \begin{cases} -|w_i\rangle, & \text{if} \quad 1 \le i \le k, \\ |w_i\rangle, & \text{if} \quad k+1 \le i \le N. \end{cases} \tag{1.6}$$

The iterations

$$U^j |s\rangle \tag{1.7}$$

are performed and stopped at $j \approx \frac{\pi}{4}\sqrt{\frac{N}{k}}$. A measurement on the quantum system will yield a state in W with large probability.

Note that the oracle function f in (1.2) is in a black box and is not known explicitly. Without a priori knowledge of the search targets, the realization of (1.5) on the quantum computer is utterly non-trivial. For complexity theorists, the use of an oracle function f is a standard practice where f is readily available as a separate computing unit and the complexity involved for the construction and operation of f is entirely ignored. However, in the context of quantum computers, in order to have a complete design which does not depend on any other stand-alone units, and to exploit the entanglement between quantum subsystems, the quantum oracle has to be integrated with other components of the system. In theory, the "standard" way to implement $\boldsymbol{I}_f$ is by the well-known Deutsch's f-c-n "gate"

$$U_f\colon\ |w\rangle|y\rangle \longrightarrow |w\rangle|y \oplus f(w)\rangle \tag{1.8}$$

where $|w\rangle \in \widehat{D}$ and $|y\rangle$, the auxiliary register, is chosen to be $|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, leading to

$$U_f\left(|w\rangle \otimes \left[\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right]\right) = (-1)^{f(w)}|w\rangle \otimes \left[\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right]. \tag{1.9}$$

However, Deutsch's gate (1.8) is not an elementary gate. The action of U_f, a linear operator, is determined by the implicitly *nonlinear* oracle function f. This approach still treats the quantum oracle as a separate module working independently, instead of an integral part of the whole quantum system. Furthermore, unless the computational structure of f is given explicitly, it is highly puzzling to us whether and how it will indeed be possible in the future to realize (1.8) quantum mechanically without the need of using elementary 1-bit and 2-bit unitary gates. As a matter of fact, all current physical implementations of quantum algorithms construct the quantum oracles via *"hard wiring"*, i.e., adapting the layout of the circuit according to the (known) distribution of the function values of f. The main thrust of this paper is to propose a "hard wiring" design to realize (1.8) with elementary gates.

In the quantum circuit design for (1.5) (or, equivalently, for (1.8)), it is totally reasonable to expect that the complexity of the "hard wiring" circuit depends on k in certain way. Therefore, for a single oracle call, there is a clear distinction between its complexity in theoretical discussion, where it is considered to be carried out in one step, and that in the practical implementation, where the *hidden complexity* associated with its construction via elementary gates must be accounted for. At present, our approach proposed here is mostly a *viability study*. The optimal design and its corresponding complexity analysis merit a separate paper, which we hope to present in the sequel.

Return to the multi-object search equations (1.3) and (1.7). In comparison with the quantum circuit design for the single-object search and in view of the commentary in the preceding two paragraphs, we understand that the main difference is in the oracle block $\mathcal{O}$ (cf. [11, Theorem 8]). In the next few sections, we ready ourselves in the redesign of this portion.

2 Circuit Design for the Multi-Object Sign-Flipping Operator

The task of $\boldsymbol{I}_f$ is to selectively flip the signs of the target states. For the single object

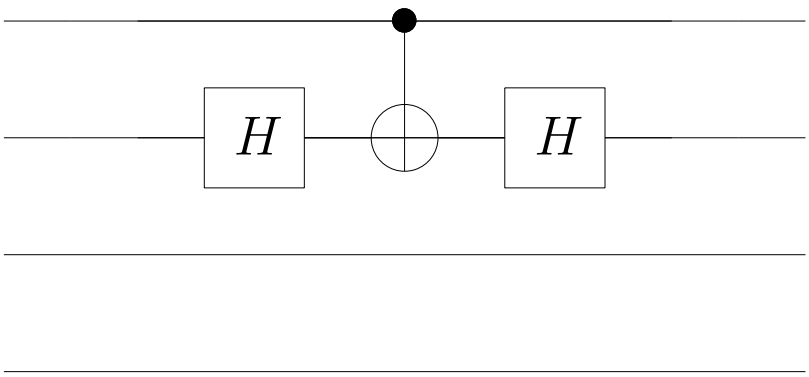

Figure 1: Circuit design of $\boldsymbol{I}_f$ for Example 2.1. H denotes the usual Hadamard transform. The concatenation of the two Hadamard gates and the CNOT gate on the first two qubits maps $|11\rangle$ to $-|11\rangle$, hence the signs of all four states in E with leading qubits $|11\rangle$ are flipped together.

case, we can construct $\boldsymbol{I}_f$ with polynomial complexity using basic 1-qubit and 2-qubit quantum gates [11]. For multi-object case, we may directly concatenate k selective sign-flipping operators of each of the k target states. However, the complexity of this construction is proportional to the number of search targets, which becomes very inefficient when k is large. A better design is to divide the targets into groups and flip the signs of states in each group together.

