Quantum Speedup and Mathematical Solutions from Implementing Bio-molecular Solutions for the Independent Set Problem on IBM's Quantum Computers

Weng-Long Chang[1]*, Ju-Chin Chen [2]*, Wen-Yu Chung [3]*, Chun-Yuan Hsiao[4]*, Renata Wong[5]* and Athanasios V. Vasilakos[6]*

***Abstract*— In this paper, we propose a bio-molecular algorithm with O(n^2 + *m*) biological operations, O(2^n) DNA strands, O(*n*) tubes and the longest DNA strand, O(*n*), for solving the independent-set problem for any graph G with *m* edges and *n* vertices. Next, we show that a new kind of the straightforward Boolean circuit yielded from the bio-molecular solutions with *m NAND* gates, (*m* + *n* × (*n* +1)) *AND* gates and ((*n* × (*n* + 1)) / 2) *NOT* gates can find the maximal independent-set(s) to the independent-set problem for any graph *G* with *m* edges and *n* vertices. We show that a new kind of the proposed quantum-molecular algorithm can find the maximal independent set(s) with the lower bound Ω($2^{n/2}$) queries and the upper bound O($2^{n/2}$) queries. This work offers an obvious evidence that to solve the independent-set problem in any graph *G* with *m* edges and *n* vertices, bio-molecular computers are able to generate a new kind of the straightforward Boolean circuit such that by means of implementing it quantum computers can give a quadratic speed-up. This work also offers one obvious evidence that quantum computers can significantly accelerate the speed and enhance the scalability of bio-molecular computers. Furthermore, to justify the feasibility of the proposed quantum-molecular algorithm, we successfully solve a typical independent set problem for a graph G with three vertices and two edges by carrying out experiments on the backend ibmqx4 with five quantum bits and the backend simulator with 32 quantum bits on IBM's quantum computer.**

***Index Terms*— data structures and algorithms, independent-set problem, molecular algorithms, molecular computing, quantum algorithms, quantum computing**

This work was submitted on October 19, 2020.

This work was supported by the National Science Foundation of the Republic of China under MOST 105-2221-E-151-040-.

W.-L. Chang is with the Department of Computer Science and Information Engineering, National Kaohsiung University of Science and Technology, No. 415, Jiangong Road, Sanmin District, Kaohsiung City, Taiwan 807-78, Republic of China (e-mail: changwl@cc.kuas.edu.tw).

J.-C. Chen is with the Department of Computer Science and Information Engineering, National Kaohsiung University of Science and Technology, No. 415, Jiangong Road, Sanmin District, Kaohsiung City, Taiwan 807-78, Republic of China (e-mail: jc.chen@nkust.edu.tw).

W.-Y. Chung is with the Department of Computer Science and Information Engineering, National Kaohsiung University of Science and Technology, No. 415, Jiangong Road, Sanmin District, Kaohsiung City, Taiwan 807-78, Republic of China (e-mail: wychung@nkust.edu.tw).

C.-Y. Hsiao is with the Department of Computer Science and Information Engineering, National Kaohsiung University of Science and Technology, No. 415, Jiangong Road, Sanmin District, Kaohsiung City, Taiwan 807-78, Republic of China (e-mail: cyhsiao@nkust.edu.tw).

R. Wong is with the Department of Computer Science and Technology, Nanjing University, 163 Xianlin Road, 210023 Nanjing, Jiangsu, China (e-mail: renata.wong@protonmail.com).

A. V. Vasilakos is with the School of Electrical and Data Engineering, University of Technology, Sydney, Australia, with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China, and with the Department of Computer Science, Electrical and Space Engineering, Lulea University of Technology, Lulea 97187, Sweden (e-mail:th.vasilakos@gmail.com).

I. Molecular Algorithm for Solving the Independent Set Problem

A. Definition of the Independent Set Problem

Let G be a graph and $G = (V, E)$, where V is a set of vertices and E is a set of edges in G. An *independent set* of graph G is a subset $V^1 \subseteq V$ of vertices such that for all $v_a, v_b \in V^1$, the edge (v_a, v_b) is *not* in E [1, 2].

Definition 1-1: The independent set problem of graph G with n vertices and m edges is to find a maximum-sized independent set in G.

B. Biomolecular Operations

Definition 1-2: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \le n\}$ and a bit x_j, the bio-molecular operation "Append-Head" appends x_j onto the head of every element in set X. Formally, Append-Head$(X, x_j) = \{x_j x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \le n$ and $x_j \in \{0, 1\}\}$.

Definition 1-3: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \le n\}$ and a bit x_j, the bio-molecular operation "Append-Tail" appends x_j onto the end of every element in set X. Formally, Append-Tail$(X, x_j) = \{x_n x_{n-1} \ldots x_2 x_1 x_j | \forall x_d \in \{0, 1\}$ for $1 \le d \le n$ and $x_j \in \{0, 1\}\}$.

Definition 1-4: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \le n\}$, the bio-molecular operation "Discard(X)" resets X to an empty set and can be represented as "$X = \varnothing$".

Definition 1-5: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$

for $1 \le d \le n\}$, the bio-molecular operation "Amplify(X, $\{X_i\}$)" creates a number of identical copies X_i of set X, and then discards X with the help of "Discard(X)".

Definition 1-6: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \le n\}$ and a bit x_j, the bio-molecular *extract* operation has two kinds of representation. The first representation is $+(X, x_j^1) = \{x_n x_{n-1} \ldots x_j^1 \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \ne j \le n\}$ and $-(X, x_j^1) = \{x_n x_{n-1} \ldots x_j^0 \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \ne j \le n\}$ if the value of x_j is equal to one. The second representation is $+(X, x_j^0) = \{x_n x_{n-1} \ldots x_j^0 \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \ne j \le n\}$and $-(X, x_j^0) = \{x_n x_{n-1} \ldots x_j^1 \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \ne j \le n\}$ if the value of x_j is equal to zero.

Definition 1-7: Given m sets $X_1 \ldots X_m$, the bio-molecular *merge* operation is $\cup(X_1, \ldots, X_m) = X_1 \cup \ldots \cup X_m$.

Definition 1-8: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \le n\}$, the bio-molecular operation "Detect(X)" returns *true* if X is not an empty tube. Otherwise, it returns *false*.

Definition 1-9: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0, 1\}$ for $1 \le d \le n\}$, the bio-molecular operation "Read(X)" describes any element in X. If X contains many different elements, this operation gives an explicit description of exactly one of them.

C. Molecular Algorithm for Solving the Independent Set Problem

From **Def. 1-1** we have that for any graph G with n vertices and m edges, all possible independent sets are the 2^n possible choices. Each possible choice corresponds to a subset of vertices in G. Therefore, it is assumed that Y is a set of 2^n possible choices, i.e., $\{y_n y_{n-1} \ldots y_2 y_1 | \forall y_d \in \{0, 1\}$ for $1 \le d \le n\}$. For the sake of presentation, we assume that y_d^0 indicates that the value of y_d is zero and y_d^1 indicates that the value of y_d is one. We propose the following molecular algorithm to solve the independent set problem for arbitrary graph G. Parameter Y_0 is an empty tube (set); n represents the number of vertices while m represents the number of edges. Each tube in the **Procedure Solve-independent-set-problem** is empty and regarded as an auxiliary storage.

