Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A hybrid quantum-classical neural network for learning to route

Marcus Rolf Peter Ritt
Universidade Federal do Rio Grande do Sul
Alexsandro Santos da Rosa Júnior
Instituto de Pesquisas Eldorado
Marcos Vinicius Reballo
Instituto de Pesquisas Eldorado
Cesar Augusto Amaral
cesar.amaral.BE@eldorado.org.br
Universidade Federal de Santa Catarina
Fernando Augusto Caletti Barros
Instituto de Pesquisas Eldorado

Research Article

Keywords:

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11538593/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A hybrid quantum-classical neural network for learning to route

Marcus Rolf Peter Ritt[1,3*], Alexsandro Santos da Rosa Júnior[1], Marcos Vinicius Reballo[1], Cesar Augusto do Amaral[1,2], Fernando Augusto Caletti de Barros[1]

[1*]Instituto de Pesquisas Eldorado, Porto Alegre, RS, Brazil.
[2]Departamento de Física, Universidade Federal de Santa Catarina, Florianópolis, 88040-900, SC, Brazil.
[3]Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil.

*Corresponding author(s). E-mail(s): marcus.ritt.BE@eldorado.org.br;
Contributing authors: alexsandro.junior@eldorado.org.br;
marcos.reballo@eldorado.org.br; cesar.amaral.BE@eldorado.org.br;
fernando.barros@eldorado.org.br;

Abstract

We study hybrid quantum-classical neural networks for learning routing heuristics. Specifically, we ask whether small quantum neural networks can replace parameter-heavy modules inside a well-established attention-based routing model while maintaining solution quality. We consider the capacitated vehicle routing problem (CVRP) and its variant with time windows (CVRPTW), evaluating instances with 10 to 100 customers. We investigate four-qubit quantum neural networks (QNNs) as replacements for encoder feed-forward modules and attention output projections, comparing them against width-matched classical bottlenecks. For these problems, encoder feed-forward replacement emerges as the most promising design, reducing the number of model parameters by 56.6% relative to the original classical architecture. Across matched comparisons, the QNN bottlenecks retain solution quality close to their classical counterparts, with differences in mean route length of approximately 0.5% or less under feed-forward replacement. Replacing the attention output projections as well increases parameter reduction to 63.2%, but leads to larger performance gaps for larger instances. The QNNs do not systematically outperform the classical controls. These results

identify encoder feed-forward replacement as a viable hybrid-module compression strategy for neural combinatorial optimization, without demonstrating a quantum-specific advantage.

Keywords: Vehicle routing, hybrid quantum-classical neural networks, neural combinatorial optimization, quantum machine learning.

1 Introduction

There is ongoing research on solving combinatorial optimization problems by learning, in particular by training neural networks to solve them from sampled instances instead of designing problem-specific algorithms manually. Current neural architectures for this task, however, have too many parameters to be replaced entirely by quantum neural networks on near-term devices.

This paper studies whether hybrid quantum-classical solutions can achieve good performance with fewer parameters, and thereby point to possible ways of replacing selected components of classical neural networks by quantum versions. This work focuses on the classical capacitated vehicle routing problem (CVRP), a strongly NP-hard routing problem with direct practical relevance [1].

Classical approaches to the CVRP range from exact branch-and-cut formulations to a wide variety of constructive heuristics, local-search methods, and metaheuristics [2, 3]. In parallel, there is growing interest in quantum optimization methods such as QAOA and VQE [4, 5], which, when applied to combinatorial optimization, can use QUBO formulations mapped to Ising Hamiltonians and solve them as variational hybrid quantum-classical programs. More recently, quantum machine learning (QML) methods have been proposed, in which parametric quantum circuits act as learnable components inside larger models. However, current quantum devices and simulators severely limit circuit width and depth, which makes it difficult to replace entire neural architectures for routing. This work therefore focuses on a narrower question: whether small QNN modules can replace parameter-heavy subcomponents of competitive classical neural models while keeping most of their solution quality.

This paper makes three main contributions. First, it develops hybrid attention models for the CVRP and CVRPTW in which small QNNs replace selected parameter-heavy encoder modules. Second, it compares these QNN modules with parameter-matched classical bottlenecks, separating the effect of quantum processing from that of compression alone. Third, it evaluates the resulting policies alongside established classical routing methods on fixed held-out instances. The single-compression hybrid reduces the model size by more than 56% while retaining solution quality close to its matched classical neural baseline.

The rest of this paper is organized as follows. Section 2 introduces the vehicle routing problem. Section 3 discusses related work. Section 4 presents the proposed hybrid quantum-classical neural network. Section 5 describes the experimental methodology. Section 6 reports the experimental results. Section 7 concludes the paper.

2 The Vehicle Routing Problem

The CVRP can be defined as follows. Let $V = \{0, 1, \ldots, n\}$ be the set of nodes, where 0 represents the depot and $N = \{1, \ldots, n\}$ denotes the customers. The distance between nodes i and j is d_{ij}, and q_i is the demand of customer i, with $q_0 = 0$. A homogeneous fleet of vehicles is available, each with capacity Q; the number of vehicles used is not fixed in advance but determined by the solution. The objective is to serve all customers with a set of routes, each starting and ending at the depot, such that every route respects the vehicle capacity and the total travelled distance of all routes is minimized.

In the capacitated vehicle routing problem with time windows (CVRPTW), each customer i additionally has a service time $p_i \geq 0$ and a time window $[a_i, b_i]$ during which service must start, and the depot imposes a scheduling horizon $[0, H]$ within which every vehicle must return. A vehicle may arrive at a customer before a_i and wait, but service cannot start before a_i nor after b_i. Setting $p_i = 0$ and $[a_i, b_i] = [0, H]$ for all i, with a sufficiently large H, recovers the plain CVRP as a special case.