Example 2.1. Let $n = 4$ and assume the search targets be $E = \{|1100\rangle, |1101\rangle\, |1110\rangle, |1111\rangle\}$. We can flip the signs of all the states in E together, without resorting to four sign-flipping operators tailored to the four targets individually. See Fig. 1 for details. □

We summarize the strategy of our design of I_f first.

1. Partition the set W of search objects into subsets W_i with proper cardinality.
2. Via permutation p_i, map each W_i onto a set E_i of states whose signs are easy to flip together, e.g., E in Example 2.1.
3. Flip the signs of states in W_i through the operations on E_i.

We start by partitioning the set W of target states into $m+1$ sets of W_i's, according to the binary expansion of k, $k = (k_m k_{m-1} \ldots k_2 k_1 k_0)_2$, i.e., $k = k_m 2^m + k_{m-1} 2^{m-1} + \ldots + k_1 2^1 + k_0 2^0$. Note that $m < n$, unless all states are search targets. Each set W_i contains $k_i 2^i$ states, for $i = 0, 1, \ldots, m$. W_i might be empty.

Example 2.2. (i) Let $k = 7$ and $W = \{|w_1\rangle, |w_2\rangle, \ldots, |w_7\rangle\}$. Then W can be partitioned to the following:

$$W = W_2 \dot{\cup} W_1 \dot{\cup} W_0$$

with $W_2 = \{|w_1\rangle, |w_2\rangle, |w_3\rangle, |w_4\rangle\}$, $W_1 = \{|w_5\rangle, |w_6\rangle\}$, and $W_0 = \{|w_7\rangle\}$, where $\dot{\cup}$ denotes disjoint union.

(ii) If $k = 10$ and $W = \{|w_1\rangle, |w_2\rangle, \ldots, |w_{10}\rangle\}$. Then

$$W = W_3 \dot{\cup} W_1,$$

where $W_3 = \{|w_1\rangle, |w_3\rangle, |w_4\rangle, |w_5\rangle, |w_7\rangle, |w_8\rangle, |w_9\rangle, |w_{10}\rangle\}$, $W_1 = \{|w_2\rangle, |w_6\rangle\}$, and $W_2 = W_0 = \emptyset$.

Note that the partition is non-unique. The only thing that matters for now is the cardinality of each set W_i, $i = 0, 1, \ldots, m$. □

We flip the signs of basis states in W by flipping the signs of the states in W_i for $i = 0, 1, 2, \ldots, m$. For each W_i, we construct a circuit block B_i. If W_i is empty, no action is needed. We now delineate the circuit design for a generic block B_i in three steps.

Step 1. Let us denote the basis states in W_i as $W_i = \{|w_{i,1}\rangle, |w_{i,2}\rangle, \ldots, |w_{i,2^i}\rangle\}$. Each $w_{i,j}$ is an n-bit string of 0 and 1's. Define E_i to be the set consisting of all the states whose first $n - i$ bits are all 1's. Clearly, $|E_i| = 2^i = |W_i|$. We construct the quantum circuit P_i which implements the permutation p_i mapping W_i onto E_i. One feasible, albeit inefficient, implementation is to pair up each $w_{i,j}$ with a state in E_i and do 2^i transpositions, as described in Table 1.

$$
\begin{array}{lll}
 & \overbrace{11\cdots1}^{(n-i)\text{ bits}}\overbrace{00\cdots00}^{i\text{ bits}}; & \\
w_{i,1} & \leftrightarrow 11\cdots1\ 00\cdots00; & (w_{i,1} \quad 11\cdots100\cdots00) \\
w_{i,2} & \leftrightarrow 11\cdots1\ 00\cdots01; & (w_{i,2} \quad 11\cdots100\cdots01) \\
w_{i,3} & \leftrightarrow 11\cdots1\ 00\cdots10; & (w_{i,3} \quad 11\cdots100\cdots10) \\
\vdots & & \vdots \\
w_{i,2^i-1} & \leftrightarrow 11\cdots1\ 11\cdots10; & (w_{i,2^i-1}11\cdots111\cdots10) \\
w_{i,2^i} & \leftrightarrow 11\cdots1\ 11\cdots11; & (w_{i,2^i} \quad 11\cdots111\cdots11).
\end{array}
$$

Table 1: The transpositions of the states in W_i with those in E_i. The left column of the table signifies that the two sides of the double arrow "$\leftrightarrow$" are mutually transposed. We use the 2-cycles on the right column to denote the corresponding transpositions on the left column.

Example 2.3. Assume that $n = 7$ and $i = 4$. For $j = 4$, say we have

$$w_{i,j} = w_{4,4} = 0001111.$$

We want to perform the permutation

$$0001111 \leftrightarrow 1110011. \tag{2.1}$$

For ease of discussion, we make the following list:

$$
\begin{array}{lll}
s_1\colon\ 0001111; & s_2\colon\ 0011111; & s_3\colon\ 0111111; \\
s_4\colon\ 1111111; & s_5\colon\ 1110111; & s_6\colon\ 1110011.
\end{array}
$$

Note that each successive pair of symbols s_i and s_{i+1} differs by only one bit.