Procedure Solve-independent-set-problem(Y_0, n, m)
(0a) Append-Tail(X_1, y_n^1). (0b) Append-Tail(X_2, y_n^0).
(0c) $Y_0 = \cup(X_1, X_2)$.
(1) **For** $d = n - 1$ **downto** 1
  (1a) Amplify(Y_0, X_1, X_2). (1b) Append-Tail(X_1, y_d^1).
  (1c) Append-Tail(X_2, y_d^0). (1d) $Y_0 = \cup(X_1, X_2)$.
End For
(2) **For** each edge, $e_k = (v_i, v_j)$, in G where $1 \le k \le m$ and bits y_i and y_j respectively represent vertices v_i and v_j.
  (2a) $P^1 = +(Y_0, y_i^1)$ and $P^3 = -(Y_0, y_i^1)$.
  (2b) $P^2 = +(P^1, y_j^1)$ and $P^4 = -(P^1, y_j^1)$.
  (2c) $Y_0 = \cup(P^3, P^4)$. (2d) Discard(P^2).
End For
(3) **For** $i = 0$ **to** n-1
(4) **For** $j = i$ **down to** 0
  (4a) $Y_{j+1}^{ON} = +(Y_j, y_{i+1}^1)$ and $Y_j = -(Y_j, y_{i+1}^1)$.
  (4b) $Y_{j+1} = \cup(Y_{j+1}, Y_{j+1}^{ON})$.
End For
End For
(5) **For** $c = n$ **down to** 1
  (5a) **If** (detect(Y_c)) **then**
    (5b) Read(Y_c) and terminate the algorithm.
  EndIf
EndFor
EndProcedure

Lemma 1-1: The independent set problem for a graph G with m edges and n vertices can be solved by the molecular algorithm **Solve-independent-set-problem**(Y_0, n, m).

Proof: Each execution of Steps (0a) and (0b), respectively, appends the value "1" for y_n as the first bit of every element in a set X_1 and the value "0" for y_n as the first bit of every element in a set X_2. Next, each execution of Step (0c) creates the set union for the two sets $X_1 = \{y_n^1\}$ and $X_2 = \{y_n^0\}$ so that $Y_0 = X_1 \cup X_2 = \{y_n^1, y_n^0\}$, $X_1 = \varnothing$ and $X_2 = \varnothing$.

Next, each execution of Step (1a) creates two identical copies, X_1 and X_2, of set Y_0, and $Y_0 = \varnothing$. Each execution of Step (1b) then appends the value "1" for y_d onto the end of $y_n \ldots y_{d+1}$ for every element in X_1. Similarly, each execution of Step (1c) also appends the value "0" for y_d onto the end of $y_n \ldots y_{d+1}$ for every element in X_2. Next, each execution of Step (1d) creates the set union for the two sets X_1 and X_2 so that $Y_0 = X_1 \cup X_2$, and $X_1 = \varnothing$ and $X_2 = \varnothing$. After repeatedly executing Steps (1a) through (1d), $Y_0 = \{y_n y_{n-1} \ldots y_2 y_1 | \forall y_d \in \{0, 1\}$ for $1 \le d \le n\}$ consists of 2^n DNA strands that encode 2^n possible choices.

Next, Step (2) is a loop that evaluates each formula of the form $\left(\overline{y_i \wedge y_j}\right)$ for the kth edge in G where $1 \le k \le m$. On each execution of Step (2a), DNA strands in tube P^1 have $y_i = 1$, DNA strands in tube P^3 have $y_i = 0$, and tube $Y_0 = \varnothing$. Next, on each execution of Step (2b), DNA strands in tube P^2 have $y_i = 1$ and $y_j = 1$, DNA strands in tube P^4 have $y_i = 1$ and $y_j = 0$, and tube $P^1 = \varnothing$. This indicates that molecular solutions in tube P^2 contain two vertices in the kth edge and are *illegal* independent sets; molecular solutions in tube P^4 only contain one vertex in the kth edge and are *legal* independent sets; and molecular solutions in tube P^3 contain one vertex or no vertices in the kth edge and are *legal* independent sets. Then, on each execution of Step (2c), tube Y_0 contains those DNA strands that encode legal independent sets, tube $P^3 = \varnothing$, and tube $P^4 = \varnothing$. Next, on each execution of Step (2d), *illegal* independent sets encoded by DNA strands in tube P^2 are discarded. After repeatedly executing Steps (2a) through (2d), tube Y_0 consists of those DNA strands that satisfy $\wedge_{k=1}^{m} \left(\overline{y_i \wedge y_j}\right)$ that is the true value for the kth edge in G for $1 \le k \le m$.

Each execution of Step (4a) at the iteration (i, j) is applied to compute the influence of y_{i+1} on the number of ones in tubes (sets) Y_{j+1} and Y_j. This is to say that tube (set) Y_{j+1}^{ON} has $y_{i+1} = 1$ and tube (set) Y_j has $y_{i+1} = 0$. This indicates that at the iteration (i, j) in the two-level nested loop, the influence of y_{i+1} on the number of ones is to record single ones in tube (set) Y_{j+1}^{ON} and also to record zero ones in tube (set) Y_j. Next, upon each execution of Step (4b) at the iteration (i, j), the *merge* operation is applied to pour the content of tube (set) Y_{j+1}^{ON} into tube (set) Y_{j+1}. This indicates that at the iteration (i, j), the influence of y_{i+1} on the number of ones is to record single ones in tube (set) Y_{j+1}. Next, from the iteration $(i, j-1)$ through the iteration $(n-1, 0)$, similar processing is applied to compute the influence of y_{i+1} through y_n on the number of ones. Hence, after each operation is completed, those DNA strands in tube Y_i for $0 \le i \le n$ have i ones that contain *i vertices*. Next, on each execution of Step (5a), if there are DNA strands in tube Y_c, a "true" is returned. Next, on each execution of Step (5b), the answer of a maximum-sized independent set is read and the algorithm terminates. ■

D. Time and Space Complexity of Molecular Algorithm for Solving the Independent Set Problem

The following lemma describes the time complexity, the volume complexity of solution space, the number of tubes used and the longest library strand in solution space for the algorithm **Solve-independent-set-problem**(Y_0, n, m).

Lemma 1-2. The independent set problem for any graph G with n vertices and m edges can be solved with $O(n^2 + m) = O(n^2)$ biological operations, $O(2^n)$ DNA strands, $O(n)$ tubes and the longest DNA strand, $O(n)$.

Proof: The above numbers follow directly from analysis of the algorithm **Solve-independent-set-problem**(Y_0, n, m).

E. The Straightforward Boolean Circuit for Determining Independent Sets from Bio-molecular Solutions

After completing Steps (0a) through (1d) in the algorithm **Solve-independent-set-problem**(Y_0, n, m), the 2^n DNA strands in tube Y_0 encode the possible choices. Bits y_i and y_j are its two inputs, and bit l_k for $1 \leq k \leq m$ is its output. If the value of bit l_k for $1 \leq k \leq m$ is equal to 1, then the corresponding subsets of vertices only contain one vertex or zero vertices in the kth edge (v_i, v_j) and are legal independent sets. Otherwise, the corresponding subsets of vertices contain two vertices in the kth edge (v_i, v_j) and are illegal independent sets. Therefore, after repeatedly executing Steps (2a) through (2d) from iteration one through iteration m, bio-molecular solutions in tube Y_0 contain one or no vertices in each edge and do not contain two vertices of any one edge. This is to say that bio-molecular solutions in tube Y_0 encode those subsets of vertices in which for all vertices v_i and v_j, the edge (v_i, v_j) is *not* in E which is the set of edges in graph G. This also implies that bio-molecular solutions in tube Y_0 satisfy the fact that each **NAND** operation of two inputs y_i and y_j has a true value. Therefore, the straightforward Boolean circuit generated from Steps (2a) through (2d) at all m iterations implements the Boolean formula ($\wedge_{k=1}^{m} (\overline{y_i \wedge y_j})$) and finds which subsets of vertices satisfy this formula.

