Ansatz Optimization using Simulated Annealing in Variational Quantum Algorithms for the TSP

Fabrizio Fagiolo
National Research Council (CNR)
Rome, Italy
fabrizio.fagiolo@istc.cnr.it

Nicolò Vescera
Università degli Studi di Perugia
Perugia, Italy
nicolo.vescera@collaboratori.unipg.it

***Abstract*—In this paper we present a Variational Quantum Algorithm (VQA) for solving the Traveling Salesman Problem (TSP) that requires only $\mathcal{O}(n \log n)$ qubits and an ansatz whose topology evolves via Simulated Annealing (SA). Instead of fixing the circuit in advance, the optimizer dynamically adds, removes, or rearranges rotation gates and entanglement gates to balance exploration and exploitation. On synthetic instances with $5-7$ cities ($7-13$ qubits, $21-39$ parameters), the algorithm identifies the optimal tour in almost all its runs within a few hundred iterations, demonstrating that co-optimization of encoding and circuit structure can efficiently address combinatorial problems.**

I. Introduction

One of the most relevant applications of quantum computing is optimization, which can be pursued through two distinct strands: the use of Quantum Annealers, specially designed hardware devices for minimizing cost functions by quantum annealing processes; or the implementation of optimization algorithms on quantum gate computers.

In this paper we explore the second approach, proposing a variational algorithm for solving the Traveling Salesman Problem (TSP) [1] that uses compact permutational encoding and an automatically optimized adaptive ansatz. In this way the algorithm requires just $\mathcal{O}(n \log n)$ qubits and keeps the circuit shallow, maximizing its performance.

The Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA) are two of the most widely recognized gate-based quantum algorithms. Both adopt a variational strategy, in which a parameterized quantum circuit is optimized by a classical continuous routine.

In the VQE we prepare a parametric ansatz $|\psi(\boldsymbol{\theta})\rangle$ and optimize the parameters $\boldsymbol{\theta}$ by minimizing

$$E(\boldsymbol{\theta}) \;=\; \langle\psi(\boldsymbol{\theta})\,|\,\hat{H}\,|\psi(\boldsymbol{\theta})\rangle,$$

thus obtaining an approximation of the ground-state energy E_{min} of $\hat{H}$ [2], [3]. The circuit's shallow depth and flexibility make the algorithm particularly suitable for NISQ hardware [4], [5].

In the QAOA the system evolves through p layers, alternating the two unitaries:

$$\begin{array}{l} U_C(\gamma_k) = e^{-i\gamma_k \hat{H}_C}, \\ U_B(\beta_k) = e^{-i\beta_k \hat{H}_B}, \end{array} \qquad k = 1, \ldots, p, \qquad \hat{H}_B = \sum_{j=1}^{n} X_j.$$

These gates build the variational state

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \Big(\prod_{k=1}^{p} U_B(\beta_k)\, U_C(\gamma_k)\Big)\, |+\rangle^{\otimes n},$$

where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ are optimized to minimize the expectation value of the Hamiltonian cost $\hat{H}_C$ [6], [7]. Because the circuit reflects $\hat{H}_C$, it directs the exploration of Hilbert space toward the states relevant for solving the underlying optimization problem.

In this paper, we present a solution for the TSP problem, which consists of finding the shortest path for a salesman, given a starting point, a number of cities and, an ending point.

TSP may be viewed formally as follows. Let $V = \{v_1, v_2, \ldots, v_n\}$ be a finite set of n cities and let $c : V \times V \to R_{\geq 0}$ assign a travel cost $c_{ij} = c(v_i, v_j)$ to every ordered pair of cities, typically interpreted as distance. A tour is a permutation π of the indices $1, \ldots, n$, which induces the closed walk $v_{\pi(1)} \to v_{\pi(2)} \to \cdots \to v_{\pi(n)} \to v_{\pi(1)}$. The total cost of this tour is

$$C(\pi) = \textstyle\sum_{k=1}^{n} c_{\pi(k)\,\pi(k+1)},$$

with the convention $\pi(n+1) = \pi(1)$.

