# Bioinspired Quantum Oracle Circuits for Biomolecular Solutions of the Maximum Cut Problem

Weng-Long Chang, Renata Wong, Yu-Hao Chen, Wen-Yu Chung, Ju-Chin Chen, and Athanasios V. Vasilakos

***Abstract*— Given an undirected, unweighted graph with n vertices and m edges, the maximum cut problem is to find a partition of the n vertices into disjoint subsets $V_1$ and $V_2$ such that the number of edges between them is as large as possible. Classically, it is an NP-complete problem, which has potential applications ranging from circuit layout design, statistical physics, computer vision, machine learning and network science to clustering. In this paper, we propose a biomolecular and a quantum algorithm to solve the maximum cut problem for any graph G. The quantum algorithm is inspired by the biomolecular algorithm and has a quadratic speedup over its classical counterparts, where the temporal and spatial complexities are reduced to, respectively, $O(\sqrt{2^n/r})$ and $O(m^2)$. With respect to oracle-related quantum algorithms for NP-complete problems, we identify our algorithm as optimal. Furthermore, to justify the feasibility of the proposed algorithm, we successfully solve a typical maximum cut problem for a graph with three vertices and two edges by carrying out experiments on IBM's quantum simulator.**

***Index Terms*— data structures and algorithms, the maximum cut problem, quantum algorithms, quantum computing, quantum speedup**

## I. Introduction

LET $G = (V, E)$ be an undirected, unweighted graph with a set of vertices $V$ is a set of edges $E$. Further, let $|V| = n$ and let $|E| = m$. A cut $\{V_1, V_2\}$ of $G$ is defined as a partition of vertices into two disjoint subsets $V_1$ and $V_2$. The size of the cut is the number of the edges between $V_1$ and $V_2$.

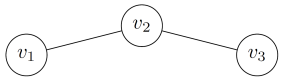


Fig. 1: Example graph.

* Corresponding author: Renata Wong (renata.wong@cgu.edu.tw)

W.-L. Chang, W.-Y. Chung and J.-C. Chen are with the Department of Computer Science and Information Engineering, National Kaohsiung University of Science and Technology, Kaohsiung City, Taiwan, R.O.C.

R. Wong is with the Department of Artificial Intelligence, College of Intelligent Computing, Chang Gung University, Taoyuan, Taiwan. She was with the Physics Division, National Center of Theoretical Sciences, National Taiwan University, Taipei, Taiwan.

Y.-H. Chen is with the Department of Physics, National Taiwan University, Taipei, Taiwan.

A. V. Vasilakos is with the Center for AI Research (CAIR), University of Agder, Grimstad, Norway.

*Example:* Consider an undirected unweighted graph $G$ that contains three vertices $\{v_1, v_2, v_3\}$ and two edges $\{(v_1, v_2), (v_2, v_3)\}$ as shown in Fig. 1. If the cut is $V_1 = \{v_1\}$ and $V_2 = \{v_2, v_3\}$, or $V_1 = \{v_1, v_2\}$ and $V_2 = \{v_3\}$, the size of the cut is 1. If it is $V_1 = \{v_1, v_3\}$ and $V_2 = \{v_2\}$, then it is 2, which is also the maximum cut size for this graph.

In what follows, we assume that $X = \{x_n \cdots x_1 | x_d \in \{0, 1\}, 1 \leq d \leq n\}$ is a set of $2^n$ possible cuts. We further assume that $x_d^0$ indicates that $x_d = 0$, while $x_d^1$ indicates that $x_d = 1$. With this, each element in $X$ is $n$ bits long and represents one of the $2^n$ possible partitions of $n$ vertices into two disjoint subsets $V_1$ and $V_2$. Furthermore, if an $x_d = 1$ in $x_n \cdots x_1 \in X$ then this indicates that the d-th vertex in graph $G$ is in $V_1$. For the same partition, if $x_d = 0$, then this indicates that the d-th vertex is in $V_2$.

The fact that an edge $(x_k, x_p) \in V_1 \times V_2$ can be verified by formula (1). Similarly, the fact that an edge is not shared between $V_1$ and $V_2$ can also be verified by formula (1).

$$f(x_k, x_p) = (x_k \wedge \overline{x_p}) \vee (\overline{x_k} \wedge x_p) \tag{1}$$

where $\wedge$ stands for the logic AND, and $\vee$ stands for the logic OR operation. Formula (1) essentially outputs a 1 if $x_k$ and $x_p$ belong to different sets of vertices, i.e. either $(x_k, x_p) \in V_1 \times V_2$ or $(x_k, x_p) \in V_2 \times V_1$. Formula (1) outputs a 0 if both $x_k$ and $x_p$ belong to the same set of vertices (either both to $V_1$ or both to $V_2$).

## II. Related Works and Motivation

Quantum computing promises to solve certain hard problems more efficiently than classical algorithms. This holds especially for the case when the input size is too large for classical algorithms to process. Some of the best known quantum algorithms that offer such a speedup are quantum integer factorization [4] which runs exponentially faster than any known classical algorithm, and quantum search algorithm [5] that offers a generic square-root speedup over classical algorithms. It has been shown [3] that classical algorithms would require $\Omega(2^n)$ queries where Grover's algorithm only requires $O(2^{n/2})$. Shor's algorithm is problem-specific. On the other hand, Grover's algorithm finds a wide range of applications as a subroutine in quantum algorithms, such as in [6], [7], [9], [11], [13].

