**Noname manuscript No.**
(will be inserted by the editor)

# Multiple Network Alignment on Quantum Computers

**Anmer Daskin · Ananth Grama · Sabre Kais**



Received: date / Accepted: date

**Abstract** Comparative analyses of graph structured datasets underly diverse problems. Examples of these problems include identification of conserved functional components (biochemical interactions) across species, structural similarity of large biomolecules, and recurring patterns of interactions in social networks. A large class of such analyses methods quantify the topological similarity of nodes across networks. The resulting correspondence of nodes across networks, also called node alignment, can be used to identify invariant subgraphs across the input graphs.

Given $k$ graphs as input, alignment algorithms use topological information to assign a similarity score to each $k$-tuple of nodes, with elements (nodes) drawn from each of the input graphs. Nodes are considered similar if their neighbors are also similar. An alternate, equivalent view of these network alignment algorithms is to consider the Kronecker product of the input graphs, and to identify high-ranked nodes in the Kronecker product graph. Conventional methods such as PageRank and HITS (Hypertext Induced Topic Selection) can be used for this purpose. These methods typically require computation of the principal eigenvector of a suitably modified Kronecker product matrix of the input graphs. We adopt this alternate view of the problem to address the problem of multiple network alignment. Using the phase estimation algorithm, we show that the multiple network alignment problem can be efficiently solved on quantum computers. We characterize the accuracy and performance of our method, and show that it can deliver exponential speedups over conventional (non-quantum) methods.

A. Daskin -A. Grama
Department of Computer Science, Purdue University, West Lafayette, IN, 47907 USA

S. Kais
Department of Chemistry, Department of Physics and Birck Nanotechnology Center,Purdue University, West Lafayette, IN 47907 USA; Qatar Environment and Energy Research Institute, Doha, Qatar

## 1 Introduction

Recent developments have shown that quantum computers can efficiently solve diverse important problems – often delivering exponential seedups compared to their classical counterparts [1,2]. Examples of such problems include finding low energy states in lattice protein folding [3], simulation of chemical dynamics [4,5,6], calculation of thermal rate constants [7], Shor's factoring technique [8], Grover's search algorithm [9] and others [10,11,12,13,14,15,16,17,18,19]. The phase estimation algorithm [1,20], used for finding eigenvalues of a matrix, has been a key ingredient of many of these quantum algorithms [14,15,16,17].

Graph structured datasets play an essential role in the representation of relationships and interactions between entities. Comparative analyses of these datasets underly diverse applications, including problems in chemoinformatics and bioinformatics. A commonly used analysis technique aims to quantify the topological similarity of nodes across a given set of graphs. Aligning nodes with high similarity, one may identify approximate invariant subgraphs among the input graphs. The approximation, in this case, is desirable, since it renders the underlying methods more robust to noise in the input datasets. This paper focuses on the problem of multiple network alignment. Specifically, it aims to develop quantum methods for computing the topological similarity of nodes across a given set of $n$ graphs.

Techniques for quantifying topological similarity of nodes can be classified as local or global. The former defines similarity on the basis of local neighborhoods of nodes, while the latter uses the entire graph to compute similarity. A commonly used global approach to computing node similarity uses the following principle: two nodes are similar if their neighbors are similar. This principle can be used to express the similarity matrix (a matrix whose $(i,j)$th element corresponds to the similarity of node $i$ in the first graph with node $j$ in the second) in an iterative form.

An alternate formulation of the same method operates on the Kronecker product of input graphs. Given two graphs $G_1$ with $r$ vertices and $G_2$ with $s$ vertices, with corresponding adjacency matrices $A_1$ of dimension $r \times r$ and $A_2$ of dimension $s \times s$, the Kronecker product of the graphs $G_1 \otimes G_2$ is computed through the adjacency matrix $K_{12} = A_1 \otimes A_2$. One may view this Kronecker product matrix as the adjacency matrix of the product graph. This graph has $r \times s$ vertices, labeled $ij$. Vertex $ij$ has an edge to vertex $i'j'$ in the product graph iff there exists an edge between vertices $i$ and $i'$ in the first graph *and* $j$ and $j'$ in the second graph. In other words, the alignment of vertex $i$ in the first graph to vertex $i'$ in the second graph and $j$ in the first graph to $j'$ in the second graph is supported by the existence of an edge between the pair of aligned vertices in *both* graphs. Now, consider powers of the matrix $K_{12}$. In particular, the $ij$th element of the matrix $K_{12} \times K_{12}$ contains the number of length two paths between vertices $i$ and $j$. This corresponds to the number of *neighbor* alignments that support the $ij$th alignment. The argument can be extended to higher powers of the matrix $K_{12}$. Stated alternately, the largest entries in the higher powers of the matrix are the best aligned nodes. If the

matrix is suitably normalized (row-stochastic), then the principal eigenvector reveals the strong alignments between the graphs $G_1$ and $G_2$. The argument can be generalized beyond two input graphs to an arbitrary number of input graphs, $G = \{G_1, \ldots, G_n\}$. Note that the size of the Kronecker product graph grows exponentially in the number of input graphs $n$. The principal eigenvector of this graph reveals the strong complete vertex alignments by computing the alignment score of each $n$-tuple [21,22,23,24].

