QAOA for Efficient Urban Logistical Ecosystem

Fabio Picariello[1,*], Gloria Turati[2,*], Riccardo Antonelli[1], Igor Bailo[1], Susanna Bonura[1], Gianmarco Ciarfaglia[1], Salvatore Cipolla[1], Paolo Cremonesi[2], Maurizio Ferrari Dacrema[2], Michele Gabusi[1], Ivan Gentile[3], Vito Morreale[1] and Antonio Noto[1]

[1]*Eng AI & Data @ Engineering Group, Piazzale dell'Agricoltura 24, 00144 Roma, Italy*
[2]*Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy*
[3]*International Foundation Big Data And Artificial Intelligence For Human Development Via Galliera n. 32 – Bologna*

Abstract

The Traveling Salesman Problem (TSP) is a cornerstone of combinatorial optimization with widespread applications in logistics and transportation. As problem sizes increase, classical algorithms often fail to deliver high-quality solutions within practical time constraints. This paper explores the use of the Quantum Approximate Optimization Algorithm (QAOA), a hybrid quantum-classical algorithm, to address TSP instances under realistic conditions. We present a QUBO-based formulation of the TSP that integrates practical constraints reflecting real-world conditions—such as vehicle capacity, road accessibility, and time windows—while maintaining compatibility with limitations of current quantum hardware. Our analysis is conducted in a simulated environment, leveraging high-performance computing (HPC) resources to evaluate the algorithm's performance across varying problem sizes and circuit depths. This approach enables a comprehensive assessment of QAOA's capabilities and limitations in solving constrained TSP scenarios, thereby laying the groundwork for its deployment on future large-scale quantum hardware.

1. Introduction

Logistics optimization plays a central role in several industrial applications, from supply chain management to urban delivery systems. Many of these problems, such as the Traveling Salesman Problem (TSP) [1], fall into the class of NP-hard problems, where finding exact solutions becomes computationally infeasible as the problem size increases [2].

To tackle this challenge, a wide range of heuristic and metaheuristic algorithms [3] have been introduced to produce solutions of acceptable quality within a reasonable time frame. However, as the size of the system grows, the quality of these approximate solutions often degrades, and the gap from the optimal solution widens.

In order to address this scalability issue, recent research has turned to quantum computing as a promising paradigm for tackling combinatorial optimization problems. Quantum algorithms, indeed, leverage quantum phenomena such as superposition and entanglement to explore large solution spaces more efficiently than their classical counterparts. By encoding optimization problems into a quantum formulation, these algorithms aim to provide high-quality approximate solutions with potentially reduced computational overhead.

In this work, we focus on the Quantum Approximate Optimization Algorithm (QAOA) [4, 5], a variational quantum algorithm specifically designed for near-term quantum devices and well-suited to tackle discrete optimization problems [6, 7, 8, 9, 10, 11, 12, 13] by alternating between quantum and classical optimization layers. We conduct an extensive analysis of QAOA's performance on both synthetic and realistic TSP instances, incorporating logistical constraints inspired by real transportation networks limitations.

Our main contributions include:

- Enforcing the one-city-per-step canonical constraint by using a Grover-inspired mixer in QAOA.

AIQxQIA 2025: International Workshop on AI for Quantum and Quantum for AI | co-located with ECAI 2025, Bologna, Italy
*Corresponding authors.
fabio.picariello@eng.it (F. Picariello); gloria.turati@polimi.it (G. Turati)
0000-0003-3773-9768 (F. Picariello); 0000-0001-5367-4058 (G. Turati)

- Proposing a constraint formulation for logistical constraints that is compatible with QAOA.
- Evaluating solution quality on a realistic dataset derived from urban transportation data.

The work is divided into the following sections. **Section 2** provides background on the TSP, QAOA, and the problem encoding used to adapt TSP for quantum optimization. **Section 3** outlines the methodology, covering the formulation of real-world logistical constraints compatible with current quantum hardware limitations (binary node compatibility, road accessibility, and time-step constraints), the construction of synthetic and realistic datasets, the adoption of a Grover-inspired mixer, the computational environment with emphasis on the CINECA infrastructure, the configuration of the QAOA algorithm, and the performance evaluation metrics. **Section 4** presents the experimental results, analyzing the performance of QAOA. **Section 5** concludes the paper and outlines directions for future research.

2. Background and Related Work

This section provides the theoretical and methodological foundations of our work. Specifically, we review the TSP and QAOA, and then present a Quadratic Unconstrained Binary Optimization (QUBO)-based encoding of the TSP that enables its integration within the QAOA framework.

2.1. The Traveling Salesman Problem (TSP)

The TSP [1] is one of the most studied problems in combinatorial optimization, with a wide range of real-world applications, including vehicle routing, logistics, circuit design, and scheduling. In its classical formulation, the problem involves a salesman who must visit a list of cities, finding the shortest possible route that visits each city exactly once and returns to the starting point. We refer to the conditions requiring that each city is visited exactly once and that each time step corresponds to exactly one visited city as the *canonical constraints*.

However, in practical scenarios, this classical formulation is often inadequate. Real-world applications typically involve additional *logistical constraints* such as:

- **Capacity constraints**, where vehicles have limited load or service capacities.
- **Time windows**, which require visits to occur within specific time intervals.
- **Road accessibility**, where certain paths may be unavailable or restricted.

Incorporating these constraints transforms the TSP into a more complex and realistic problem, which is even more challenging to solve [14].

Recent advances in quantum computing have opened new avenues for addressing such complex optimization tasks. Quantum algorithms, particularly those based on quantum annealing and variational approaches, have been explored for solving both symmetric and asymmetric versions of the TSP, including constrained variants. For instance, quantum annealing has been applied to TSP instances by mapping the problem into a QUBO formulation and embedding it onto quantum hardware such as D-Wave systems [15]. These approaches have shown promising results for small- to medium-sized instances, particularly when constraints are encoded directly into the cost function [16].

While quantum annealing has been widely explored for solving the TSP, recent research has increasingly shifted toward gate-based quantum algorithms, which are more compatible with universal quantum computers. Notably, Lytrosyngounis et al. [17] proposed a hybrid quantum-classical framework that integrates QAOA with classical machine learning techniques to enhance scalability and resilience to noise. Their results show that, although purely quantum approaches still lag behind classical solvers, hybrid methods significantly reduce the performance gap and offer promising scalability for future applications.

These gate-based approaches are particularly well-suited for constrained TSP variants, as they allow flexible encoding of problem-specific constraints directly into the quantum circuit. Building on this line of work, we apply QAOA to both synthetic and realistic TSP instances, incorporating practical constraints inspired by real-world logistical limitations (see Subsection 3.1).

2.2. The Quantum Approximate Optimization Algorithm (QAOA)

The QAOA [4] is a hybrid quantum-classical algorithm developed to tackle combinatorial optimization problems. It is particularly well-suited for near-term quantum devices, thanks to its shallow circuit depth and resilience to noise [18].

The algorithm constructs a parameterized quantum state by alternating two types of unitary operators: one derived from the cost Hamiltonian H_c, which encodes the objective function, and one from the mixing Hamiltonian H_m, which enables a broader exploration of the solution space. These operators are applied in a sequence of p layers, where p controls the trade-off between approximation accuracy and circuit complexity.