Then the transposition (2.1) can be achieved through the following sequence of transpositions (cf. the notation used in Table 1):

$$(s_1\ s_2)(s_2\ s_3)(s_3\ s_4)(s_4\ s_5)(s_5\ s_6)(s_4\ s_5)(s_3\ s_4)(s_2\ s_3)(s_1\ s_2). \tag{2.2}$$

Note that through the above permutations, s_1 becomes s_6 and s_6 becomes s_1, achieving (2.1), while $s_2, s_3, \ldots, s_5$ remain unchanged. Several permutations in (2.2) are duplicated. Thus we only need to construct $(s_1 \quad s_2)$, $(s_2 \quad s_3)$, $(s_3 \quad s_4)$, $(s_4 \quad s_5)$ and $(s_5 \quad s_6)$ in order to achieve (2.1).

The circuit design in Fig. 2 realizes the permutation $(s_1\ s_2) = (0001111 \quad 0011111)$. The circuit diagrams for any other $(s_i\ s_{i+1})$ in (2.2) are similar. □

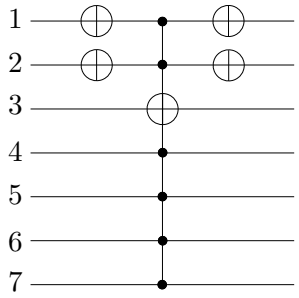

Figure 2: Circuit diagram for the permutation $(s_1 \quad s_2) = (0001111 \quad 0011111)$. Note that the third bit is flipped when and only when the remaining bits are, respectively, 0,0,1,1,1,1, in sequential order.

Step 2. Construct $\mathcal{O}_i$, which flips the signs of any states whose first $n-i$ leading bits are all 1's, i.e., states in E_i. The circuit block is given in Fig. 3.

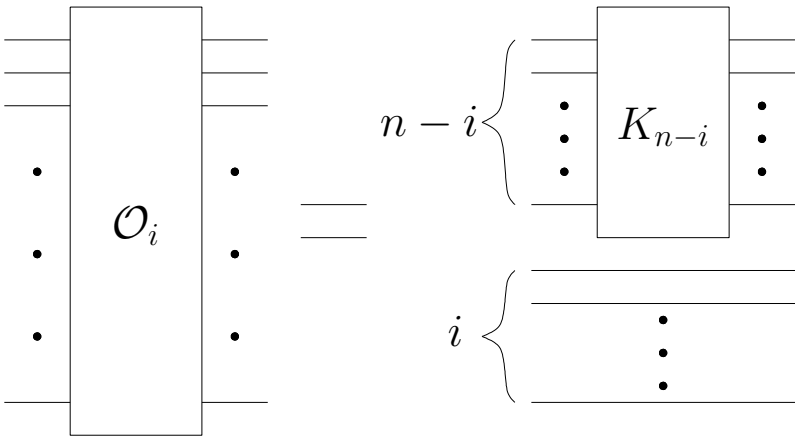

Figure 3: The $\mathcal{O}_i$ block, which flips the signs of any states whose first $n-i$ bits are all 1's. K_{n-i} is the key transformation for the first $n-i$ bits.

Recall from [11] that K_{n-i} is the key transformation on (the first) $n-i$ bits, defined by

$$K_{n-i} = \mathbf{1}_{n-i} - 2|\overbrace{11\cdots1}^{n-i}\rangle\langle 11\cdots 1|,$$

where $\mathbf{1}_{n-i}$ is the identity operator on the first $n-i$ bits. Its construction in terms of elementary gates is given in [11, Fig. 9]. The $\mathcal{O}_i$ block in Fig. 3 thus represents the unitary transformation

$$K_{n-i} \otimes \mathbf{1}_i, \text{ where } \mathbf{1}_i \text{ is the identity operator on the last } i \text{ bits.}$$

Step 3. Piece together $\mathcal{O}_i$, P_i, and P_i^{-1} (the circuit implementing p_i^{-1}, the inverse of p_i), to obtain B_i, for $i = 0, 1, \ldots, m$. See Fig. 4.

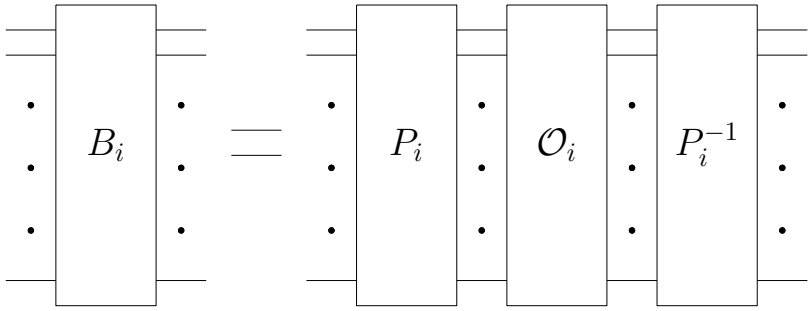

Figure 4: The block $B_i, i = 0, 1, \ldots, m$.