The optimization task is to find a permutation π^* whose cost $C(\pi^*)$ is minimal among all $n!$ possible permutations, thereby yielding the shortest possible closed route that starts and ends at the same city while visiting every other city exactly once. Because determining π^* belongs to the class of NP-hard problems, exact solutions become computationally prohibitive as n grows.

An important point of decision when approaching TSP resolution using QAOA or VQE is how to represent a solution of the problem. Since it is necessary to express the distance (or other objective functions) as a Hamiltonian, a very large number of qubits is usually required.

For example, in [8], different encoding methods of the TSP are analyzed for its resolution by variational quantum algorithms, such as the QAOA and the VQE. The study compares the performance and scalability of different encodings, providing clear guidance on which one to choose based on specific needs.

A critical design choice for variational quantum TSP solvers is the mapping of tours to qubit registers. The straightforward QUBO or HUBO encodings require $\mathcal{O}(n^2)$ qubits and additional penalty terms. By contrast, permutation encodings such as Lehmer codes achieve $\mathcal{O}(n \log n)$ qubits while ensuring that every basis state represents a valid tour, as shown by Schnaus et al. [8]. We adopt this compact encoding in order to fit larger TSP instances.

Existing studies typically assume a fixed circuit structure to solve the TSP problem. We depart from this practice by optimizing the ansatz itself. Inspired by evolutionary algorithms, we iteratively modify the circuit topology by adding, removing or permuting gates, and accept or reject mutations through a Simulated Annealing (SA) [9] algorithm that balances exploration and exploitation.

The result is an adaptive ansatz that evolves to architectures best suited to the TSP instance in question, while remaining shallow enough for NISQ execution. To calculate the mean value of the objective function, we adopt an empirical estimation consisting of three steps: (i) we generate N solutions using the best ansatz identified, (ii) we calculate the objective function for each solution, (iii) we calculate the average of the obtained values. Using this approach, it is possible to design an ansatz that generates only valid solutions. Thus, the objective function used in the optimization process corresponds to the empirical average of the costs of the produced solutions.

The paper is organized as follows. Section II describes the related work. Section III introduces the scheme of the proposed approach. Section IV describes and comments the experimental results. Section V ends the paper by providing some concluding remarks as well as possible future lines of research.

II. Related work

In variational quantum algorithms, the ansatz determines which quantum states can be visited by the circuit, and thus limits the solution space that the algorithm is able to explore. In quantum chemistry, for example, it has already been seen that different choices of ansatz, from the hardware-efficient circuits of Kandala et al. [3], decisively affect the trade-off between circuit depth and quality of results.

The same is true for combinatorial problems such as TSP. Comparing architectures (annealer vs gate-based) and forms of the ansatz, Venkat et al. [10] point out that both choices are decisive in quality and solution time. Murhaf et al. [11] take advantage of D-Wave's Pegasus and a QUBO reformulation that eliminates the initial node: with fewer qubits they get better tours than D-Wave's ”standard” TSP solver and solve problems up to 20 cities. The opposite approach of Jain et al. [12], without optimizations, stops at 8 cities and does not report good performance compared to the previous approach.

Despite few qubits, gate-based computers can still compete with quantum annealers through targeted encoding and ansatz. Ruan et al. [13] integrate TSP constraints into the QAOA mixer, halve the qubits and double the probability of finding the optimal tour. Qian et al. [14] test three alternative mixers, improved edge-based encoding and layered learning, showing the best depth-quality trade-off even with realistic noise.

Instead, Meng Shi et al. [15] propose APIs that automatically transform any problem on graphs (including TSP) into circuits that are solved by VQA, obtaining with a not-too-high number of qubits solutions equal to or better than specialized algorithms.

All these evidences clearly indicate that an ansatz designed on the problem is crucial. In our work, we explore different internal configurations, such as different spin gates and different entanglement models, using SA to find the most effective configuration to solve the TSP problem.