Starting from an initial quantum state $|s\rangle$, the circuit evolves according to the parameter vectors $\vec{\gamma} = (\gamma_1, \dots, \gamma_p)$ and $\vec{\beta} = (\beta_1, \dots, \beta_p)$, reaching the final state:

$$|\vec{\gamma}, \vec{\beta}\rangle = e^{-i\beta_p H_m} e^{-i\gamma_p H_c} \dots e^{-i\beta_1 H_m} e^{-i\gamma_1 H_c} |s\rangle. \tag{1}$$

This quantum state is measured multiple times in order to estimate the expected cost:

$$C_{\gamma,\beta} = \langle \vec{\gamma}, \vec{\beta} | H_c | \vec{\gamma}, \vec{\beta} \rangle, \tag{2}$$

and a classical optimizer iteratively adjusts $\vec{\gamma}$ and $\vec{\beta}$ to minimize $C_{\gamma,\beta}$, thereby allowing the circuit to produce a final state that minimizes such cost function (the ground state).

A commonly used choice for the mixing Hamiltonian, particularly effective for unconstrained problems, is the X-Mixer:

$$H_m = \sum_{i=1}^{n} X_i. \tag{3}$$

This initialization and mixer configuration enable exploration over the entire solution space.

However, for constrained problems, an alternative choice is the Grover mixer [19]. The key idea behind this approach is to initialize the quantum system in a uniform superposition over feasible solutions and construct a mixer that preserves transitions within this subspace. Let U_s be a unitary operator that maps the all-zero state to a uniform superposition over the set of feasible solutions F:

$$U_s |0\rangle^{\otimes n} = \frac{1}{\sqrt{|F|}} \sum_{x \in F} |x\rangle. \tag{4}$$

The associated Grover mixer is defined as:

$$U_m(\beta_i) = U_s \left(I_d - \left(1 - e^{-i\beta_i}\right) |0\rangle\langle 0| \right) U_s^{\dagger}. \tag{5}$$

This operator replaces $e^{-i\beta_i H_m}$ in Equation 1 for each β_i. The advantage of the Grover mixer is that it restricts the quantum evolution to the subspace of feasible solutions, significantly reducing the effective size of the search space and improving ease of trainability.

QAOA offers several advantages: it is versatile, compatible with NISQ-era hardware [20], and capable of improving solution quality by increasing the circuit depth p. Nonetheless, its performance is sensitive to the choice of parameter initialization and to the complexity of the classical optimization process, especially when strong constraints are embedded in H_c.

A notable class of problems that can be addressed using QAOA is the family of QUBO [21] problems. These are NP-hard combinatorial optimization problems, whose general form is given by:

$$\min_{x \in \{0,1\}^n} x^{\top} Q x, \tag{6}$$

where $x \in \{0, 1\}^n$ is a binary vector, and $Q \in \mathbb{R}^{n \times n}$ is a symmetric (or upper triangular) matrix that defines the cost landscape.

QUBO problems can be mapped to the task of finding the ground state of a corresponding Hamiltonian via the Ising model representation [22], enabling their solution through quantum algorithms such as QAOA. In this case, the resulting Hamiltonian is diagonal in the computational basis, meaning that its ground state corresponds to one of the basis states. Consequently, the measurement process yields a bitstring that directly encodes a candidate solution to the original QUBO problem.

2.3. TSP Encoding for QAOA

The QAOA can be adapted to solve instances of the TSP by reformulating the problem as a QUBO model [23]. In this formulation, binary variables indicate which city is visited at each time step t of a candidate tour. The total travel cost, measured in terms of distance or time, serves as the objective function to be minimized within the QAOA framework.

The TSP involves canonical constraints: each time step must correspond to exactly one visited city, and each city must be visited exactly once. Since the QUBO formulation is inherently unconstrained, these conditions are enforced by incorporating suitable penalty terms into the cost function, thereby discouraging infeasible solutions.

Consider a TSP instance involving n cities. In this QUBO formulation, a total of n^2 binary variables are required. Each variable $x_{i,t}$ indicates whether city i is visited at step t in the tour, and is defined as:

$$x_{i,t} = \begin{cases} 1 & \text{if city } i \text{ is visited at step } t \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In this formulation, the Hamiltonian encoding the objective function is given by:

$$D(x) = \sum_{i,j=1}^{n} \omega_{i,j} \sum_{t=1}^{n-1} x_{i,t} x_{j,t+1}, \tag{8}$$

where ω denotes the *cost matrix*, whose entries represent the cost $\omega_{i,j}$ of traveling from city i to city j.

To enforce the TSP canonical constraints according to which each city must be visited exactly once and only one city is visited at each time step, we introduce penalty terms to penalize invalid configurations:

$$P(x) = \lambda_p \underbrace{\sum_{i} \left(\sum_{t} x_{i,t} - 1 \right)^2}_{\text{each city once}} + \lambda_q \underbrace{\sum_{t} \left(\sum_{i} x_{i,t} - 1 \right)^2}_{\text{one city per time step}}, \tag{9}$$

where the first term ensures that each city is visited exactly once, and the second term enforces that each time step corresponds to a visit to exactly one city. The factors λ_p and λ_q are penalty weights introduced to penalize infeasible solutions by increasing their associated cost, while leaving the cost of feasible solutions unchanged.

In our approach, the constraint corresponding to the second penalty term is not enforced via the cost function but rather through a strategy based on the use of the Grover mixer [19], as described in Subsection 3.3. Consequently, the only penalty term explicitly included in the cost function is:

$$\tilde{P}(x) = \lambda_p \sum_{i} \left(\sum_{t} x_{i,t} - 1 \right)^2, \tag{10}$$

which ensures that each city is visited exactly once.

As a result, the complete cost function $C(x)$ for a TSP instance without logistical constraints is given by:

$$C(x) = D(x) + \tilde{P}(x). \tag{11}$$

3. Methodology

This section outlines the experimental setup used to evaluate the performance of QAOA on TSP instances, describing the logistical constraints, the computational environment, the datasets, the use of the Grover mixer, the algorithmic configurations, and the performance metrics adopted in our study.

3.1. Logistical Constraints

While the classical TSP has been extensively studied as a combinatorial optimization problem [23, 24, 25, 26], its direct application to real-world scenarios is often limited by practical logistical constraints. To ensure the relevance and applicability of quantum algorithms to real-world logistics and routing problems, it is essential to incorporate such constraints into the quantum formulation. However, this must be done carefully to minimize the impact on circuit depth, qubit count, and overall algorithmic complexity.

The definition and formulation of the constraints adopted in this work are inspired by [14, 27], but have been adapted to a binary representation for ease of implementation. This adaptation reduces the number of required qubits, thereby lowering the overall demand for computational resources.

Binary Node Compatibility

To enable the simulation of a capacitated TSP within the limitations of current quantum hardware and simulators, we propose a simplified and tractable formulation that approximates capacity constraints, referred to as *Binary Node Compatibility* (BNC).