Fig. 1-1 shows a flowchart for recognizing independent sets of the independent-set problem for a graph G with n vertices and m edges. In statement S_1, variable k is set to one (1) and o_0 is set to one (1). Next, in statement S_2, if the value of k is less than or equal to the value of m, then *next executed* instruction is statement S_3. Otherwise, in statement S_6, an *End* instruction is executed to terminate the task of recognizing independent sets.

In statement S_3, a ***NAND*** gate "$l_k \leftarrow \overline{y_i \wedge y_j}$" is implemented. Bits (Boolean variables) y_i and y_j respectively encode vertex v_i and vertex v_j that are connected by the kth edge in a graph G with n vertices and m edges. Bit (Boolean variable) l_k with $1 \leq k \leq m$ stores the result of implementing ($\overline{y_i \wedge y_j}$) (the kth ***NAND*** gate). Next, in statement S_4, a logical **AND** operation "$o_k \leftarrow l_k \wedge o_{k-1}$" is executed that is the *kth* clause in ($\wedge_{k=1}^{m}(\overline{y_i \wedge y_j})$). Bit (Boolean variable) l_k stores the result of implementing the kth ***NAND*** gate and is the first operand of the logical **AND** operation. Bit (Boolean variable) o_{k-1} with $1 \leq k \leq m$ is the second operand of the logical **AND** operation and stores the result of the previous logical **AND** operation. Bit (Boolean variable) o_k with $1 \leq k \leq m$ stores the result of implementing $l_k \wedge o_{k-1}$ (the kth clause that is the kth ***AND*** gate). Next, in statement S_5, variable k is incremented.

Repeat to execute statements S_2 through S_5 until in statement S_2 the conditional judgement returns a *false* value. From Figure 1-1 it follows that the total number of ***NAND*** gates is m. The total number of logical **AND** operations corresponds to m ***AND*** gates. Therefore, the cost of recognizing independent set(s) corresponds to m ***NAND*** gates and m ***AND*** gates. We use **Lemma 1-3** to show that the straightforward Boolean circuit in Fig. 1-1 for recognizing independent sets of the independent set problem for a graph G is the *best* Boolean circuit *known* for the problem.

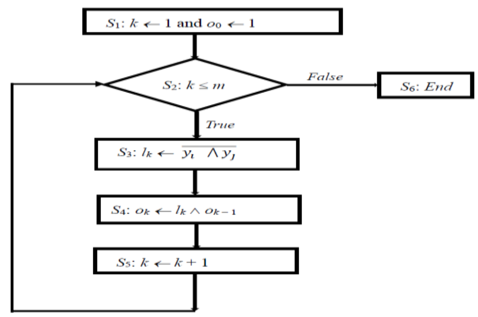

Fig. 1-1: Recognizing independent-sets of the independent-set problem for a graph G with n vertices and m edges.

Lemma 1-3: For the independent-set problem for any graph G with n vertices and m edges, in Fig. 1-1, the Boolean circuit with m ***NAND*** gates and m ***AND*** gates generated from Step (2a) through (2d) at all m iterations in the molecular algorithm **Solve-independent-set-problem**(Y_0, n, m) is the best Boolean circuit known for recognizing independent-set(s) among 2^n possible choices.

Proof: Please refer to the content of this subsection. ■

F. The Straightforward Boolean Circuit for Computing the Number of Vertices in Independent Sets from Bio-molecular Solutions

For computing the number of vertices, auxiliary Boolean variables $w_{i+1, j}$ and $w_{i+1, j+1}$ with $0 \leq i \leq n - 1$ and $0 \leq j \leq i$ are set to the initial value 0 (zero). Boolean variable $w_{i+1, j+1}$ stores the number of vertex in a solution after figuring out the influence of Boolean variable y_{i+1} that encodes the (i + 1)th vertex in the number of ones (vertices). If the value of Boolean variable $w_{i+1, j+1}$ is equal to 1 (one), then this indicates that there are (j + 1) ones (vertices) in the solution. Boolean variable $w_{i+1, j}$ stores the number of vertex in a solution after figuring out the influence of Boolean variable y_{i+1} that encodes the (i + 1)th vertex on the number of ones (vertices). If the value of Boolean variable $w_{i+1, j}$ is equal to 1 (one), then this indicates that there are j ones (vertices) in the solution.

In a solution (an independent-set) that has the value of bit o_m equal one, bit y_1 encodes the first vertex v_1. If the value of bit y_1 is equal to one (1), then the first vertex v_1 appears in the solution and it increments the number of vertices (the number of ones) for the solution. Otherwise, the first vertex v_1 does not appear in the solution and it preserves the number of vertices (the number of ones) for the solution. In the molecular algorithm

Solve-independent-set-problem(Y_0, n, m), on the execution of Step (4a) in the iteration ($i = 0$, $j = 0$), the DNA strands in tube Y_1^{ON} have $y_1 = 1$ and contain vertex v_1 and the DNA strands in tube Y_0 have $y_1 = 0$ and do not contain vertex v_1. This is to say that the influence of y_1 (the influence of vertex v_1) on the number of ones (the number of vertices) is recorded as single ones in tube Y_1^{ON} and to record zero ones in tube Y_0. Next, on the execution of Step (4b) in the same iteration ($i = 0$, $j = 0$), the influence of y_1 on the number of ones is recorded as single ones in tube (set) Y_1. Therefore, for the influence of the first vertex v_1, incrementing the number of vertices in each solution is to satisfy the formula ($o_m \wedge y_1$) and preserving the number of vertices is to satisfy the formula ($o_m \wedge \overline{y_1}$).

Similarly, the influence of the (i + 1)th vertex with $1 \le i \le n - 1$ is to decide whether in each solution the number of vertices (the number of ones) is incremented or is preserved. In order to increment the number of vertices (the number of ones) in each solution two conditions must be satisfied. The *first* condition is that the (i + 1)th vertex is within the solution and the *second* condition is that each solution currently has j vertices. In order to preserve the number of vertices (the number of ones) in each solution two conditions must be satisfied. The *first* condition is that the (i + 1)th vertex is not within the solution and the *second* condition is that each solution currently also has j vertices. Next, on each execution of Step (4a) in the iteration (i, j), the DNA strands in tube Y_{j+1}^{ON} encode each solution that has $y_{i+1} = 1$ and contains vertex v_{i+1}. The DNA strands in tube Y_j on the other hand encode each solution that has $y_{i+1} = 0$ and does not contain vertex v_{i+1}. This indicates that in the iteration (i, j), the influence of y_{i+1} on the number of ones (the number of vertices) is recorded as (j + 1) ones in tube Y_{j+1}^{ON} and also as j ones in tube Y_j. Next, on each execution of Step (4b) in the iteration (i, j), the influence of y_{i+1} on the number of ones (the number of vertices) is recorded as having (j + 1) ones in tube Y_{j+1}. Therefore, for the influence of the (i + 1)th vertex for $1 \le i \le n - 1$ in each solution, the two conditions for incrementing the number of vertices (the number of ones) in each solution are to satisfy the Boolean formula ($y_{i+1} \wedge w_{i,j}$). The two conditions for preserving the number of vertices in each solution are to satisfy the Boolean formula (($\overline{y_{i+1}}$) $\wedge$ $w_{i,j}$).

