Towards Quantum Machine Learning for Constrained Combinatorial Optimization: a Quantum QAP Solver

Xinyu Ye [1] **Ge Yan** [1] **Junchi Yan** [1]

Abstract

Combinatorial optimization (CO) on the graph is a crucial but challenging research topic. Recent quantum algorithms provide a new perspective for solving CO problems and have the potential to demonstrate quantum advantage. Quantum Approximate Optimization Algorithm (QAOA) is a well-known quantum heuristic for CO constructed by a parametric quantum circuit. However, QAOA is originally designed for unconstrained problems and the circuit parameters and solutions are jointly solved with time-consuming iterations. In this paper, we propose a novel quantum neural network (QNN) for learning CO problems in a supervised manner to achieve better and faster results. We focus on the Quadratic Assignment Problem (QAP) with matching constraints and the node permutation invariance property. To this end, a quantum neural network called QAP-QNN is devised to translate the QAP into a constrained vertex classification task. Moreover, we study two QAP tasks: Graph Matching and Traveling Salesman Problem on TorchQauntum simulators, and empirically show the effectiveness of our approach.

1. Introduction

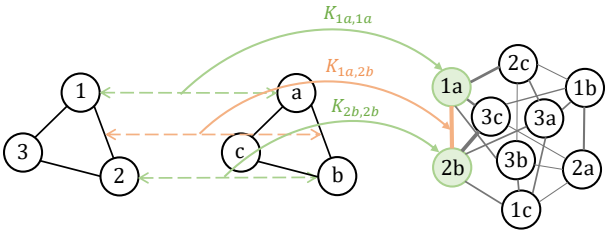
Quantum computing attracts increasing attention (Arute et al., 2019; Preskill, 2018), with the potential benefits from the properties of quantum systems, e.g. superposition, interference, and entanglement. Quantum computing hardware is also stepping into the Noisy Intermediate-Scale Quantum (NISQ) era, facilitating the near-term application of quantum computing in many fields, such as machine learning (Biamonte et al., 2017) and combinatorial optimization (CO) (Farhi et al., 2014).

[1]MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai, China. Correspondence to: Junchi Yan <yanjunchi@sjtu.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

Quantum Approximate Optimization Algorithm (QAOA) (Farhi et al., 2014) is one of the best-known quantum heuristic algorithms on NISQ computers for solving quadratic unconstrained binary optimization (QUBO) problems. The QAOA consists of two main components: a cost Hamiltonian that encodes the problem to be solved and a mixing Hamiltonian that is used to explore the solution space. However, for quadratic *constrained* binary optimization such as the quadratic assignment problem (QAP), it remains a relatively open problem in quantum computing. For example, QAOA needs to add a soft constraint penalty to the objective, making a bounded violation of the constraints difficult to guarantee, which is unfriendly for practical constraint-sensitive scenarios. The Quantum Alternating Operator Ansatz (Hadfield et al., 2019) was proposed to mitigate this issue, which starts with a state in the constrained subspace and guarantees that the quantum state evolves in the constrained space by designing a quantum circuit of mixer Hamiltonians. Quantum Alternating Operator Ansatz is termed C-QAOA to distinguish it from QAOA in this paper. Both QAOA and C-QAOA are problem-inspired ansatzes that contain adjustable parameters. Nevertheless, each problem instance corresponds to different model structures and optimal parameters, and finding them often requires time-consuming classical optimizers or grid searches.

This paper lies in the intersection of quantum machine learning (QML) and CO, for developing QML approaches for solving CO problems, especially for constrained cases. We develop a quantum network paradigm for *constrained* CO, which could be regarded as a quantum counterpart for the emerging paradigm for classic machine learning for CO (Bengio et al., 2021). Our learning-based quantum *constrained* CO solver is designed to bear the following features: **i) consisting of two separate phases for training and inference:** this is in contrast to variational quantum eigensolver (VQE) (Peruzzo et al., 2014) and specifically QAOA-like methods that adjust the trainable circuit parameters and find solution simultaneously. We believe such a design has three merits. First, the circuit parameter training and solution finding can be decoupled to make the optimization easier in each subtask; Second, more instances can be utilized during offline training, which in fact provides a way of utilizing the instances (with given solutions as supervi-



(a) graph matching (b) association graph

Figure 1. The QAP instance by transforming two graphs for matching into one association graph (Wang et al., 2022b), whose vertices denote the node-to-node correspondences in the original two graphs, and its weighted adjacency matrix refers to $\mathbf{K}$ in Eq. 1.

sion); Third, the problem can be efficiently solved through one forward computation of the trained quantum network. **ii) solving *constrained* CO problems** via naturally encoding the constraints into the quantum circuits instead of as a less-controllable penalty term in the objective.

As a concrete case for constrained CO, the quadratic assignment problem (QAP) will be the object of study in this paper, which is defined as (Cho et al., 2010):

$$\begin{aligned} &\operatorname{vec}(\mathbf{X})^\top \mathbf{K} \operatorname{vec}(\mathbf{X}) \\ s.t. \quad &\mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}; \mathbf{X}^\top \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}; \mathbf{X} \in \{0,1\}^{n_1 \times n_2}, \end{aligned} \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ is a (partial) permutation matrix and $\operatorname{vec}(\mathbf{X})$ is column-vectorized version of $\mathbf{X}$. Here $\mathbf{1}_{n_1}$ means a column vector of length n_1 whose elements are all equal to 1. For convenience, we only consider $n_1 = n_2$ in this paper, and then the constraint becomes $\mathbf{X}\mathbf{1}_{n_1} = \mathbf{1}_{n_1}$, $\mathbf{X}^\top \mathbf{1}_{n_1} = \mathbf{1}_{n_1}$. The QAP can be regarded as a combinatorial problem on a graph. If variables of $\mathbf{X}$ are used for indicating vertices of a graph, then matrix $\mathbf{K}$ represents the affinity matrix that captures the relationship between vertices (Wang et al., 2022b). In particular, the Traveling Salesman Problem (TSP) (Arora, 1996) and Graph Matching (GM) problem (Wang et al., 2023) can both be formulated as QAP which will be studied in the experiment part of our paper. Fig. 1 shows how GM is converted to a QAP formulation via the concept of association graph from two input graphs for matching (Wang et al., 2022b).