Further, concatenate all the B_i blocks for $i = 0, 1, \ldots, m$ to form the $\mathcal{O}$ (oracle) block. See Fig. 5.

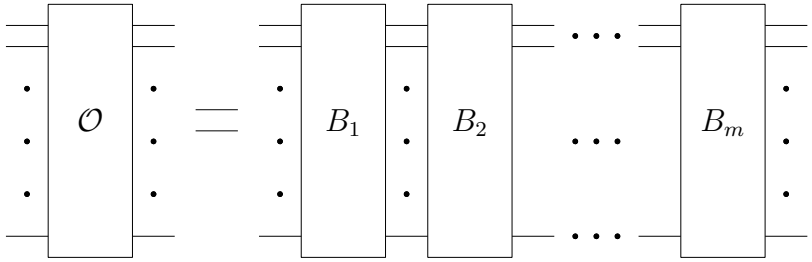

Figure 5: The oracle block $\mathcal{O}$, formed by concatenating $B_0, B_1, \ldots, B_m$.

Example 2.4. Let $n = 2$ and assume the search targets be $W = \{|00\rangle, |01\rangle\}$. Then $k = 2$ and only one block B_1 for $W_1 = \{|00\rangle, |01\rangle\}$ is needed. See Fig. 6 for the circuit design of B_1.

If, as in [11], what we have available are the following elementary gates:

- 1-bit unitary gates

$$U_{\theta,\phi} = \begin{bmatrix} \cos\theta & -ie^{-i\phi}\sin\theta \\ -ie^{-i\phi}\sin\theta & \cos\theta \end{bmatrix}, \quad 0 \le \theta, \phi \le 2\pi. \tag{2.3}$$

- 2-bit quantum phase gates

$$Q_\eta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\eta} \end{bmatrix}, \quad 0 \le \eta \le 2\pi. \tag{2.4}$$

The circuit design for B_1 and, consequently, $\mathcal{O}$, is given in Fig. 7.
Note that P_1 here performs the permutations (00 11)(01 10).

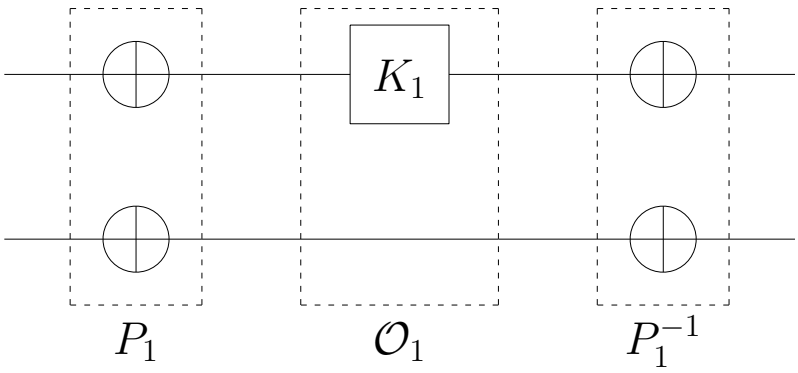

Figure 6: The circuit design of B_1 for Example 2.4.

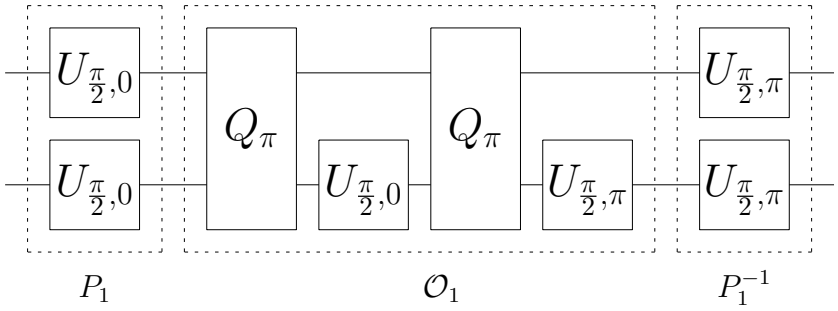

Figure 7: The circuit design of B_1 and $\mathcal{O}$ for Example 2.4, using $U_{\theta,\phi}$ and Q_η as elementary gates.

Theorem 2.1. *Let $U_{\mathcal{O}}$ denote the unitary operator corresponding to the operation performed by $\mathcal{O}$. Then*

$$U_{\mathcal{O}}|w_j\rangle = \begin{cases} (-1)|w_j\rangle, & \text{if } |w_j\rangle \in W, \\ |w_j\rangle, & \text{if } |w_j\rangle \notin W. \end{cases}$$

Proof. If $|w_j\rangle \in W$, then $|w_j\rangle \in W_{i_0}$ for some unique i_0, $i_0 \in \{0, 1, \ldots, m\}$. Therefore

$$B_i|w_j\rangle = \begin{cases} -|w_j\rangle, & \text{if } i = i_0, \\ |w_j\rangle, & \text{if } i \neq i_0. \end{cases}$$

Thus $U_{\mathcal{O}}|w_j\rangle = B_m B_{m-1} \cdots B_1 B_0 |w_j\rangle = -|w_j\rangle$.