While the opposite problem of finding a minimum cut in a graph is efficiently solvable by the Ford–Fulkerson algorithm [10], the maximum cut problem is known to be NP-hard. This means that there are no known polynomial classical algorithms for the problem in general graphs. The max-cut problem in planar graphs can be however solved in polynomial time [12]. As planar graphs constitute only a small subset of the graph family, it is vital to research on finding ways to solve the maximum cut problem efficiently.

There exist classical approximation algorithms, the best of which runs in polynomial time and has an approximation ratio of $\approx 0.878$ [8]. In contrast to that, the algorithm presented in the present work is exact.

### A. Main Contributions and Novelty

In this work, we have devised a biomolecular and a quantum algorithm for the maximum cut problem for arbitrary undirected graphs. We show how certain quantum operations are inspired by biomolecular operations. We improve the system size of the quantum algorithm by a significant factor compared to the algorithm in [1]. Our quantum algorithm has a quadratic speedup over comparable classical algorithms for the problem.

## III. Biomolecular algorithm for the maximum cut problem

In this section, we introduce the biomolecular operations necessary to carry out our biomolecular algorithm for the maximum cut problem. Then, we introduce the algorithm itself.

### A. Biomolecular operations

Biomolecular operations employed in this paper were originally introduced in [2]. Below we present them briefly based on [9] for completeness. In the following definitions it is assumed that experimental lab tubes $X = \{x_n x_{n-1} ... x_1 | 1 \leq d \leq n, x_d \in \{0,1\}\}$:

1) Given a tube $X$ and a strand $x_j$, the operation *Append-Tail* appends $x_j$ onto the end of every element in $X$, and the operation *Append-Head* appends $x_j$ onto the front of every element in $X$. Formally: $Append\_Tail(X, x_j) = \{x_n x_{n-1} ... x_1 x_j\}$. $Append\_Head(X, x_j) = \{x_j x_n x_{n-1} ... x_1\}$. This is achieved by means of denaturation and annealing.
2) Given $m$ tubes $X_1, ..., X_m$, the *Merge* operation unifies their content: $Merge(X_1, ..., X_m) = X_1 \cup ... \cup X_m$. This is achieved by pouring the contents of the tubes into a single tube.
3) Given a tube $X$, the operation $Amplify(X, \{X_i\})$ generates a number of identical copies $X_i$ of $X$ and then discards $X$. This is achieved by polymerase chain reaction.
4) Given a tube $X$ and a strand $x_j$, if $x_j = 1$ then the *Extract* operation creates two new tubes $+(X, X_j^1) = \{x_n ... x_j^1 ... x_1\}$ and $-(X, X_j^1) = \{x_n ... x_j^0 ... x_1\}$. This is achieved by affinity chromatography.
5) Given a tube $X$, the operation $Detect(X)$ returns a True if $X \neq \emptyset$. Otherwise, it returns a False.
6) Given a tube $X$, the bio-molecular operation $Read(X)$ describes any element in $X$. Even if $X$ includes many different elements, this operation can give an explicit description of exactly one of them.

### B. Biomolecular algorithm

In the following, we present a molecular algorithm, Algorithm 1, to solve the maximum cut problem for an undirected unweighted graph $G$ with $n$ vertices and $m$ edges. The first parameter is an empty tube $X_0$ that is regarded as the input tube. Each tube $T, P$ in the algorithm is initially empty and is regarded as an auxiliary storage. Note that in the following code, bits $x_a$ and $x_b$ encode vertices $v_a$ and $v_b$ for an edge $e_j = (v_a, v_b)$ in $G$. Auxiliary bits $s_{j,1}, 1 \leq j \leq m$ store the result of evaluating formula 1. Since there are 4 possible input combinations, $s_{j,1}$ store the corresponding 4 outputs. $s_{j,1}^0$ indicates that the corresponding edge $e_j \in V_1 \times V_2$ or $e_j \in V_2 \times V_1$. Similarly, $s_{j,1}^1$ indicates that the corresponding edge $e_j \notin V_1 \times V_2$ and $e_j \notin V_2 \times V_1$.

**Data:** $X_0$, $n$, $m$
**Result:** a maximum cut
$Append\_Tail(T_1, x_n^1)$;
$Append\_Tail(T_2, x_n^0)$;
$X_0 = Merge(T_1, T_2)$;
**for** $d = n - 1$ *down to* $1$ **do**
  $Amplify(X_0, T_1, T_2)$;
  $Append\_Tail(T_1, x_d^1)$;
  $Append\_Tail(T_2, x_d^0)$;
  $X_0 = Merge(T_1, T_2)$;
**end**
**for** $j = 1$ *to* $m$ **do**
  $P^1 = +(X_0, x_a^1)$ and $P^3 = -(X_0, x_a^1)$;
  $P^2 = +(P^1, x_b^1)$ and $P^4 = -(P^1, x_b^1)$;
  $P^6 = +(P^3, x_b^1)$ and $P^8 = -(P^3, x_b^1)$;
  $Append\_Head(P^8, s_{j,1}^0)$;
  $Append\_Head(P^6, s_{j,1}^1)$;
  $Append\_Head(P^4, s_{j,1}^1)$;
  $Append\_Head(P^2, s_{j,1}^0)$;
  $X_0 = Merge(P^8, P^6, P^4, P^2)$;
**end**
**for** $i = 0$ *to* $m - 1$ **do**
  **for** $j = i$ *down to* $0$ **do**
    $X_{j+1}^{ON} = +(X_{j,s_{i+1,1}}^1)$ and $X_j = -(X_{j,s_{i+1,1}}^1)$;
    $X_{j+1} = Merge(X_{j+1}, X_{j+1}^{ON})$;
  **end**
**end**
**for** $c = m$ *to* $1$ **do**
  **if** $Detect(X_c)$ **then**
    $Read(X_c)$ and terminate algorithm
  **end**
**end**