It is important to note that real-world graphs in bioinformatics and chemoinformatics are typically large ($10^4$ vertices, $10^5$ edges and beyond, and 10 graphs and beyond). The Kronecker product of these graphs can be viewed as combining local operators to represent them as a global operator in quantum circuits. Furthermore, the sparsity of these graphs renders then well-suited to efficient simulation on quantum computers [25]. Motivated by these considerations, we focus on the problem of multiple graph alignment using the principal eigenvector formulation of the implicit Kronecker product matrix. We show that for the types of the problems where the dominant eigenvalue is known (1 in our case) and the corresponding eigenvector is the solution to the problem, one can efficiently produce the solution as a quantum state using the quantum phase estimation procedure. We show that in the case of stochastic matrices, one can generate the solution as a quantum state with the success probability 1. We present the quantum simulation algorithm, quantify its cost, and show that in some cases, we can achieve exponential improvement w.r.t. corresponding non-quantum methods.

In the following sections, we first briefly discuss the classical network alignment method based on the PageRank and HITS ranking algorithms. We then describe the phase estimation algorithm, and show how it can be adapted to the network alignment problem. Finally, we discuss the representation of networks on quantum computers and the complexity of our method to find the eigenvector (and consequently the node alignments).

## 2 Principal Eigenvectors and Ranking Nodes in Networks

Consider the problem of computing the principal eigenvector of a matrix using a random walk on a suitably specified transition matrix. Assume, for generality that the graph is directed; the case for undirected graphs is a special (symmetric) case of the directed case.

Given a graph (in our case, the Kronecker product graph), the transition matrix $P$ is constructed by setting value $p_{ij} = 1/deg(i)$, where $deg(i)$ is the out-degree of node $i$, for each $j$ that node $i$ is connected to, and 0 otherwise. Note that this matrix is not stochastic, since there may be nodes with no out-edges; i.e., their transition probabilities sum to 0. A number of solutions have been proposed to deal with this. Perhaps, the most commonly used is the PageRank formulation [26], used in web ranking. Pagerank deals with the problem by specifying a vector of a-priori probabilities to which a walker jumps to if there are no out edges from a node. In this case, the new transition matrix

is defined as $\hat{P} = P + \mathbf{d}\mathbf{w}^T$, where $\mathbf{w}$ is the a-priori probability vector, and elements of $\mathbf{d}$ are defined to be 1 if $deg(i) = 0$, and 0 otherwise. Although this matrix is stochastic, it is reducible; i.e., there may be multiple eigenvectors on the unit circle. To address this, $\hat{P}$ is replaced by the matrix $\tilde{P}$ given by [27]:

$$\tilde{P} = \alpha\hat{P} + (1 - \alpha)E. \tag{1}$$

Here, $\alpha \in [0, 1]$ and $E = \mathbf{e}\mathbf{v}^T$: $\mathbf{e} = [1, \ldots, 1]^T$, and the vector $\mathbf{v}$ is called the personalization vector. It adds to all nodes a new set of outgoing transitions with small probabilities. The power iterations for matrix $\tilde{P}$, $\mathbf{r} = \tilde{P}^T\mathbf{r}$, converge to a unique vector, which is the eigenvector corresponding to the dominant eigenvalue of $\tilde{P}$, which is 1.

## 2.1 Example: PageRank inspired Protein-Protein Interaction (PPI) Network Similarity

Advances in high-throughput methodologies, supplemented with computational approaches have resulted in large amounts of protein interaction data. This data is typically represented as a protein-protein interaction (PPI) network – an undirected graph, $G(V, E)$, in which $V$ represents the set of proteins and edge $(v_i, v_j) \in E$ represents observed interaction between proteins $v_i$ and $v_j \in V$. Comparative analyses of PPI networks of different species help in identifying conserved functional components across species. The most common comparative analysis technique is based on the alignment of PPI networks, where correspondences between nodes in different networks are used to maximize an objective function. [21,22,23,24]

Singh et al.[24] proposed an iterative global algorithm called IsoRank, in which the similarity measure between two nodes is determined by the similarity of their neighbors. For two graphs, the iterative relation for pairwise similarity of nodes follows:

$$R_{ij} = \sum_{u \in N(i)} \sum_{v \in N(j)} \frac{1}{|N(u)||N(v)|} R_{uv}. \tag{2}$$

