**3 A Series of Quantum Algorithms with Varying Complexity**

We provide three quantum algorithms with increasing algorithmic code complexity and decreasing complexity with respect to resource consumption.

**3. 1. A First Attempt**

Before presenting the algorithm, we need to explain some quantum states (M and b(\*, \*) values are as in the algorithm in section 2):

The algorithm, now, goes as follows:



α = n!; N = N\_A; n = N\_B

**repeat**

|y> = 1/sqrt(α) **Σ**\_{u, C} |t\_u> **χ** |C>

Measure the superposed states and let the resulting state be:



where **χ** and **Ε** denotes tensor product and inclusion in set, respectively, u **Ε** U = {u | u = k\_1 + k\_2 + ... +k\_M <= M and k\_j **Ε** {0, 1, ..., M}, 1 <= j <= M}, C **Ε** μ = {[β\_1, β\_2, ..., β\_\_{n - 1}]^T | β\_j **Ε** {0, 1}^M,1 <= i <= n - 1} and **Π** denotes composition of transpositions.

**if** γ\_{(u, C} is not in STATES **then**

Construct the permutation matrix M\_γ from the permutation γ\_{u, C}

S\_n = S\_n U **Π**\_ν(A^{β\_{1,1}}\_ν Α^{β\_{2,1}}\_ν ... A^{β\_{n - 1,ν}}\_{(n - 1)})^{τ^ν\_u}, (1 <= ν <= Μ)

/\* the following nested loop computes all subsets of n vertices in the total of N vertices of graph A so that (m\_1, m\_2, ..., m\_n) with m\_1 < m\_2 < ...< m\_n and m\_j is in {1,2, ..., N}, j in {1,2, ..., n}. \*/

/\* Initialize STATES \*/

**for** m\_n = N, N - 1, ..., n **do**

**for** m\_{n - 1} = N - 1, ..., n - 1 **and** m\_{n - 1} < m\_n **do**

**for** m\_1 = N - n + 1, ..., 1 **and** m\_1 < m\_2 **do**

STATES\_{m\_1,m\_2, ...,m\_n} = {}

**end**

**end**

**end**

**for** m\_n = N, N - 1, ..., n **do**

**for** m\_{n - 1} = N - 1, ..., n - 1 **and** m\_{n - 1} < m\_n **do**

**for** m\_1 = N - n + 1, ..., 1 **and** m\_1 < m\_2 **do**

STATES\_{m\_1,m\_2, ...,m\_n} = STATES\_{m\_1, m\_2, ...,m\_n} U {γ\_{(u, C)}}

Apply INDEX\_d\_{γ}(γ\_{u, C}(m\_1, m\_2, ..., m\_n), A, B, d\_γ) from the 2-Dimensional scratch tape of all possible TMs of the Quantum Machine (QM) as in section 1. It actually performs the piece of commands of the algorithm on page 20 of the paper by N. Mariella and A. Simonetto, "A Quantum Algorithm for the Sub-Graph Isomorphism Problem", (2021)

**if** d\_γ = 0 **then**

Output the permutation γ\_{u, C}(m\_1, m\_2, ..., m\_n) at the QM's output tape

**break**

**end**

α = α - 1

**end**

**end**

**end**

**until** α = 0

Now #B = 2^{(n - 1)\*M}, #U = o(2^M) and O(M^M) (# denotes "number of"). We activate #B \* #U states totally, that is, o(2^{n \* M}) and O((2^n \* M)^M), and the length of qubit strings is o(n \* M) and O(n \* M + MlogM).

Now, the time complexity of this algorithm is obviously exponential in the worst-case scenario; e.g., if there is no isomorphism, then for all the related inputs the running time is exponential most probably. The number of iterations on the {m\_1, m\_2, ..., m\_n} is the binomial coefficient (N over n), which is polynomial. Further, we can see that due to superposition and assuming the QM algorithm works on an input data augmented also by the recursive trees k\_1 + k\_2 + ...+ k\_M = ν, 1 <= ν <= M plus the matrix C, so that paths to the leaves from the roots of the recursive summation trees and, as well as, access to the elements of C, can be performed in O(1) time by the QM's input head, and if #(d\_γ = 0) = ο(#(d\_γ <> 0)) where # denotes "number of", for at least one subgraph of A on all possible sets of vertices {m\_1, m\_2, ..., m\_n}, then T(ω) = Ο(ω^c), c positive constant and ω the input length, which is defined by the N\_A and N\_B (or, samely N and n, respectively). Such a case implies that graph A has a subgraph on some vertices {m\_1, m\_2, ..., m\_n} isomorphic to graph B. Then by the below lemma if there exists an isomorphism, the time complexity of this quantum algorithm is polynomial in the worst case.

**Lemma 3.1*.*** *All permutations on the vertices on the 2-Dimensional cartesian space of graph A result geometrically in isomorphic graphs to A.*

*Proof:* By induction on the n vertices of A we see that the lemma holds. We start with n = 3. Without loss of generality, the vertices are arranged on the plain and number-labeled as below.



We can see all possible graphs of 3 vertices with regard to the possible edges, if permuted in any manner, they give geometrically isomorphic to A. Assume now that all possible graphs on some n > 3 are permuted by any manner give isomorphic-to-A graphs. Assume we take graph A on n + 1 vertices. Now A is generated by a subgraph B of A on n vertices plus the (n + 1)st vertex and the related edges from it to B. Any permutation on A that does not move the (n + 1)st vertex is a permutation on B which gives, by the induction hypothesis, a subgraph of A isomorphic to B. Now putting all the vertices from the (n + 1)st vertex to the resulting subgraph is an isomorphic graph to A. The case that a permutation moves the (n + 1)st vertex is straightforward. Therefore any permutation on every graph on n vertices, n >= 3, results geometrically to an isomorphic graph. QED