We introduce a binary vector $k \in \{0,1\}^n$, where each element k_i represents the operational state of node (or city) i:

$$k_i = \begin{cases} 1 & \text{if node } i \text{ belongs to state A,} \\ 0 & \text{if node } i \text{ belongs to state B.} \end{cases} \tag{12}$$

This abstraction models scenarios in which nodes are categorized into two distinct types, and transitions between nodes of the same type incur a penalty. Examples include electric vehicle routing, where customer visits (1) alternate with charging stops (0), and waste collection, where pickups (1) alternate with depot unloading (0).

The penalty term can be modeled as:

$$K(x) = \lambda_k \sum_{\substack{i,j,t \\ i \neq j}} x_{i,t} x_{j,t+1} (1 - k_i \oplus k_j), \tag{13}$$

where $\oplus$ denotes the binary sum.

By examining Equation 8, we observe that it shares the same structure as Equation 13. Therefore, the BNC constraint can be directly incorporated into the cost matrix, avoiding any increase in circuit complexity:

$$\omega_{i,j}^{(\text{with BNC constraint})} \rightarrow \omega_{i,j} + \lambda_k (1 - k_i \oplus k_j). \tag{14}$$

Road-Related Constraints

Another class of constraints considered in this work involves the possibility that certain pairs of nodes may not be directly connected, simulating real-world scenarios such as road closures or inaccessible paths.

To model this, we define a binary road constraint matrix $R_{i,j}$, where:

$$R_{i,j} = \begin{cases} 1 & \text{if node } i \text{ can not be directly connected to node } j \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

The corresponding penalty term in the Hamiltonian is given by:

$$R(x) = \lambda_R \sum_{\substack{i,j,t \\ i \neq j}} x_{i,t}\, x_{j,t+1}\, R_{i,j}. \tag{16}$$

However, since this penalty term shares the same structure as Equation 13, it can be directly incorporated into the cost matrix. This allows us to avoid increasing the quantum circuit complexity:

$$\omega_{i,j}^{(\text{with road constraint})} \rightarrow \omega_{i,j} + \lambda_R R_{i,j}. \tag{17}$$

Time-step Constraints

To incorporate time-related constraints into the TSP while maintaining compatibility with current quantum hardware limitations, we introduce a simplified variant inspired by the Vehicle Routing Problem with Time Windows (VRPTW) [28]. In VRPTW, certain nodes must be visited within predefined time windows due to operational or business requirements. However, a full VRPTW formulation introduces a large number of additional variables, making it impractical for current quantum hardware [14]. To address this limitation, we propose a lightweight alternative that enforces simplified time constraints without increasing the qubit count. Specifically, we define *time-step constraints*, which restrict certain cities to appear at predetermined positions in the tour.

To achieve this, we define a binary time constraint matrix $T_{i,t}$, where:

$$T_{i,t} = \begin{cases} 1 & \text{if city } i \text{ is not allowed to be visited at step } t \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Based on this matrix, we introduce a penalty term in the Hamiltonian to discourage violations of the time constraints:

$$T(x) = \lambda_T \sum_{i,t} T_{i,t} x_{i,t}. \tag{19}$$

Consequently, the resulting cost matrix becomes:

$$C(x) = D(x) + \tilde{P}(x) + T(x). \tag{20}$$

Unlike BNC and road constraints, time-step constraints cannot be directly embedded into the cost matrix, resulting in a slight increase in the complexity of the quantum circuit.

3.2. TSP Problem Instances

To assess the performance of QAOA under diverse conditions, we consider both synthetic and realistic TSP datasets. This subsection provides a detailed description of the two datasets.

3.2.1. Synthetic Dataset

The first dataset employed is a synthetic one, generated by sampling from a random distribution. Specifically, the cost matrix is constructed as an $n \times n$ matrix with entries randomly sampled from the interval $[0, 10]$. The diagonal elements are set to zero, representing the zero cost of remaining at the same node. The matrix is allowed to be asymmetric to reflect unbalanced travel costs.

This dataset serves as a benchmark to assess algorithm performance in the absence of inherent structure. Since the data lacks any natural ordering or patterns, finding optimal solutions is generally more challenging compared to more realistic datasets.

3.2.2. Realistic Dataset: Milan Subway Network

To evaluate the algorithms in a real-world context, we employed a dataset based on the actual geographic positions of subway stops in the city of Milan. Specifically, we utilized open-source data provided in the General Transit Feed Specification (GTFS) format[1]. Through this dataset, we extracted the precise locations of Milan's subway stops, focusing particularly on the most frequently used stations.

After acquiring the stop locations, we constructed a cost matrix using data from OpenStreetMap (OSM)[2], which provides satellite-based geographic data, including the full urban street network. Rather than relying on straight-line distances between latitude-longitude coordinates, we leveraged OSM to construct a realistic road network graph that accounts for actual intersections, road segments, and

[1]Available at https://dati.comune.milano.it/gtfs.zip.

[2]The cost matrix was computed using the `osmnx` Python library, available at https://github.com/gboeing/osmnx.

one-way constraints. On this graph, we computed the pairwise distances between subway stops by applying Dijkstra's algorithm [29], ensuring that the computed paths reflect real-world accessibility and routing.

To better reflect real-world travel conditions, we converted the distance-based cost matrix into a time-based one. This was achieved by dividing each edge's distance by the maximum allowed speed on the corresponding road segment, as provided by OSM metadata. The resulting cost matrix is asymmetric, capturing the variability in travel times due to differing traffic conditions and road types.

This realistic dataset enables the exploration of practical applications, such as:

- **Inspection and Maintenance Planning:** Prioritizing interventions on the most critical or frequently used subway stops.
- **Network Vulnerability Analysis:** Identifying shortest paths between major nodes to support emergency planning, detour strategies, and infrastructure improvements.

Furthermore, we generated multiple datasets of varying sizes using the GTFS data. While the complete dataset includes up to 130 subway stops, we created smaller subsets by selecting the top-n most visited stops. This allows for scalability testing and performance evaluation of the algorithms under different problem sizes.

3.3. Grover Mixer

In the context of the TSP, many bitstrings represent invalid solutions: for example, visiting multiple cities at the same time step or revisiting the same city. A uniform exploration of all bitstrings, as done with standard mixers, therefore may waste computational resources on infeasible candidates.

To mitigate this, we adopt a Grover-inspired mixer [19] that restricts mixing to states that (partially) satisfy the canonical constraints. Although this approach increases the complexity of the initialization step, it substantially reduces the effective search space and improves the probability of sampling high-quality solutions.

While one could ideally prepare a superposition of only fully feasible states, the corresponding unitary becomes intractable to construct, as the number of feasible states grows factorially with n. We therefore consider a more tractable initialization strategy.