III. Proposed solution

The proposed method introduces two key innovations: compact permutation-based coding and an adaptive ansatz whose structure evolves during the optimization process. Differently from most existing studies that rely on a predefined circuit topology, we co-optimize the ansatz layout itself to better fit the problem instance. This is achieved by applying SA to iteratively search for the most effective sequence of port layers, balancing rotation operations and entanglement patterns.

A. Encoding

As shown in [8], encoding allows reducing the space required for a tour. In practice, we start from the permutation π of n cities and convert it into a single integer P_π between 0 and $n! - 1$ using permutation encode. To re-obtain the original permutation, we can apply the decoding procedure outlined in Algorithm 1, which requires only $\mathcal{O}(n)$ operations and is thus highly efficient even for large values of n.

Algorithm 1: Permutation decoding function

```
function PermutationDecoding(P) is
    R ← [1, 2, ..., n];
    for i ← n downto 1 do
        j ← 1 + (P mod i);
        π(n + 1 − i) ← R_j;
        remove R_j from R;
        L ← ⌊P/i⌋;
    end
    return π;
end
```

The number of qubits required by encoding is

$$\lceil \log_2(n!) \rceil = \mathcal{O}(n \log n).$$

For example, with $n = 5$ cities there are $5! = 120$ permutations; since $(\log_2(120) \simeq 6.91)$, are sufficient $(\lceil 6.91 \rceil = 7)$ qubits to represent any tour, regardless of the number of machines or processing times.

B. Ansatz Optimization

Another key node of our work is the use of the SA optimization algorithm, to find the best Ansatz for the TSP. We adopt it because of its ability to find an approximate global optimum over a large search space, avoiding local optimum. We opted for a Single-Individual Algorithm over a Population-based one because it can devote all its resources to thoroughly exploring and refining a single solution, enabling faster convergence and more in-depth local search. Our method is given in Algorithm 2.

Algorithm 2: SA for Ansatz Optimization

```
Data: tsp_instance_file
Result: Best Ansatz with parameters
1  T ← 1.0;
2  CoolingRate ← 0.999;
3  min_T ← 1 × 10^{-3};
4  max_iter ← 500;
5  iteration ← 0;
6  S_best ← generate a random solution;
7  S_current ← S_best;
8  while T > min_T and iteration < max_iter do
9      S_neighbor ← Perturbate(S_current);
10     ΔE ← Fitness(S_neighbor) − Fitness(S_current);
11     if ΔE > 0 or RandomNumber() < e^{ΔE/T} then
12         S_current ← S_neighbor;
13         if Fitness(S_current) > Fitness(S_best) then
14             S_best ← S_current;
15         end
16     end
17     T ← T × CoolingRate;
18     iteration ← iteration +1;
19 end
```

Selecting the optimal ansatz size is a delicate task, since a overly large circuit leads to excessive computational costs, whereas a undersized one yield suboptimal results. That being said, we parameterize a solution S using a sequence of 5 non-negative integers between $[0, 7]$.

$$S = < x_0, x_1, x_2, x_3, x_4 >,$$
$$x_i \in \begin{cases} \{0, 1, 2\} & \text{if } i \text{ is even} \\ \{3, 4, 5, 6, 7\} & \text{if } i \text{ is odd} \end{cases} \tag{1a}$$

Elements with even indices represent the possible types of rotation (R_x, R_y, R_z), while those in odd positions will define the entanglement types obtained through the gate CX (see [16]). The associations between the numerical value and its meaning are listed in Table I. There can be no solution where rotation blocks are replaced by entanglement blocks, or vice versa. The solution S will then be converted into a quantum circuit consisting of 5 blocks, where blocks with even indices will be composed of rotation gates, while those with odd indices will be entanglement blocks. A rotation block consists of a single type of spinning gate, with no mixing of different types allowed. Each gate within a block has its own parameter $\Theta_i[j]$, where i is the block index and j is the qubit index.

Let us take into account the following example:

$$S = < 1, 3, 2, 7, 2 > \tag{2}$$

This solution is translated into the quantum circuit showed in Figure 1. We can see how the first element x_0 has a value of 1, so it will be translated into a block composed only by R_y gates, while the second element x_1 has a value of 3 and will be converted into an entanglement block composed by CX gates arranged in $linear$ entanglement order, and so on.