As a problem on graphs, we first consider the basic permutation invariance property w.r.t. the order of the vertex, *i.e.*, the network needs to output the *consistent result* regardless of the permutation of the vertices. Specifically, we design a permutation-invariance quantum neural network (QNN) as shown in Fig. 2 with three main components: i) input encoding layer, ii) auxiliary constraint layer, and iii) quantum perceptron layer. The *input encoding layer* treats the interaction of qubits as the vertices connected by edges, thereby the graph information can be encoded in the quantum circuit. Moreover, the input encoding layer is repeated in our framework for information interaction among vertices. Followed by the encoding layer, the parametric *constraint layer* is operated on the circuit to restrict interaction among vertices according to the constraint subspace of the problem. Then, one *quantum perceptron module* acts on multiple qubits to learn the feature representation of one vertex, and all vertices share the same quantum perceptron module to ensure permutation invariance. Here quantum perceptron means a parameterized quantum circuit, which can be viewed as a universal function approximator over vector space (Schuld et al., 2021), akin to classical multilayer perceptrons (Hornik et al., 1989). After stacking these three modules several times, a pooling layer is finally used to reduce the dimensions of each vertex to one output qubit. In addition, the result obtained from the measurement employs the Sinkhorn normalization (Adams & Zemel, 2011) to further incorporate the matching constraint. In the end, a classical optimizer is used to optimize the parameters of the proposed quantum neural network. **The highlights of this paper are:**

1) This paper strikes an initiative for solving of constrained combinatorial optimization problem via separately training and testing a parameterized quantum circuit, whereby the solved CO instances as training samples. This is in contrast to QAOA-like methods that jointly find the solution and adjust the trainable parameters, which could be more time-consuming at the inference (problem-solving) stage.

2) Specifically for QAP, we propose a quantum neural network (*i.e.*, parameterized quantum circuits) that both fulfills the node permutation invariance property as well as the matching constraint. Both of them are vital for learning a general quantum model to solve constrained combinatorial problems.

3) We study two specific forms of QAP, including the Graph Matching Problem and the Traveling Salesman Problem, to demonstrate the wide applicability. Numerical experiments show the effectiveness of the proposed approach compared with the classical and quantum-based methods.

2. Related Work

Quantum Computing for CO. A few CO problems are known can be solved by exploiting well-designed quantum algorithms. For example, Grover's algorithm (Grover, 1996) can be used to solve the Boolean satisfiability problem (Cerf et al., 1998). Srinivasan et al. (2018) proposed a quantum algorithm to solve the traveling salesman problem (TSP) using the quantum phase estimation technique. With the development of the parameterized quantum circuit, QAOA was proposed to solve the quadratic unconstrained binary optimization. Although QAOA contains the adjustable model parameter, it is not a unified model that can be trained on

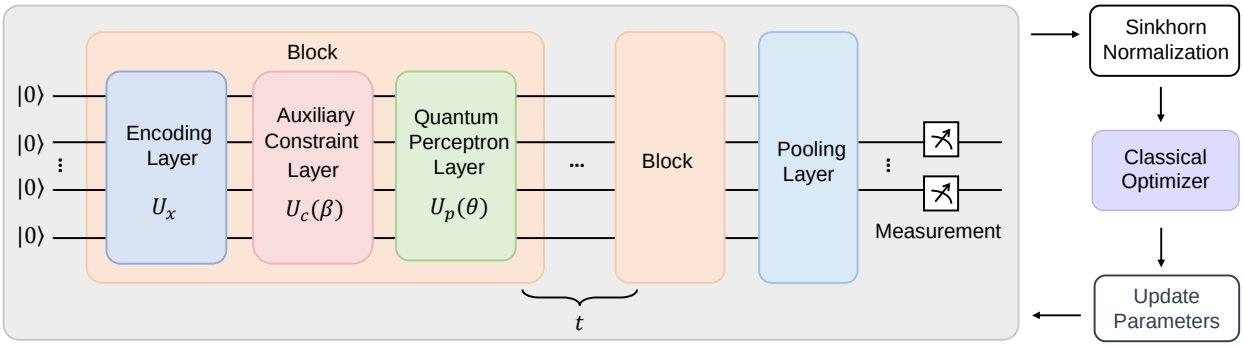


Figure 2. Overview of the proposed QAP-QNN for solving the constrained CO problem (specifically QAP). The encoding layer encodes graph information into the quantum circuit, whereby it is specifically designed (see Fig. 3) for ensuring the node permutation invariance property for the input graph. The auxiliary constraint layer is used to promote the matching constraint in QAP. The quantum perceptron layer is a core learning component for learning the embedding of each vertex. Note the constraint and perceptron layers both naturally fit with the node permutation invariance. The three layers form a block, which is repeated for t times, followed by a pooling layer and measurement to transform the quantum information into the classical one. The output results are first normalized to achieve a doubly stochastic matrix using the Sinkhorn layer before being used to calculate the loss. Note that the node permutation invariance is strictly enforced while the matching constraint is softly obeyed via both the constraint layer as well as the Sinkhorn operator.

a dataset. To obtain the generalization on unseen test instances and find high-quality solutions of QAOA, Khairy et al. (2020) formulated the problem of finding optimal QAOA parameters as a learning task. They employed two classic machine learning techniques: deep reinforcement learning and kernel density estimation to learn the optimal QAOA parameters of different samples. However, this is still limited to the framework of QAOA, and difficult to solve constrained CO problems. In contrast, our approach is a new quantum paradigm for learning to solve constrained CO.