**Algorithm 1:** Overview of the biomolecular algorithm for the maximum cut problem

Algorithm 1 proceeds as follows. Each execution of steps 1-2 appends the value 1 for $x_n$ as the first bit of every element in

a set $T_1$ and the value 0 as the first bit of every element in a set $T_2$. Hence, $T_1 = \{x_n^1\}$ and $T_2 = \{x_n^0\}$. In step 3, a set union is performed that results in $X_0 = T_1 \cup T_2 = \{x_n^1, x_n^0\}$. After that, the contents of $T_1$ and $T_2$ are discarded, i.e., $T_1 = T_2 = \emptyset$. Next, in each execution of step 5, two ideantical copies $T_1$ and $T_2$ of tube $X_0$ are created and then the content of $X_0$ is discarded, resulting in $X_0 = \emptyset$. In each step 6, the value 1 is appended for $x_d$ onto the end of $x_n \dots x_{d+1}$ for every element in $T_1$. And similarly, each execution of step 7 appends the value 0 for $x_d$ onto the end of $x_n \dots x_{d+1}$ for every element in $T_2$. After that, the two tubes $T_1$ and $T_2$ are merged in step 8 to $X_0 = T_1 \cup T_2$, and $T_1 = T_2 = \emptyset$. Having executed steps 4-9, $X_0 = \{x_n x_{n-1} \dots x_2 x_1 | x_d \in \{0,1\}, 1 \leq d \leq n\}$. This indicates that $2^n$ DNA strands in tube $X_0$ encode $2^n$ cut candidates.

In steps 10-19, we evaluate formula 1 for $j$-th edge $e_j = (v_a, v_b)$. Upon each execution of step 11, tube $P^1$ contains those DNA strands that have $x_a = 1$, while tube $P^3$ contains those DNA strands that have $X_a = 0$. The contents of $X_0$ is discarded. Upon each execution of step 12, tube $P^2$ contains those DNA strands that have $x_a = 1$ and $x_b = 1$, while tube $P^4$ contains those DNA strands that have $X_a = 1$ and $x_b = 0$. Tube $P^1 = \emptyset$. Similarly, upon each execution of step 13, tube $P^6$ contains those DNA strands that have $x_a = 0$ and $x_b = 1$, while tube $P^8$ contains those DNA strands that have $x_a = 0$ and $x_b = 0$. Tube $P_3 = \emptyset$. Next, in steps 14-17 the value 1 is appended for $s_{j,1}$ onto the head of every element in $P^6$ and $P^4$. Similarly, the value 0 is appended for $S_{j,1}$ onto the head of every element in $P^8$ and $P^2$. This indicates that the molecular solutions in tubes $P^4$ and $P^6$ contain those partitions for which the $j$-th edge is either in $V_1 \times V_2$ or in $V_2 \times V_1$. Likewise, the molecular solutions in tubes $P^2$ and $P^8$ contain those partitions for which the $j$-th edge is neither in $V_1 \times V_2$ nor in $V_2 \times V_1$.

Next, step 22 is used to judge the influence of $s_{i+1,1}$ on the number of 1s in tubes $X_{j+1}$ and $X_j$ at iteration $(i, j)$. Upon each execution of this step, tubes $X_{j+1}^{ON}$ and $X_j$ are formed from $X_j$. Therefore, $X_{j+1}^{ON}$ has $s_{i+1,1} = 1$ and $X_j$ has $s_{i+1,1} = 0$. This means that at iteration $(i, j)$ $s_{i+1,1}$ records single 1s in tube $X_{j+1}^{ON}$ and 0s in $X_j$. Next, in step 23, the $Merge$ operation is used to pour the content of tube $X_{j+1}^{ON}$ into tube $X_{j+1}$. This implies that at iteration $(i, j)$, $s_{i+1,1}$ records single 1s in tube $X_{j+1}$. From iteration $(i, j-1)$ through $(m-1, 0)$ similar processing is used to compute the influence of $s_{i+1,1}$ through $s_{m,1}$ on the number of 1s. Therefore, after each operation has been completed, the DNA strands in tube $X_i$ for $0 \leq i \leq m$ have $i$ 1s and contain $i$ edges.

In steps 26-30 molecular solutions representing a maximum-sized cut are read out. If there are DNA strands in tube $X_c$, a "true" is returned. In this case, the solution is read out and the algorithm terminates.

### C. Time and space complexity

The maximum cut problem for any undirected, unweighted graph $G$ with $n$ vertices and $m$ edges can be solved with $O(n + m^2)$ biomolecular operations, $O(2^n)$ DNA strands, $O(m)$ tubes and the longest DNA strand of $O(n + m)$ base pairs. This analysis follows directly from the structure of Algorithm 1.