Fig. 1-2 shows the logical flowchart for counting the number of vertices in each solution. In statement S_1, a logical **AND** operation "$w_{1,1} \leftarrow o_m \wedge y_1$" is implemented. Boolean variable $w_{1,1}$ stores the result of implementing one ***AND*** gate ($o_m \wedge y_1$). Next, in statement S_2, a logical **AND** operation "$w_{1,0} \leftarrow o_m \wedge \overline{y_1}$" is implemented. Boolean variable $w_{1,0}$ stores the result of implementing one ***AND*** gate ($o_m \wedge \overline{y_1}$).

Next, in statement S_3, variable i is set to one. Then, in statement S_4, if the value of i is less than or equal to the value of ($n - 1$), then *next executed* instruction is statement S_5. Otherwise, in statement S_{11}, an *End* instruction is executed to terminate the task of counting the number of vertices in each solution. In statement S_5, variable j is set to the value of the variable i. Next, in statement S_6, if the value of j is greater than or equal to zero, then the next executed instruction is statement S_7. Otherwise, the next executed instruction is statement S_{10}.

In statement S_7, a logical **AND** operation "$w_{i+1,j+1} \leftarrow y_{i+1} \wedge w_{i,j}$" is implemented. Boolean variable $w_{i,j}$ stores the number of vertex in a solution after determining the influence of Boolean variable y_i that encodes the ith vertex on the number of ones (vertices). Boolean variable $w_{i+1,j+1}$ stores the result of implementing the logical **AND** operation "$w_{i+1,j+1} \leftarrow y_{i+1} \wedge w_{i,j}$". This is to say that $w_{i+1,j+1}$ stores the number of vertex in a solution after determining the influence of Boolean variable y_{i+1} that encodes the (i + 1)th vertex on the number of ones (vertices).

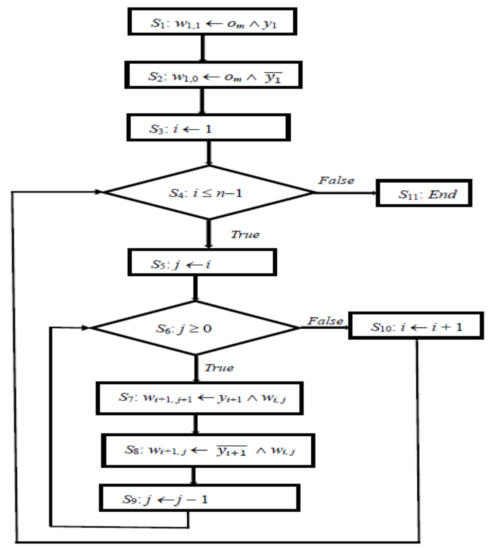

Fig. 1-2: Flowchart for computing the number of vertices in each solution (independent set).

Next, in statement S_8, a logical **AND** operation "$w_{i+1,j} \leftarrow \overline{y_{i+1}} \wedge w_{i,j}$" is implemented. For Boolean variable y_{i+1}, its negation $\overline{y_{i+1}}$ is the first operand of the logical **AND** operation. Boolean variable $w_{i+1,j}$ stores the result of implementing the logical **AND** operation "$w_{i+1,j} \leftarrow \overline{y_{i+1}} \wedge w_{i,j}$". This indicates that $w_{i+1,j}$ stores the number of vertex in a solution after determining the influence of Boolean variable y_{i+1} that encodes the (i + 1)th vertex on the number of ones (vertices).

Next, in statement S_9, variable j is decremented. Repeat to execute statement S_6 through statement S_9 until in statement S_6 the conditional judgement attains a *false* value. Next, in statement S_{10}, variable i is incremented. Repeat to execute statements S_4 through S_{10} until in S_4 the conditional judgement attains a *false* value. When this happens, the next executed statement is S_{11}. In S_{11}, an *End* instruction is executed to terminate the task of counting the number of vertices in each solution. The cost of each operation in Fig. 1-2 is ($n \times (n + 1)$) ***AND*** gates and ($\frac{n\times(n+1)}{2}$) ***NOT*** gates. Therefore, the cost of counting the number of vertices for each solution is to implement ($n \times (n + 1)$) ***AND*** gates and ($\frac{n\times(n+1)}{2}$) ***NOT*** gates. We use **Lemma 1-4** to show that in Fig. 1-2 the straightforward Boolean circuit for counting the number of vertices in each solution is the *best* Boolean circuit *known* for the problem.

Lemma 1-4: In Fig. 1-2, the Boolean circuit with ($n \times (n + 1)$) ***AND*** gates and ($\frac{n\times(n+1)}{2}$) ***NOT*** gates generated from Steps (4a) through (4b) in each iteration in the molecular algorithm **Solve-independent-set-problem**(Y_0, n, m) is the *best* Boolean circuit *known* for counting the number of vertices in each solution.

Proof: Please refer to the content of this subsection. ■

II. Quantum Algorithms for Implementing the Straightforward Boolean Circuits from Molecular Solutions for Solving the Independent Set Problem

A. *Computational State Space of Molecular Solutions for the Independent Set Problem*

We use a unique *computational basis vector* with 2^n-tuples of binary numbers to represent each element in set (tube) Y_0. The first corresponding computational basis vector for the *first* element $y_n^0\ y_{n-1}^0 \ldots y_2^0\ y_1^0$ is $([1 \quad 0 \quad \cdots \quad 0]^T_{1\times 2^n})$. And so on, with the *last* corresponding computational basis vector for the *last* element $y_n^1\ y_{n-1}^1 \ldots y_2^1\ y_1^1$ being $([0 \quad 0 \quad \cdots \quad 1]^T_{1\times 2^n})$. Therefore, the set of the corresponding computational basis vectors that span the space C^{2^n} is $D = \{ [1 \quad 0 \quad \cdots \quad 0]^T_{1\times 2^n}, [0 \quad 1 \quad \cdots \quad 0]^T_{1\times 2^n}, \ldots, [0 \quad 0 \quad \cdots \quad 1]^T_{1\times 2^n} \}$ and. This is to say it forms an *orthonormal* basis of a 2^n dimensional Hilbert space.

B. *Quantum Circuits and Mathematical Solutions for Computational State Space of Molecular Solutions for the Independent Set Problem*

Each possible molecular solution corresponds to an element in an orthonormal basis of a Hilbert space (C^{2^n}). A quantum register of n bits, $(\otimes_{p=n}^{1}|y_p\rangle)$, is set to $(\otimes_{p=n}^{1}|y_p^0\rangle)$. We assume that $|\lambda_0\rangle = (\otimes_{p=n}^{1}|y_p^0\rangle)$ and the initial quantum state vector is $(|\lambda_0\rangle)$. Using n Hadamard gates to operate on the initial quantum state vector $(|\lambda_0\rangle)$, 2^n possible molecular solutions are encoded by the following new state vector $(|\lambda_{5-1}\rangle)$

$$|\lambda_{5-1}\rangle = (H^{\otimes n})|\lambda_0\rangle = \frac{1}{\sqrt{2^n}}(\otimes_{p=n}^{1}(|y_p^0\rangle + |y_p^1\rangle)) = \frac{1}{\sqrt{2^n}}(\textstyle\sum_{y=0}^{2^n-1}|y\rangle). \quad (2\text{-}1)$$

In the new state vector $(|\lambda_{5-1}\rangle)$, state $|y_n^0\ y_{n-1}^0 \ldots y_2^0\ y_1^0\rangle$ with the amplitude $(\frac{1}{\sqrt{2^n}})$ encodes the *first* element $y_n^0\ y_{n-1}^0 \ldots y_2^0\ y_1^0$ of the molecular solution space that does not contain any vertices. And so on, with state $|y_n^1\ y_{n-1}^1 \ldots y_2^1\ y_1^1\rangle$ with the amplitude $(\frac{1}{\sqrt{2^n}})$ encoding the *last* element $y_n^1\ y_{n-1}^1 \ldots y_2^1\ y_1^1$ of the molecular solution space containing n vertices $\{v_n\ v_{n-1} \ldots v_2\ v_1\}$. Using one Hadamard gate on the state $|1\rangle$ gives the new quantum state vector $(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle))$ that labels the amplitude of the answer(s) among the 2^n states.