Introducing a binary decision variable x_{ij}, which assumes 1 if a vehicle travels directly from node i to node j and 0 otherwise, both problems can be captured by a single compact arc-based formulation. Degree and depot-balance constraints fix the route structure, a continuous load variable u_i enforces the capacity restriction and eliminates subtours, and an optional continuous time variable t_i, present only when time windows apply, propagates schedules along each route:

$$\min \sum_{\substack{(i,j)\in V^2 \\ i\neq j}} d_{ij} x_{ij}, \tag{1}$$

$$\text{s.t.} \sum_{j\in V\setminus\{i\}} x_{ij} = \sum_{j\in V\setminus\{i\}} x_{ji} = 1, \qquad \forall i \in N, \tag{2}$$

$$\sum_{j\in N} x_{0j} = \sum_{j\in N} x_{j0}, \tag{3}$$

$$u_j \geq u_i + q_j - Q(1 - x_{ij}), \qquad \forall i, j \in N,\ i \neq j, \tag{4}$$

$$q_i \leq u_i \leq Q, \qquad \forall i \in N, \tag{5}$$

$$t_j \geq t_i + p_i + d_{ij} - M_{ij}(1 - x_{ij}), \qquad \forall i, j \in N,\ i \neq j, \tag{6}$$

$$t_j \geq d_{0j} - M_{0j}(1 - x_{0j}), \qquad \forall j \in N, \tag{7}$$

$$t_i + p_i + d_{i0} \leq H + M_{i0}(1 - x_{i0}), \qquad \forall i \in N, \tag{8}$$

$$a_i \leq t_i \leq b_i, \qquad \forall i \in N, \tag{9}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in V,\ i \neq j. \tag{10}$$

Here, (1) minimizes total distance; (2) ensures that each customer is entered and left exactly once; (3) balances the number of vehicles leaving and returning to the depot, leaving the fleet size free; (4)–(5) are Miller–Tucker–Zemlin-style load-propagation constraints, where u_i tracks the cumulative load carried up to and including customer i, so that they enforce capacity feasibility along each route and, being strictly increasing along any used arc, also rule out subtours that do not include the depot; and (10)

enforces integrality. Constraints (6)–(9), included only for CVRPTW instances, are the time-window analogue of (4)–(5): the variable t_i denotes the service start time at customer i, propagated forward along each route by (6), with (7) handling the first leg from the depot and (8) the return leg, while (9) restricts t_i to the customer's time window. The constants M_{ij} are big-M values, e.g., $M_{ij} = H + \max_k p_k + \max_{k,l} d_{kl}$, chosen large enough to deactivate the corresponding constraint whenever $x_{ij} = 0$.

Because (4)–(5) and (6)–(9) both replace an exponential family of cuts by a compact set of big-M constraints, this formulation is polynomial in size for either problem. Its linear relaxation is, however, considerably weaker than that of the classical rounded-capacity-cut formulations solved via branch-and-cut [2]; here, it serves primarily to give a precise, self-contained problem definition and to provide exact baselines at the small and medium instance sizes considered for the learning approach, rather than as a candidate for large-scale exact solving.

3 Related work

There is a large body of work on learning to solve combinatorial optimization problems, including constructive, search-based, and end-to-end approaches. A recent survey is given by Chung et al. [6]. A related direction explores large language models (LLMs) as end-to-end solvers that map a natural-language problem description directly to a solution, bypassing intermediate code generation or solver calls. Jiang et al. [7] fine-tune a 7B-parameter LLM, first by supervised learning and then by reinforcement learning, to generate solutions for seven combinatorial problems. On graph instances with at most 100 nodes and scheduling instances with at most 20×20 operations, best-of-eight decoding attains average gaps of 1.03–8.20%, outperforming much larger general-purpose LLMs and several lightweight constructive heuristics in solution quality, but requiring longer inference times than the latter.

Among these specialized, reinforcement-learning-trained architectures, *constructive* solvers build a route incrementally, one node at a time. For routing problems, this line of work goes back to sequence-to-sequence learning and pointer networks [8], which Bello et al. [9] first paired with a REINFORCE-trained policy gradient to construct tours directly from tour length, without labelled optimal solutions, and includes reinforcement-learning approaches for the CVRP such as Nazari et al. [10].

Kool et al. [11] replace this recurrent pointer-network architecture with a fully attention-based encoder-decoder: an encoder embeds all nodes in parallel through several self-attention and feed-forward layers, and a decoder attends, from a single context vector updated as the tour is built, to these node embeddings to select the next node at each step, with the whole model trained end-to-end using REINFORCE and a greedy-rollout baseline.

The attention model (AM) has inspired several variants. Lei et al. [12] add edge embeddings and residual connections in a residual edge-graph attention network, shifting a larger share of the computation into the encoder. Luo et al. [13] move complexity into the decoder instead, with a "light encoder, heavy decoder" (LEHD) design whose heavier multi-layer decoder re-embeds only the first and last node of the partial tour at each step, letting a model trained on small instances generalize to ones an order of

magnitude larger. Fang et al. [14] restrict each decision to several fixed-size k-nearest-neighbour views and normalize their coordinates, so that a model trained at $n = 100$ never processes the complete large instance at once. They evaluate this approach up to $n = 10{,}000$ for TSP and $n = 5{,}000$ for CVRP. Chen et al. [15] retain this local decomposition and optimize the coordinate projection used at test time, reporting experiments with up to 100,000 nodes. In the latter experiments, each neural call considers only $k = 100$ nearby nodes; the reported scale refers to the surrounding instance, which is still traversed through $O(n)$ sequential decisions. Thus, scalability comes primarily from imposing a fixed local candidate set, while the learned model ranks candidates within that restricted view. These variants push complexity to opposite ends of the model: Lei et al. [12] into the encoder, the LEHD family into the decoder. Since the QNN replacement studied here specifically targets the encoder, the encoder-light, decoder-heavy LEHD family is harder to adapt to it, and Lei et al.'s heavier encoder would make the replacement more costly still—which is why this paper builds on the original, more balanced AM architecture instead.

A separate line of work keeps the constructive setup but targets constraints for which the usual feasibility mask is itself hard to compute. Bi et al. [16] study routing variants with time windows and draft limits, where checking whether a candidate move keeps the rest of the route feasible is NP-hard in general; rather than an exact mask, they score violations with a Lagrangian penalty term and add a proactive infeasibility-prevention (PIP) step that looks one move ahead and forbids actions that would make an unvisited customer infeasible later in the same route, with a learned variant (PIP-D) that keeps this lookahead cheap. Bi et al. [17] build on this with a lightweight refinement stage: a handful of learned local-search moves repair remaining violations after construction, and successful repairs are then fed back as imitation targets so the constructor itself gradually absorbs them. Because we assume an unlimited homogeneous fleet, a customer that cannot be added to the current route can still be served by a newly opened route. The specific PIP mechanism of Bi et al. [16] is therefore unnecessary in our setting. Their broader point that some constraints are cheap to mask while others require dedicated architectural or training support is a useful complement to the parameter-efficiency question addressed in this paper.