## IV. Bioinspired Quantum Algorithm for the Maximum Cut Problem

In this section, we present our quantum algorithm that was inspired by the biomolecular algorithm described in Section III.

### A. Deciding to which cut an edge belongs

After completion of steps 10-19 in Algorithm 1, tube $P^2$ contains those DNA strands that have $x_a = x_b = 1$ and $s_{j,1} = 0$, tube $P^4$ contains those DNA strands that have $x_a = 1, x_b = 0$ and $s_{j,1} = 1$, tube $P^6$ includes those DNA strands that have $x_a = 0, x_b = 1$ and $s_{j,1} = 1$ and tube $P^8$ consists of those DNA strands that have $x_a = x_b = 0$ and $s_{j,1} = 0$. Hence, the bioinspired truth table generated from these steps at the same iteration is the same as the truth table for formula 1.

We use auxiliary bits $r_{j,1}$ and $r_{j,2}$, where $1 \leq j \leq m$, to store the result of evaluating the first term $x_k \wedge \overline{x_p}$ and the second term $\overline{x_k} \wedge x_p$ of formula 1. Further auxiliary bits $s_{j,1}, 1 \leq j \leq m$ are used to store the result of evaluating $r_{j,1} \vee r_{j,2}$ in formula 1. As assumed previously, $s_{j,1}^1$ indicates that $j$-th edge $(v_a, v_b)$ is in $V_1 \times V_2$ or in $V_2 \times V_1$, while $s_{j,1}^0$ stands for the fact that $j$-th edge is in $V_1$ or $V_2$. Flowchart in Fig. 2 shows the procedure for determining to which cut an edge belongs step-by-step.

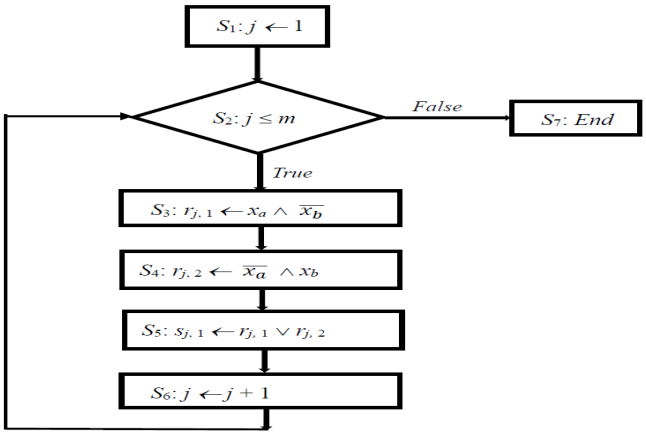


Fig. 2: Flowchart for deciding to which cut an edge belongs.

### B. Computing number of edges in a cut

In order to compute the number of edges in each cut, we introduce auxiliary Boolean variables $z_{i+1,j}$ and $z_{i+1,j+1}$, $1 \leq i \leq m, 0 \leq j \leq i$. All the variables are initialized to 0. $z_{i+1,j+1}$ stores the number of edges in a cut after determining the influence of bits $(x_k, x_p)$ encoding the $(i+1)$-th edge $(v_k, v_p)$ on the number of edges (this corresponds to the number of 1s). Hence, $z_{i+1,j+1} = 1$ indicates that there are $j+1$ edges in the cut. Likewise, $z_{i+1,j}$ stores the number of edges in a cut after determining the influence of bits $(x_k, x_p)$

encoding the $(i+1)$-th edge $(v_k, v_p)$ on the number of edges. $z_{i+1,j} = 1$ indicates that there are $j$ edges in the cut.

In the molecular Algorithm 1, upon each execution of step 22 at iteration $(i = 0, j = 0)$, the *extract* operation forms tubes $X_{j+1}^{ON}$ and $X_j$ from tube $X_j$. This indicates that $X_{j+1}^{ON}$ has $s_{1,1} = 1$ and $X_j$ has $s_{1,1} = 0$. Therefore, $s_{1,1}$ records single 1s in tube $X_{j+1}^{ON}$ and zero 1s in tube $X_j$. Then, upon each execution of step 23 at iteration $(i = 0, j = 0)$, the *merge* operation is used to pour the content of tube $X_1^{ON}$ into tube $X_1$. This implies that at this iteration $s_{1,1}$ records single 1s in $X_1$. Therefore, incrementing the number of 1s in each solution is to satisfy the following bioinspired Boolean formula:

$$s_{1,1} \tag{2}$$

Preserving the number of 1s is to satisfy the following bioinspired Boolean formula

$$\overline{s_{1,1}} \tag{3}$$

Next, the *extract* operation and the *merge* operation at each execution of steps 22-23 at iterations other than $(i = 0, j = 0)$ is to determine the influence of auxiliary bit $s_{i+1}$ on the number of 1s. The biological operations indicate that increasing the number of 1s in a cut corresponds to satisfying the condition that the cut currently has to have $j$ 1s and $s_{i+1} = 1$. The bioinspired Boolean formula for increasing the number of 1s in a cut is

$$s_{i+1,1} \wedge z_{i,j} \tag{4}$$

The biological operations also indicate that preserving the number of 1s in a cut is to satisfy that the cut currently has $j$ 1s and $s_{i+1} = 0$. The bioinspired Boolean formula for preserving the number of 1s in a cut is