Here, $N(w)$ is the set of neighbors of node $w$; $|N(w)|$ is the size of this set; $V_1$ and $V_2$ are the set of nodes for networks $G_1$ and $G_2$; and $i \in V_1$ and $j \in V_2$. $R$ defines the functional similarity matrix whose stationary state is used to find the solution for the alignment problem. Eq.(2) can be written in matrix form as:

$$R = \tilde{A}R, \tag{3}$$

where $\tilde{A}$ is a stochastic matrix constructed from the Kronecker product of the input graphs with principal eigenvalue of one, and is defined as $\tilde{A} = \tilde{A}_1 \otimes \tilde{A}_2$. $\tilde{A}_i$ represents the modified adjacency matrix for the graph $G_i$. The matrix $R$ is the stationary distribution of the random walk on the Kronecker product graph. Since $\tilde{A}$ has all positive entries, the infinite product of the matrix

will have a limit. The limiting matrix $R$ is the matrix with every row equaling the left eigenvector associated with eigenvalue one. Therefore, it can be solved using the power method [30]. A-priori information regarding similarity of nodes (for example, what proteins in one species are functionally related to proteins in other species) can also be integrated into the iterative form. This a-priori information may be derived from the Bit-Score of the BLAST (sequence) alignments [31]. The following equation integrates this a-priori information in matrix $H$ as:

$$R = \alpha \tilde{A} R + (1 - \alpha) H. \tag{4}$$

Here, entries of matrix $H$ define the Bit-Score between two nodes (proteins) and $\alpha$ is a parameter that controls the weight of the network data relative to the a-priori node-similarity data. This procedure can be written in an iterative form as:

$$R(k+1) = \alpha \tilde{A} R(k) + (1 - \alpha)\boldsymbol{h}. \tag{5}$$

This equation, in the limit, simplifies to the following [22]:

$$R(\infty) = (1 - \alpha) \sum_{k=0}^{\infty} \alpha^k \tilde{A}^k \boldsymbol{h}. \tag{6}$$

The $R$ matrix can then be used in conjunction with a bipartite matching process to identify a set of maximally aligned nodes across the networks. While our discussion has been in the context of two networks, the procedure can be generalized to the multiple network alignment problem [21,32,33].

## 2.2 Example: PageRank based Molecular Similarity

Structurally similar chemical compounds generally exhibit similar properties. Analyses of similarity of molecules plays an important role in infering properties of compounds, and in designing new materials with desired characteristics [34]. Graph kernels can be used to compute similarity of molecules. In this approach, molecules are represented as undirected graphs, also called molecular graphs. Vertices in these graphs correspond to atoms and edges correspond to covalent bonds. In molecular graphs, vertices and edges are annotated with element and bond types. These graphs can be large – for instance, a muscle protein titin has $4.23 \times 10^5$ atoms. Different graph-based approaches have been proposed for molecular similarity.

Rupp et al.[35] describe a technique based on iterative graph similarity. Given molecular graphs $G_1$ and $G_2$ represented by their adjacency matrices $A_1$ and $A_2$, the following update equation is used for computing the pairwise similarity vector $x$:

$$\boldsymbol{x_{i+1}} = (A_1 \otimes A_2)\boldsymbol{x_i}. \tag{7}$$

In order to include more molecular graph properties (e.g. number of bonds), this formula is subsequently modified as:

$$\boldsymbol{x_{i+1}} = (1 - \alpha)k_v + \alpha \times \max_P P\boldsymbol{x_i}, \tag{8}$$

where $k_v$ is the vector of kernel functions between vertices (similarity functions measuring the similarity of pairs of atoms in graphs by including information on their certain substructures), and $P$ is a square matrix, compliant with the neighborhood structure, whose rows represent the possible neighbor assignments. For detailed explanation, refer to Rupp et al. [35].

## 2.3 Ranking Nodes in a Graph: The HITS Algorithm

Consider a directed graph $G(V, E)$ with $m$ vertices. Each node now has two parameters – an authority weight and a hub weight. These weights are determined by the inherent quality of the node and the number of edges to other authoritative nodes, respectively. The HITS (Hypertext Induced Topic Selection) algorithm of Kleinberg [36] associates a non-negative authority weight $a_i$ and a non-negative hub weight $h_i$ with node $i$. Since the algorithm was originally proposed for ranking web pages, nodes correspond to web pages and edges to hyperlinks. HITS computes numerical estimates of hub and authority scores using an iterative procedure: if a node is pointed to by many good hubs, its authority is increased in the next iteration. For a node $i$, the value of $a_i$ is updated to be the sum of $h_i$ over all nodes that point to $i$:

$$a_i = \sum_{j:j\to p} h_j. \tag{9}$$

The hub weight of the page is also increased in a similar way:

$$h_i = \sum_{j:p\to j} a_j. \tag{10}$$

As a result, the update rules for the vectors of the authority weights and the hub weights of the pages, respectively $\boldsymbol{a}$ and $\boldsymbol{h}$, can be written as:

$$\begin{aligned} \boldsymbol{a} &= A^T\boldsymbol{h} = A^TA\boldsymbol{a} = (A^TA)\boldsymbol{a} \\ \boldsymbol{h} &= A\boldsymbol{a} = AA^T\boldsymbol{h} = (AA^T)\boldsymbol{h}. \end{aligned} \tag{11}$$

Vectors $\boldsymbol{a}$ and $\boldsymbol{h}$ converge to the principal eigenvectors of $A^TA$ and $AA^T$, respectively.