A complementary family of methods decouples solution construction from a learned edge heatmap, using the heatmap to guide a generic search procedure such as Monte Carlo tree search. These methods differ mainly in how that heatmap is obtained: supervised graph-convolutional learning from sampled subgraphs [18], REINFORCE learning of sparse edge preferences followed by active search [19], graph diffusion [20], or unsupervised optimization of a surrogate objective [21]. All scale to instances with $n = 10K$ nodes or more. However, Pan et al. [22] show, in a systematic re-evaluation of this heatmap-plus-MCTS pipeline, that once the MCTS/k-opt search itself is properly tuned, the marginal contribution of the learned heatmap is small and inconsistent: at $n = 10K$, the best result in their comparison has a relative deviation of only 2.13% from the optimal tour length, and comes not from a learned model but from a fixed, parameter-free heatmap derived from nearest-neighbour rank. Xia et al. [23] reach the same conclusion independently: their own parameter-free heatmap, a plain softmax over distances, already rivals or beats these learned ones, and the whole

heatmap-plus-search paradigm is in turn beaten by the classical LKH-3 heuristic. This indicates that, in this family of methods, the strong instance-agnostic local search rather than the learned component accounts for most of the solution quality. Quantum-enhanced attention models have also been investigated for vehicle routing. Correll et al. [24] integrate quantum orthogonal neural networks into the attention mechanism of a reinforcement-learning model for supply-chain logistics. In contrast, we investigate replacing encoder feed-forward modules and attention output projections with small variational QNNs, comparing them against width-matched classical bottlenecks for CVRP and CVRPTW.

4 A hybrid quantum-classical neural network

Our approach takes inspiration from Kool et al. [11], who proposed a classical neural network for learning to route two-dimensional Euclidean instances. Their model consists of an encoder that takes the positions of the depot and customers, together with the customer demands, and produces a representation for each node. These representations are processed by several layers, each combining an attention mechanism, following the transformer architecture of Vaswani et al. [25], with a feed-forward layer. After the encoder, a decoder, which also employs an attention mechanism, iteratively extends partial routes until complete routes are obtained. For the CVRP, the approach produced good results on problem instances from 20 up to 100 customers, which covers the small-to-medium regime considered in this work.

Figure 1 shows the encoder and Figure 2 shows the decoder. The encoder first embeds each node into a latent space of dimension d_h. Customers use their coordinates and demand, while the depot, whose demand is always zero, uses a separate coordinate projection:

$$h_i^{(0)} = \begin{cases} W^x [x_i \; y_i \; q_i]^\top + b^x, & i \neq 0, \\ W^0 [x_0 \; y_0]^\top + b^0, & i = 0. \end{cases} \tag{11}$$

followed by L layers, each combining a multi-head attention (MHA) sublayer with $M = 8$ heads and a feed-forward (FF) sublayer, both wrapped in a residual connection. Writing $d_k = d_h/M = 16$ for the per-head dimension, and $W_m^Q, W_m^K, W_m^V \in \mathbb{R}^{d_k \times d_h}$ and $W_m^O \in \mathbb{R}^{d_h \times d_k}$ for the per-head query, key, value, and output projections, $m = 1, \ldots, M$,

$$h'_{im} = \sum_{j \in V} \mathrm{softmax}_j \left(\frac{(W_m^Q h_i)^\top (W_m^K h_j)}{\sqrt{d_k}} \right) W_m^V h_j, \tag{12}$$

$$\mathrm{MHA}_i(h) = \sum_{m=1}^{M} W_m^O h'_{im}. \tag{13}$$

Equivalently, the M head outputs may be concatenated into $[h'_{i1}; \ldots; h'_{iM}] \in \mathbb{R}^{d_h}$ and multiplied by the block output matrix $W^O = [W_1^O \; \cdots \; W_M^O] \in \mathbb{R}^{d_h \times d_h}$. This is the encoder attention output projection considered for replacement below. For each layer

$\ell = 1, \ldots, L$,

$$\hat{h}_i^{(\ell)} = \mathrm{BN}_1^{(\ell)}\left(h_i^{(\ell-1)} + \mathrm{MHA}_i\big(h^{(\ell-1)}\big)\right), \tag{14}$$

$$h_i^{(\ell)} = \mathrm{BN}_2^{(\ell)}\left(\hat{h}_i^{(\ell)} + FF\big(\hat{h}_i^{(\ell)}\big)\right), \tag{15}$$

where $FF(x) = W_2\,\mathrm{ReLU}(W_1 x + b_1) + b_2$ is a two-layer feed-forward network with hidden dimension $d_{ff} = 512$. Following Kool et al. [11], each residual addition is followed by an independent batch-normalization layer with learned scale and shift.

The encoder produces a final node embedding $h_i = h_i^{(L)}$ for $i \in V$ as well as a graph embedding $\overline{h} = \sum_{i \in V} h_i / |V|$.

The decoder builds a route by repeatedly attending from a single context query to the node embeddings. At step t, with π_t the last visited node and ρ_t the remaining vehicle capacity, the context query combines the graph embedding with the current state through two separate linear projections that are summed rather than concatenated,

$$q_c = W^Q_{\text{graph}}\,\overline{h} + W^Q_{\text{step}} \begin{bmatrix} h_{\pi_t} \\ \rho_t \end{bmatrix}, \tag{16}$$

This is the CVRP-specific context of Kool et al. [11]: because a route may be reopened from the depot at any time, only the last visited node and the remaining capacity are needed. The query is refined by one further application of the same multi-head attention operator as the encoder, attending over all node embeddings,

$$\tilde{q}_c = \mathrm{MHA}_c(q_c, h), \tag{17}$$

and a final single-head compatibility layer produces masked, tanh-clipped logits over the candidate nodes,

$$u_j = \begin{cases} C \cdot \tanh\left(\dfrac{\tilde{q}_c^{\top} k_j}{\sqrt{d_k}}\right), & j \text{ feasible}, \\ -\infty, & \text{otherwise}, \end{cases} \qquad p_j = \frac{e^{u_j}}{\sum_{j'} e^{u_{j'}}}, \tag{18}$$

where $k_j = W^K h_j$ and $C = 10$ clips the logits, following Kool et al. [11]. Already visited customers, as well as customers whose demand would exceed the remaining capacity, are masked before the softmax. A route is constructed iteratively from p, either greedily or by sampling; when no feasible successor remains, a new route is opened from the depot. This is repeated until all customers are served.