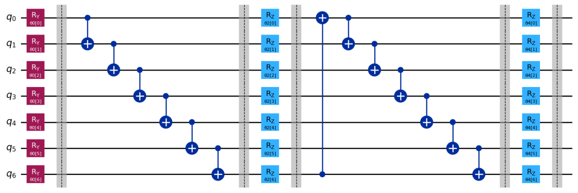

Figure 1: Quantum Circuit associated to example in (2)

The perturbations of the current solution, implemented by the $Perturbate$ function (line 9), involve randomly selecting an element $t \in \{0, 1, 2, 3, 4\}$ from S and replacing it with a legal distinct value. In this way, we can alter S, generating $S_{neighbor}$, without violating the rules defined previously. The quantum circuit will be updated accordingly. Formally, we can define this operation as:

$$\text{Randomly choose } t \in \{0, 1, 2, 3, 4\}$$
$$S_{neighbor} = \{x'_0, x'_1, x'_2, x'_3, x'_4\},$$
$$\text{where } x'_j = \begin{cases} x_j & \text{if } j \neq t \\ f(x_t) & \text{else} \end{cases} \tag{3a}$$

$$f : \{0, 1, 2, 3, 4, 5, 6, 7\} \to \{0, 1, 2, 3, 4, 5, 6, 7\},$$
$$f(x_t) = \begin{cases} v \in \{0, 1, 2\} \setminus \{x_t\} & \text{if } t \text{ is even} \\ v \in \{3, 4, 5, 6, 7\} \setminus \{x_t\} & \text{if } t \text{ is odd} \end{cases} \tag{3b}$$

The fitness value for a given solution is calculated based on the result of executing the associated quantum circuit. First of all, S is transformed into its corresponding quantum circuit. Then, 100 random vectors of angular parameters in the range $[0, 2\pi]$ are generated and their energy is calculated. The 10 vectors with the lowest energy values are then selected and used as a starting point for the next phase. Using the Powell algorithm [17], we optimize the parameters to be applied to our quantum circuit. Powell algorithm has been selected because, after extensive experimentation conducted with both gradient-based and gradient-free optimization algorithms, it results the best in terms of effectiveness and reliability.

Subsequently, the circuit with optimized parameters is executed N times and the results are recorded. Finally, the fitness value is calculated as the ratio between the number of times the optimal permutation has been found and the number of times the circuit was executed. All this is summarized by the $Fitness$ procedure in Algorithm 3.

Algorithm 3: $Fitness$ procedure

```
1  function Fitness(S, T: int) → float is
2      execute $RunVQA(S)$ T-times and store the results;
3      f ← compute the average value over stored results;
4      return f;
5  end
6  function RunVQA(S) → list[float] is
7      $\mathcal{A}$ ← convert S to Quantum Circuit;
8      samples ← generate 100 random vector in range $[0, 2\pi]$;
9      compute $energy$ for each sample;
10     starting_point ← find first 10 vectors with lowest energy;
11     results ← empty list;
12     for $n$-times do
13         optimize $\mathcal{A}$ parameters using Powell's method starting
             from starting_point;
14         apply found parameters to $\mathcal{A}$;
15         res ← execute VQA using $\mathcal{A}$ ($N = 1024$) and store
             the results;
16         $P_{opt}$ ← $\frac{\text{number of correct solution found}}{N}$;
17         add $P_{opt}$ to results;
18     end
19     return results;
20 end
```

Finally, at line 11 of Algorithm 2, we can see how the current solution $S_{current}$ is replaced with $S_{neighbor}$ using the Metropolis criterion [18]. This criterion is a key component of SA, enabling the algorithm to efficiently search for global optima in complex optimization problems, balancing exploitation, exploration and hopefully avoiding local optima.