C. *Quantum Circuits and Mathematical Solutions for Implementing Molecular Solutions for legal Independent Sets among 2^n Possible Choices*

The straightforward Boolean circuit for labelling legal independent sets among the 2^n possible choices in Fig. 1-1 is

$$(\wedge_{k=1}^{m}(\overline{y_i \wedge y_j})), \quad (2\text{-}2)$$

where bits y_i and y_j respectively represent vertices v_i and v_j in the kth edge, $e_k = (v_i, v_j)$, in G for $1 \le k \le m$. The Boolean formula $(\wedge_{k=1}^{m}(\overline{y_i \wedge y_j}))$ consists of m **NAND** operations and m **AND** operations. The **NAND** operation and the **AND** operation are, respectively, implemented by quantum circuits in Figures 2-1(a) and 2-1(b). The initial state for each quantum bit in the second quantum register $|l_m\ l_{m-1} \ldots l_1\rangle$ is prepared in state $|1\rangle$. The kth quantum bit $|l_k\rangle$ for $1 \le k \le m$ stores the result of evaluating the kth **NAND** gate of the form $(\overline{y_i \wedge y_j})$. The mth quantum bit $|l_m\rangle$ stores the result of the evaluating computation for the *last* **NAND** operation.

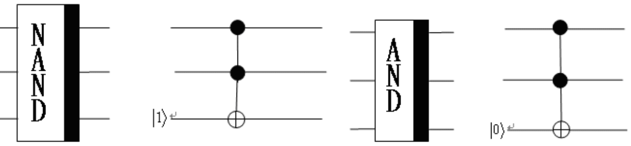

Fig. 2-1: (a) **NAND** operation of two Boolean variables, and (b) **AND** operation of two Boolean variables.

Next, the first quantum bit $|o_0\rangle$ in the third quantum register $|o_m\ o_{m-1} \ldots o_1\ o_0\rangle$ is initially prepared in state $|1\rangle$ and the other m bits are initially in state $|0\rangle$. The kth quantum bit $|o_k\rangle$ for $1 \le k \le m$ stores the result of evaluating the **AND** operation of the previous clause (the $(k-1)$th clause) and the current clause (the kth clause). The $(m + 1)$th quantum bit $|o_m\rangle$ stores the result of evaluating the **AND** operation of the *last two* clauses. This indicates that the $(m + 1)$th quantum bit $|o_m\rangle$ stores the result of the evaluating computation for all of the clauses. We use the quantum circuit **LIS** in Fig. 2-2 with $(2 \times m)$ ***CCNOT*** gates to implement the straightforward Boolean circuit $(\wedge_{k=1}^{m}(\overline{y_i \wedge y_j}))$ in equation (2-2).

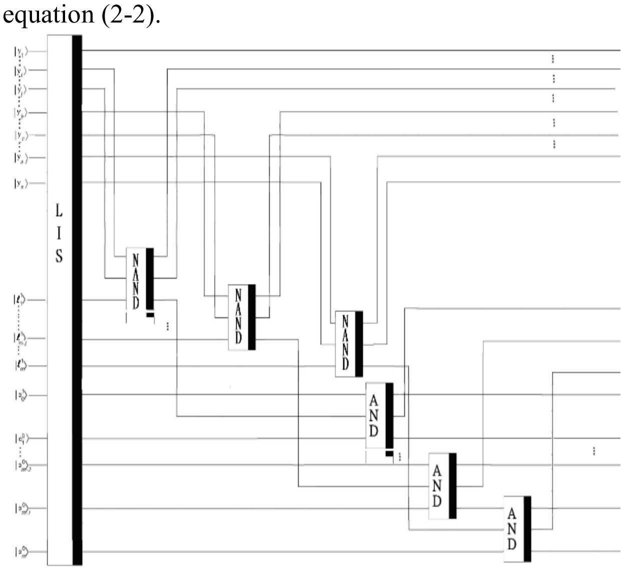

Fig. 2-2: The quantum circuit, LIS, used to label legal independent sets among 2^n possible choices.

D. *Quantum Circuits and Mathematical Solutions of Molecular Solutions to the Maximum-sized Independent Sets*

The straightforward Boolean circuits in Figure 1-2 for counting the number of vertices in each legal independent sets are

$(w_{1,1} \leftarrow o_m \wedge y_1)$ and $(w_{1,0} \leftarrow o_m \wedge \overline{y_1})$ and (2-3)

$(w_{i+1,j+1} \leftarrow y_{i+1} \wedge w_{i,j})$ and $(w_{i+1,j} \leftarrow \overline{y_{i+1}} \wedge w_{i,j})$ for $1 \le i \le n-1$ and $0 \le j \le i$. (2-4)

For $0 \le i \le n-1$ and $0 \le j \le i$, each auxiliary quantum bit in $|w_{i+1,j}\rangle$ and $|w_{i+1,i+1}\rangle$ is initially prepared in state $|0\rangle$. Quantum bit $|w_{i+1, j+1}>$ will record the status of tube (set) Y_{j+1} that has $(j + 1)$ ones after the influence of y_{i+1} on the number of ones. Quantum bit $|w_{i+1, j}>$ will record the status of tube (set) Y_j that has j ones after the influence of y_{i+1} on the number of ones. We use the quantum circuit **CFFV** in Fig. 2-3 to implement the straightforward Boolean circuit $(w_{1,1} \leftarrow o_m \wedge y_1)$ and $(w_{1,0} \leftarrow o_m \wedge \overline{y_1})$ in equation (2-3). We also use the quantum circuit **CMO** in Fig. 2-4 to implement the straightforward Boolean circuit $(w_{i+1, j+1} \leftarrow y_{i+1} \wedge w_{i, j})$ and $(w_{i+1, j} \leftarrow \overline{y_{i+1}} \wedge w_{i, j})$ for $1 \leq i \leq n - 1$ and $0 \leq j \leq i$ in equation (2-4).

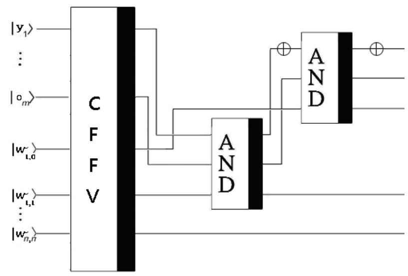

Fig. 2-3: Implementation of the first and the second conditions of equation (2-3) using the quantum circuit CFFV.

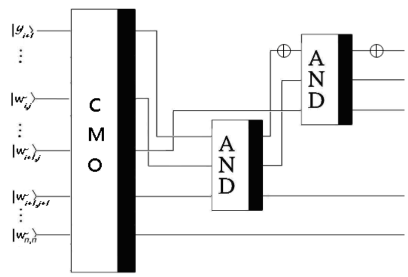

Fig. 2-4: Implementation of the first and the second conditions of (2-4) using the quantum circuit CMO.