If $|w_j\rangle \notin W$, then $B_i|w_j\rangle = |w_j\rangle$ for all $i \in \{0, 1, \ldots, m\}$. Therefore $U_{\mathcal{O}}|w_j\rangle = B_m B_{m-1} \cdots B_1 B_0 |w_j\rangle = |w_j\rangle$. □

Hence $U_{\mathcal{O}}$ indeed corresponds to the sign-flipping operator $\boldsymbol{I}_f$ in (1.8).

Finally, the overall circuit blocks are given in Fig. 8.

3 Additional Discussion

Any universal set of quantum gates [1] can be used to construct the component circuitries required in Section 2. In particular, the 1-bit and 2-bit gates $U_{\theta,\phi}$ and Q_η in (2.3) and (2.4)

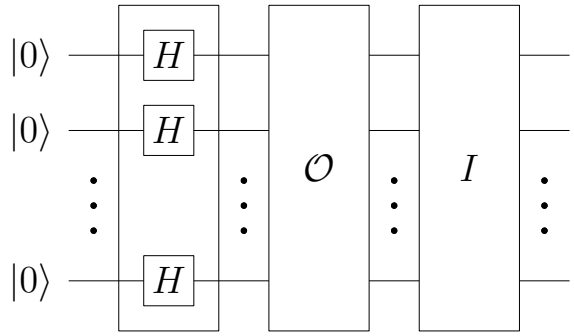

Figure 8: The block diagram for the multi-object search iteration (1.7). The block I performs the inversion about average operation $\boldsymbol{I}_s$, whose circuit design is the same as in [11].

are universal. Therefore, they can be used for the purpose of this paper. See some relevant results in [11].

But there is a special case we need to address. That is, when $n - i = 1$. We need to construct K_1, which is simply the transformation $|0\rangle \to |0\rangle$ and $|1\rangle \to -|1\rangle$. However, this is not directly constructible with the 1-bit gates $U_{\theta,\phi}$, as they are *special unitary*, i.e., all $U_{\theta,\phi}$ have determinant equal to 1.

Two solutions are possible:

1. Use an auxiliary qubit which is set to $|1\rangle$. Bind this auxiliary qubit with the first work qubit with a phase shift gate Q_π. If the leading work qubit is $|1\rangle$, then Q_π maps $|11\rangle$ to $-|11\rangle$. Otherwise, Q_π leaves $|10\rangle$ unchanged. Ignore the auxiliary qubit, the sign of the leading work qubit is flipped. In other words, we have the equivalent network as shown in Fig. 9.

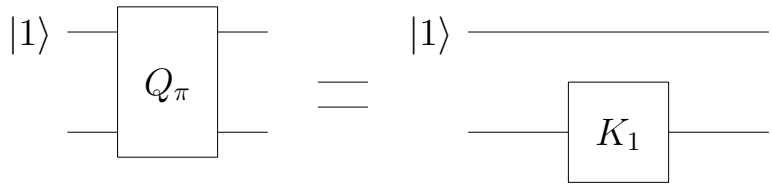

Figure 9: Construction of K_1 using Q_π and an auxiliary qubit.

2. This one is slightly more complicated than the previous one, but no auxiliary qubit is needed. The idea is to use the first qubit to flip the sign of the second qubit, no matter what it is, so that the sign of the overall state is flipped. See the captions and circuits in Fig. 10.

The second solution, in particular, points out one possible realization of the 1-bit phase shift operator

$$\begin{bmatrix} e^{i\phi} & 0 \\ 0 & e^{i\phi} \end{bmatrix}, \tag{3.1}$$

which was not possible using the $U_{\theta,\phi}$ gates alone. The circuit is given in Fig. 11.

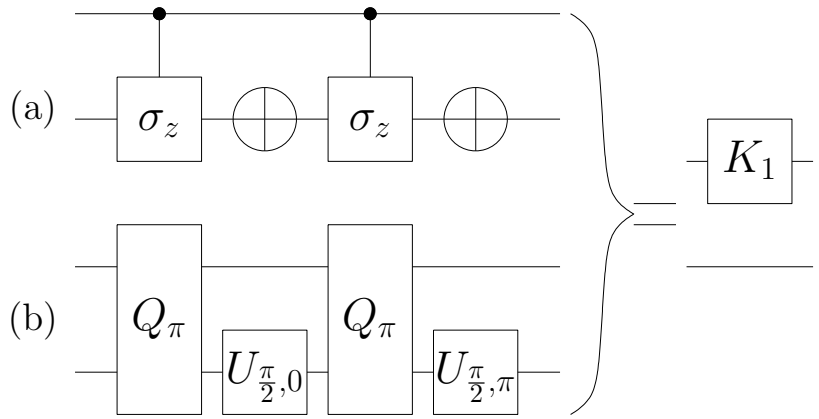

Figure 10: (a) Construction of K_1 without auxiliary qubits, where σ_z is the standard Pauli matrix $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. If the first qubit is $|0\rangle$, then the NOT gate will be applied on the second qubit twice. Hence nothing is changed. If the first qubit is $|1\rangle$, then no matter what the second qubit is, its sign is going to be flipped exactly once. So the function of this circuit is exactly what we expected. (b) The components circuits in (a) are rewritten in terms of U and Q gates.