Table I: Mapping Solution element to gate

x_i	**Gate**
0	R_x gate
1	R_y gate
2	R_z gate
3	CX linear
4	CX reverse linear
5	CX full
6	CX circular
7	CX sca

Table II: SA Parameters

Parameter	**Value**
T	1.0
$CoolingRate$	0.999
min_T	1×10^{-3}
max_iter	500

IV. Experiments

This section presents the experimental results of applying our VQA approach to the TSP.

A. Problem setup

We generate TSP instances using a controlled procedure, where each instance is represented by a symmetric $n \times n$ distance matrix, with n denoting the number of cities. The distances between cities are randomly generated integers within a predefined range (e.g., 1 to 100). To ensure validity, the self-distance of each city is set to 0. This generation modality allows to efficiently build instances to test and compare algorithms on different levels of complexity and problem size.

We generated a total of 9 TSP instances, consisting of 3 independent instances for each of 3 city sizes: 5, 6, and 7. This allows us to evaluate the algorithms' performance with increased robustness.

In this way, the optimal tour cost can also be calculated through a complete enumeration of all city permutations, providing an exact reference solution. This optimal value can be used to evaluate the performance of our VQA approach, comparing the cost of the found tours with the known minimum value.

The number of qubits and ansatz parameters required for each instance size is showed in Table III.

Table III: Qubit count required for different city sizes.

City nodes	**Qubits used**	**Ansatz parameters**
5	7	21
6	10	30
7	13	39

B. Experimental Setup

The Algorithm 2 is implemented in Python with Qiskit [16] library.

For each number of cities n considered, we conducted five independent runs of the VQA procedure. In each run we first computed the final average tour cost, denoted by T, obtained from the best optimized quantum circuit and evaluated on 1024 shots. We then stored the entire distribution of distance values produced in these 1024 shots and, finally, counted how many of the measured bit-strings denoted by P_{opt} reach a distance exactly equal to the optimal value e_{min}, determined by exhaustive enumeration of permutations. We let SA runs for 500 iterations.

C. Results

In this section we describe in detail the results with respect to the experimental method adopted to evaluate the effectiveness of our VQA on the TSP problem. The goal is to demonstrate how the proposed method behaves as the problem size varies (5 to 7 cities) and to quantify the average cost of the tour and the probability of finding the optimal solution.

1) Convergence to the optimal solution: For each of the 9 TSP problems (3 instances of 5, 6 and 7 cities) we report the evolution, of the average probability of obtaining the optimal solution, denoted as P_{opt}. Observing the trend of this metric allows us to understand how quickly VQA approaches the minimum tour and how long it takes to consolidate the optimum.

The results in 5 cities, showed in Figure 2, are very similar each other. With only 5 nodes, the algorithm immediately approaches the optimum and then slowly files down the last decimal places until P_{opt} reaches an average of 100%. This highlights that the algorithm for such a small problem reaches the best result almost automatically and then consumes iterations to get to the optimum.

The chart corresponding to the instances with 6 cities tell almost the same story as seen with 5 cities. Immediately after the first few steps, the algorithm again catapults itself from an approximate starting solution to well over 95% of the optimum. In two of the runs this jump occurs in less than 10 iterations; in the third instance, which was already starting from an higher P_{opt}, the jump is almost instantaneous. What changes, with that extra city, is what happens next: the curve flattens out much longer. For dozens, sometimes a couple of hundred iterations, the best-fitness line is a straight plateau, showing that the research is busy exploring micro-movements that still do not improve the score. Eventually a second, smaller jump occurs, bringing the P_{opt} to $\sim 99.8 - 99.99\%$. In the case of instances with 7 cities, showed in Figure 2, the situation becomes even more complicated. The behavior of the curves is less uniform and shows greater variability across runs. These trends suggest that, as the number of cities increases, the difficulty of the problem increases significantly so the algorithm may take many more iterations to get out of optimal locales, and it may not always do so successfully.

2) Best Ansatz found: Figure 3 shows the variational circuit that, after the optimization phase, provided the lowest tour value for the second instance with 5 cities. The architecture employs all 7 available qubits and is divided into 3 parametric blocks separated by barriers that highlight the logical structure.