Lempel and Moran [37] modified the HITS algorithm to a random walk on a graph: for the adjacency matrix $A$, a stochastic matrix $W_r$ is constructed by dividing each entry of $A$ by its row sum. Similarly, another stochastic matrix $W_c$ is generated by dividing each entry of $A$ by its column sum. Then, the iterations for the vectors of the authority and hub weights are computed as defined [38]:

$$\begin{aligned} \boldsymbol{a} &= W_c^TW_r\boldsymbol{a} \\ \boldsymbol{h} &= W_r^TW_c\boldsymbol{h}. \end{aligned} \tag{12}$$

Initially, $\boldsymbol{a}$ and $\boldsymbol{h}$ are unit vectors. Since $(W_c^TW_r)$ and $(W_cW_r^T)$ are stochastic matrices, $\boldsymbol{a}$ and $\boldsymbol{h}$ converge to the principal eigenvectors of, respectively, $(W_c^TW_r)$ and $(W_cW_r^T)$ associated with the eigenvalue 1.

### 2.4 Network Similarity using the HITS Algorithm

Blondel et al. [39] propose an iterative network similarity algorithm based on the HITS method. For directed graphs $G_1$ and $G_2$ with adjacency matrices $A_1$ and $A_2$, the iterative equation for the algorithm is given by:

$$X_{k+1} = A_2 X_k A_1^T + A_1 X_k A_2^T, \tag{13}$$

where $X_0$ is the matrix of all ones. This equation is converted to the vector form:

$$\boldsymbol{x_{k+1}} = (A_1 \otimes A_2 + A_1^T \otimes A_2^T)\boldsymbol{x_k}, \tag{14}$$

where $\boldsymbol{x_k}$ is the vector form of $X_k$.

Having established the network similarity computation problem as one of computing the principal eigenvector of a suitably defined matrix (with dominant eigenvalue 1), we now focus on finding the steady state – the principal eigenvector, of the iterations defined in Sec.II and Sec.III on quantum computers.

## 3 Phase Estimation Process and Eigenvector Generation

The phase estimation algorithm [1,20] is a quantum algorithm for estimating the eigenphase corresponding to a given approximate eigenvector of a unitary matrix. For the eigenvalue equation: $U|\mu_j\rangle = e^{i2\pi\phi_j}|\mu_j\rangle$, it finds the value of $\phi_j$ for a given approximate eigenvector $|\mu_j\rangle$. The algorithm uses two quantum registers: $|reg1\rangle$ and $|reg2\rangle$. While $|reg1\rangle$ is initially on zero state, $|reg2\rangle$ holds the eigenvector of the unitary matrix. After putting $|reg1\rangle$ into the superposition, we apply a sequence of operators, $U^{2^j}$, controlled by the $j$th qubit of $|reg1\rangle$, to $|reg2\rangle$. This generates the Fourier transform of the phase on $|reg1\rangle$. The application of the inverse quantum Fourier transform makes $|reg1\rangle$ hold the binary value of the phase.

Assume that we have the operator $U = e^{i2\pi\tilde{A}}$, where $\tilde{A}$ is the Kronecker product matrix for which we are trying to compute ranks. The eigenvalues of the ranking matrix $\tilde{A}$ are known to be $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{N-1} < \lambda_N = 1$, associated with eigenvectors $|\mu_1\rangle, \ldots, |\mu_N\rangle$, where $N$ is the size of $\tilde{A}$. Consequently, in the ranking problem, we need to find the eigenvector associated with the eigenvalue $\lambda_N = 1$.

The above problem is, in some sense, the inverse of the standard phase estimation algorithm. In the phase estimation process, instead of a particular eigenvector, if the initial state of $|reg2\rangle$ is set to a superposition of the eigenvectors (not necessarily uniform), as in Shor's factoring algorithm[8], $|reg1\rangle$ holds the superposition of the eigenvalues of $\tilde{A}$ in the final state. Since the principal eigenvalue $\lambda_N$ is known, in the sate where we have $\lambda_N$ on $|reg1\rangle$ (for the eigenvalue 1, the state where $|reg1\rangle = |0\rangle$), we have the corresponding eigenvector on $|reg2\rangle$, which is the solution to the network alignment problem.

## 3.1 Steps of the Algorithm

Here, we give the states in each step of the algorithm applied to the ranking matrix $\tilde{A}$ with the eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{N-1} < \lambda_N = 1$, associated with eigenvectors $|\mu_1\rangle, \ldots, |\mu_N\rangle$:

1. Find the unitary operator $U = e^{i2\pi\tilde{A}}$, which can be found easily if $\tilde{A}$ is sparse (see Section 3.4).
2. Initialize the quantum registers $|reg1\rangle$ and $|reg2\rangle$ as $|reg1\rangle = |0\rangle$ and $|reg2\rangle = |\mu\rangle$, which is the superpostion of the eigenvectors (choosing $|\mu\rangle = H^{\otimes}|0\rangle$ makes the success probability of the algorithm equal to 1, see Section 3.2.)
3. Apply the quantum Fourier transform to $|reg1\rangle$, which produces the state:

$$\frac{1}{\sqrt{\kappa}} \sum_{j=0}^{\kappa-1} |j\rangle |\mu\rangle . \tag{15}$$