Existing Equivariant Quantum Neural Network. Recent works on equivariant quantum neural networks have been developed. Mernyei et al. (2022) proposed equivariant Hamiltonian quantum graph circuits (EH-QGCs) and equivariantly diagonalizable unitary quantum graph circuits (EDU-QGCs) as special subclasses of equivariant quantum graph circuits (EQGCs) from the perspective of quantum graph representation learning. The works (Ragone et al., 2022; Nguyen et al., 2022) presented an introduction to representation theory tools from the optics of quantum learning and laid the theoretical foundation of Geometric Quantum Machine Learning, such as the group-invariant QML model (Larocca et al., 2022) and the QML model with $SU(d)$ symmetry (Zheng et al., 2022). Among these methods, EQGCs (Mernyei et al., 2022) are the most relevant to our work because both methods target graph-related problems. However, the differences lie in the fact that EQGCs focus on graph classification tasks, while our model is more focused on node-level tasks. In fact, our paper is aimed at addressing constrained CO on graphs via a QNN, which is a more complex problem than node classification. Another difference is that EQGC only considers the graph structure information and has not yet considered the problem of encoding node and edge features into the model.

3. Methodology

We show how to develop a combinatorial neural solver for QAP using a quantum neural network.

3.1. Approach Overview

The idea is to translate the CO problem into a (constrained) vertex classification problem. As described in Eq. 1, the decision variable of QAP is binary, *i.e.*, $x \in \{0, 1\}$. Given the problem instances, it discovers a feature representation for each decision variable in the matching matrix $\mathbf{X}$. As illustrated in Fig. 2, our variational quantum pipeline consists of three components: feature encoding layer, auxiliary constraint layer, and quantum perceptron layer. The three layers form a block and are repeated t times. At the end of the circuit, there are a pooling layer and a measurement layer to output the result. For training, a classical optimizer is employed to learn the trainable parameters of the circuit.

3.2. Feature Encoding Layer

In the formulation of QAP, each decision variable can be associated with a vertex of the graph. And $\mathbf{K}$ contains the affinity information between vertices. The feature encoding layer aims to encode classical affinity information of different instances into the quantum states by the quantum circuit. Suppose that there are n vertices, and each vertex is able to be represented by m qubits. Fig. 3 shows a case of the representations of $m = 1$, and the unitary matrix of the encoding layer is defined as $U_x = \exp(-\mathbf{i}H^{node}) \cdot \exp(-\mathbf{i}H^{edge})$,

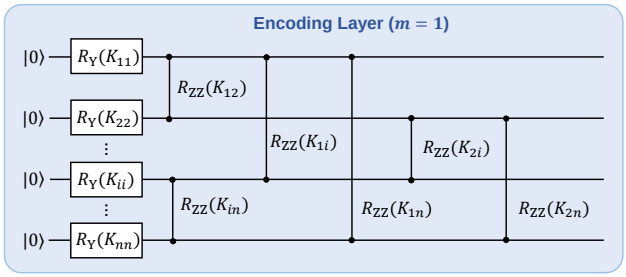


Figure 3. The encoding layer. It encodes the diagonal and off-diagonal elements of the affinity matrix, respectively. The diagonal elements serve as the parameter of R_Y gate, and off-diagonal elements serve as the parameter of the two-qubit quantum gate R_{ZZ}.

where $H^{node} = \sum_i \frac{\mathbf{K}_{ii}}{2} Y_{(i)}$ and $\sum_{i,j} \frac{\mathbf{K}_{ij}}{2} Z_{(i)} Z_{(j)}$, which are Hermitian matrices over one and two-node state spaces, respectively. Y and Z represent the Pauli Y gate and Pauli Z gate, and $Y_{(i)}$ refers to the Y operators applied at the specified node i. For example, $Y_{(3)} = I \otimes I \otimes Y \otimes I$ in the case of $n = 4$ nodes. The rotation gate $R_Y(\mathbf{K}_{ii}) = \exp(-\mathbf{i}\frac{\mathbf{K}_{ii}}{2} Y_{(i)})$ acts on the single qubit with the diagonal elements of $\mathbf{K}$ as parameters. The rotation gate $R_{ZZ}(\mathbf{K}_{ij}) = \exp(-\mathbf{i}\frac{\mathbf{K}_{ij}}{2} Z_{(i)} \otimes Z_{(j)})$ is operated on the two qubits to encode the non-diagonal elements of the $\mathbf{K}$ matrix, which represents the information interaction between vertices. As for the case of $m > 1$, the parameter of each gate is replaced by the elements times m to enhance the expression of high-dimensional feature representations. Note that the U_x repeatedly appears on the circuit. This structure can emphasize the relationships between vertices by entangling the qubits with correlation.

3.3. Auxiliary Constraint Layer

In the constraints of QAP, the sum of the row or column of the matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_1}$ is required to be one. Benefiting from the properties of quantum operators, we can explicitly model the constraints on the quantum circuit. Inspired by the partial mixing operator (Hadfield et al., 2019), the auxiliary constraint layer is built by multiply-controlled X operators. Take the element x_{ij} of $\mathbf{X}$ as a target vertex, and then other elements in the row and column in which x_{ij} is located are viewed as the neighbors of x_{ij}. As can be seen, there is a restrictive relationship between the target and its neighbors. Let $|\mathbf{v}_{x_{ij}}\rangle$ denote the state of the neighbors of x_{ij}, and the unitary operator for the target x_{ij} is defined as:

$$U_{c_{ij}}(\beta) = \sum_{|\mathbf{v}_{x_{ij}}\rangle \neq |0\ldots00\rangle} |\mathbf{v}_{x_{ij}}\rangle\langle\mathbf{v}_{x_{ij}}| \otimes I + \sum_{|\mathbf{v}_{x_{ij}}\rangle = |0\ldots00\rangle} |\mathbf{v}_{x_{ij}}\rangle\langle\mathbf{v}_{x_{ij}}| \otimes R_X(\beta), \tag{2}$$

where β is a trainable parameter of the auxiliary constraint layer. Its overall unitary matrix is given as:

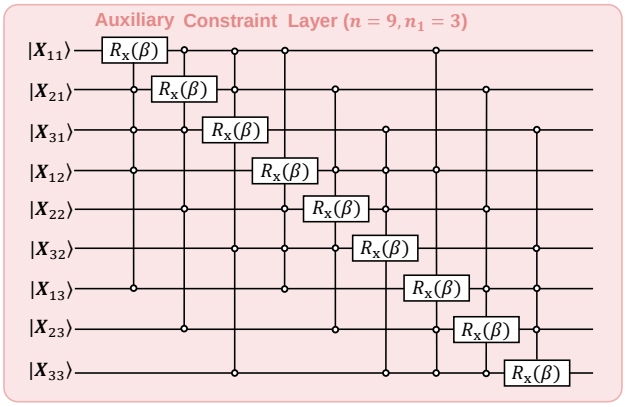


Figure 4. An example of the auxiliary constraint layer with nine vertices on the graph. In the circuit, the multi-qubit control gates are operated on the qubits with a constraint relationship.

$$U_c(\beta) = \prod_{ij} U_{c_{ij}}(\beta). \tag{3}$$

Fig. 4 shows an example of an auxiliary constraint layer when the number of variables n is nine and n_1 equals three. Considering that the strength of constraints imposed on each variable should be consistent, the multiply-controlled gates adopt the same parameter β for the same group. In the case where the state $|x_{ij}\rangle$ is characterized as m qubits, the auxiliary constraint layer employs m different trainable parameters $\boldsymbol{\beta}$.

3.4. Quantum Perceptron Layer

As mentioned before, it is vital that the permutation invariance of input for training a universal quantum variational method. When the order of the vertices encoded by the quantum circuit changes, the order of the output result will vary accordingly. To preserve the permutation invariance, we perform the same quantum perceptron for the m qubits of each vertex. Concretely, the quantum perceptron indicates a m-qubit parametric quantum circuit in the proposed framework. The larger the value of m, the more the number of trainable parameters, and the richer the learned vertex representation. m can be appropriately selected according to the requirement of the practical problem. Fig. 5 illustrates the case of the quantum perceptron layer with different m. The unitary matrix of the quantum perceptron layer is:

$$U_p(\boldsymbol{\theta}) = \bigotimes_{i=1}^{n} U_{rzy}(\boldsymbol{\theta})\, U_{ent}, \tag{4}$$

where $U_{rzy}(\boldsymbol{\theta}) = \otimes_{j=1}^{m} R_Z(\boldsymbol{\theta}_{j1}) R_Y(\boldsymbol{\theta}_{j2})$.

A quantum perceptron layer contains parametric rotation gates and entanglement gates. For each qubit, we perform R_Z and R_Y gates to learn the feature representation and CNOT gates to correlate different feature dimensions of

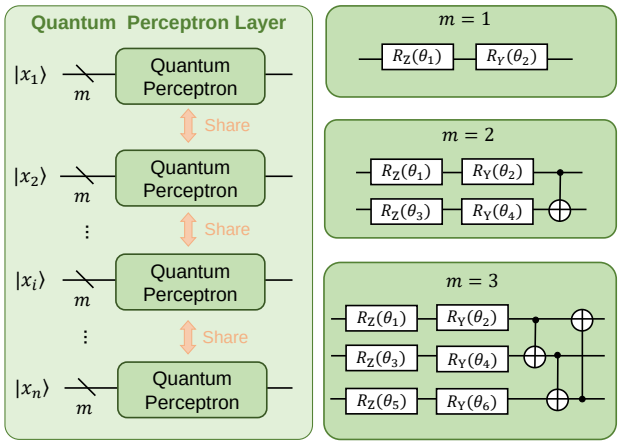


Figure 5. The structure quantum perceptron layer. The quantum perceptron is shared on different vertices. The right shows the structures of a single perceptron module with different m.

a vertex. The quantum perceptron module with the same structure and parameters is shared between different vertices. In this way, the proposed approach is able to keep the property of permutation invariance. Therefore, we can obtain the unitary matrix of the block:

$$U_b(\beta, \boldsymbol{\theta}) = U_x\, U_c(\beta)\, U_p(\boldsymbol{\theta}). \tag{5}$$

3.5. Pooling and Measurement

After the quantum circuit iterates the block comprised of the above three layers t times, a pooling layer is employed to reduce the feature dimension of each vertex to one. We adapt the structure of the Matrix Product State (MPS) circuit (Bhatia et al., 2019) to the pooling layer, which applies two-qubit controlled gates with a ladder-like architecture, as shown in Fig. 6. As we can see, a pooling layer contains $2m-1$ trainable parameters $\boldsymbol{\alpha}$. In the end, the last qubit of m qubits is measured using Pauli Z operators and the circuit outputs the confidence of the current vertex. Similar to the quantum perceptron layer, all vertices share the same structure and parameters of the pooling layer.

3.6. Training and Testing

We can translate the information on quantum states to classical data through the measurement operation. For n vertex, there are the predict result $\mathbf{y} = \{y_i\}_{i=1}^{n}$, where $y_i \in [0, 1]$ represents the classification score of vertex i. Given that the optimal permutation matrix $\mathbf{X}^* \in \mathbb{R}^{n_1 \times n_1}$, we reshape $\mathbf{y}$ into the same size, followed by Sinkhorn normalization (Adams & Zemel, 2011; Wang et al., 2019) to obtain a double-stochastic matrix $\mathbf{Y} \in \mathbb{R}^{n_1 \times n_1}$. Moreover, the binary cross-entropy loss is used for end-to-end learning:

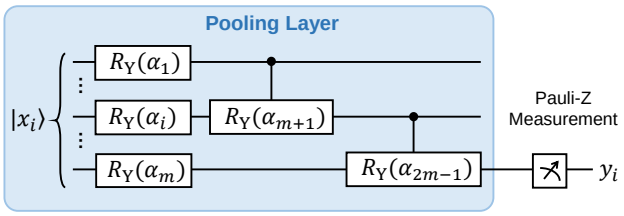


Figure 6. Schematic illustration of the pooling layer and Pauli-Z measurement for the qubits that represent a vertex.