E. *Quantum Circuits and Mathematical Solutions from Reading Molecular Solutions for the Maximum-sized Independent Sets*

The 2^n possible molecular solutions that are created by Steps (0a) through (1d) in the algorithm **Solve-independent-set-problem** are initialized in the distribution: $(\frac{1}{\sqrt{2^n}} \frac{1}{\sqrt{2^n}} \frac{1}{\sqrt{2^n}} \ldots \frac{1}{\sqrt{2^n}})$. This indicates that there is the same amplitude in each of the 2^n possible molecular solutions. The previously proposed quantum circuits have labelled the answer(s), but the amplitude or probability of finding the answer(s) will decrease exponentially. Hence, based on [2], the diffusion operator is applied to increase exponentially the amplitude or probability of finding the answer(s), and is defined by matrix G as follows: $G_{i, j} = (2 / 2^n)$ if $i \neq j$ and $G_{i, i} = (-1 + (2 / 2^n))$. **Algorithm 2-1** is used to measure the answer(s) that are generated by Steps (5a) and (5b) in the algorithm **Solve-independent-set-problem**.

For convenience of presentation, we assume that $|y_b{}^1>$, $|l_k{}^1>$, $|o_k{}^1>$, $|w_{i+1, j}{}^1>$ and $|w_{i+1, i+1}{}^1>$ for $1 \leq b \leq n$, $0 \leq k \leq m$, $0 \leq i \leq n - 1$, and $0 \leq j \leq i$, subsequently, represent the fact that the value of their corresponding quantum bits is 1. We further assume that $|y_b{}^0>$, $|l_k{}^0>$, $|o_k{}^0>$, $|w_{i+1, j}{}^0>$ and $|w_{i+1, i+1}{}^0>$ for $1 \leq b \leq n$, $0 \leq k \leq m$, $0 \leq i \leq n - 1$, and $0 \leq j \leq i$, subsequently, represent tha fact that the value of their corresponding quantum bits is 0. Furthermore, we have made use of the notation from **Algorithm 2-1** below in previous subsections. We use the first parameter t in **Algorithm 2-1** to represent the maximum size of vertex sets among legal answers, and the execution of Step (1a) in **Algorithm 2-2** in the next subsection passes its value.

Algorithm 2-1 (t): Mathematical solutions obtained by reading molecular solutions of the maximum-sized independent sets for any graph G with m edges and n vertices.

(0) A unitary operator, $\mathbf{U}_{init} = (H) (\otimes_{i=n}^{1} \otimes_{j=i}^{0} I_{2 \times 2}) (\otimes_{k=m}^{1} I_{2 \times 2}) (I_{2 \times 2}) (\otimes_{k=m}^{1} I_{2 \times 2}) (H^{\otimes n})$, operates on an initial quantum state vector, $(|1>) (\otimes_{i=n}^{1} \otimes_{j=i}^{0} |w_{i, j}{}^0>) (\otimes_{k=m}^{1} |o_k{}^0>) (|o_0{}^1>) (\otimes_{k=m}^{1} |l_k{}^1>) (\otimes_{b=n}^{1} |y_b{}^0>)$, and the 2^n possible choices of n bits (containing all possible independent sets) are

$|\varphi_{0,0}> = (\frac{1}{\sqrt{2}} (|0> - |1>)) \frac{1}{\sqrt{2^n}} (\otimes_{i=n}^{1} \otimes_{j=i}^{0} |w_{i, j}{}^0>) (\otimes_{k=m}^{1} |o_k{}^0>) (|o_0{}^1>) (\otimes_{k=m}^{1} |l_k{}^1>) (\otimes_{b=n}^{1} (|y_b{}^0> + |y_b{}^1>))$.

(1) For labeling which among the 2^n possible choices are legal independent sets and which are not answers, a quantum circuit in Figure 2-2, $(I_{2 \times 2}) (\otimes_{i=n}^{1} \otimes_{j=i}^{0} I_{2 \times 2})$ **(LIS)**, is used to operate on the quantum state vector $|\varphi_{0,0}>$, and the following new quantum state vector is obtained

$|\varphi_{1,0}> = (\frac{1}{\sqrt{2}} (|0> - |1>)) \frac{1}{\sqrt{2^n}} (\otimes_{i=n}^{1} \otimes_{j=i}^{0} |w_{i, j}{}^0>) (\sum_{y=0}^{2^n - 1} (\otimes_{k=m}^{1} |o_k{}^0 \oplus l_k \bullet o_{k-1}>) (|o_0{}^1>) (\otimes_{k=m}^{1} |l_k{}^1 \oplus y_i \bullet y_j>) (|y>))$.

(2) For implementing $(w_{1,1} \leftarrow o_m \wedge y_1)$ and $(w_{1,0} \leftarrow o_m \wedge \overline{y_1})$ in equation (2-3), a quantum circuit in Figure 2-3, $(I_{2 \times 2})$ **(CFFV)**, is applied to the quantum state vector $|\varphi_{1,0}>$, and the following new quantum state vector is

$|\varphi_{2,0}> = (\frac{1}{\sqrt{2}} (|0> - |1>)) \frac{1}{\sqrt{2^n}} (\otimes_{i=n}^{2} \otimes_{j=i}^{0} |w_{i, j}{}^0>) (\sum_{y=0}^{2^n - 1} (|w_{1, 1}{}^0 \oplus o_m \bullet y_1>) (|w_{1, 0}{}^0 \oplus o_m \bullet \overline{y_1}>) (\otimes_{k=m}^{1} |o_k>) (|o_0{}^1>)(\otimes_{k=m}^{1} |l_k>) (|y>))$.

(3) **For** $i = 1$ **to** $n - 1$

(4) **For** $j = i$ **down to** 0

(4a) A quantum circuit in Fig. 2-4, $(I_{2 \times 2})$ **(CMO)**, is to determine the number of vertices among the legal independent sets and operates on the quantum state vector $(|\varphi_{2+\left(\sum_{\theta_1=0}^{i-1}(\theta_1+1)\right)+(i-j),0}>)$. Since Step (4a) is embedded in the only loop, after repeatedly executing the quantum circuit in Fig. 2-4, $(I_{2 \times 2})$ **(CMO)**, the resulting state vector for calculating the number of vertices in each legal independent set is

$\left|\varphi_{2+\frac{n^2+n-2}{2},0}\right\rangle = (\frac{1}{\sqrt{2}}(|0>-|1>)) \frac{1}{\sqrt{2^n}} (\sum_{y=0}^{2^n-1} (\otimes_{i=n}^{1} \otimes_{j=i}^{0} |w_{i,j}\rangle) (\otimes_{k=m}^{1} |o_k>) (|o_0{}^1>) (\otimes_{k=m}^{1} |l_k>) (|y>))$.

End For

End For

(5) A ***CNOT*** gate $(\frac{|0>-|1>}{\sqrt{2}} \oplus w_{n, t})$ with the target bit $|\frac{|0>-|1>}{\sqrt{2}}>$ and the control bit $|w_{n, t}>$ labels the legal independent set(s) with the maximum number of vertices in the quantum state vector $(|\varphi_{2+\frac{n^2+n-2}{2},0}>)$, and the following new quantum state vector is

$|\varphi_{2+\frac{n^2+n-2}{2}+1,0}> = (\frac{1}{\sqrt{2}} (|0> - |1>)) \frac{1}{\sqrt{2^n}} \times (-1)^{w_{n,t}} (\sum_{y=0}^{2^n-1} (\otimes_{i=n}^{1} \otimes_{j=i}^{0} |w_{i,j}>) (\otimes_{k=m}^{1} |o_k>) (|o_0{}^1>) (\otimes_{k=m}^{1} |l_k>) (|y>))$.