$$\overline{s_{i+1,1}} \wedge z_{i,j} \tag{5}$$

A flowchart on the calculation of the number of edges in a cut is given in Fig. 3. Recall that the number of edges corresponds to the number of 1s. Boolean variable $z_{1,1}$ in $S_1$ stores the result of implementing formula (2). If $z_{1,1} = 1$, the number of edges is incremented so that the number of edges in each cut with the first edge $(v_a, v_b)$ is 1. In $S_2$, variable $z_{1,0}$ stores the result of implementing formula (3). If $z_{1,0} = 1$, then the number of edges is preserved so that the number of edges in each cut with two vertices $v_a$ and $v_b$ of the first edge $(v_a, v_b)$ is 0. $S_3$ sets the index of the first loop variable to 1. $S_4$ checks the condition if $i$ is smaller than $m$. If so, $S_5$ is executed, otherwise the procedure of counting the number of edges is terminated. In $S_5$, the index variable $j$ of the second loop is set to the value of $i$. $S_6$ checks if $j \geq 0$. If so, $S_7$ is executed. Otherwise, the next executed instruction is $S_{10}$. In $S_7$, Boolean variable $z_{i,j}$ stores the number of edges in a cut after determining the influence of the $i$-th edge on the number of 1s (edges). $z_{i,j} = 1$ indicates that there are $j$ edges in the cut. Boolean variable $z_{i+1,j+1}$ stores the number of edges in a cut after determining the influence of $s_{i+1,1}$ on the number of edges. $z_{i+1,j+1} = 1$ indicates that there are $j+1$ edges in the cut.

In $S_8$, Boolean variable $z_{i,j}$ stores the number of edges in a cut after determining the influence of the $i$-th edge on the number of edges. $z_{i,j} = 1$ indicates that there are $j$ edges in the cut. Variable $z_{i+1,j}$ stores the number of edges in a cut after determining the influence of $s_{i+1,1}$ on the number of edges. $z_{i+1,j} = 1$ indicates that there are $j$ edges in the cut. $S_9$ decrements the value of the index variable $j$ in the second loop. Execute repeatedly $S_6$ through $S_9$ until $S_6$ results in a False. Then, $S_{10}$ increments the value of the index variable $i$ in the first loop. Loop over $S_4$ through $S_{10}$ until a False in obtained in $S_4$. When this happens, $S_{11}$ is executed and the procedure terminates. The total cost for Fig. 3 is 2 CNOT gates, $m(m+1)$ AND gates and $m(m+1)/2$ NOT gates. This is the cost of counting the number of edges for each cut.

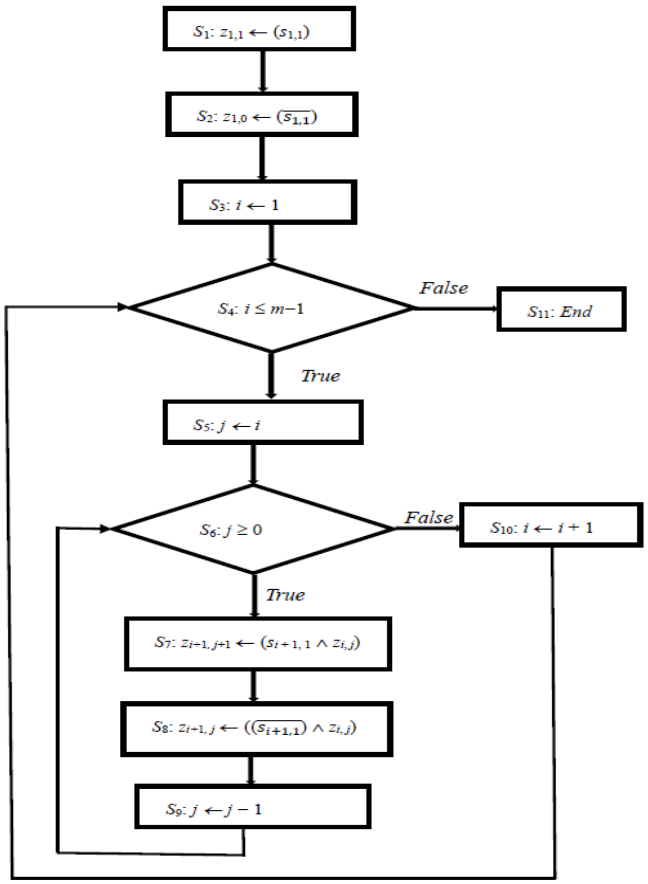


Fig. 3: Flowchart for calculating the number of edges in a cut.

### C. Bioinspired quantum circuits for calculating to which cut an egde belongs

We use auxiliary quantum bits $|r_{j,1}\rangle$ and $|r_{j,2}\rangle$, where $1 \leq j \leq m$ to respectively store the result of evaluating the two disjunctions in equation 1. The initial state of each auxiliary quantum bit $r_{j,k}$ is set to $|0\rangle$. We further use auxiliary quantum bits $|s_{j,1}\rangle$ to respectively store the result of evaluating $|r_{j,1}\rangle \vee |r_{j,2}\rangle$ in equation 1. The initial state of each $|s_{j,1}\rangle$ is set to $|1\rangle$. The quantum circuit in Fig. 4 determines whether an edge belongs to a cut or not.