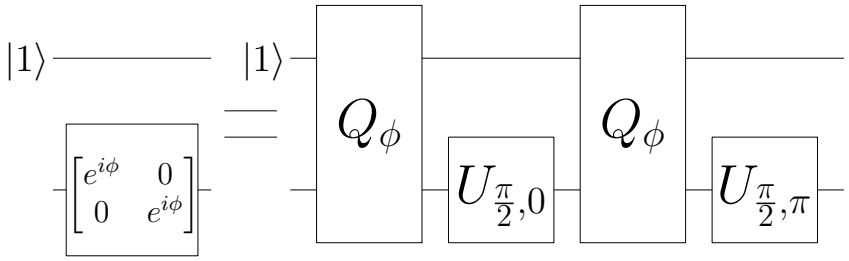

Figure 11: The 1-bit phase gate in equation (3.1) can be realized using the elementary gates $U_{\theta,\phi}$ and Q_η. The first auxiliary qubit is always set to $|1\rangle$. The net operation done on the second qubit is exactly (3.1) by ignoring the auxiliary qubit.

4 Complexity Issues

Following the analysis of the quantum circuit design for the single-object search ([11]), we know that we have linear circuit complexity to construct the I block using elementary 1-bit and 2-bit gates. To be exact, using $U_{\theta,\phi}$ and Q_η gates, the total number of gates needed is $24n - 74$, where n is the number of qubits involved. However, the construction of the $\mathcal{O}$ block for the multi-object case is more complicated than that of the single-object case. Since we have broken up the $\mathcal{O}$ block into $m + 1$ blocks B_0, B_1, ..., B_m, where m can be as large as $n - 1$, and each B_i utilizes a K_{n-i} block, which requires linear complexity itself, we would not expect our design to have linear complexity as in the single object case. Even so, it is still highly desirable if we can achieve the design with as much simplicity as possible. As suggested by the summary of our design in Section 2, there are several flexibilities that we can exploit in order to achieve optimal complexity.

1. The partition of W into W_i's is not unique. We only enforce the cardinality requirement.

2. The choice of permutation $p_i : W_i \to E_i$ is not unique. In fact, there are $2^i!$ of them.

3. The implementation of each permutation p_i or p_i^{-1} in terms of elementary 1-bit and 2-bit gates is not unique.

Taking all these factors into account, we can formulate the optimal design of the block $\mathcal{O}$ as a minimization problem:

$$\min_{\mathbb{P}} \sum_i \min_{p_i : W_i \to E_i} \{c(p_i) + c(p_i^{-1})\}, \tag{4.1}$$

where $\mathbb{P}$ denotes all possible partitions of W into W_i's; $c(p_i)$ and $c(p_i^{-1})$ denote the complexities of p_i and p_i^{-1}, respectively. We have omitted the complexity of $\mathcal{O}_i$, since it stays the same in our design. Because the complexity of the permutations p_i and p_i^{-1} constitutes the basic elements of our complexity analysis, we elaborate on this issue in the following.

The block P_i implements the permutation which maps the 2^i states in W_i onto those in E_i. A "brute-force" way of constructing P_i is to pair up the states in W_i and E_i, implement 2^i transpositions separately following the approach given in Example 2.2, and then concatenate them together. In total, the number of transposition blocks to construct for the overall search circuit will be linear in k, the number of target states. When k is much smaller than N, the total number of items in the database, as in the cases when quantum search algorithm is most powerful, the complexity of the circuit is still quite satisfactory, since each transposition requires only $O(\log^2 N)$ elementary gates ([11]). Nevertheless, unfortunately, when k is large, the complexity of this kind of construction becomes unacceptable.

We should note that, in general, the "brute-force" approach is far from being optimal and there is much room for improvement. For example, in Example 2.4, we did not use this approach to implement the permutation $(00 \quad 11)(01 \quad 10)$. Instead, we use two NOT gates to realize the product of those two transpositions in one step. The resultant circuit is much simpler than the one by concatenation of the two transpositions constructed separately. Let us look at another example.