(6) Because quantum operations are reversible by nature, reversing all the operations carried out by Steps (4a), (2) and (1) can restore the auxiliary quantum bits to their initial states.

(7) Apply the diffusion operator to the quantum state vector produced in Step (6).

(8) Repeatedly execute Step (1) to Step (7) at most $O(\sqrt{2^n / R})$

times, where the value of R is the number of solutions and can be determined with the quantum counting algorithm [2].

(9) The answer is obtained with a probability of success of at least (1 / 2) after a measurement is completed.

End Algorithm

Lemma 2-1: The output of **Algorithm 2-1** are mathematical solutions obtained by reading molecular solutions of the maximum-sized independent sets for any graph G with m edges and n vertices.

Proof: Since there are 2^n possible choices (including all possible independent sets) to the independent set problem for any graph G with m edges and n vertices, a quantum register of n bits ($\otimes_{b=n}^{1}|y_b>$) can represent 2^n choices with initial state vector ($\otimes_{b=n}^{1}|y_b{}^0>$). The independent set problem for any graph G with m edges and n vertices requires finding a maximum-sized independent set in G, so those auxiliary quantum registers are necessary. By executing Step (0), an initial state vector $|\Omega>$ = ($|1>$) ($\otimes_{i=n}^{1}\otimes_{j=i}^{0}|w_{i,j}{}^0>$) ($\otimes_{k=m}^{1}|o_k{}^0>$) ($|o_0{}^1>$) ($\otimes_{k=m}^{1}|l_k{}^1>$) ($\otimes_{b=n}^{1}|y_b{}^0>$) starts the quantum computation of the independent set problem. A unitary operator, $\mathbf{U}_{init}$ = (H) ($\otimes_{i=n}^{1}\otimes_{j=i}^{0}I_{2\times 2}$) ($\otimes_{k=m}^{1}I_{2\times 2}$) ($I_{2\times 2}$) ($\otimes_{k=m}^{1}I_{2\times 2}$) ($H^{\otimes n}$), operates on the initial state vector $|\Omega>$, and the resulting state vector becomes $|\varphi_{0,0}>$ with 2^n choices. This indicates that 2^n possible molecular choices generated by Steps (0a) through (1d) in the algorithm **Solve-independent-set-problem** can be implemented by Step (0) in **Algorithm 2-1**.

Next, Step (1) in **Algorithm 2-1** acts as the unitary operator **LIS** in Fig. 2-2. On the execution of Step (1) in **Algorithm 2-1**, those choices among the 2^n possible that satisfy the *straightforward* Boolean circuit ($\wedge_{k=1}^{m}(\overline{y_i \wedge y_j})$) in equation (2-2) are labeled. After the execution of Step (1) has been completed, the resulting state vector $|\varphi_{1,0}>$ is obtained, containing those choices with $|o_m{}^1>$ that indicate them to be legal independent sets and those illegal choices with $|o_m{}^0>$ that do not satisfy the condition. Hence, the straightforward Boolean circuit ($\wedge_{k=1}^{m}(\overline{y_i \wedge y_j})$) in equation (2-2) generated by Steps (2a) through (2d) in the algorithm **Solve-independent-set-problem** can be implemented by Step (1) in **Algorithm 2-1**.

Next, Step (2) in **Algorithm 2-1** acts as the unitary operator **CFFV** in Fig. 2-3. On the execution of Step (2) in **Algorithm 2-1**, the number of ones from the influence of the first vertex in each legal independent set is computed. After the execution of Step (2), the state vector $|\varphi_{2,0}>$ is obtained, which includes those legal independent sets with $|w_{1,1}{}^1>$ that have *one* ones and contain the *first* vertex and those legal independent sets with $|w_{1,0}{}^1>$ that have *zero* ones and do not contain the first vertex. This implies that the straightforward Boolean circuit ($w_{1,1} \leftarrow o_m \wedge y_1$) and ($w_{1,0} \leftarrow o_m \wedge \overline{y_1}$) in equation (2-3) generated by Steps (4a) and (4b) in the *first* iteration (0, 0) in **Solve-independent-set-problem** can be implemented by Step (2) in **Algorithm 2-1**.

Next, Step (4a) in **Algorithm 2-1** works as the unitary operator **CMO** in Fig. 2-4. This step is to determine the number of ones (the number of vertices) among the legal independent sets. Steps (3) and (4) consist each of a two-level loop. When the value of the index variable i is equal to one and the value of the index variable j is from one down to zero, Step (4a) is executed repeatedly two times. Similarly, when the value of the index variable i is equal to two and the value of the index variable j is from two down to zero, Step (4a) is executed repeatedly three times. Similarly, when the value of the index variable i is equal to (n – 1) and the value of the index variable j is from (n – 1) down to zero, Step (4a) is repeatedly executed n times. This is to say that the total number of executions of Step (4a) is (2 + 3 + … n) = ($n^2 + n - 2$) / 2. Because the state vector $|\varphi_{2,0}>$ is generated from Step (2) and its index is 2 (two), after repeatedly executing Step (4a), we use 2 + (($n^2 + n - 2$) / 2) as the index of the resulting state and the resulting state vector $|\varphi_{2+\frac{n^2+n-2}{2},0}>$ is obtained in which the number of vertices in each legal independent set is calculated. This indicates that the straightforward Boolean circuit ($w_{i+1,\,j+1} \leftarrow y_{i+1} \wedge w_{i,\,j}$) and ($w_{i+1,\,j} \leftarrow \overline{y_{i+1}} \wedge w_{i,\,j}$) for $1 \le i \le n - 1$ and $0 \le j \le i$ in equation (2-4) generated in Steps (4a) and (4b) in the same iteration (i, j) in **Solve-independent-set-problem** can be implemented by Step (4a) in **Algorithm 2-1**.

Next, one **CNOT** gate, ($\frac{|0>-|1>}{\sqrt{2}} \oplus w_{n,\,t}$) with the target bit $|\frac{|0>-|1>}{\sqrt{2}}>$ and the control bit $|w_{n,\,t}>$, in Step (5) of **Algorithm 2-1** labels the answer(s) with the phase (–1). The resulting state vector $|\varphi_{2+\frac{n^2+n-2}{2}+1,0}>$ consists of the part of the answer with the phase (–1) and the other part with the phase (+1). Because quantum operations are reversible by nature, the execution of Step (6) will reverse all these operations completed by Step (4a), Step (2) and Step (1) that can restore the auxiliary quantum bits to their initial states. Next, on the execution of Step (7) in **Algorithm 2-1**, the diffusion operator is applied to complete the task of increasing the probability of success in measuring the answer(s). In Step (8) in **Algorithm 2-1**, after repeatedly executing Steps (1) through (7) of $O(\sqrt{2^n/R})$ times, a maximum probability of success is generated. Next, by executing Step (9) in **Algorithm 2-1**, a measurement is obtained and the answer(s) is/are returned to **Algorithm 2-2**. Because the result produced by each step in **Algorithm 2-1** is a unit vector in a finite-dimensional Hilbert space, therefore, we at once infer that the output of **Algorithm 2-1** are the mathematical solutions obtained by reading molecular solutions of the maximum-sized independent sets to any graph G with m edges and n vertices. ■

F. Solving the Independent Set Problem on any Graph G with m Edges and n Vertices

The following algorithm solves the independent-set problem for any graph G with m edges and n vertices. We have used the notations used in **Algorithm 2-2** in the previous subsections.

Algorithm 2-2: Solving the independent set problem for any Graph G with m edges and n vertices.