4. Apply $U^{2^j}$ controlled by the $j$th qubit of $|reg1\rangle$ to $|reg2\rangle$. If we only consider the $j$th qubit of $|reg1\rangle$, then the following quantum state is obtained:

$$\frac{1}{\sqrt{\kappa}}(|0\rangle |\mu\rangle + |1\rangle U^{2^j} |\mu\rangle). \tag{16}$$

   $U^{2^j}|\mu\rangle$ generates a superposition of the eigenvectors with the coefficients determined by the eigenvalues. If we assume $|\mu\rangle = \frac{1}{\sqrt{N}} \sum_i^N |\mu_i\rangle$, then for the eigenvector $|\mu_i\rangle$, we have the coefficient $\lambda_i^{2^j}/\sqrt{N}$ in the state $U^{2^j}|\mu\rangle$:

$$\frac{1}{\sqrt{\kappa}}(|0\rangle |\mu\rangle + |1\rangle \frac{1}{\sqrt{N}} \sum_i^N U^{2^j} |\mu_i\rangle) = \frac{1}{\sqrt{\kappa}}(|0\rangle |\mu\rangle + |1\rangle \frac{1}{\sqrt{N}} \sum_i^N \lambda_i^{2^j} |\mu_i\rangle). \tag{17}$$

   Note that since the principal eigenvalue is 1, the largest coefficient is $\lambda_N^j/\sqrt{N} = 1/\sqrt{N}$ and so the dominant term in $U^{2^j}|\mu\rangle$ is the principal eigenvector $|\mu_N\rangle$.
5. Apply the inverse Fourier transform to obtain the superposition of the binary form of the phases in $|reg1\rangle$.
6. Finally, apply conditional measurement to $|reg1\rangle$ to produce the eigenvector $|\mu_N\rangle$ on $|reg2\rangle$ corresponding the principal eigenvalue $\lambda_N = 1$: i.e., if $|reg1\rangle = |0\rangle$, $|reg2\rangle = |\mu_N\rangle$. However, as discussed in Section 3.2 and Section 3.3, the success probability for the stochastic matrices is 1. Therefore, there is no need for conditional measurement. After Step 5 (the application of the inverse Fourier transform), $|reg1\rangle = |0\rangle$. Thus, $|reg2\rangle$ holds the principal eigenvector.

### 3.2 Success Probability

The success probability of the algorithm is the probability of observing the principal eigenvalue on $|reg1\rangle$, which is related to the closeness of the input to the principal eigenvector. If we have $|\mu\rangle = H^{\otimes n}|0\rangle$ as the initial input on $|reg2\rangle$, the amplitudes of the eigenvalues on $|reg1\rangle$ change depending on the closeness of the eigenvectors to this input. We measure the closeness between an eigenvector and the input vector by using the dot product of these vectors: i.e., the cosine of the angle between these vectors:

$$\begin{pmatrix} \langle\mu_1|\mu\rangle \\ \langle\mu_2|\mu\rangle \\ \vdots \\ \langle\mu_N|\mu\rangle \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{N}}\sum_j \mu_{1j} \\ \frac{1}{\sqrt{N}}\sum_j \mu_{2j} \\ \vdots \\ \frac{1}{\sqrt{N}}\sum_j \mu_{Nj} \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_N \end{pmatrix} \tag{18}$$

When the angle between two vectors is small, $\beta_i$s get larger. The squares of the amplitudes in the above vector give us the success probability for finding an eigenvector on $|reg2\rangle$ and the corresponding eigenvalue on $|reg1\rangle$. For instance, the probability of observing the eigenvector $|\mu_N\rangle$ on $|reg2\rangle$ is $\beta_N^2$. Fig.1 and Fig.2 show the comparison of the expected probabilities computed from Eq.(18) with the probabilities found in the phase estimation algorithm for random matrices with the dominant eigenvalue 1. The following Perron-Frobenius theorem [40] provides the basis for comparing the probability for the principal eigenvalue, $\beta_N^2$, with the others:

**Theorem 1** *For an irreducible non-negative square matrix, the dominant eigenvalue is positive and has multiplicity one. The eigenvector (unique up to scaling) corresponding to this eigenvalue is also positive, and there are no other non-negative eigenvectors for this matrix.*

Based on the above theorem, the vector $|\mu_N\rangle$ must be positive. Therefore, the cosine of the angle between the input and the principal eigenvector, $\beta_N = \langle\mu_N|\mu\rangle = \frac{1}{\sqrt{N}}\sum_j \mu_{Nj}$, can be bounded by:

$$1 \geq \beta_N > \frac{1}{\sqrt{N}}. \tag{19}$$

Here $\beta_N$ is $\frac{1}{\sqrt{N}}$ only when an element of the eigenvector is one and the rest of the elements are zero. Since the principal eigenvector is positive, and all the other eigenvectors include negative elements; $\beta_N > |\beta_j|$, $1 \leq j \leq N-1$. Fig.1 shows the success probabilities for a collection of $32 \times 32$ random matrices, while Fig.2 shows success probabilities for random matrices of various dimensions. Random matrices used in these experiments are symmetric positive and generated using the Wishard method [41,42]. Here, one creates a random matrix $X$ and uses the product $XX^T$ to generate a symmetric matrix. We also scale the matrices so that the largest eigenvalue is one. We observe, in these experiments, that success probability is very high and grows sharply with the size of the system.