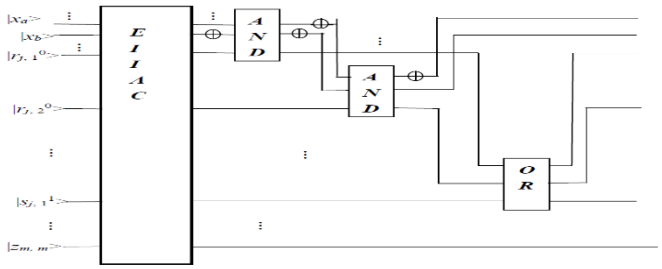


Fig. 4: Quantum circuit EIIAC is used to implement bioinspired Boolean circuits from $S_3$ through $S_5$ in Fig. 2 and to determine if $k$-th edge $(v_a, v_b)$ is in a cut or not.

### D. Bioinspired quantum circuits for computing the number of edges

The bioinspired circuits in instructions $S_1$, $S_2$, $S_7$ and $S_8$ of Fig. 3 for counting the number of 1s in a cut are respectively

$$\begin{aligned} z_{1,1} &\leftarrow s_{1,1} \\ z_{1,0} &\leftarrow \overline{s_{1,1}} \\ z_{i+1,j+1} &\leftarrow s_{i+1,1} \wedge z_{i,j} \\ z_{i+1,j} &\leftarrow \overline{s_{i+1,1}} \wedge z_{i,j} \end{aligned} \tag{6}$$

The outcomes of the operations in (6) are stored in auxiliary qubits $|z_{i+1,j}\rangle$ and $|z_{i+1,i+1}\rangle$, where $0 \leq i \leq m-1, 0 \leq j \leq i$. Each of these qubits is initially prepared in the state $|0\rangle$. We assume that $|z_{i+1,i+1}\rangle$ store the number of edges in a cut after determining the influence of Boolean variable $s_{i+1,1}$ that increases the number of 1s. We also assume that $|z_{i+1,j}\rangle$ store the number of edges in a cut after determining the influence of Boolean variable $s_{i+1,1}$ that preserves the number of 1s.

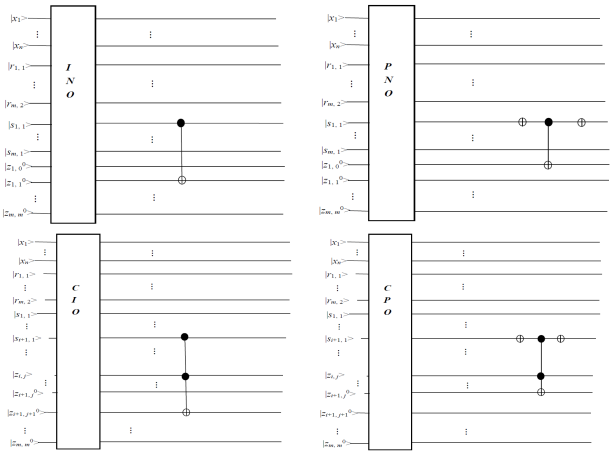


Fig. 5: Top left: Increasing the number of 1s for the influence of $s_{1,1}$ in each cut using the quantum circuit INO. Top right: Preserving the number of 1s for the influence of $\overline{s_{1,1}}$ in each cut using the quantum circuit PNO. Bottom left: Increasing the number of 1s for the influence of $s_{i+1,1}$ in each cut using the quantum circuit CIO. Bottom right: Preserving the number of 1s for the influence of $\overline{s_{i+1,1}}$ in each cut using the quantum circuit CPO.

In the quantum circuit INO in Fig. 5, a CNOT gate with target qubit $|z^0_{1,1}\rangle$ and control qubit $|s_{1,1}\rangle$ implements the formula $z_{1,1} \leftarrow s_{1,1}$ in (6) and copies the value of qubit $|s_{1,1}\rangle$ to $|z_{1,1}\rangle$. Next, in the quantum circuit PNO in Fig. 5, a NOT gate on $|s_{1,1}\rangle$ and a CNOT gate with target qubit $|z^0_{1,0}\rangle$ and control qubit $|\overline{s_{1,1}}\rangle$ implement the formula $z_{1,0} \leftarrow \overline{s_{1,1}}$ in (6) and copy the value of $|\overline{s_{1,1}}\rangle$ to $|z_{1,0}\rangle$. Then, another NOT gate restores $|s_{1,1}\rangle$ to its original state. In the quantum circuit CIO of Fig. 5, a CCNOT gate with target qubit $|z^0_{i+1,j+1}\rangle$ and two control qubits $|z_{i,j}\rangle$ and $|s_{i+1,1}\rangle$ implements the formula $z_{i+1,j+1} \leftarrow s_{i+1,1} \wedge z_{i,j}$ in (6). Next, in the quantum circuit CPO in Fig. 5, a NOT gate on qubit $|s_{i+1,1}\rangle$ and a CCNOT gate with target qubit $|z^0_{i+1,j}\rangle$ and two control qubits $|z_{i,j}\rangle$ and $|\overline{s_{i+1}}\rangle$ implement the last operation in (6). Then, another NOT gate restores qubit $|s_{i+1,1}\rangle$ to its original state.