$$l = -\sum_{a=1}^{n_1}\sum_{b=1}^{n_1} \mathbf{X}^*_{a,b} \log \mathbf{Y}_{a,b} + (1 - \mathbf{X}^*_{a,b}) \log(1 - \mathbf{Y}_{a,b}). \tag{6}$$

Note that Sinkhorn layer also imposes a matching constraint, which encourages the continuous prediction into a doubly stochastic matrix *i.e.*, the convex hull of the matching matrix (a permutation matrix). The proposed quantum model namely QAP-QNN can be learned via backpropagation and gradient descent due to the fact that all the components are differentiable. During the training process, a classical optimizer Adam (Kingma & Ba, 2014) with an initial learning rate of 0.1 is used to find the optimal parameters of quantum circuits, including $\boldsymbol{\beta}, \boldsymbol{\theta}$, and $\boldsymbol{\alpha}$. In the testing, as a common post-processing step, the gap between doubly stochastic matrix $\mathbf{Y}$ and the predicted optimal permutation matrix $\mathbf{Y}^*$ is fulfilled by the Hungarian algorithm (Burkard & Dell'Amico, 2009).

4. Analysis of Permutation Invariance

We further elaborate on how our circuit guarantees permutation invariance, with a concrete example of a graph consisting of four nodes labeled as a, b, c, and d. If the input order is $abcd$, the output state after the encoding layer U_{abcd} should be $|\psi_{abcd}\rangle = U_{abcd}|0\rangle^{\otimes 4}$ (let $|0\rangle^{\otimes 4}$ be the initial state). If the order is changed to $cbad$, the encoding layer will also change accordingly, resulting in $|\psi_{cbad}\rangle = U_{cbad}|0\rangle^{\otimes 4}$. The permutation invariance means $|\psi_{cbad}\rangle = SWAP_{1,3}|\psi_{abcd}\rangle$, which refers to the fact that when the input node order changes, the previously output qubit results will also change accordingly. This means the unitary matrix of the quantum circuit is required to satisfy:

$$U_{cbad} = SWAP_{1,3} U_{abcd}, \tag{7}$$

where $SWAP_{1,3}$ is a $2^n \times 2^n$ unitary as a swap gate on qubit 1 and qubit 3. Furthermore, the encoding layer can be decomposed into the unitary acting on the single-qubit and two-qubits: $U_{abcd} = U^S_{abcd} U^D_{abcd}$. For sigle qubit gates, we can easily find $U^S_{cbad} = SWAP_{1,3} U^S_{abcd}$. For example, we choose the RY gate as the single qubit gate, then

$$\begin{aligned} &RY(\theta_c) \otimes RY(\theta_b) \otimes RY(\theta_a) \otimes RY(\theta_d) \\ &= SWAP_{1,3}(RY(\theta_a) \otimes RY(\theta_b) \otimes RY(\theta_c) \otimes RY(\theta_d)). \end{aligned}$$

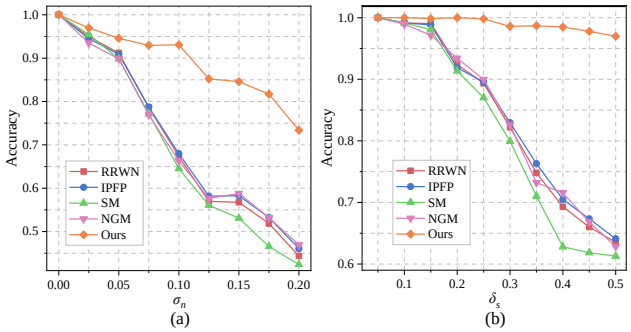


Figure 7. Accuracy of graph matching using the synthetic dataset with varying deformation level $\sigma_n(a)$ and $\delta_s(b)$, comparing with classic GM methods: RRWM, IPFP, SM, NGM.

Therefore, we can derive $U^D_{abcd} = U^D_{cbad}$, which indicates the two-qubit gates need to be selected carefully. Specifically, suppose that there are three edges: $\{a,b\},\{b,c\},\{c,d\}$, and the unitary matrix is defined: $U^W_{abcd} = (W(\theta_{a,b}) \otimes I \otimes I)(I \otimes W(\theta_{b,c}) \otimes I)(I \otimes I \otimes W(\theta_{c,d}))$, where $W(\theta_{a,b})$ is the two-qubit parametric gate encoded the information of edge $\{a,b\}$. However, the order in which two-qubit gates act can be changed arbitrarily, and it means that these three elements must be commutative. In our model design, we have chosen R_{ZZ} as the two-qubit gate, which is a diagonal matrix, and R_{ZZ} tensor product with I is also a diagonal matrix. Diagonal matrices are commutative with each other.

Note the quantum perceptron layer also satisfies the permutation invariance property. Suppose the unitary matrix of a perceptron module is U_{cp}. For a graph with four nodes, the perceptron layer can be represented as $U_p = U_{cp} \otimes U_{cp} \otimes U_{cp} \otimes U_{cp}$, and we can see the unitary matrix is independent of the order of the nodes because all nodes use the same U_{cp}. Similar to quantum perceptron layer, the auxiliary constraint layer is also independent of the order of the input nodes and preserve permutation invariance.

5. Experiments

We solve two popular forms of QAP: graph matching (GM) and Traveling Salesman Problem (TSP), in comparison to both classic and quantum methods. All the experiments are performed on a workstation with a single machine with four physical CPUs with 224 cores Intel(R) Xeon(R) Platinum 8276 CPU @ 2.20GHz, and a GPU (NVIDIA A100). Source code is written using TorchQauntum (Wang et al., 2022a), which is a Pytorch-based library for quantum computing. By its virtue, we can scale up the simulation of 20+ qubits with our GPU, with batch size 16 for training.