The total system of n^2 qubits is partitioned into n registers of n qubits each, denoted $|q_i\rangle$ for $i = 1, \dots, n$. Each register encodes the position of the i-th city in the tour and must contain exactly one qubit in the $|1\rangle$ state, with the others in $|0\rangle$, which correspond to the Dicke state $|D_1^n\rangle$. Therefore, each register is initialized in an equal superposition of all valid one-hot encoded states. For example, when $n = 3$ (corresponding to a system of $3^2 = 9$ qubits), each register $|q_k\rangle$ is initialized as:

$$|q_k\rangle = |D_1^3\rangle = U_s^k |0\rangle^{\otimes 3} = \frac{1}{\sqrt{3}} \left(|100\rangle + |010\rangle + |001\rangle\right), \tag{21}$$

where U_s^k denotes the unitary operator that prepares an equal superposition over all feasible states of the k-th register. Based on Equation 5, the corresponding mixer unitary for the k-th register is defined as:

$$U_m^k(\beta_i) = U_s^k \left(I_d - \left(1 - e^{-i\beta_i}\right) |0\rangle\langle 0|\right) (U_s^k)^\dagger. \tag{22}$$

Since each $U_m^k(\beta_i)$ acts on a distinct register, the overall mixer unitary $U_m(\beta_i)$ applied to the full system can be written as the tensor product:

$$U_m(\beta_i) = \bigotimes_{k=1}^{n} U_m^k(\beta_i). \tag{23}$$

By using the mixer Hamiltonian defined according to Equation 5, we restrict the quantum evolution to partially feasible subspaces. This construction guarantees that each time step corresponds to exactly one city being visited, although the final state may still include invalid tours where some cities are

visited multiple times. This approach significantly reduces the size of the search space from 2^{n^2} to n^n, in contrast to using a standard X-Mixer.

Finally, we observe that, due to the construction which guarantees that exactly one city is visited at each time step, the second term of Equation 9 can be omitted when applying this mixer Hamiltonian. As a result, the simplified cost in Equation 10 can be used instead.

3.4. Computational Environment

The simulated experiments were conducted using the `Qiskit` framework. All simulations for problem sizes up to $n = 5$ were executed on a local high-performance workstation equipped with 96 CPU cores. The computational capabilities of this system significantly reduced the overall simulation time, enabling the execution of a large number of distinct simulations within a feasible timeframe.

As previously discussed, the QUBO formulation of the TSP requires n^2 qubits, where n is the number of cities in the problem instance. Simulating such a quantum system entails storing the full quantum state, which consists of 2^{n^2} complex amplitudes. Assuming double-precision floating-point representation, the memory requirements grow exponentially with n, as shown in Table 1.

Table 1
Estimated memory requirements for storing QUBO-based TSP instances. Values are computed assuming double-precision complex amplitudes.

n	RAM Required [GiB]
4	9.8×10^{-4}
5	0.5
6	1024

As shown in Table 1, simulations are feasible on local machines only up to $n = 5$. For larger instances, such as $n = 6$, the exponential growth in memory requirements necessitates the use of more powerful high-performance computing (HPC) resources.

CINECA Infrastructure

To address these computational demands, we leveraged the Leonardo supercomputing infrastructure to simulate the TSP instance with $n = 6$, corresponding to a 36-qubit quantum system. Leonardo is a pre-exascale supercomputer hosted by the CINECA consortium in Italy, designed to support large-scale scientific computing and artificial intelligence workloads[3].

For our experiments, we utilized 8 compute nodes, each equipped with 4 NVIDIA A100 GPUs (64 GB of memory per GPU) and a single CPU. The statevector representation of the quantum system was distributed across all the involved GPUs, enabling large-scale simulation through the software stack's distributed architecture. The use of GPUs was essential for exploiting the distributed simulation capabilities provided by `Qiskit`'s `cusvaer` backend, integrated within the NVIDIA cuQuantum Appliance [30], which supports partitioning the statevector across multiple GPUs and compute nodes.

This functionality allowed us to simulate a 36-qubit system by distributing the memory load across 32 GPUs. Due to the internal memory management strategy of the cuQuantum Appliance, each GPU was configured to allocate up to 32 GB of its 64 GB memory for storing a portion of the statevector. The remaining memory was reserved for auxiliary operations such as inter-GPU communication and memory transfers, which are critical for maintaining consistency and performance in distributed quantum simulations.

All quantum circuits were transpiled using `Qiskit` with optimization level 3, the highest available level, to reduce circuit depth and improve simulation performance.

[3] https://leonardo-supercomputer.cineca.eu/

3.5. QAOA Configuration

Following the precedent set by several studies in the literature [13, 31], we choose the COBYLA optimizer [32] due to its favourable balance between solution quality and computational efficiency [33, 34]. The initial values of the parameters γ_i and β_i for $i = 1, \ldots, p$ are randomly sampled from the interval $[0, 2\pi]$ and subsequently refined by the classical optimizer. Classical optimization was carried out using the COBYLA algorithm with a maximum of 200 iterations, which exceeded the convergence threshold in all cases.

Finally, for the penalty coefficients associated with the canonical and logistical constraints, we set the values of λ_p, λ_k, λ_R, and λ_T in Equations 10, 13, 16, and 19 as follows:

$$\lambda = \max_{i,j}(\omega_{i,j}) \cdot n.$$

This choice follows a common strategy in QUBO formulations, where penalty weights are set so that any feasible solution has a lower cost than any infeasible one [21, 22, 35]. Specifically, using a value proportional to the maximum edge weight and the problem size provides a conservative bound that guarantees any constraint violation incurs a higher cost than any feasible solution.

Alternative approaches in the literature include setting penalties based on the full range of the objective function or tuning them empirically [35]. However, excessively large penalties can distort the energy landscape, potentially degrading performance. Choosing an appropriate penalty value remains a challenging task and may be better addressed in future work.

3.6. Performance Evaluation Metrics

The most widely adopted metric in the literature to assess solution quality is the *approximation ratio* (AR) [36], defined as the ratio between the cost of the solution obtained by the algorithm and the cost of the optimal solution (C_{opt}).

Given the probabilistic nature of QAOA, each execution yields a distribution over possible solutions. Therefore, we consider two distinct forms of the approximation ratio:

$$\tilde{AR}_{\text{exp}} = \frac{\langle C \rangle}{C_{\text{opt}}}, \tag{24}$$

$$\tilde{AR}_{\text{min}} = \frac{C_{\text{min}}}{C_{\text{opt}}}, \tag{25}$$

where $\langle C \rangle$ denotes the expected cost at the end of the optimization process (i.e., the average value of the cost function over sampled solutions), and C_{min} is the minimum cost observed throughout the entire optimization.

These metrics capture complementary aspects of solution quality:

- $\tilde{AR}_{\text{exp}}$ reflects the expected quality of a solution sampled from the output distribution, providing insight into how concentrated the distribution is around low-cost solutions.
- $\tilde{AR}_{\text{min}}$ evaluates the best solution sampled during a run, offering a direct measure of the algorithm's ability to find near-optimal solutions.

However, in constrained optimization problems such as ours, the output distribution may exhibit high variance. To account for this, we introduce a normalized version of the approximation ratio by subtracting the worst possible cost (C_{worst}) from both the numerator and the denominator. This normalization bounds the metric within the interval $[0, 1]$, where a value of 1 corresponds to an optimal solution and 0 to the worst-case outcome:

$$AR_{\text{exp}} = \frac{\langle C \rangle - C_{\text{worst}}}{C_{\text{opt}} - C_{\text{worst}}} \in [0, 1] \tag{26}$$

$$AR_{\min} = \frac{C_{\min} - C_{\text{worst}}}{C_{\text{opt}} - C_{\text{worst}}} \in [0, 1]. \tag{27}$$

This adjusted formulation provides a more robust and interpretable metric for evaluating QAOA performance, especially in the presence of hard constraints and noisy solution landscapes.