### E. Putting the algorithm together

The pieces of our quantum algorithm described above need to be put together and combined with Grover's algorithm for amplification of solutions. This is shown by the pseudo-code in Algorithm 2. The input to the algorithm are: number of vertices $n$, number of edges $m$ in graph $G$, and the maximum number of edges among the $2^n$ possible cuts. We note that $R$ in line (24) of the algorithm is the number of maximum cuts. $R$ can be determined by the quantum counting algorithm [3].

In Algorithm 2 the initial state is

$$|\psi_0\rangle = |1\rangle \bigotimes_{i=m}^{1} \bigotimes_{j=i}^{0} |z^0_{i,j}\rangle \bigotimes_{k=m}^{1} |s^1_{k,1}\rangle \bigotimes_{k=m}^{1} \bigotimes_{a=2}^{1} |r^0_{k,a}\rangle \bigotimes_{d=n}^{1} |x^0_d\rangle$$

The first register ($|1\rangle$) is a standard auxilliary register used in Grover's routine for both the oracle and the diffusion operator. We will refer to it as *aux*.

## V. Complexity Assessment

As Grover's algorithm has the complexity of $O(2^{n/2})$, this is also the complexity of the present algorithm.

## VI. Experimental Validation

We have coded [15] and executed our algorithm on IBM Quantum qasm simulator for the example graph given in Fig. 1. The statistical outcome is shown in Fig. 6. The maximum cut $V_1 = \{v_1, v_3\}, V_2 = \{v_2\}$ (or *vice versa*) is measured with a probability greater than 1/2.

The circuit was executed using IBM Quantum Qiskit platform. For the experiment, we used qasm-simulator with 1024 shots.

Fig. 7 shows the circuit used to produce the outcome in Fig. 6. The gates applied before the first occurrentce of EIIAC circuit belong to the initialization step. The circuit is run once only in accordance with the formula $\frac{\pi}{4}\sqrt{\frac{2^n}{R}}$, where $R = 2$ since the maximum cut sets are counted twice, as indicated in Fig. 6. The single run consists of a block of two IEEAC circuits, one for each edge, followed by a CNOT gate that flips the phase of the oracle qubit *aux* for the case where $aux = 1$. After the CNOT gate, all the gates inserted after the initialization step must be uncomputed to free qubits for eventual further runs in the case such runs are specified by

**Data:** quantum system in state $|\psi_0\rangle$
**Result:** a maximum cut
1 Apply Hadamard gates to $|aux\rangle$ and $|x\rangle$ to set the auxiliary register into superposition and to generate the search space over register $|x\rangle$;
2 **for** *edge* $e = 1$ *to* $m$ **do**
3 | Apply EIIAC to $e$
4 **end**
5 **if** $t > 0$ **then**
6 | Apply INO to implement $z_{1,1} \leftarrow s_{1,1}$
7 **end**
8 **if** $t < m$ **then**
9 | Apply PNO to implement $z_{1,0} \leftarrow \overline{s_{1,1}}$
10 **end**
11 **for** $i = 1$ *to* $m - 1$ **do**
12 | **for** $j = i$ *down to* $0$ **do**
13 | | **if** $j + 1 <= t$ *and* $m - i + j = t$ **then**
14 | | | Apply CIO to implement $z_{i+1,j+1} \leftarrow s_{i+1,1} \wedge z_{i,j}$
15 | | **end**
16 | | **if** $j <= t$ *and* $m - i + j - 1 = t$ **then**
17 | | | Apply CPO to implement $z_{i+1,j} \leftarrow \overline{s_{i+1,1}} \wedge z_{i,j}$
18 | | **end**
19 | **end**
20 **end**
21 Apply CNOT on control $|z_{m,t}\rangle$ and target $|aux\rangle$ to label the cuts with the largest number of edges;
22 Reverse the operations from row 20 down to 2 to restore auxiliary qubits to original state;
23 Apply diffusion operator;
24 Repeat rows 2 through 23 at most $\sqrt{\frac{2^n}{R}}$ times. ;
25 Measure to obtain a solution with a probability $\geq \frac{1}{2}$.

**Algorithm 2:** Overview of the quantum algorithm for the maximum cut problem

the formula $\frac{\pi}{4}\sqrt{\frac{2^n}{R}}$. And lastly, we have a diffusion block that amplifies the solution in each iteration/run of the Grover routine.

## VII. Conclusions

In the present paper, we have devised both a biomolecular and a a quantum algorithm for the max-cut problem and have shown how a quantum algorithm can be inspired by biomolecular operations.

The quantum algorithm offers a quadratic speedup over its classical, exact counterparts. We have further successfully executed an instance of the proposed quantum algorithm using IBM's Qiskit SDK [14]. This version of the algorithm is an improvement over the quantum algorithm by Chang et al. [1]. It reduces the number of qubits from $\frac{5m^2+9m+1}{2} + n$ to $m^2 + 5m + n + 1$, where $n$ is the number of vertices, and $m$ is the number of edges.

## Code Availability

The Python/Qiskit code for the proposed algorithm can be obtained from R. Wong's GitHub repository https://github.com/renatawong/quantum-maxcut [15].