5.1. Case Study I: Graph Matching

Problem definition. Graph matching plays a vital role in computer vision, pattern recognition, and bioinformatics fields (Zhang et al., 2019). The problem seeks to establish node correspondences between two graphs, according to the node-to-node and edge-to-edge affinity (Cho et al., 2010). The difference between graph matching and point-based matching (Zhang, 1994) is that the latter does not consider edge information. In this case study, we only discuss the two-graph matching problem, which can be written as the form of QAP. Given that two weighted graph $G_s = (V_s, E_s)$ and $G_t = (V_t, E_t)$, where V is the node set and $E \subseteq V \times V$ is the edge set. Suppose that the number of nodes $|V_s| = n_1$, $|V_s| = n_2$, and we only consider the case of $n_1 = n_2$ in this experiment. The permutation binary matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_1}$ of Eq. 1 encodes the node-to-node correspondence, *i.e.*, $\mathbf{X}_{ia}$ indicates whether node V_t^i in G_v matches node V_s^a in G_t. Two-graph matching can be translated into an association graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, as illustrated in Fig. 1. Then, the problem of two-graph matching can be viewed as the vertex classification problem of the association graph. The vertices of the association graph $\mathcal{V} = V_s \times V_t$, and the vectorized assignment matrix vec$(\mathbf{X})$ is equivalent to the vertex set of the association graph. The edges of the association graph $\mathcal{E}$ denote the agreement between two pairs of correspondence, such as $\mathcal{E}^{ia,jb} = (E_s^{ij}, E_t^{ab}) = \{(V_s^i, V_t^j), (V_s^a, V_t^b)\}$. Therefore, the off-diagonal part of affinity matrix $\mathbf{K} \in \mathbb{R}^{n_1 n_1 \times n_1 n_1}$ of Eq. 1 can be viewed as the adjacency matrix of the association graph, representing the edge-to-edge similarity between G_v and G_s. In addition, the diagonal elements of $\mathbf{K}$ are assigned as the node-to-node affinity between G_v and G_s.

Datasets. We perform the experiment on synthetic 2-D point sets following the protocol of (Wang et al., 2022b). The dataset is established to match the 2-D random point sets. First of all, we build 10 sets of ground truth points distributed in the plane $U(0,1) \times U(0,1)$, where U means the uniform distribution. Then, each set of synthetic points is distorted by the random scaling from $U(1-\delta_s, 1+\delta_s)$ and additive random noise $N(0, \sigma_n^2)$, where N is the Gaussian noise function. The distorted points are used to construct a reference graph, and a set of ground truth points form a target graph. For each target graph, we sample 320 reference graphs for training and 100 for testing. In addition, the target graph is fully connected and the reference graph is constructed employing Delaunay triangulation. The affinity matrix $\mathbf{K}$ is computed by $\mathbf{K}_{ia,jb} = \exp(-(f_{ij} - f_{ab})^2/\gamma^2)$, where f_{ij} represents the edge length or node coordinate information, and γ is empirically set as 1e-2.

Results. We first compare the performance between the proposed method and the classical learning-free and learning-based methods that are designed for graph matching, including 1) RRWM (Reweighted Random Walk Matching) (Cho et al., 2010) adopts random-walk to match nodes in two different graphs by reweighting edges based on the similarity of matched nodes; 2) IPFP (Integer Projected Fixed Point method) (Leordeanu et al., 2009) finds the optimal match-

Table 1. Comparison of the optimal gap and runtime between quantum-based methods under different deformation levels.

	QAOA	C-QAOA	**QAP-QNN** (m=1)	**QAP-QNN** (m=2)
$\delta_s = 0.4, \sigma_n = 0$	4.211%	3.419%	0.0%	0.0%
$\delta_s = 0.3, \sigma_n = 0.1$	13.21%	4.236%	3.153%	2.879%
$\delta_s = 0.2, \sigma_n = 0.2$	4.307%	5.941%	3.481%	0.521%
$\delta_s = 0.1, \sigma_n^2 = 0.3$	6.325%	2.729%	0.270%	0.045%
Runtime (seconds)	2507.3	120.34	0.1981	0.6284

ing of nodes between two graphs by iteratively updating the matching based on integer projection; 3) SM (Spectral-technique Matching) (Leordeanu & Hebert, 2005) considers graph matching as discovering graph cluster by spectral numerical technique; 4) NGM (Neural Graph Matching network) (Wang et al., 2022b) utilizes graph convolution network with matching-aware embedding modules to learn the correspondences. All comparing methods are implemented by using Pygmtools (Python Graph Matching Tools) [1]. In addition, the number of qubits of quantum simulation is limited to the memory of machines. Therefore, in this experiment, we generate a small-scale dataset that is used for matching two graphs with four nodes, *i.e.*, there need 16 qubits to represent vertices of the association graph.

Fig. 7 shows the evaluation results on graph matching, where the horizontal axis indicates the level of scale σ_n and noise δ_s deformation of datasets. From the results, we see that the proposed method outperforms other competitors in both scale and noise deformation. Moreover, our method can better cope with scale distortion than noise distortion. The promising performance may benefit from the huge Hilbert space constructed by the quantum circuit and the power of supervised learning. The inferior performance of the learning-based classical method NGM may lie in the model being too complicated for the small-scale dataset.

Next, we compare the performance and runtime between quantum-based methods, as shown in Table 1. In this experiment, the adopted synthetic dataset is built for two graphs with three nodes, namely, the number of variables of QAP is 9. We choose four evenly distributed noise levels to evaluate the optimal gap of the objective scores of different methods. The reason for adopting the optimal gap as a metric is that QAOA is an approximate algorithm and its solutions may not satisfy the constraint conditions. Moreover, we test our method in the case of $m = 1$ and $m = 2$, where m represents the number of qubits for each vertex. The larger m, the more network parameters, and the better the learning ability of the network. As we can see in Table 1, the case of $m = 2$ achieves better performance than the case of $m = 1$, but runtime increases accordingly. Therefore, m should be selected according to the trade-off between performance and runtime in practice. In addition, we can see that the performance of C-QAOA is better and faster than QAOA since QAOA is hard to optimize when it solves constrained problems by adding penalty terms. Moreover, QAOA and C-QAOA are slower than our training-based model due to the fact that they need to employ the classical optimizer to optimize specific instances. From this experiment, we can see that the proposed method can surpass QAOA and C-QAOA in terms of efficiency and effectiveness.