For the CVRPTW variant, time windows are enforced as hard constraints via masking, not as a penalty in the objective. Writing t for the vehicle's current time and π_t for its current location, a candidate customer j is masked as infeasible, in addition to already visited or over-capacity customers, whenever

$$\max(t + d_{\pi_t j},\, a_j) > b_j \qquad \text{or} \qquad \max(t + d_{\pi_t j},\, a_j) + p_j + d_{j0} > H, \tag{19}$$

i.e. whenever service could not start before the due date b_j, or serving j would prevent the vehicle from returning to the depot before the horizon H closes. A vehicle arriving before a_j simply waits, so only late arrival, never early arrival, is restricted. Because the fleet is unlimited, masking a customer out of the current route costs nothing beyond ending that route: every instance is generated so that each customer is individually feasible for a direct depot–customer–depot trip (Section 5), so a fresh route from the depot can always serve whatever remains once the current route closes. A feasible complete solution is therefore always reached without ever violating a window, which is why hard masking, rather than the soft time-window penalties more common elsewhere in the literature (Section 3), is a natural fit for this unlimited-fleet setting.

Kool et al. [11] propose a hidden size of $d_h = 128$ and a hidden layer of size 512 in the feed-forward network. Therefore, the sizes and evaluation counts per forward step of the components are as shown in Table 1. Note that the decoder does not use the usual all-to-all attention mechanism, but rather a one-to-all attention of the context node to all other nodes. This is for performance reasons, since the decoder step must be repeated $n + k$ times until a complete routing solution with k vehicles is produced. Nevertheless, this remains the hottest path, requiring $O(|V|^2)$ attention evaluations over the construction of a complete solution.

Table 1 Component sizes and number of evaluations during the construction of a complete solution in the classical attention model.

Component	#parameters	#evaluations
Encoder		
Initial embeddings	896	$\|V\|$
Attention queries, keys, values	$49{,}152\,L$	$L\|V\|$
Attention output W^O	$16{,}384\,L$	$L\|V\|$
Feed-forward	$131{,}712\,L$	$L\|V\|$
Batch normalization	$512\,L$	$L\|V\|$
Decoder		
Attention layer	$98,432$	$\|V\|^2$
Totals	692,608	$O(\|V\|^2 + L\|V\|)$

For $L = 3$, the feed-forward layers contain 395,136 parameters and therefore account for more than half of the model. The encoder attention output projections contribute another 49,152 parameters. In particular, each feed-forward sublayer maps $128 \to 512 \to 128$ and contains 131,712 parameters, whereas each W^O maps the concatenated 128-dimensional head representation back to 128 dimensions and contains 16,384 parameters. Thus W^O is smaller than the feed-forward sublayer, rather than a 512×512 matrix, but it is still a repeated dense encoder component outside the iterative decoder path.

We therefore study two replacement levels. *Single compression* replaces every encoder feed-forward sublayer by a narrow classical or quantum $128 \to q \to 128$ bottleneck module. *Double compression* additionally replaces every encoder attention output

projection W^O by a module with the same bottleneck. For each level we compare a classical bottleneck with a QNN bottleneck at matched width $q = 4$. As illustrated in Figure 3, the QNN module first projects the d_h-dimensional input to q values, which are encoded through R_X rotations. A trainable unitary $U(\boldsymbol{\theta})$ implements a brickwall ansatz consisting of R_X, R_Y, and R_Z rotations and CNOT entangling gates. The resulting Pauli-Z expectation values are projected back to dimension d_h. Selecting $q \ll d_h$ imposes the desired strong bottleneck while allowing the same module to replace either dense component.

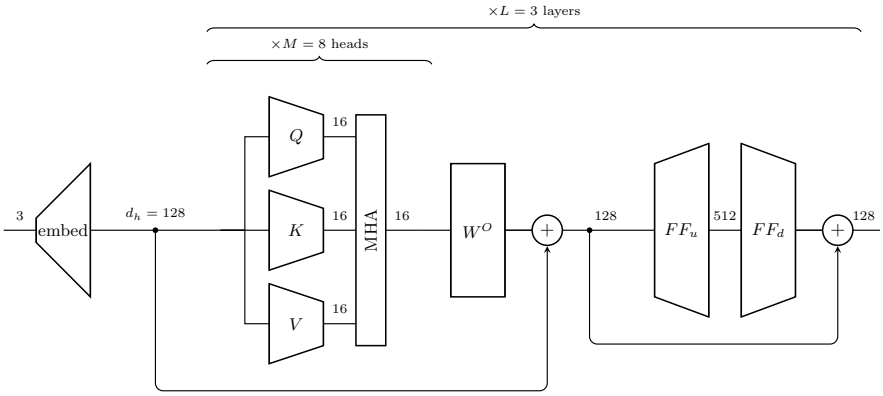

Fig. 1 Encoder architecture of the classical attention model. The figure shows the node embedding and one layer of a total of L layers.

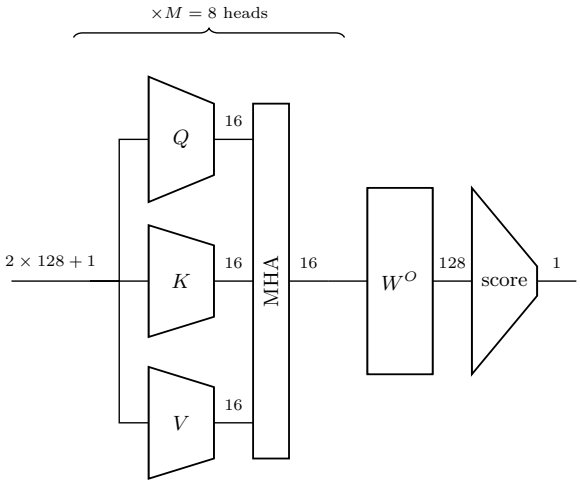

Fig. 2 Decoder architecture of the classical attention model.