(1) **For** $t = n$ **to** 1

(1a) Call **Algorithm 2-1**(t).

(1b) **If** the answer is obtained from the tth execution of Step (1a) **then**

(1c) Terminate **Algorithm 2-2**.

End If

End For

End Algorithm

Lemma 2-2: **Algorithm 2-2** obtains the maximum-sized independent sets for any graph G with m edges and n vertices.

Proof: In each execution of Step (1a) in **Algorithm 2-2**, it calls **Algorithm 2-1** to find the answer(s) with t vertices. Next, in each execution of Step (1b) in **Algorithm 2-2**, if the answer(s) is (are) found, then the tth execution of Step (1c) in **Algorithm 2-2** will terminate **Algorithm 2-2**. Otherwise, repeatedly execute Steps (1a) through (1c) until the answer(s) is (are) found. ■

III. Complexity Assessment

A. *The Time and Space Complexity of Algorithm 2-2*

Lemma 3-1: The best case time complexity for **Algorithm 2-2** involves $((2^{n/2} \times (2 \times n)) + (n + 1))$ **Hadamard** gates, $(2^{n/2} \times (2 \times (n^2 + n)))$ **NOT** gates, $(2^{n/2})$ **CNOT** gates, $(2^{n/2} \times (4 \times m + (2 \times (n^2 + n))))$ **CCNOT** gates, $(2^{n/2})$ phase shift gates of n quantum bits and a quantum measurement.

Proof: Please refer to **Algorithms 2-1** and **2-2**. ■

Lemma 3-2: The worst case time complexity for **Algorithm 2-2** is $(n \times ((2^{n/2} \times (2 \times n)) + (n + 1)))$ **Hadamard** gates, $(n \times (2^{n/2} \times (2 \times (n^2 + n))))$ **NOT** gates, $(n \times 2^{n/2})$ **CNOT** gates, $(n \times (2^{n/2} \times (4 \times m + (2 \times (n^2 + n)))))$ **CCNOT** gates, $(n \times 2^{n/2})$ phase shift gates of n quantum bits and (n) quantum measurements.

Proof: Please refer to **Algorithms 2-1** and **2-2**. ■

Lemma 3-3: The worst and the best case spatial complexity for solving the independent set problem for any graph G are the same: $((2 \times m + 2 \times n + 2) + ((n \times (n + 1)) / 2))$ quantum bits.

Proof: Please refer to **Algorithms 2-1** and **2-2**. ■

B. *Proof of a Quadratic Speedup for Solving the Independent Set Problem for any Graph G with m Edges and n Vertices*

Lemma 3-4: **Algorithm 2-2** gives a quadratic speed-up for solving the independent set problem for any graph G. This speedup is the *best* speed-up *known* for the problem.

Proof: From [2] and **Lemma 3-2**, we immediately derive that **Algorithm 2-2** gives a quadratic speed-up, which is the *best* speed-up *known* for solving the problem. ■

IV. Experimental Results of Finding the Maximum-Sized Independent Sets in a Graph with Three Vertices and Two Edges

In Fig. 4-1, graph G^2 consists of three vertices and two edges. The independent sets in G^2 are $\{v_2, v_3\}$, $\{v_1\}$, $\{v_2\}$, $\{v_3\}$ and $\{\}$. The *maximum-sized* independent set for G^2 is $\{v_2, v_3\}$. We write the program in OpenQASM ver. 2.0 to find the *maximum-sized* independent set $\{v_2, v_3\}$ of graph G^2. Fig. 4-2 is the corresponding quantum circuit.

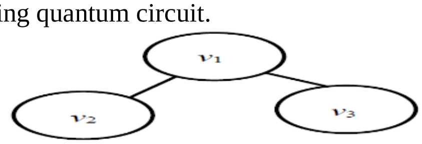

Fig. 4-1: Graph G^2 with three vertices and two edges.

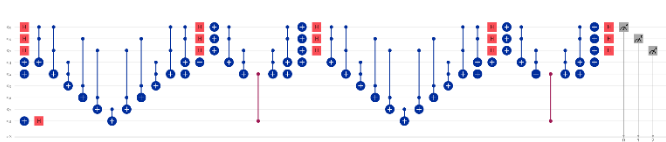

Fig. 4-2: The corresponding quantum circuit for finding the answer $\{v_2, v_3\}$.

The program labels the amplitude of the answer(s) by (−1) and amplifies the amplitude of the answer(s) twice. It declares nine quantum bits with the initial state |0> and three classical bits with the initial value 0. Quantum bit q[2] encodes vertex v_3, quantum bit q[1] encodes vertex v_2 and quantum bit q[0] encodes vertex v_1. Next, we use the statements "h q[0]; h q[1]; h q[2]; x q[8]; h q[8]; x q[3]; x q[4];" to generate all possible solutions and to set the initial state of the auxiliary quantum bits. The next eleven statements label the amplitude of the answer(s) by (−1). Then, the amplitude amplification is executed by "h q[0]; h q[1]; h q[2]; x q[0]; x q[1]; x q[2]; x q[3]; x q[4]; ccx q[0],q[1],q[3]; ccx q[3],q[2],q[4]; cz q[4],q[8]; ccx q[3],q[2],q[4]; ccx q[0],q[1],q[3]; x q[0]; x q[1]; x q[2]; x q[3]; x q[4]; h q[0]; h q[1]; h q[2]". After that, the next eleven statements will again label the amplitude of the answer(s) by (−1). And the remaining gates will again execute the amplitude amplification of the answer(s). The measurement is carried out by the last three statements that are "measure q[0] -> c[0]; measure q[1] -> c[1]; measure q[2] -> c[2];". We use the command "simulate" to execute the quantum circuit in Fig. 4-2 on the simulator backend. Fig. 4-3 shows the measured results for the program. The state q[2] q[1] q[0] = 110 is observed with the highest probability of 0.55. This state corresponds to the answer $\{v_2, v_3\}$.

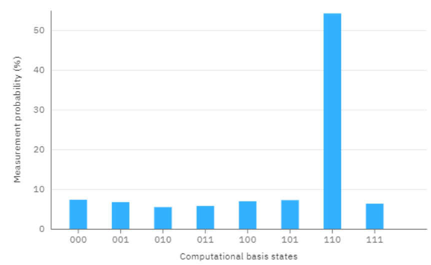

Fig. 4-3: The measured result of finding the answer $\{v_2, v_3\}$ on the backend Simulator.

V. Conclusion

We show that the independent set problem for any graph can be solved by the algorithm **Solve-independent-set-problem** with $O(n^2+m)$ biological operations, $O(2^n)$ DNA strands, $O(n)$ tubes and the longest DNA strand, $O(n)$. **Lemma 2-1** to **Lemma 2-2** show that the same problem can be solved with a quadratic speed-up by **Algorithm 2-2** and **Algorithm 2-1** which implement the straightforward Boolean circuits generated from the algorithm **Solve-independent-set-problem**. In **Lemma 3-1** to **Lemma 3-4**, we show that **Algorithm 2-2** and **Algorithm 2-1** give a quadratic speed-up which is the *best* speed-up *known* for dealing with the problem.

References

[1] Chang, W.-L. and Vasilakos, A.V. Molecular Computing: Towards a Novel Computing Architecture for Complex Problem Solving, Springer, ISBN-13: 978-3319051215, ISBN-10: 3319051210, June 2014.

[2] Chang, W.-L. and Vasilakos, A.V. *Fundamentals of Quantum Programming in IBM's Quantum Computers*, Springer, ISBN 13: 978-3030635824, ISBN 10: 3030635821, January 2021.