[1] https://pygmtools.readthedocs.io/en/latest/index.html

Table 2. Performance of our technique compared to classical baselines and quantum methods for various TSP instance sizes.

Method	TSP-4		TSP-5	
	Tour Len.	Opt. Gap.	Tour Len.	Opt. Gap.
Concorde	1.7075	0.00%	2.1180	0.00%
Nearest Ins.	1.7075	0.00%	2.1605	1.98%
Random Ins.	1.7075	0.00%	2.1180	0.00%
Farthest Ins.	1.7075	0.00%	2.1313	0.63%
QAOA	1.8167	6.47%	2.3594	11.4%
C-QAOA	1.7507	2.53%	2.2997	8.58%
QAP-QNN (Ours)	1.7205	0.76%	2.2574	6.30%

5.2. Case Study II: Traveling Salesman Problem

Problem definition. The Traveling Salesman Problem (TSP) is a well-known combinatorial optimization problem: given a set of cities and the distances between each pair of cities, find the shortest possible route that visits each city exactly once and returns to the starting city. TSP can be formulated as a Quadratic Assignment Problem (Goh et al., 2022) by defining a set of variables that represent the order in which the cities are visited. Specifically, the variable $X_{v,j}$ is an indicator variable that the city v is the j-th city to be visited, and there are n_1^2 variables for an n_1-city instance.

$$\begin{aligned} &\min \sum_{(u,v)\in E} \sum_{j=1}^{n_1} d_{uv} \mathbf{X}_{u,j} \mathbf{X}_{v,\text{mod}(j+1,n_1)} \\ &s.t. \quad \mathbf{X}\mathbf{1}_{n_1} = \mathbf{X}^\top \mathbf{1}_{n_1} = \mathbf{1}_{n_1},\ \mathbf{X} \in \{0,1\}^{n_1 \times n_1}, \end{aligned} \tag{8}$$

where d_{uv} indicates the distance between city u and city v, and mod$(j+1, n_1)$ means the $(n_1$+1)-th city of the tour is the first city. The constraint implies a city can only be visited once during the tour. According to the above formulation, we can translate TSP into the form of QAP.

Datasets and Metrics. In this experiment, we focus on the 2D Euclidean TSP. For each TSP instance, the n-node locations are sampled uniformly at random in the unit square $S = \{x_i\}_{i=1}^n$ where each $x_i \in [0,1]^2$. We employ Concorde TSP solver (David et al., 2006) to find the optimal tour as ground-Truth tours. For each trial, we sample 3200 (1000) pairs of problem instances and solutions as the training set (testing set). The optimality gap is an evaluation metric, which is defined as the average percentage ratio of predicted tour length l^{TSP} relative to the optimal solution $\hat{l}^{TSP}$ over

Table 3. Ablation study. The optimal gap of the TSP-4 is tested to study the effect of encoding and auxiliary constraint layers.

Repeated Encoding	Auxiliary Constraint	Optimal Gap
✓		2.46%
	✓	12.56%
✓	✓	0.76%

Table 4. Optimal gap of TSP-4 dataset by different layer numbers.

Layer	**2**	**4**	**6**	**8**	**10**
Optimal Gap	7.32%	4.21%	0.76%	0.44%	1.01%

the test instances, computed as $\frac{1}{m}\sum_{i=1}^{m}(l^{TSP}/\hat{l}^{TSP}-1)$.

Evaluation and Results. We generate two TSP datasets with $n_1 = 4$ (TSP-4) and $n_1 = 5$ (TSP-5), respectively (for the shortage of computing resources since the requirement for qubits is n_1^2). The tour length and optimal gap of all methods are reported in Table 2. The compared classical methods include Nearest insertion, Random insertion, as well as Farthest insertion. They can be regarded as a different insertion heuristic algorithm, which represents a partial solution as a tour, and extends it by inserting one node at a time in a different way. For TSP-4, all the classical methods achieve the best result that is consistent with the Concorde solver, but the quantum-based methods are inferior to classical methods. This may be because TSP-4 is easy for classical heuristics designed for routing problems but a bit more involved when TSP is translated into the formulation of QAP. In the TSP-5 dataset, the optimal gap of both classical heuristic and quantum-based methods becomes larger. Nevertheless, compared with other quantum-based methods, our model can outperform them on both datasets.

In addition, we conduct an ablation study on the TSP-4, as reported in Table 3. It shows that if we only encode graph information in the first block, then the performance will significantly drop, which manifests that the repeated encoding layer is crucial. This is because the repeated encoding layer plays an important role in the interaction of information between vertices. Moreover, the performance will degrade when the auxiliary constraint layer is removed in the proposed method, but the decrease was moderate and still within an acceptable range. As the auxiliary constraint layer only serves as a soft constraint, our model still works without it. This is useful if our model is deployed on real quantum devices. We can sacrifice some accuracy by omitting the auxiliary constraint layer to ensure feasibility, considering the issue of multi-qubit gates decomposing into single- and two-qubit gates and causing depth problems.

To explore the impact of the different numbers of layers on the performance of our methods, Table 4 reports the optimal gap under the different numbers of layers. As we can see, the performance shows a trend of first increasing and then decreasing as the number of layers grows, and achieves the best with 8 layers. However, as the number of layers increases, the number of parameters also increases, making the network slower and harder to optimize. Therefore, one needs to make a trade-off between performance and speed by choosing the number of layers.

Table 5. Performance changes as the level of noise increases, where noise refers to use the bit-phase flip as the readout noise.

Bit-phase flip error	**0**	**0.01**	**0.02**	**0.03**	**0.04**	**0.05**
Precision	0.845	0.832	0.813	0.801	0.797	0.780

5.3. Study on the Effect of Simulated Noise

There are many types of noise on quantum devices, such as bit flip and phase flip, depolarization, amplitude damping, and phase damping. In this subsection, we test different levels of bit-phase flip as readout noise, which is relatively easy to implement in our code to study the performance of our algorithm under simulated noise. Specifically, the experiment uses a graph matching dataset with $\sigma_n = 0.15$ and $\delta_s = 0$ and tests the precision of the graph matching task. We add bit-phase flips at the end of our circuit with a probability ranging from 0 to 0.05. The experimental results are shown in the following Table 5. As the bit-phase flip error increases, our method experiences a certain degree of decline, but the decline is not very sharp and remains within an acceptable range.