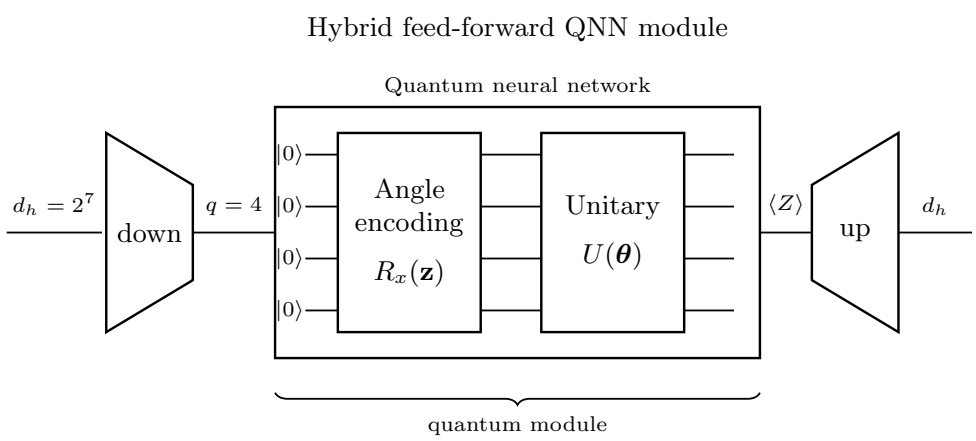

Fig. 3 Hybrid replacement of the encoder feed-forward block.

5 Methodology

5.1 Problem Instances

Training is performed on randomly generated Euclidean instances with varying numbers of nodes $n \in \{10, 20, 50, 100\}$, where node coordinates of the customers and the depot are sampled uniformly from the unit square $[0, 1]^2$. Customer demands are drawn as random integers from $\{1, 2, \ldots, 9\}$, and vehicle capacities are set to 20, 30, 40, and 50 for instances with $n = 10$, 20, 50, and 100 customers, respectively.

For the CVRPTW variant, the same underlying instances are extended with a per-customer service time and time window, following the classical distinction of Solomon [26] into families R1 (tight time windows) and R2 (wide time windows). Every customer has a fixed service time of 0.1, and the scheduling horizon is set to $H = 3$ for R1 instances and $H = 10$ for R2 instances; both values are rescaled from Solomon's original benchmarks to the unit square used here, and chosen large enough that every customer remains individually feasible for a direct depot–customer–depot round trip even when the depot falls near a corner. For each customer i, a window centre is drawn uniformly at random from the interval of times for which such a round trip remains feasible, and a window half-width is drawn from a folded normal distribution, $|\mathcal{N}(0.1, 0.05^2)|$ for R1 and $|\mathcal{N}(0.5, 0.25^2)|$ for R2, following Solomon's original construction; the half-width is lower-bounded at 10^{-3} to avoid degenerate windows and then capped by the largest individually feasible half-width, so that the resulting window $[a_i, b_i]$ stays within $[0, H]$.

5.2 Models and Training Setup

All training and evaluation experiments were performed on a workstation equipped with an Intel Core i7-13700KF processor, 32 GiB of RAM, and an NVIDIA GeForce RTX 3060 GPU with 12 GiB of memory. Neural training and evaluation use CUDA, while the classical routing solvers use Release-mode CPU implementations.

Both the classical and hybrid neural networks are trained using a batch size of 2^7 samples and 2^8 batches per epoch, resulting in an epoch size of 2^{15} samples, and a total of 2^7 training epochs. This setup deviates from the training regime of Kool et al. [11] in two respects. First, the number of samples per epoch is reduced by approximately a factor of 40 compared to the 1,280K samples used there. Second, preliminary experiments indicate that improved performance is often achieved in later epochs, motivating an increase in the total number of training epochs.

Following Kool et al. [11], the policy is trained end-to-end using the REINFORCE algorithm [27], combined with a greedy rollout baseline, which demonstrated superior performance in prior work. Optimization is carried out using the Adam optimizer with a learning rate of 10^{-4}. The QNN components are simulated and trained via back-propagation. A compact QNN configuration is adopted, consisting of $q = 4$ qubits and $L_q = 2$ layers. Each QNN bottleneck contains 1,024 classical projection weights and 24 trainable circuit parameters. With $L = 3$ encoder layers, single compression therefore contains 300,544 classical and 72 quantum parameters, for 300,616 trainable parameters in total. This is a reduction of 391,992 (56.6%) from the 692,608-parameter uncompressed model. Double compression uses six QNN bottlenecks and contains 254,464 classical and 144 quantum parameters, or 254,608 in total, reducing the original model by 438,000 parameters (63.2%). The corresponding matched classical single- and double-compression models contain 300,940 and 254,860 parameters, respectively.

Each model in the broad comparison is trained once, using training seed 1234, following a curriculum over $n = 10 \rightarrow 20 \rightarrow 50 \rightarrow 100$. At each size, the checkpoint with the lowest objective on a fixed set of 1,000 validation instances generated with seed 4321 is selected. Its model and Adam state initialize the next size, while the rollout baseline is rebuilt for the new instance size. Model selection never uses the held-out test instances. Due to the computational cost of training the simulated QNNs, we use a single training seed for each configuration. Consequently, the comparisons characterize performance under a fixed training protocol rather than variability across independent training runs.

The selected models are evaluated using both greedy decoding and stochastic sampling on a fixed held-out set of $N = 1{,}000$ instances generated with seed 9876. For the sampling-based evaluation, the best solution among 1,280 samples is reported.

5.3 Baseline Algorithms

Performance is compared against GOT, the routing solver from Google OR-Tools v9.15 [28], and LKH3, the Lin–Kernighan–Helsgaun heuristic [29, 30]. For plain CVRP, we reproduce the LKH3 protocol released by Kool et al. [11]. Because that implementation leaves the number of vehicles unspecified, LKH3 fixes it to the aggregate-demand lower bound $K = \lceil \sum_i q_i/Q \rceil$. This baseline therefore solves a fixed-minimum-fleet restriction of our unlimited-fleet CVRP. We use one run with 1,000 trials per instance.

For CVRPTW we additionally use Solomon-I1, the classical sequential-insertion heuristic of Solomon [26]. Each route starts from one unrouted seed customer. For a feasible insertion of customer u between consecutive nodes i and j, I1 combines the added distance $c_{11} = d_{iu} + d_{uj} - d_{ij}$ with the resulting delay c_{12} at the successor into $c_1 = \alpha c_{11} + (1 - \alpha)c_{12}$. For each customer, the insertion position minimizing c_1

is retained, and the customer maximizing $c_2 = \lambda d_{0u} - c_1$ is inserted. Thus α trades distance against schedule disruption, while λ controls the preference for customers far from the depot. When no further feasible insertion exists, the route is closed and a new seed is selected. We run all combinations of $\alpha \in \{1, 0.75, 0.5\}$, $\lambda \in \{1, 2\}$, and either farthest-depot or earliest-due-date seeding, retaining the best of the resulting 12 solutions. Since selecting an unrestricted fleet size directly with LKH3 is not scalable, our CVRPTW hybrid first obtains a feasible GOT solution with K_{GOT} routes, then runs LKH3 for 1,000 trials at each of K_{GOT}, $K_{\mathrm{GOT}} + 1$, and $K_{\mathrm{GOT}} + 2$. The best valid solution among these three runs and the original GOT solution is reported as GOT–LKH3.