Example 4.1. Let $n = 4$ and assume the search targets be $W = \{|0000\rangle, |0001\rangle, |0010\rangle, |0011\rangle, |0100\rangle, |0101\rangle, |0110\rangle, |0111\rangle\}$, i.e., all states with leading qubit being 0. Clearly, $W_0 = W_1 = W_2 = \emptyset$ and $W_3 = W$. If we had followed the approach as given above, we would have to construct 8 transpositions, namely, $(0000 \quad 1000)$, $(0001 \quad 1001)$, $(0010 \quad 1010)$, $0011 \quad 1011)$, $(0100 \quad 1100)$, $(0101 \quad 1101)$, $(0110 \quad 1110)$, and $(0111 \quad 1111)$. However, there exists a much more elegant way to implement P_3. All we need to do here is to negate the first qubit and the correctness of this approach is trivial to verify. This cuts down the circuit complexity to a constant for this example. □

We can rephrase our discussion of implementing permutation p_i via elementary gates under the framework of group theory. Now this task is reduced to a special case of the optimal generation of finite symmetric groups using a set of generators, e.g., but not limited to, the set of transpositions:

Problem: Let S_{2^n} be the symmetric group on 2^n elements, and let $G = \{g_1, g_2, \ldots, g_L\}$ be a set of generators for S_{2^n}. Define $c(p)$, the complexity of a permutation $p \in S_{2^n}$, by

$$c(p) = \min_{p=g_{i_1} g_{i_2} \cdots g_{i_l}} l. \tag{4.2}$$

Given p, what is $c(p)$, the minimum number of g_i's (repetition counted) needed to generate p, and what is the best (shortest) generation? □

With our problem in mind, we can encode the 2^n elements by their binary representation, and formulate the operations of the elementary gates by permutations on these elements. We may take the generating set G to be the permutations resulted from any sets of universal gates, in particular, the following fundamental gates:

NOT-gate: it flips the value of one qubit from 0 to 1, and 1 to 0.

Controlled-NOT-gate: it flips the value of a designated qubit depending on other control qubits.

It is well-known that these gates form a universal generating set of S_{2^n}. However, it is not clear what is the most efficient way to generate any given permutation $P \in S_{2^n}$ using the permutations induced by them.

Example 4.2. Let $n = 2$. Consider the minimum generation of S_4 via the following 4 permutations

NOT-gate on bit 1: $N(1) = (00 \quad 01)(10 \quad 11)$,
NOT-gate on bit 2: $N(2) = (00 \quad 10)(01 \quad 11)$,

Controlled-NOT-gate, bit 1 controlling bit2:
$\Lambda_1(2) = (01 \quad 11)$,
Controlled-NOT-gate, bit 2 controlling bit1:
$\Lambda_2(1) = (10 \quad 11)$.

We may generate all the $4! = 24$ permutations with these four permutations. Fig. 12 gives a minimum generation using breadth first search [9, p. 469].

We can see that the depth of this tree is 4. That is, we need at most 4 permutations (elementary gates) to implement any permutation in S_{2^2}. And we can read out the optimal generation by following the branches of the tree.

The analogous group generation problem has been studied in [15]. It is well known that the set $B = \{(1 \quad i) \,|\, 2 \le i \le n\}$ generates S_n. Given $\alpha \in S_n$, let $l_B(\alpha)$ be the complexity of α using generators from B. Let l_B be the largest $l_B(\alpha)$ of all elements in S_n. One can show that $l_B = 4$ in S_4. See Table 2, where the left-hand-side is written in terms of the list notation of a permutation, e.g., $(3, 1, 4, 2)$ stands for the permutation $1 \to 3$, $2 \to 1$, $3 \to 4$, and $4 \to 2$. [15] has shown that l_B is $3m$ in S_{2m+1}. It seems that a formula of l_B in S_{2m} is an interesting open question. □

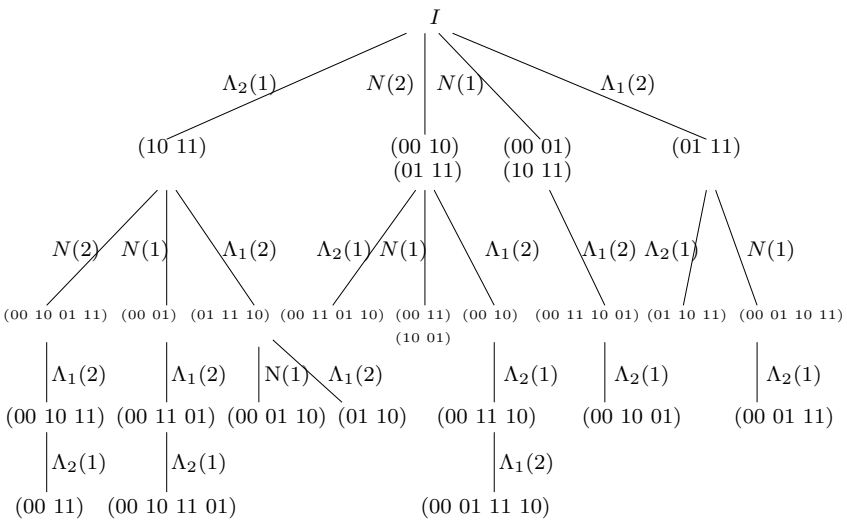

Figure 12: Generation tree of the symmetric group S_4.