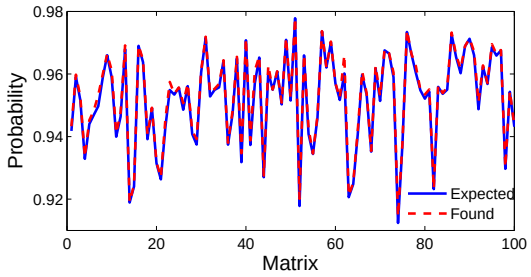


Fig. 1: The success probabilities for 32x32 random symmetric positive matrices.

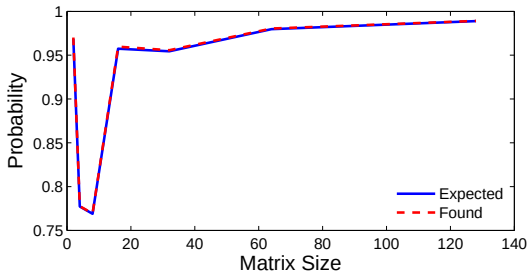


Fig. 2: The success probabilities for random symmetric positive matrices of different sizes.

### 3.3 Success Probability in the Case of Stochastic Matrices

For a row stochastic matrix $A$ with the eigenvalue $\lambda_k$, and the associated eigenvector $|\mu_k\rangle$, the eigenvalue equation can be written as:

$$A\,|\mu_k\rangle - \lambda_k\,|\mu_k\rangle = 0, \tag{20}$$

or more explicitly,

$$\begin{pmatrix} a_{11} & \dots & a_{1M} \\ \vdots & \dots & \vdots \\ a_{M1} & \dots & a_{MM} \end{pmatrix} \begin{pmatrix} \mu_{k1} \\ \vdots \\ \mu_{kM} \end{pmatrix} - \lambda_k \begin{pmatrix} \mu_{k1} \\ \vdots \\ \mu_{kM} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$
$$\left(\mu_{k1}\boldsymbol{a_1} + \dots + \mu_{k1}\boldsymbol{a_M}\right) - \lambda_k \begin{pmatrix} \mu_{k1} \\ \vdots \\ \mu_{kM} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{21}$$

Here, $\boldsymbol{a_j}$s are the column vectors. If we sum the rows on either side of the equality, since $\sum_j a_{kj} = 1$, we get the following:

$$\begin{aligned} \sum_j \mu_{kj} - \lambda_k \sum_j \mu_{kj} &= 0 \\ (1-\lambda_k) \sum_j \mu_{kj} &= 0 \end{aligned} \tag{22}$$

Hence, for $k \neq N$; since $\lambda_k \neq 1$, $\beta_k = \sum_j \mu_{kj} = 0$. Since there is only one nonzero value in the amplitude vector $|\beta\rangle$ defined in Eq.(18), $\beta_N = \sum_j \mu_{Nj}$ has to be 1. Thus, the success probability of the algorithm is 1 for stochastic matrices, which is the case for the network alignment problem.

## 3.4 Representation of Networks

A network of $M_1$ nodes can be represented using an $M_1 \times M_1$ adjacency matrix $A_1$. In this case, $log(M_1) = m_1$ qubits are needed to represent this network on a quantum computer. If there are $k$ such networks, the total number of qubits for $|reg2\rangle$ is $\sum_{j=1}^{k} m_j$.

For each $U_j = e^{-iA_j t}$, one can find a quantum circuit design with $O(M_j^2)$ number of operations. In network alignment, the Kronecker product of the adjacency matrices is used: $A = A_1 \otimes \cdots \otimes A_k$. Hence, $U = e^{-iAt}$ can be defined as $U = U_1 \otimes \cdots \otimes U_k$. Therefore, the total number of operations is the sum of operations needed for each $U_j$, which is $\sum_{j=1}^{k} O(M_j^2)$. If the networks are of the same size (in terms of number of vertices) $M$, then this sum becomes equal to $O(kM^2)$.

However, these matrices are typically sparse and so can be simulated on quantum computers with an exponential speed-up [44,45]. The operator $U_j = e^{-iA_j t}$ is the same as the operator used in the continuous time quantum walk on a graph defined by the adjacency matrix $A_j$. It has been shown that continuous time random walks can be performed on quantum computers efficiently [25]. The efficiency of the quantum walk has been studied for different types of the graphs, and for a class of graphs it has been shown that traversing is exponentially faster[46]. The exponential efficiency, in general, can be observed when the circuit design for the adjacency matrix or the Laplacian operator of a graph on $m_j$ qubits requires $O(poly(m_j))$ number of one- and two-qubit operations. Lemma 1 in Ref.[25] states that if $A_j$ is a row-sparse (the number of nonzero entries is bounded by $poly(m_j)$) and $||A_j|| \leq poly(m_j)$, then $A_j$ is efficiently simulatable. A Hamiltonian acting on $m_j$ number of qubits is said to be efficiently simulatable if there is a quantum circuit using $poly(m_j, t, 1/\epsilon)$ one- and two-qubit gates that approximates the evolution of $A_j$ for time $t$ with error at most $\epsilon$ [45]. Berry et al.[44] present an algorithm that can simulate $A_j$ with computational complexity bounded by $O((d^4 \times m_j^* ||A_j t||)^{1+o(1)})$, where $d$ is the maximum degree of a vertex in the graph represented by $A_j$. This complexity bound is further improved to $O(d^2(d + m_j^*)||A_j t||)^{1+o(1)}$ by