Both C_{opt} and C_{worst} were determined using an exhaustive brute-force approach. Specifically, we evaluated all possible bitstrings of length n^2, recording the absolute minimum value as C_{opt} and the maximum value as C_{worst}.

4. Experimental Results and Analysis

We conducted a detailed analysis to observe how the performance metrics vary with two key hyperparameters:

1. The number of shots used in the quantum circuit execution (see Subsection 4.1).
2. The depth parameter p of the QAOA circuit (see Subsection 4.2).

For the synthetic dataset, all problem constraints defined in the previous sections are explicitly incorporated into the analysis. A similar evaluation was conducted using the realistic dataset derived from the Milan subway network. In this case, only the BNC and time-step constraints were explicitly enforced during the optimization process. Road-related constraints, such as route accessibility, are inherently embedded in the structure of the cost matrix, as it was constructed using real-world data from OSM. This implicit incorporation ensures that the cost matrix reflects realistic travel conditions without requiring additional constraint modeling.

The results obtained on the local workstation for problem sizes $n = 4$ and $n = 5$ are presented separately from those for $n = 6$, which were obtained using the HPC infrastructure. The latter are discussed in dedicated paragraphs within both the shot analysis and depth analysis subsections.

Given the limited complexity of problem instances with $n = 3$, the corresponding results are not reported.

4.1. Shot Analysis

To evaluate the impact of measurement statistics on solution quality, we conducted experiments with a fixed QAOA depth of $p = 1$, varying the number of shots.

It is important to analyze the behavior of the algorithm under these conditions for two main reasons. First, evaluating performance at $p = 1$ allows us to determine whether the system can identify the optimal solution using minimal circuit depth, a desirable property for near-term quantum devices. Second, by varying the number of shots, we gain insight into the trade-off between solution quality and computational cost. While increasing the number of shots generally improves solution quality, it also results in longer simulation times and higher resource usage. Therefore, identifying the minimum number of shots required to obtain high-quality solutions is crucial for optimizing overall efficiency.

In this analysis, we examine how $AR_{\min}$ evolves as a function of the number of shots, with the goal of identifying the optimal balance between solution quality and computational effort. For the instances with $n = 4$ and $n = 5$, we vary the number of shots across the set shots $= [10,\ 100,\ 500,\ 1000,\ 2000,\ 5000]$.

Tables 2 and 3 present the analysis of the approximation ratio of the solution with the minimum cost sampled, $AR_{\min}$, for $n = 4$ and $n = 5$, respectively, across different constraint configurations. It is worth noting that, in the presented tables, the "Runs" column indicates the number of problem instances solved under identical constraints and hyperparameters, but with different initial cost matrices.

Table 2
Shot analysis for $n = 4$ and $p = 1$. Each entry in the "Shots" column denotes a range of shot counts (e.g., 10-5000) over which the corresponding data have been averaged. The column $\langle AR_{\text{min}} \rangle$ reports the mean value of AR_{min} for all runs within each shot range.

Dataset	Constraint	Shots	Runs	$\langle AR_{\text{min}} \rangle$
Random	-	10-5000	5	1.000
Realistic	-	10-5000	5	1.000
Random	BNC	10-5000	15	1.000
Realistic	BNC	10-5000	15	1.000
Random	road	10-5000	25	1.000
Random	time	10	15	0.998
Random	time	100-5000	15	1.000
Realistic	time	10-5000	15	1.000

Table 3
Shot analysis for $n = 5$ and $p = 1$. Each entry in the "Shots" column denotes a range of shot counts (e.g., 10-5000) over which the corresponding data have been averaged. The column $\langle AR_{\text{min}} \rangle$ reports the mean value of AR_{min} for all runs within each shot range.

Dataset	Constraint	Shots	Runs	$\langle AR_{\text{min}} \rangle$
Random	-	10	5	0.999
Random	-	100-5000	5	1.000
Realistic	-	10	5	0.9990
Realistic	-	100-5000	5	1.000
Random	BNC	10	20	0.987
Random	BNC	100-5000	20	1.000
Realistic	BNC	10	20	0.992
Realistic	BNC	100-5000	20	1.000
Random	road	10	25	0.998
Random	road	100-5000	25	1.000
Random	time	10	25	0.996
Random	time	100	25	0.999
Random	time	500-5000	25	1.000
Realistic	time	10-100	25	0.999
Realistic	time	500-5000	25	1.000

The analysis of the results highlights a key insight into the performance of QAOA: the algorithm is capable of consistently identifying the optimal solution with a relatively low number of measurement shots. Specifically, the best approximation ratio, AR_{min}, reaches the optimal value of 1 under all tested constraints—requiring as few as 100 shots for $n = 4$ and 500 shots for $n = 5$. Furthermore, a key observation is that the optimal solution is always found even with $p = 1$ and a relatively small number of shots.

Results for $n = 6$ from Simulations on the Leonardo HPC Infrastructure

Similarly, we conducted experiments on TSP instances with $n = 6$ under various constraint configurations, using the Leonardo high-performance computing infrastructure. These experiments were carried out with a fixed QAOA depth of $p = 1$, testing shot counts of 500 and 2000. The results are presented in Table 4.

Table 4
Shot analysis for $n = 6$ and $p = 1$. Each entry in the "Shots" column denotes a range of shot counts (e.g., 500-2000) over which the corresponding data have been averaged. The column $\langle AR_{\text{min}} \rangle$ reports the mean value of AR_{min} for all runs within each shot range.

Dataset	Constraint	Shots	Runs	$\langle AR_{\text{min}} \rangle$
Random	-	500	1	0.999
Random	-	2000	1	1.000
Realistic	-	500-2000	1	1.000
Random	BNC	500-2000	1	1.000
Realistic	BNC	500	1	0.999
Realistic	BNC	2000	1	1.000
Random	road	500-2000	1	1.000
Random	time	500-2000	1	1.000
Realistic	time	500-2000	1	1.000

For the $n = 6$ instance, 2000 shots are sufficient to sample the optimal solution during the algorithm execution (i.e., to obtain a value of $\langle AR_{\text{min}} \rangle$ equal to 1). This shows that high-quality solutions can be achieved with limited sampling effort. This observation reinforces the finding that QAOA remains efficient across varying problem sizes and constraint configurations.

Notably, a circuit depth of $p = 1$ proves sufficient to reach optimal solutions in all tested cases with $n \leq 6$. This is a particularly encouraging result, as it suggests that shallow circuits can deliver strong performance while keeping computational and hardware demands low. Given that deeper circuits are more susceptible to noise and decoherence on real quantum hardware, the ability to achieve optimal performance with minimal depth is a significant advantage. It supports the practical viability of QAOA in near-term quantum devices, where circuit depth remains a critical constraint.