6 Experimental results

This section reports computational results comparing the proposed learning-based approach with classical algorithms. All reported solutions are explicitly checked for feasibility. For the neural models, decoder masking enforces feasibility by construction, and no invalid routes were observed in our experiments.

Table 2 reports the matched single- and double-compression models under sample-1,280 decoding together with the classical baselines. Every panel uses the same 1,000 seed-9876 held-out instances for all methods. "Obj." is mean route length, $\bar{r}$ is mean instance-wise relative deviation, and t is total evaluation time in minutes.

For each problem family and size, let v_{im} denote the objective obtained by method m on instance i, let I_m be the instances on which that method returns a valid solution, and let $\mu_i = \min_m v_{im}$ over the valid results in the table. We report $\bar{r}_m = 100|I_m|^{-1} \sum_{i \in I_m} (v_{im}/\mu_i - 1)$. Thus each relative-deviation pool contains exactly the methods displayed in the corresponding panel.

We begin with an examination of overall expected objective values. Haimovich and Rinnooy Kan [31] give an asymptotic estimate for the expected optimal CVRP tour length, combining the cost of returning to the depot with the cost of local routing among customers:

$$L_{\mathrm{CVRP}} \approx \frac{2n\bar{q}}{Q}\,\bar{d}_0 + \beta_2\sqrt{n}, \tag{20}$$

where $\bar{q}$ is the expected customer demand, $\bar{d}_0$ is the expected distance between the depot and a customer, and $\beta_2 \approx 0.71$ is the Beardwood–Halton–Hammersley constant [32]. For our instances, both the depot and the customers are drawn uniformly from the unit square, so the expected distance between them is $\bar{d}_0 \approx 0.5214$. With $\bar{q} = 5$ and capacities 20, 30, 40, and 50 for $n = 10$, 20, 50, and 100, respectively, Equation (20) yields estimates of 4.85, 6.65, 11.54, and 17.53. These track the objective values in Table 2 closely across all four instance sizes, suggesting that the observed CVRP objective values lie within the expected range.

At $n = 10$ and 20, the matched classical and quantum bottlenecks differ by at most 0.35%, without a consistent ordering. Single compression remains within approximately 0.5% at every size, with differences changing sign across problem families and sizes. A clearer size-dependent pattern appears under double compression: at $n = 50$ and 100, the QNN objective is higher in all six comparisons, by 0.29%–0.97%. Thus, replacing only the encoder feed-forward blocks preserves the quality of the matched

Table 2 Held-out CVRP and CVRPTW results. Objective values are route lengths in the unit square; $\bar{r}$ is the mean relative deviation from the best displayed method per instance; and t is total evaluation time in minutes. Neural rows use the best of 1,280 samples. GOT–LKH3 reports the best of GOT and three 1,000-trial LKH3 runs at $K_{\mathrm{GOT}}, K_{\mathrm{GOT}}+1, K_{\mathrm{GOT}}+2$. †At CVRP $n = 10$, fixed-minimum-fleet LKH3 returned 993 valid solutions; all other entries contain 1,000.

	$n = 10$			$n = 20$			$n = 50$			$n = 100$		
Method	Obj.	$\bar{r}$ (%)	t (min)	Obj.	$\bar{r}$ (%)	t (min)	Obj.	$\bar{r}$ (%)	t (min)	Obj.	$\bar{r}$ (%)	t (min)
CVRP												
Classical (single)	4.639	1.95	0.2	6.370	2.44	0.4	10.776	3.36	1.4	16.556	4.82	4.2
QNN (single)	4.623	1.65	0.3	6.370	2.45	0.5	10.810	3.67	1.5	16.639	5.33	4.3
Classical (double)	4.625	1.67	0.2	6.362	2.32	0.4	10.810	3.69	1.4	16.630	5.31	4.2
QNN (double)	4.615	1.47	0.4	6.370	2.44	0.6	10.841	3.99	1.6	16.747	6.04	4.4
GOT	4.650	2.31	< 0.1	6.523	4.95	0.1	11.270	8.13	0.3	17.242	9.23	0.9
LKH3 (1K)†	4.561	0.38	1.9	6.221	0.06	7.1	10.428	0.01	22.5	15.797	0.00	40.2
CVRPTW–R1												
Classical (single)	5.903	1.41	0.3	9.413	2.28	0.5	17.927	4.64	1.7	29.316	6.97	5.0
QNN (single)	5.903	1.42	0.4	9.432	2.48	0.6	17.898	4.46	1.8	29.374	7.17	5.2
Classical (double)	5.897	1.32	0.3	9.412	2.28	0.5	17.927	4.66	1.7	29.532	7.76	5.0
QNN (double)	5.896	1.32	0.5	9.413	2.31	0.7	18.036	5.31	1.9	29.702	8.40	5.2
GOT	5.886	1.19	< 0.1	9.325	1.29	0.1	17.609	2.71	0.3	28.367	3.40	1.4
GOT–LKH3 (1K)	5.828	0.11	7.6	9.214	0.09	14.1	17.153	0.06	28.4	27.440	0.03	50.1
Solomon-I1	5.967	2.70	< 0.1	9.630	4.84	< 0.1	18.506	8.27	< 0.1	30.108	10.13	0.1
CVRPTW–R2												
Classical (single)	5.017	2.75	0.2	7.499	4.93	0.5	13.722	7.64	1.6	21.791	10.36	4.7
QNN (single)	5.030	3.05	0.4	7.476	4.61	0.6	13.669	7.22	1.7	21.804	10.43	4.8
Classical (double)	5.001	2.47	0.2	7.452	4.28	0.5	13.639	6.99	1.6	21.902	10.94	4.7
QNN (double)	5.011	2.65	0.5	7.441	4.12	0.7	13.752	7.89	1.8	22.115	12.02	4.9
GOT	5.058	3.66	< 0.1	7.601	6.39	0.1	14.179	11.21	0.3	22.442	13.66	1.0
GOT–LKH3 (1K)	4.882	0.03	9.7	7.146	0.01	13.7	12.755	0.05	32.4	19.757	0.04	63.3
Solomon-I1	5.241	7.66	< 0.1	8.205	15.03	< 0.1	15.427	21.20	< 0.1	24.277	23.15	0.2

classical bottleneck more consistently, whereas also replacing the attention output projections produces a small but consistent degradation in the reported experiments at larger sizes. The additional replacement increases the parameter reduction from 56.6% to 63.2%, but does not improve solution quality.