6. Conclusion and Outlook

We have presented a quantum neural network with the node permutation invariance for learning to solve constrained CO. The proposed quantum framework encodes the graph information and the matching constraint and shares the same perceptron module on different vertices to learn the representation of each vertex. The parameters of the quantum circuit are optimized by the classic optimizer in a supervised learning manner. Numerical Experiments on the two specific applications of QAP verified the effectiveness of the proposed QAP-QNN model. However, we currently only deal with the QAP form, and it is a long-standing effort for seeking more general solvers for CO even in the classic ML community remains open (Wang et al., 2023), and a quantum mixture integer programming neural solver could be of an interesting direction in the future.

Acknowledgments.

The work was supported in part by National Key Research and Development Program of China (2020AAA0107600), NSFC (62222607) and Science and Technology Commission of Shanghai Municipality (22511105100).

References

Adams, R. and Zemel, R. Ranking via sinkhorn propagation. *arXiv:1106.1925*, 2011.

Arora, S. Polynomial time approximation schemes for euclidean tsp and other geometric problems. In *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 2–11. IEEE, 1996.

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G., Buell, D. A., et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *EJOR*, 2021.

Bhatia, A. S., Saggi, M. K., Kumar, A., and Jain, S. Matrix product state–based quantum classifier. *Neural computation*, 31(7):1499–1517, 2019.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

Burkard, R. and Dell'Amico, S. Martello assignment problems. *SIAM, Society for Industrial and Applied Mathematics: Philadelphia, PA, USA*, 2009.

Cerf, N. J., Grover, L. K., and Williams, C. P. Nested quantum search and np-complete problems. *arXiv preprint quant-ph/9806078*, 1998.

Cho, M., Lee, J., and Lee, K. M. Reweighted random walks for graph matching. In *ECCV*, pp. 492–505. Springer, 2010.

David, A., Robert, B., Vasek, C., and William, C. Concorde tsp solver, 2006. URL https://www.math.uwaterloo.ca/tsp/concorde.html.

Farhi, E., Goldstone, J., and Gutmann, S. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

Goh, S. T., Bo, J., Gopalakrishnan, S., and Lau, H. C. Techniques to enhance a qubo solver for permutation-based combinatorial optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2223–2231, 2022.

Grover, L. K. A fast quantum mechanical algorithm for database search. In *STOC*, 1996.

Hadfield, S., Wang, Z., O'gorman, B., Rieffel, E. G., Venturelli, D., and Biswas, R. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, 2019.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Khairy, S., Shaydulin, R., Cincio, L., Alexeev, Y., and Balaprakash, P. Learning to optimize variational quantum circuits to solve combinatorial problems. In *AAAI*, volume 34, pp. 2367–2375, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Larocca, M., Sauvage, F., Sbahi, F. M., Verdon, G., Coles, P. J., and Cerezo, M. Group-invariant quantum machine learning. *PRX Quantum*, 3(3):030341, 2022.

Leordeanu, M. and Hebert, M. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, volume 2, pp. 1482–1489. IEEE, 2005.

Leordeanu, M., Hebert, M., and Sukthankar, R. An integer projected fixed point method for graph matching and map inference. *NeurIPS*, 22, 2009.

Mernyei, P., Meichanetzidis, K., and Ceylan, I. I. Equivariant quantum graph circuits. In *ICML*, pp. 15401–15420, 2022.

Nguyen, Q. T., Schatzki, L., Braccia, P., Ragone, M., Coles, P. J., Sauvage, F., Larocca, M., and Cerezo, M. Theory for equivariant quantum neural networks. *arXiv preprint arXiv:2210.08566*, 2022.

Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., and Obrien, J. L. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):1–7, 2014.

Preskill, J. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

Ragone, M., Braccia, P., Nguyen, Q. T., Schatzki, L., Coles, P. J., Sauvage, F., Larocca, M., and Cerezo, M. Representation theory for geometric quantum machine learning. *arXiv preprint arXiv:2210.07980*, 2022.

Schuld, M., Sweke, R., and Meyer, J. J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3): 032430, 2021.

Srinivasan, K., Satyajit, S., Behera, B. K., and Panigrahi, P. K. Efficient quantum algorithm for solving travelling salesman problem: An ibm quantum experience. *arXiv preprint arXiv:1805.10928*, 2018.

Wang, H., Ding, Y., Gu, J., Lin, Y., Pan, D. Z., Chong, F. T., and Han, S. Quantumnas: Noise-adaptive search for robust quantum circuits. In *HPCA*, pp. 692–708, 2022a.

Wang, R., Yan, J., and Yang, X. Learning combinatorial embedding networks for deep graph matching. In *ICCV*, pp. 3056–3065, 2019.

Wang, R., Yan, J., and Yang, X. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE TPAMI*, 2022b.

Wang, R., Shen, L., Chen, Y., Yang, X., Tao, D., and Yan, J. One-shot neural combinatorial optimization solvers: Theoretical and empirical notes on the cardinality-constrained case. In *ICLR*, 2023.

Wang, R., Yan, J., and Yang, X. Combinatorial learning of robust deep graph matching: an embedding based approach. *IEEE TPAMI*, 2023.

Zhang, Z. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 13(2):119–152, 1994.

Zhang, Z., Xiang, Y., Wu, L., Xue, B., and Nehorai, A. Kergm: Kernelized graph matching. *NeurIPS*, 32, 2019.

Zheng, H., Li, Z., Liu, J., Strelchuk, S., and Kondor, R. On the super-exponential quantum speedup of equivariant quantum machine learning algorithms with SU (d) symmetry. *arXiv preprint arXiv:2207.07250*, 2022.