4.2. Circuit Depth Analysis

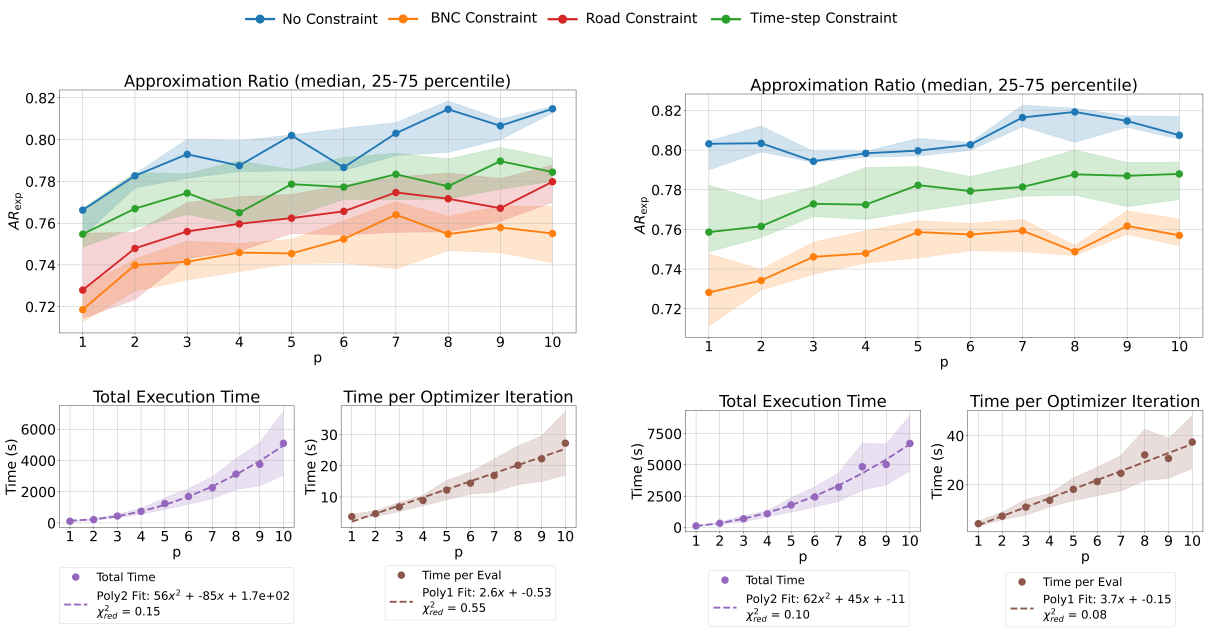
We investigated the impact of varying the number of QAOA layers p on the approximation ratio of the expectation value, AR_{exp}, while fixing the number of shots to 500 for instances with $n = 4$ and $n = 5$. The results are presented in Figures 1a, 1b,1c and 1d, along with an analysis of the scaling behavior of the simulation time.

This analysis is essential to determine whether increasing the circuit depth leads to a meaningful improvement in solution quality. While deeper circuits may enhance the expressivity of the ansatz, they also introduce greater computational complexity and increased susceptibility to noise and decoherence, factors that pose significant challenges when using real quantum devices.

By observing how AR_{exp} evolves with increasing p, we aim to assess whether the probability of sampling low-cost (i.e., optimal or near-optimal) solutions improves significantly. If such an improvement is observed, it would justify the additional computational overhead introduced by deeper circuits; otherwise, it would suggest that shallow circuits are sufficient to achieve satisfactory performance under constrained resources.

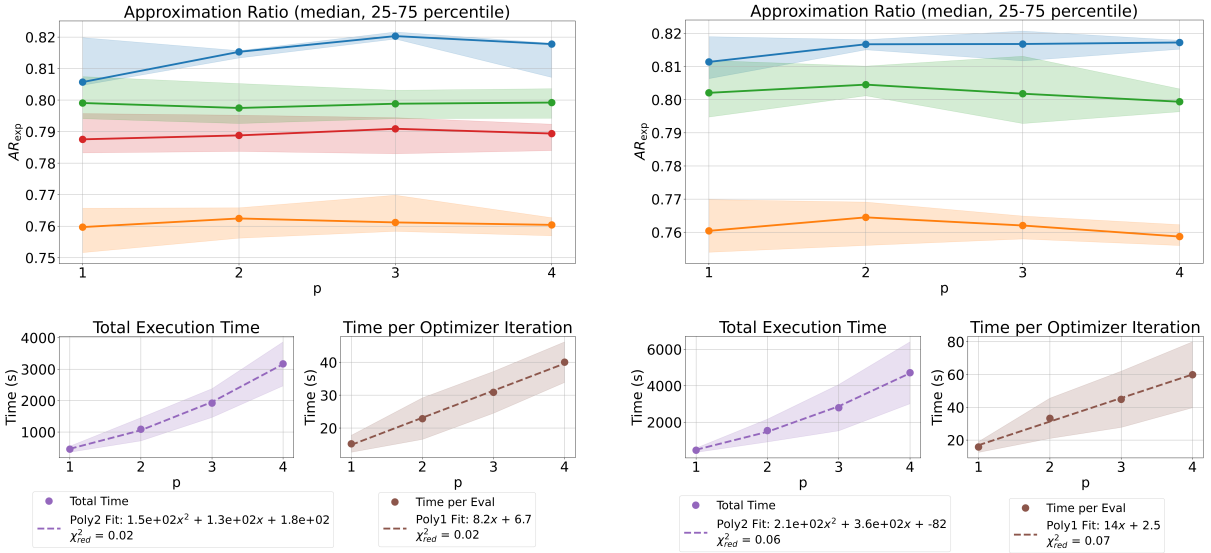
The values of p tested range from 1 to 10 for the $n = 4$ problem instances, and from 1 to 4 for the $n = 5$ instances. The reduced range in the latter case is due to the prohibitively high computational cost associated with high-depth circuits as the problem size increases.

The analysis of AR_{exp} reveals that, as the number of layers p increases, AR_{exp} tends to improve, indicating a higher probability of sampling low-cost bitstrings. This effect is particularly evident for $n = 4$, where the relatively small solution space allows even modest increases in circuit depth to yield noticeable gains. For larger problem sizes ($n = 5$), the improvement is less evident, likely due to the exponential growth of the solution space. Nonetheless, the general trend confirms that deeper circuits enhance the quality of the sampled solutions, even if the optimal solution is already accessible at lower



(a) Random dataset, $n = 4$.

(b) Milan Subway Network dataset, $n = 4$.



(c) Random dataset, $n = 5$.

(d) Milan Subway Network dataset, $n = 5$.

Figure 1: QAOA performance across random and realistic datasets based on the Milan subway system. Each plot shows AR_{exp} (top), total QAOA execution time (bottom left), and execution time per iteration (bottom right).