Since any computable function can be embedded into a reversible function, which can be viewed as a permutation, we can reformulate the computation of functions by generating permutations. For the complexity issues, we also consider the basic bit operations as in arithmetic complexity theory [13]. The difference is, we have translated all the basic bit operations into permutations in a certain symmetric group, and the complexity is considered in the context of group generation. This kind of symmetric group framework has also been used in the enumeration problems in combinatorics [10, 14]. The difficulty of our problem lies in the fact that symmetric groups are non-abelian. More research is needed in order to understand better ways to do multi-object quantum search.

References

[1] A. Barenco, C.H. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator and H. Weinfurter, Elementary gates for quantum computation, *Phys. Rev.* **A52**, 3457 (1995).

[2] E. Biham, O. Biham, D. Biron, M. Grassl and D.A. Lidar, Grover's quantum search algorithm for an arbitrary initial amplitude distribution, *Phys. Rev.* **A60** (1999), 2742–2745.

[3] O. Biham, E. Biham, D. Biron, M. Grassl and D.A. Lidar, Generalized Grover search algorithm for arbitrary initial amplitude distribution, in *Quantum Computing and Quantum Communications*, Lecture Notes in Comp. Sci., vol. 1509, Springer–Verlag, New York, 1998, pp. 140–147.

[4] M. Boyer, G. Brassard, P. Høyer and A. Tapp, Tight bounds on quantum searching, *Fortsch, Phys.* **46** (1998), 493–506.

[5] G. Brassard, P. Høyer and A. Tapp, Quantum counting, `quant-ph/9805082`, May 1998.

(2, 1, 3, 4)	=	(1 2)
(3, 2, 1, 4)	=	(1 3)
(4, 2, 3, 1)	=	(1 4)
(1, 2, 3, 4)	=	(1 2)(1 2)
(3, 1, 2, 4)	=	(1 3)(1 2)
(4, 1, 3, 2)	=	(1 4)(1 2)
(2, 3, 1, 4)	=	(1 2)(1 3)
(4, 2, 1, 3)	=	(1 4)(1 3)
(2, 4, 3, 1)	=	(1 2)(1 4)
(3, 2, 4, 1)	=	(1 3)(1 4)
(1, 3, 2, 4)	=	(1 2)(1 3)(1 2)
(4, 1, 2, 3)	=	(1 4)(1 3)(1 2)
(1, 4, 3, 2)	=	(1 2)(1 4)(1 2)
(3, 1, 4, 2)	=	(1 3)(1 4)(1 2)
(4, 3, 1, 2)	=	(1 4)(1 2)(1 3)
(2, 4, 1, 3)	=	(1 2)(1 4)(1 3)
(1, 2, 4, 3)	=	(1 3)(1 4)(1 3)
(3, 4, 2, 1)	=	(1 3)(1 2)(1 4)
(2, 3, 4, 1)	=	(1 2)(1 3)(1 4)
(4, 3, 2, 1)	=	(1 4)(1 2)(1 3)(1 2)
(1, 4, 2, 3)	=	(1 2)(1 4)(1 3)(1 2)
(2, 1, 4, 3)	=	(1 3)(1 4)(1 3)(1 2)
(3, 4, 1, 2)	=	(1 3)(1 2)(1 4)(1 2)
(1, 3, 4, 2)	=	(1 2)(1 3)(1 4)(1 2)

Table 2: Generation table of S_4 with 2-cycles (1 2), (1 3), and (1 4).

[6] G. Chen and Z. Diao, Quantum multi-object search algorithm with the availability of partial information, *Z. Naturforsch.* **A56a** (2001), 879–888.

[7] G. Chen, S.A. Fulling and J. Chen, Generalization of Grover's algorithm to multi-object search in quantum computing, Part I: continuous time and discrete time, in Chap. 6 of "*Mathematics of Quantum Computation*", edited by R.K. Brylinski and G. Chen, CRC Press, Boca Raton, Florida, 2002, 135–160.

[8] G. Chen and S. Sun, ibid, Part II: general unitary transformations, in Chap. 7 of the same book, 161–168.

[9] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to algorithms*, MIT Press, 1989.

[10] N.B. De Bruijn, Polya's theory of counting, in *Applied Combinatorial Mathematics*, E.F. Beckenbach ed., John Wiley and Sons, 1964.

[11] Z. Diao, M.S. Zubairy and G. Chen, A quantum circuit design for Grover's algorithm, *Z. Naturforsch.* **A57a** (2002), 701–708.

[12] L.K. Grover, Quantum mechanics helps in searching for a needle in a haystack, *Phys. Rev. Lett* **78** (1997), 325–328.

[13] D. Knuth, *The art of computer programming*, v.2, Addison-Wesley, 1997.

[14] J.H. Kwak and J. Lee, Enumeration of graph covering, surface branched coverings and related group theory, *Combinatorial and Computational Mathematics, present and future*, World Scientific, 2001, 97–161.

[15] I. Pak, Reduced decompositions of permutations in term of star transposition, generalized Catalan numbers, and k-ary trees, *Discrete Math.* **204** (1999), 329–335.