Childs and Kothari [45]. Therefore, when $A_j$s are row-sparse, the implementation of the operator $U_j = e^{-iA_j t}$ requires $O(poly(m_j))$ number of operations. Thus, the number of operations for $U = e^{-iAt}$ is bounded by $O(k \times poly(m))$, where the networks are assumed to have the same sizes. For dense matrices, although exponential efficiency has not been demonstrated, polynomial efficiency is achievable.

Please note that ranking algorithms operate on a modified matrix $\tilde{A}$, instead of $A$. These modifications can be mapped to the local and global rotation matrices, which eases the difficulty of finding a circuit design.

### 3.5 Algorithmic Complexity

The complexity of the algorithm is dominated by the complexity of the phase estimation algorithm, which depends on the number of operations needed to implement the adjacency matrices. Assuming all networks have the same size $M$, as shown before, the total number of gates in the circuit implementing the evolution of the product of the adjacency matrices $A$ is bounded by $O(kM^2)$. If there are $\kappa$ qubits in the first register, then the phase estimation algorithm requires $O(\kappa k M^2)$ operations excluding the quantum Fourier transform. For a general case, this is more efficient than the number of operations required by the classical algorithms that are based on the power iterations [30]. However, as shown in the previous section, when $A_j$s are row-sparse, they can be efficiently simulatable. In this case, the computational complexity is bounded by $O(poly(m)\kappa k)$, which gives us an exponential efficiency over the classical case. Here, the eigenvector is produced as a quantum state.

### 3.6 Precision and Eigenvalue Gap

In our test cases, using six qubits in $|reg1\rangle$ gave us enough precision to get accurate results. However, when there are other eigenvalues close to one or zero ($e^{i2\pi 0} = e^{i2\pi 1}$), then one must make the size of $|reg1\rangle$ sufficiently large to distinguish the principal eigenvalue from the rest. When the eigenvalue gap between the first and the second eigenvalues is small, the algorithm may generate a vector combination of the eigenvectors corresponding to the second and the first eigenvalues.

## 4 Conclusion

In this paper, we consider the problem of multiple network alignment. We formulate the problem as one of ranking nodes of the Kronecker product graph of the input networks. We use conventional PageRank [26] and HITS [36] algorithms for computing the node rankings. We solve this problem on quantum computers by modifying the well-known quantum phase estimation algorithm

to generate the principal eigenvector of a given operator. We discuss the computational complexity and show that our algorithm has significantly lower computational complexity than classical algorithms. We also show that if the adjacency matrices for the networks are sparse, exponential efficiency is possible. Our proposed framework provides a roadmap for solving numerous other problems that can be formulated as Markovian processes or ranking problems, on quantum computers.

## References


1. D.S. Abrams, S. Lloyd, Phys. Rev. Lett. **83**(24), 5162 (1999). DOI 10.1103/PhysRevLett.83.5162
2. A. Papageorgiou, C. Zhang, Quantum Information Processing **11**(2), 541 (2012)
3. A. Perdomo-Ortiz, N. Dickson, M. Drew-Brook, G. Rose, A. Aspuru-Guzik, Scientific reports **2** (2012)
4. B.C. Sanders, Appl. Math. Inf. Sci **3**(2), 117 (2009)
5. S. Raeisi, N. Wiebe, B.C. Sanders, New Journal of Physics **14**(10), 103017 (2012)
6. I. Kassal, S.P. Jordan, P.J. Love, M. Mohseni, A. Aspuru-Guzik, Proceedings of the National Academy of Sciences **105**(48), 18681 (2008)
7. D. Lidar, H. Wang, Phys. Rev. E **59**, 2429 (1999)
8. P.W. Shor, SIAM Review **41**(2), 303 (1999). DOI 10.1137/S0036144598347011
9. L.K. Grover, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (ACM, New York, NY, USA, 1996), STOC '96, pp. 212–219
10. K.L. Brown, W.J. Munro, V.M. Kendon, Entropy **12** (2010)
11. D. Lu, B. Xu, N. Xu, Z. Li, H. Chen, X. Peng, R. Xu, J. Du, Phys. Chem. Chem. Phys. **14**, 9411 (2012). DOI 10.1039/C2CP23700H
12. I. Kassal, J.D. Whitfield, A. Perdomo-Ortiz, M.H. Yung, A. Aspuru-Guzik, Annual Review of Physical Chemistry **62**(1), 185 (2011). DOI 10.1146/annurev-physchem-032210-103512. PMID: 21166541
13. K.C. Young, M. Sarovar, J. Aytac, C. Herdman, K.B. Whaley, Journal of Physics B: Atomic, Molecular and Optical Physics **45**(15), 154012 (2012)
14. A. Aspuru-Guzik, A. Dutoi, P. Love, M. Head-Gordon, Science **309**, 1704 (2005)
15. H. Wang, S. Kais, A. Aspuru-Guzik, M. Hoffmann, Phys.Chem.Chem.Phys. **10**, 5388 (2008)
16. L. Veis, J. Pittner, J. Chem. Phys. **133**, 194106 (2010)
17. A. Daskin, S. Kais, J. Chem. Phys. **134**(14), 144112 (2011). DOI 10.1063/1.3575402
18. A. Daskin, A. Grama, S. Kais, Quantum Information Processing **13**(2), 333 (2014). DOI 10.1007/s11128-013-0654-1