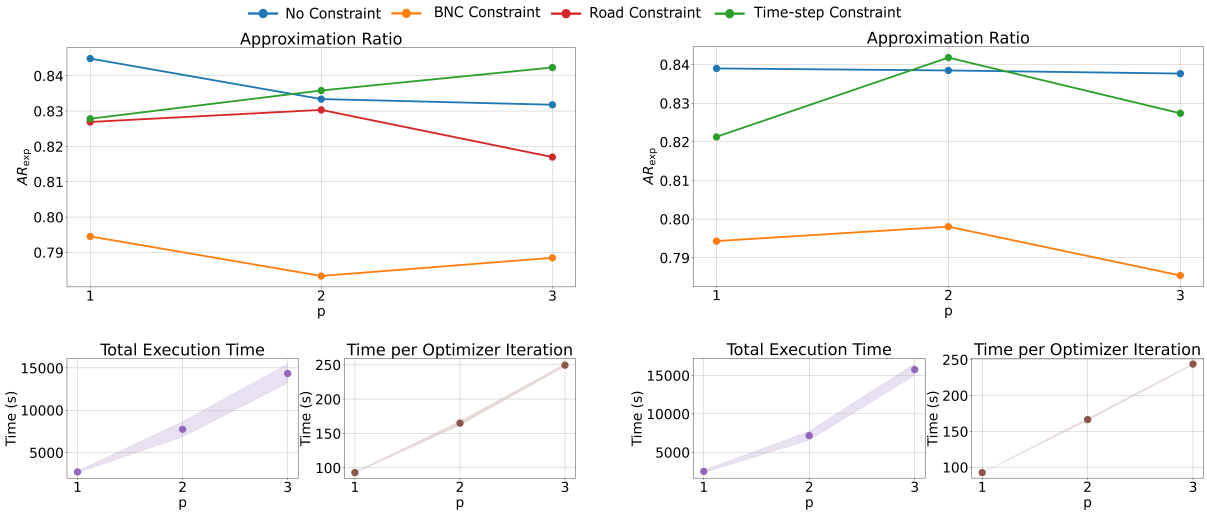
depths, as shown in the shots analysis.

In parallel, we observe that the total computational time grows approximately quadratically with the circuit depth p. While the theoretical time complexity of the quantum circuit itself scales linearly with p, practical considerations introduce additional overhead. In particular, the classical optimizer must handle a larger number of parameters as p increases, which often leads to a greater number of iterations required to converge. This results in a superlinear increase in total runtime. These findings underscore the trade-off between solution quality and computational cost when tuning the depth of QAOA circuits.

Results for $n = 6$ from Simulations on the Leonardo HPC Infrastructure

Here we present the analysis of the AR_{exp} metric for the $n = 6$ instance, performed on the Leonardo HPC infrastructure. The evaluation is conducted by varying the circuit depth p from 1 to 3, with the

number of shots fixed at 2000, under different constraint configurations, as illustrated in Figures 2a and 2b.



(a) Random dataset, $n = 6$.

(b) Milan Subway Network dataset, $n = 6$.

Figure 2: QAOA performance across random and realistic datasets based on the Milan subway system. Each plot shows AR_{exp} (top), total QAOA execution time (bottom left), and execution time per iteration (bottom right).

The analysis of AR_{exp} for the $n = 6$ instance aligns with the trends observed for $n = 4$ and $n = 5$, indicating that the need for deeper quantum circuits does not significantly increase with problem size in the tested configurations. This suggests that the proposed approach remains scalable, as high-quality solutions can be obtained with relatively shallow circuits even for larger instances.

5. Conclusion and Future Work

This work investigated the application of the Quantum Approximate Optimization Algorithm (QAOA) to the Traveling Salesman Problem (TSP), incorporating realistic constraints such as Binary Node Compatibility (BNC), road accessibility, and time-step constraints. We introduced a QUBO-based formulation capable of encoding these constraints without increasing the complexity of the quantum system. To further reduce the search space, we employed a Grover-inspired mixer, tailored to guide the algorithm more efficiently toward feasible solutions.

Our experiments, conducted on both synthetic and real-world datasets, demonstrated that QAOA is capable of finding optimal solutions for problem instances up to $n = 5$, and preliminary results for $n = 6$, obtained using the Leonardo HPC infrastructure, further support this capability. Notably, optimal solutions were consistently found using a single QAOA layer and a relatively small number of measurement shots, regardless of the constraint type.

The analysis of the approximation ratio relative to the expectation value, AR_{exp}, revealed that increasing the circuit depth p leads to modest improvements in solution quality. Moreover, unconstrained TSP instances generally achieved higher AR_{exp} values compared to their constrained counterparts. This discrepancy is likely due to the fact that constraints are enforced through penalty terms in the cost function, and thus infeasible solutions are not explicitly excluded from the quantum state. Each time a constraint is violated, the expectation value of the problem Hamiltonian increases significantly, thereby substantially reducing AR_{exp}. The stricter the constraint, the more pronounced this effect becomes.

Looking ahead, several promising directions emerge for future research. One avenue involves enhancing the current framework to support the representation of more general real-world constraints (e.g., those that are not binary in nature), while still preserving efficiency in the use of computational resources. In parallel, the framework could be generalized to address more complex routing problems,

such as the Vehicle Routing Problem (VRP). Another direction is the exploration of alternative problem encodings, such as binary one-hot representations, that could reduce the number of required qubits and enable the simulation of larger instances, taking inspiration from the approach proposed in [37]. Finally, QAOA-based methods could be developed to address larger TSP instances. This could be achieved by integrating QAOA with classical algorithms (e.g., clustering techniques) to create hybrid quantum-classical workflows, thereby helping to overcome current hardware limitations that hinder scalability.

Collectively, these directions aim to bridge the gap between theoretical quantum algorithms and their practical applicability to real-world problems, paving the way for quantum-enhanced optimization in complex logistical systems.

Acknowledgements

The authors acknowledge CINECA for providing computational resources used to carry out the simulations for the $n = 6$ instances.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT-4 in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] D. L. Applegate, The traveling salesman problem: a computational study, volume 17, Princeton university press, 2006.

[2] G. J. Woeginger, Exact algorithms for np-hard problems: A survey, in: Combinatorial Optimization—Eureka, You Shrink!, Springer, 2003, pp. 185–207.

[3] I. Boussaïd, L. Julien, P. Siarry, Metaheuristic research: a comprehensive survey, Artificial Intelligence Review 37 (2013) 663–686. doi:10.1007/s10462-013-9361-3.

[4] E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm, arXiv preprint arXiv:1411.4028 (2014).

[5] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, A. Summer, A review on quantum approximate optimization algorithm and its variants, 2023. arXiv:2306.09198.

[6] G. E. Crooks, Performance of the quantum approximate optimization algorithm on the maximum cut problem, 2018. arXiv:1811.08419.

[7] M. Willsch, D. Willsch, F. Jin, H. D. Raedt, K. Michielsen, Benchmarking the quantum approximate optimization algorithm, Quantum Information Processing 19 (2020). URL: https://doi.org/10.1007%2Fs11128-020-02692-8. doi:10.1007/s11128-020-02692-8.

[8] J. Cook, S. Eidenbenz, A. Bärtschi, The quantum alternating operator ansatz on maximum k-vertex cover, in: 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), 2020, pp. 83–92. doi:10.1109/QCE49297.2020.00021.