The optimal ansatz follows a simple but effective scheme:

$$\begin{aligned} &\mathrm{R}_z \text{ layer} \longrightarrow \text{full entanglement} \longrightarrow \mathrm{R}_z \text{ layer} \\ &\quad \longrightarrow \text{linear entanglement} \longrightarrow \mathrm{R}_z \text{ layer.} \end{aligned}$$

Concretely, each qubit is first rotated around the z axis (R_z). These single qubit phases are then mixed by a fully connected CNOT block, which connects each qubit with all others. A second layer of R_x refines the phases introduced so far, after which a CNOT scheme provides linear entanglement. The circuit closes with the R_z layer.

V. Conclusions and Future Work

In this paper, we introduced a VQA for the TSP problem that combines two key ingredients. On the one hand, the use of compact permutational encoding reduces the qubit requirement to $\mathcal{O}(n \log n)$ and ensures that every state of the computational basis represents a valid tour, thus eliminating the need for heavy penalty terms. On the other hand, the topology of the circuit is not predetermined, but evolves along with its continuous parameters via SA algorithm. The Metropolis criterion balances exploration and exploit, allowing the circuit to be kept shallow while still identifying highly performing configurations for the reference instance.

Tests on synthetic 5, 6, and 7-city instances show that the algorithm quickly reaches the optimal round with a probability close to unity (probability $\simeq 1$ for $n = 5$ and > 0.998 for $n = 6$), and maintains good performance even in the 7-city case, although with greater variability in the results and longer convergence times. The algorithm is also shown to be robust to growth in the number of qubits and parameters (from $7/21$ to $13/39$) by identifying optimized gate schemes for the problem. These results confirm that optimizing the circuit architecture with SA can be a decisive factor in finding good solutions to combinatorial problems.

Although the results so far are encouraging, several avenues still need to be explored. A possible future development is to run the algorithm on TSP instances with 10 or more cities, to assess how the quality of the solution is affected by longer circuits and to see if increasing the number of qubits and circuit depth due to larger problem sizes still leads to good results. Another possible development is to discover a high-performance ansatz on a small training set, "freeze" its gate sequence, and then re-optimize only the continuous parameters when facing new instances. This will allow us to quantify the time saved and the stability of the quality of the solution. In the future, one can also expand the search space of SA with more expressive operations, such as swapping entire blocks of gates, inserting additional controlled gates, or replacing gates with SWAP operations, the

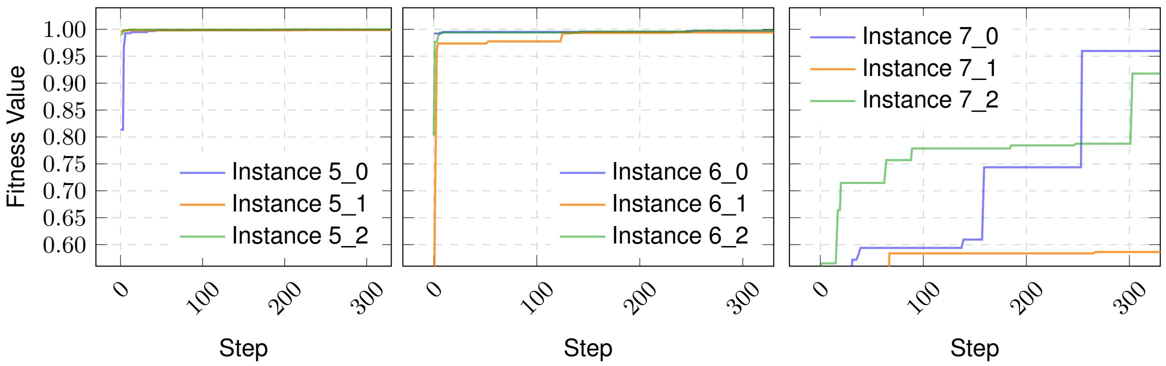

Figure 2: Average probability of obtaining an optimal solution for the three TSP instances with 5, 6, 7 cities. The images highlight the first 300 iterations of SA, displaying only fitness values greater than 0.6.