19. A.M. Childs, W. van Dam, Rev. Mod. Phys. **82**, 1 (2010). DOI 10.1103/RevModPhys.82.1
20. A. Kitaev, Electronic Colloquium on Computational Complexity (ECCC) **3**(3) (1996)
21. S. Mohammadi, A. Grama, in *Functional Coherence of Molecular Networks in Bioinformatics* (Springer, 2012), pp. 97–136
22. G. Kollias, S. Mohammadi, A. Grama, Knowledge and Data Engineering, IEEE Transactions on **24**(12), 2232 (2012). DOI 10.1109/TKDE.2011.174
23. M. Koyutürk, Y. Kim, U. Topkara, S. Subramaniam, W. Szpankowski, A. Grama, Journal of Computational Biology **13**(2), 182 (2006)
24. R. Singh, J. Xu, B. Berger, in *Research in Computational Molecular Biology*, *Lecture Notes in Computer Science*, vol. 4453 (Springer Berlin Heidelberg, 2007), pp. 16–31
25. D. Aharonov, A. Ta-Shma, in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing* (ACM, New York, NY, USA, 2003), STOC '03, pp. 20–29. DOI 10.1145/780542.780546
26. L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab (1999). Previous number = SIDL-WP-1999-0120
27. C. Brezinski, M. Redivo-Zaglia, SIAM Journal on Matrix Analysis and Applications **28**(2), 551 (2006). DOI 10.1137/050626612
28. D.F. Waugh, Advances in protein chemistry **9**, 325 (1954)
29. S. Jones, J.M. Thornton, Proceedings of the National Academy of Sciences **93**(1), 13 (1996)
30. G.H. Golub, C.F. Van Loan, *Matrix computations (3rd ed.)* (Johns Hopkins University Press, Baltimore, MD, USA, 1996)
31. T.A. Tatusova, T.L. Madden, FEMS microbiology letters **174**(2), 247 (1999)
32. R. Singh, J. Xu, B. Berger, et al., in *Pac Symp Biocomput*, vol. 13 (2008), vol. 13, pp. 303–314
33. C.S. Liao, K. Lu, M. Baym, R. Singh, B. Berger, Bioinformatics **25**(12), i253 (2009)
34. A. Bender, R.C. Glen, Org. Biomol. Chem. **2**, 3204 (2004). DOI 10.1039/B409813G
35. M. Rupp, E. Proschak, G. Schneider, Journal of chemical information and modeling **47**(6), 2280 (2007)
36. J.M. Kleinberg, Journal of the ACM (JACM) **46**(5), 604 (1999)
37. R. Lempel, S. Moran, Computer Networks **33**(1), 387 (2000)
38. A. Farahat, T. LoFaro, J.C. Miller, G. Rae, L.A. Ward, SIAM Journal on Scientific Computing **27**(4), 1181 (2006)
39. V.D. Blondel, A. Gajardo, M. Heymans, P. Senellart, P. Van Dooren, SIAM review **46**(4), 647 (2004)
40. C. Meyer, *Matrix analysis and applied linear algebra book and solutions manual*, vol. 2 (Society for Industrial and Applied Mathematics, 2000)
41. J. Wishart, Biometrika **20**(1/2), 32 (1928)

42. M.L. Mehta, *Random matrices*, vol. 142 (Academic press, 2004)
43. D. Biron, O. Biham, E. Biham, M. Grassl, D.A. Lidar, in *Quantum Computing and Quantum Communications* (Springer, 1999), pp. 140–147
44. D. Berry, G. Ahokas, R. Cleve, B. Sanders, Communications in Mathematical Physics **270**(2), 359 (2007). DOI 10.1007/s00220-006-0150-x
45. A.M. Childs, R. Kothari, in *Theory of Quantum Computation, Communication, and Cryptography* (Springer, 2011), pp. 94–103
46. J. Kempe, Contemporary Physics **44**(4), 307 (2003)

This figure "differentsizes.png" is available in "png" format from:

http://arxiv.org/ps/1307.7220v2

This figure "random100.png" is available in "png" format from:

http://arxiv.org/ps/1307.7220v2