[9] S. Brandhofer, D. Braun, V. Dehn, G. Hellstern, M. Hüls, Y. Ji, I. Polian, A. S. Bhatia, T. Wellens, Benchmarking the performance of portfolio optimization with QAOA, Quantum Information Processing 22 (2022) 25. URL: https://doi.org/10.1007/s11128-022-03766-5. doi:10.1007/s11128-022-03766-5.

[10] Z. Tabi, K. H. El-Safty, Z. Kallus, P. Haga, T. Kozsik, A. Glos, Z. Zimboras, Quantum optimization for the graph coloring problem with space-efficient embedding, in: 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2020, p. 56–62. URL: http://dx.doi.org/10.1109/QCE49297.2020.00018. doi:10.1109/qce49297.2020.00018.

[11] C. Y.-Y. Lin, Y. Zhu, Performance of QAOA on typical instances of constraint satisfaction problems with bounded degree, 2016. arXiv:1601.01744.

[12] K. Kurowski, T. Pecyna, M. Slysz, R. Różycki, G. Waligóra, J. Weglarz, Application of quantum approximate optimization algorithm to job shop scheduling problem, European Journal of Operational Research 310 (2023) 518–528. URL: https://www.sciencedirect.com/science/article/pii/S0377221723002072. doi:https://doi.org/10.1016/j.ejor.2023.03.013.

[13] G. Turati, M. F. Dacrema, P. Cremonesi, Feature Selection for Classification with QAOA , in: 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE Computer Society, Los Alamitos, CA, USA, 2022, pp. 782–785. URL: https://doi.ieeecomputersociety.org/10.1109/QCE53715.2022.00117. doi:10.1109/QCE53715.2022.00117.

[14] H. Irie, G. Wongpaisarnsin, M. Terabe, A. Miki, S. Taguchi, Quantum annealing of vehicle routing problem with time, state and capacity, in: Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings 1, Springer, 2019, pp. 145–156.

[15] C. Silva, A. Aguiar, P. M. Lima, I. Dutra, Mapping a logical representation of tsp to quantum annealing, Quantum Information Processing 20 (2021) 386.

[16] R. H. Warren, Solving the traveling salesman problem on a quantum annealer, SN Applied Sciences 2 (2020) 75.

[17] A. Lytrosyngounis, et al., Hybrid quantum-classical optimisation of traveling salesperson problem, arXiv preprint arXiv:2503.00219 (2025).

[18] N. Yanakiev, N. Mertig, C. K. Long, D. R. Arvidsson-Shukur, Dynamic-adapt-QAOA: An algorithm with shallow and noise-resilient circuits, arXiv preprint arXiv:2309.00047 (2023).

[19] A. Bärtschi, S. Eidenbenz, Grover mixers for QAOA: Shifting complexity from mixer design to state preparation, in: 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2020, pp. 72–82.

[20] J. Preskill, Noisy intermediate-scale quantum computers, Quantum 2 (2018) 79. URL: https://doi.org/10.22331/q-2018-08-06-79. doi:10.22331/q-2018-08-06-79.

[21] F. Glover, G. Kochenberger, R. Hennig, Y. Du, Quantum bridge analytics i: A tutorial on formulating and using QUBO models, Annals of Operations Research 314 (2022) 141–183. URL: https://doi.org/10.1007/s10479-022-04634-2. doi:10.1007/s10479-022-04634-2.

[22] A. Lucas, Ising formulations of many np problems, Frontiers in physics 2 (2014) 5.

[23] M. Cattelan, S. Yarkoni, Modeling routing problems in qubo with application to ride-hailing, 2022. URL: https://arxiv.org/abs/2212.04894. arXiv:2212.04894.

[24] U. Azad, B. K. Behera, E. A. Ahmed, P. K. Panigrahi, A. Farouk, Solving vehicle routing problem using quantum approximate optimization algorithm, IEEE Transactions on Intelligent Transportation Systems 24 (2023) 7564–7573. URL: http://dx.doi.org/10.1109/TITS.2022.3172241. doi:10.1109/tits.2022.3172241.

[25] N. Mohanty, B. K. Behera, C. Ferrie, Solving the vehicle routing problem via quantum support vector machines, Quantum Machine Intelligence 6 (2024). URL: http://dx.doi.org/10.1007/s42484-024-00161-4. doi:10.1007/s42484-024-00161-4.

[26] M. Radzihovsky, J. Murphy, M. Swofford, A QAOA solution to the traveling salesman problem using pyquil, 2019.

[27] C. Papalitsas, T. Andronikos, K. Giannakis, G. Theocharopoulou, S. Fanarioti, A qubo model for the traveling salesman problem with time windows, Algorithms 12 (2019). URL: https://www.mdpi.com/1999-4893/12/11/224. doi:10.3390/a12110224.

[28] J.-F. Cordeau, Q. Groupe d'études et de recherche en analyse des décisions (Montréal, The VRP with time windows, Groupe d'études et de recherche en analyse des décisions Montréal, 2000.

[29] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271. doi:10.1007/BF01386390.

[30] H. Bayraktar, A. Charara, D. Clark, S. Cohen, T. Costa, Y.-L. L. Fang, Y. Gao, J. Guan, J. Gunnels, A. Haidar, et al., cuquantum sdk: A high-performance library for accelerating quantum science, in: 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 1,

IEEE, 2023, pp. 1050–1061.
[31] C. Campbell, E. Dahl, QAOA of the highest order, in: 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C), IEEE, 2022, pp. 141–146.
[32] M. J. D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in: Advances in Optimization and Numerical Analysis, Springer Netherlands, 1994, pp. 51–67. URL: https://doi.org/10.1007%2F978-94-015-8330-5_4. doi:10.1007/978-94-015-8330-5_4.
[33] H. Singh, S. Majumder, S. Mishra, Benchmarking of different optimizers in the variational quantum algorithms for applications in quantum chemistry, The Journal of Chemical Physics 159 (2023) 044117. URL: https://doi.org/10.1063/5.0161057. doi:10.1063/5.0161057.
[34] M. Fernández-Pendás, E. F. Combarro, S. Vallecorsa, J. Ranilla, I. F. Rúa, A study of the performance of classical minimizers in the quantum approximate optimization algorithm, Journal of Computational and Applied Mathematics 404 (2022) 113388. URL: https://www.sciencedirect.com/science/article/pii/S0377042721000078. doi:https://doi.org/10.1016/j.cam.2021.113388.
[35] M. Ayodele, Penalty weights in qubo formulations: Permutation problems, in: European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar), Springer, 2022, pp. 159–174.
[36] J. Choi, J. Kim, A tutorial on quantum approximate optimization algorithm (QAOA): Fundamentals and applications, in: 2019 International Conference on Information and Communication Technology Convergence (ICTC), 2019, pp. 138–142. doi:10.1109/ICTC46691.2019.8939749.
[37] B. Bakó, A. Glos, O. Salehi, Z. Zimborás, Prog-QAOA: Framework for resource-efficient quantum optimization through classical programs, Quantum 9 (2025) 1663. URL: http://dx.doi.org/10.22331/q-2025-03-20-1663. doi:10.22331/q-2025-03-20-1663.