To assess the robustness of the central single-compression CVRP comparison, we repeated the complete $n = 10 \to 20 \to 50 \to 100$ curriculum for five training seeds. Under sample-1,280 decoding, the paired QNN-minus-classical differences in mean route length range from -0.05% to $+0.15\%$ across sizes; every seed-level difference is within 0.54%. Thus, the practical parity between the feed-forward QNN and its matched classical bottleneck is robust across these five training seeds. This replication concerns CVRP single compression only; the CVRPTW and double-compression rows in Table 2 remain single-seed comparisons.

The specialized classical methods provide a stronger solution-quality reference. Fixed-minimum-fleet LKH3 has the smallest mean CVRP objective at every size. On CVRPTW, the GOT–LKH3 protocol dominates the mean objective and is within 0.12% mean relative deviation of the per-instance pool minimum in every R1 and R2 column. GOT alone is substantially faster but weaker, while Solomon-I1 degrades most rapidly with size. These comparisons do not support solver-dominance or quantum-advantage claims; they delimit the quality retained by the compressed learned policies.

In runtime terms, GOT and Solomon-I1 are inexpensive, neural sample-1,280 decoding takes seconds to a few minutes per test set, and the multi-start GOT–LKH3 protocol is deliberately more computationally intensive in exchange for stronger solutions.

Taken together, these results give a qualified positive answer to the central design question: a four-qubit QNN can replace the encoder feed-forward layers while preserving the performance of the corresponding narrow classical model to within approximately 0.5%, reducing the original model by 56.6%. Extending the replacement to the attention output projections increases the reduction to 63.2%, but is less robust at larger sizes. Crucially, the matched classical bottlenecks achieve essentially the same quality. Most of the parameter reduction therefore comes from the narrow bottleneck itself rather than from quantum processing; the result demonstrates a viable quantum replacement, not a quantum-specific compression advantage.

7 Conclusion

We asked whether small QNN bottlenecks can replace parameter-heavy modules in an attention-based routing policy. Our answer is a qualified yes. Replacing the encoder feed-forward layers with four-qubit QNNs reduces the model by 56.6%, while their objectives remain within approximately 0.5% of those of the corresponding narrow classical models across all tested problem families and sizes. Replacing the attention output projections as well increases the reduction to 63.2%, but produces a consistent degradation at larger sizes in the reported experiments. Single compression is therefore the better-supported design.

The central qualification is that matched classical bottlenecks perform equally well while containing nearly the same total number of parameters. The observed compression consequently comes primarily from imposing a narrow intermediate representation, not from quantum processing. Our results establish that a compact QNN can serve as a viable module within a routing policy, but they do not demonstrate a quantum-specific performance or compression advantage. Specialized classical routing methods also remain stronger solution-quality references.

This outcome is consistent with broader QML evidence that practical advantages remain scarce and often weaken under carefully tuned, capacity-matched classical controls [33–38]. Rather than being exceptional, the present result extends that evidence to a matched comparison of compressed routing policies.

Because the QNN components are simulated, these results concern model quality and parameterization rather than execution speed or resource advantages on quantum hardware. The next question is whether quantum-specific properties—such as inductive bias, sample efficiency, or hardware execution—can provide a benefit unavailable to an equally compact classical component.

Acknowledgments. This work was executed under the TIC26 – Brazil Quantum Camp project, funded within the scope of the Prioritized Informatics Programs and Projects (PPI), Process No. 01245.008254/2025-22, under the responsibility of the Ministry of Science, Technology and Innovation (MCTI), with operational coordination by the Association for the Promotion of Brazilian Software Excellence (SOFTEX),

and executed by CESAR and the Instituto de Pesquisas Eldorado. The authors also want to thank VCI Vanguard Confecções Importadas S.A. (Aramis) for their collaboration on solving practical vehicle routing problems with hybrid approaches.

Declarations

Data Availability. The CVRP and CVRPTW instances used in this study were synthetically generated following the procedures described in Section 5. The generated instances and experimental results are not publicly available due to confidentiality restrictions of the Instituto de Pesquisas Eldorado, but may be obtained from the corresponding author on reasonable request, subject to institutional approval.

Code Availability. The source code and scripts developed for this study are proprietary to the Instituto de Pesquisas Eldorado and are not publicly available. Further details may be obtained from the corresponding author on reasonable request, subject to institutional approval.

Competing Interests. The authors declare no competing interests.

References

[1] Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management Science **6**(1), 80–91 (1959) https://doi.org/10.1287/mnsc.6.1.80

[2] Toth, P., Vigo, D. (eds.): Vehicle Routing: Problems, Methods, and Applications. SIAM, Philadelphia, PA (2014)

[3] Bogyrbayeva, A., Meraliyev, M., Mustakhov, T., Dauletbayev, B.: Machine learning to solve vehicle routing problems: A survey. IEEE Transactions on Intelligent Transportation Systems **25**(6), 4754–4772 (2024)

[4] Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm (2014) https://doi.org/10.48550/ARXIV.1411.4028 arXiv:1411.4028

[5] Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P.J., Aspuru-Guzik, A., O'Brien, J.L.: A variational eigenvalue solver on a quantum processor (2013) https://doi.org/10.1038/ncomms5213 arXiv:1304.3061

[6] Chung, K.T., Lee, C.K.M., Tsang, Y.P.: Neural combinatorial optimization with reinforcement learning in industrial engineering: a survey **58**(5) https://doi.org/10.1007/s10462-024-11045-1

[7] Jiang, X., Wu, Y., Li, M., Cao, Z., Zhang, Y.: Large language models as end-to-end combinatorial optimization solvers (2025) https://doi.org/10.48550/ARXIV.2509.16865 arXiv:2509.16865