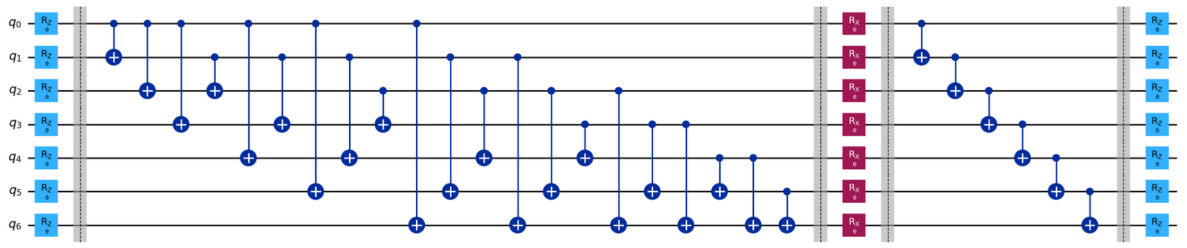

Figure 3: Best Ansatz found for second instance with 5 nodes

different structure should help the optimizer escape local minima and discover better circuit architectures.

A systematic study of these lines of research will elucidate how scalable the algorithm is, how fast it converges, and which circuit structures fit one instance or another more effectively, insights that will be crucial for implementing the method on upcoming NISQ hardware.

References

[1] M. Jünger, G. Reinelt, and G. Rinaldi, "Chapter 4 the traveling salesman problem," in *Network Models*, vol. 7 of *Handbooks in Operations Research and Management Science*, pp. 225–330, Elsevier, 1995.

[2] A. Peruzzo, J. Mcclean, P. Shadbolt, M. H. Yung, X. Zhou, P. Love, A. Aspuru-Guzik, and J. O'Brien, "A variational eigenvalue solver on a quantum processor," *Nature communications*, vol. 5, 04 2013.

[3] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, p. 242–246, Sept. 2017.

[4] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[5] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, "Noisy intermediate-scale quantum algorithms," *Reviews of Modern Physics*, vol. 94, Feb. 2022.

[6] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014.

[7] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, Jan. 2018.

[8] M. Schnaus, L. Palackal, B. Poggel, X. Runge, H. Ehm, J. M. Lorenz, and C. B. Mendl, "Efficient encodings of the travelling salesperson problem for variational quantum algorithms," in *2024 IEEE International Conference on Quantum Software (QSW)*, pp. 81–87, July 2024.

[9] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical science*, vol. 8, no. 1, pp. 10–15, 1993.

[10] V. Padmasola, Z. Li, R. Chatterjee, and W. Dyk, "Solving the traveling salesman problem via different quantum computing architectures," 2025.

[11] M. Alawir, M. A. Alatasi, H. Salloum, and M. Mazzara, "Enhanced quantum annealing tsp solver (eqats): Advancements in solving the traveling salesman problem using d-wave's quantum annealer," 12 2024.

[12] S. Jain, "Solving the traveling salesman problem on the d-wave quantum computer," *Frontiers in Physics*, vol. 9, 2021.

[13] Y. Ruan, S. Marsh, X. Xue, Z. Liu, and J. Wang, "The quantum approximate algorithm for solving traveling salesman problem," *Computers, Materials & Continua*, vol. 63, no. 3, pp. 1237–1247, 2020.

[14] W. Qian, R. A. M. Basili, M. M. Eshaghian-Wilner, A. Khokhar, G. Luecke, and J. P. Vary, "Comparative study of variations in quantum approximate optimization algorithms for the traveling salesman problem," *Entropy*, vol. 25, no. 8, 2023.

[15] M. Shi, S. Wu, Y. Li, G. Yuan, C. Yao, and G. Chen, "A quantum framework for combinatorial optimization problem over graphs," *Data Science and Engineering*, vol. 10, no. 2, pp. 246–257, 2025.

[16] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with qiskit," 2024.

[17] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, pp. 155–162, 01 1964.

[18] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 06 1953.