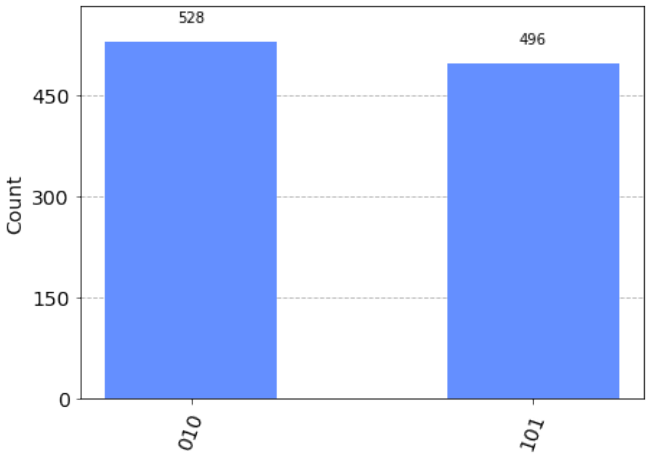


Fig. 6: The maximum cut found for the example graph in Fig.1. Note that both 010 and 101 indicate equivalent maximum cuts. Hence, the maximum cut for Fig.1 is obtained with probability 1.

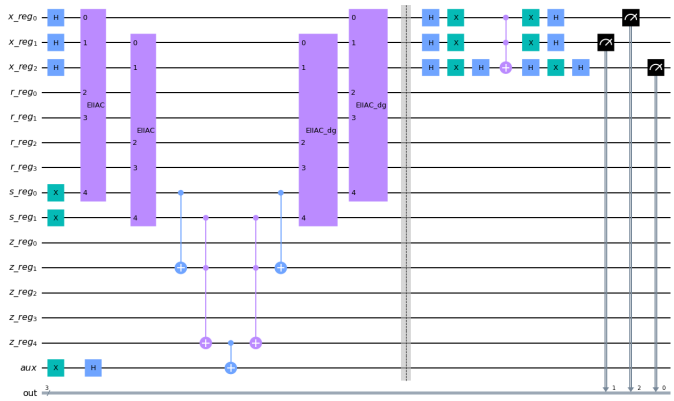


Fig. 7: The circuit for the example graph in Fig.1.

## Acknowledgment

This work was supported by the National Science Foundation of the Republic of China under MOST 105-2221-E-151-040. R. Wong was supported by the National Science and Technology Council, the Ministry of Education (Higher Education Sprout Project NTU-111L104022), and the National Center for Theoretical Sciences of Taiwan. This work is an extended and updated version of the conference paper [1].

## References

[1] W.-L. Chang et al., "Quantum speedup for the maximum cut problem," The 28th Workshop on Compiler Techniques and System Software for High-Performance and Embedded Computing (CTHPC 2023), 2023, DOI: 10.48550/arXiv.2305.16644.
[2] L. Adleman, "Molecular Computation of Solutions to Combinatorial Problems". Science, vol. 266, pp. 1021-1024, 1994.
[3] G. Brassard, P. Hoyer, A. Tapp, "Quantum counting", in: Larsen, K.G., Skyum, S., Winskel, G. (eds) Automata, Languages and Programming. ICALP 1998. Lecture Notes in Computer Science, vol 1443. Springer, Berlin, Heidelberg, DOI: 10.1007/BFb0055105.

[4] P.W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc. Press: 124–134, 1994, DOI:10.1109/sfcs.1994.365700.

[5] L.K. Grover, "A fast quantum mechanical algorithm for database search", Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery: 212–219, 1996, DOI: 10.1145/237814.237866.

[6] W.-L. Chang et al., "Quantum Speedup for Inferring the Value of Each Bit of a Solution State in Unsorted Databases Using a Bio-Molecular Algorithm on IBM Quantum's Computers," IEEE Transactions on NanoBioscience 21(2):286-293, 2022, DOI: 10.1109/TNB.2021.3130811.

[7] W.-L. Chang et al., "Quantum Speedup and Mathematical Solutions from Implementing Bio-molecular Solutions for the Independent Set Problem on IBM's Quantum Computers," IEEE Transactions on NanoBioscience 20(3):354-376, 2021, DOI: 10.1109/TNB.2021.3075733.

[8] M.X. Goemans and D.P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming", Journal of the ACM, 42 (6): 1115–1145, 1995, DOI: 10.1145/227683.227684.

[9] R. Wong, W.-L. Chang, W.-Y. Chung, and A.V. Vasilakos, "Biomolecular and quantum algorithms for the dominating set problem in arbitrary networks," Scientific Reports 13:4205, 2023, DOI: 10.1038/s41598-023-30600-4.

[10] L.R. Ford and D.R. Fulkerson, "Maximal flow through a network", Canadian Journal of Mathematics 8: 399–404, 1956, DOI: 10.4153/CJM-1956-045-5.

[11] R. Wong, and W.-L. Chang, "Quantum speedup for protein structure prediction," IEEE Transactions on NanoBioscience 20(3):323-330, 2021, DOI: 10.1109/TNB.2021.3065051.

[12] F. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time", SIAM J. Comput., 4 (3): 221–225, 1975, DOI: 10.1137/0204019.

[13] R. Wong, and W.-L. Chang, "Fast quantum algorithm for protein structure prediction in hydrophobic-hydrophilic model," Journal of Parallel and Distributed Computing 164:178-190, 2022, DOI: 10.1016/j.jpdc.2022.03.011.

[14] Qiskit contributors, "Qiskit: An Open-source Framework for Quantum Computing," 2023, DOI: 10.5281/zenodo.2573505.

[15] R. Wong, "Quantum maximum cut algorithm", Zenodo, 2023, DOI: 10.5281/zenodo.7790804.