[8] Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks (2015) https://doi.org/10.48550/ARXIV.1506.03134 arXiv:1506.03134

[9] Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning (2016) https://doi.org/10.48550/ARXIV.1611.09940 arXiv:1611.09940

[10] Nazari, M., Oroojlooy, A., Snyder, L., Takáč, M.: Reinforcement learning for solving the vehicle routing problem. In: Advances in Neural Information Processing Systems, vol. 31 (2018)

[11] Kool, W., Hoof, H., Welling, M.: Attention, learn to solve routing problems! (2018) https://doi.org/10.48550/ARXIV.1803.08475 arXiv:1803.08475

[12] Lei, K., Guo, P., Wang, Y., Wu, X., Zhao, W.: Solve routing problems with a residual edge-graph attention neural network. Neurocomputing (2021) https://doi.org/10.1016/j.neucom.2022.08.005

[13] Luo, F., Lin, X., Liu, F., Zhang, Q., Wang, Z.: Neural combinatorial optimization with heavy decoder: Toward large scale generalization (2023) https://doi.org/10.48550/ARXIV.2310.07985 arXiv:2310.07985

[14] Fang, H., Song, Z., Weng, P., Ban, Y.: Invit: A generalizable routing problem solver with invariant nested view transformer (2024) https://doi.org/10.48550/ARXIV.2402.02317 arXiv:2402.02317

[15] Chen, Y., Chen, R., Luo, F., Wang, Z.: Improving generalization of neural combinatorial optimization for vehicle routing problems via test-time projection learning (2025) https://doi.org/10.48550/ARXIV.2506.02392 arXiv:2506.02392

[16] Bi, J., Ma, Y., Zhou, J., Song, W., Cao, Z., Wu, Y., Zhang, J.: Learning to handle complex constraints for vehicle routing problems (2024) https://doi.org/10.48550/ARXIV.2410.21066 arXiv:2410.21066

[17] Bi, J., Cao, Z., Zhou, J., Song, W., Wu, Y., Zhang, J., Ma, Y., Wu, C.: Towards Efficient Constraint Handling in Neural Solvers for Routing Problems (2026)

[18] Fu, Z.-H., Qiu, K.-B., Zha, H.: Generalize a small pre-trained model to arbitrarily large tsp instances (2021) https://doi.org/10.1609/aaai.v35i8.16916 arXiv:2012.10658

[19] Qiu, R., Sun, Z., Yang, Y.: Dimes: A differentiable meta solver for combinatorial optimization problems (2022) https://doi.org/10.48550/ARXIV.2210.04123 arXiv:2210.04123

[20] Sun, Z., Yang, Y.: Difusco: Graph-based diffusion solvers for combinatorial optimization (2023) https://doi.org/10.48550/ARXIV.2302.08224 arXiv:2302.08224

[21] Min, Y., Bai, Y., Gomes, C.P.: Unsupervised learning for solving the travelling salesman problem (2023) https://doi.org/10.48550/ARXIV.2303.10538 arXiv:2303.10538

[22] Pan, X., Wang, C., Ying, C., Xue, Y., Yu, T.: Beyond the heatmap: A rigorous evaluation of component impact in mcts-based tsp solvers (2024) https://doi.org/10.48550/ARXIV.2411.09238 arXiv:2411.09238

[23] Xia, Y., Yang, X., Liu, Z., Liu, Z., Song, L., Bian, J.: Position: Rethinking post-hoc search-based neural approaches for solving large-scale traveling salesman problems (2024) https://doi.org/10.48550/ARXIV.2406.03503 arXiv:2406.03503

[24] Correll, R., Weinberg, S.J., Sanches, F., Ide, T., Suzuki, T.: Quantum neural networks for a supply chain logistics application. Advanced Quantum Technologies **6**(7), 2200183 (2023) https://doi.org/10.1002/qute.202200183

[25] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)

[26] Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research **35**(2), 254–265 (1987) https://doi.org/10.1287/opre.35.2.254

[27] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8**(3–4), 229–256 (1992) https://doi.org/10.1007/BF00992696

[28] Furnon, V., Perron, L.: OR-Tools Routing Library v9.15 (2024). https://developers.google.com/optimization/routing/

[29] Helsgaun, K.: An effective implementation of the lin–kernighan traveling salesman heuristic. European Journal of Operational Research **126**(1), 106–130 (2000) https://doi.org/10.1016/s0377-2217(99)00284-2

[30] Helsgaun, K.: An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, Roskilde (2017). https://forskning.ruc.dk/en/publications/an-extension-of-the-lin-kernighan-helsgaun-tsp-solver-for-constra/

[31] Haimovich, M., Rinnooy Kan, A.H.G.: Bounds and heuristics for capacitated routing problems. Mathematics of Operations Research **10**(4), 527–542 (1985) https://doi.org/10.1287/moor.10.4.527

[32] Beardwood, J., Halton, J.H., Hammersley, J.M.: The shortest path through many points. Mathematical Proceedings of the Cambridge Philosophical Society **55**(4), 299–327 (1959) https://doi.org/10.1017/s0305004100034095

[33] Schuld, M., Petruccione, F.: Machine Learning with Quantum Computers. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83098-4

[34] Hibat-Allah, M., Mauri, M., Carrasquilla, J., Perdomo-Ortiz, A.: A framework for demonstrating practical quantum advantage: Racing quantum against classical generative models (2023) https://doi.org/10.1038/s42005-024-01552-6 arXiv:2303.15626

[35] Jerbi, S., Gyurik, C., Marshall, S.C., Briegel, H.J., Dunjko, V.: Parametrized quantum policies for reinforcement learning (2021) https://doi.org/10.48550/ARXIV.2103.05577 arXiv:2103.05577

[36] Peters, E., Caldeira, J.a., Ho, A., Leichenauer, S., Mohseni, M., Neven, H., Spentzouris, P., Strain, D., Perdue, G.N.: Machine learning of high dimensional data on a noisy quantum processor (2021) https://doi.org/10.48550/ARXIV.2101.09581 arXiv:2101.09581

[37] Franz, M., Winker, T., Groppe, S., Mauerer, W.: Hype or heuristic? quantum reinforcement learning for join order optimisation (2024) https://doi.org/10.1109/qce60285.2024.00055 arXiv:2405.07770

[38] Bowles, J., Ahmed, S., Schuld, M.: Better than classical? the subtle art of benchmarking quantum machine learning models (2024) arXiv:2403.07059