Quantum Optimization for Energy-Efficient Route-Finding

Cherilyn Christen, Nathaniel Pacey, Lindon Zymberi
Quantum Engineering Section, EPFL

Abstract

This report compares the energy consumption of classical brute-force, Held–Karp, and Quantum Approximate Optimization Algorithm (QAOA) approaches for solving the Traveling Salesman Problem (TSP). Using closed-form scaling formulas and execution data from IBM's Sherbrooke quantum processor, the analysis identifies crossover points where quantum methods become more energy-efficient than classical counterparts. While classical algorithms scale factorially or exponentially, quantum variants maintain near-constant energy use. These results highlight the potential of quantum optimization to cut computational energy demands in large-scale applications, supporting UN Sustainable Development Goals 9, 11, and 12, despite current hardware and modeling limitations.

1 Data Centers and their Energy Usage

Data centers are specialized facilities that host large networks of servers, storage systems, and networking equipment. They support the digital backbone of our world, powering services like Google Maps, Uber, Netflix, AI models, national infrastructure, and financial systems. As global reliance on cloud computing and AI continues to grow, the energy impact of data centers becomes increasingly critical [1, 2].

For instance, the total energy consumed by Google searches worldwide in a single day, estimated at roughly 4.2 gigawatt-hours (GWh), could power around 1,200 average European households for an entire year, assuming an annual consumption of about 3,500 kilowatt-hours (kWh) per household [3]. Although a single search requires only a fraction of a watt-hour, the scale of global usage, exceeding 14 billion searches daily, accumulates into a substantial energy demand, emphasizing the often overlooked environmental footprint of routine digital activities.

According to the International Energy Agency (IEA), data centers currently consume about 1–2% of global electricity, a figure projected to rise to 6–8% by 2030 [2]. A significant portion of this energy is consumed by compute-heavy hardware, with servers, CPUs, and GPUs accounting for 40–60% of total use.

The graph in Fig. 1 illustrates the projected growth in electricity consumption by data centers through 2030, segmented by region. The United States and China are expected to account for nearly 80% of the global increase, with both nations showing steep upward trends. Europe and the rest of Asia follow with moderate yet consistent growth. If current regulatory and industry trends continue, global data center electricity usage is forecast to surpass 1,000 TWh by 2030, highlighting a pressing need for more energy-efficient computing infrastructure [2].

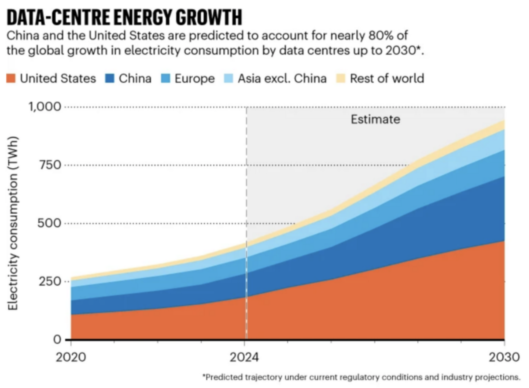

Figure 1: Projected energy growth of data centers by region. Source: IEA [2]. CC BY 4.0

2 Route Optimization Algorithms

Route optimization algorithms are at the core of modern logistics, enabling efficient decisions in scenarios ranging from web search and train scheduling to parcel delivery. These algorithms analyze millions of possible paths and select those that best satisfy specific objectives, such as minimizing travel time, distance, or energy consumption. The complexity of the underlying task determines the computational resources required—simple cases may be solvable in milliseconds, while others become exponentially harder [4].

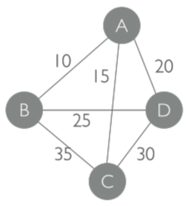

Figure 2: Graph representation of delivery locations with travel times

A common example is the daily challenge faced by UPS delivery trucks illustrated in Fig. 2. Given a set of delivery locations (node B, C, D), the vehicle must start and end its journey at the post office (node A) and visit every destination exactly once. Each edge is labeled with the travel time or distance between two locations. Finding the most efficient route is crucial for saving time, fuel, and computational resources, especially when scaled to thousands of deliveries per day [4].

3 Traveling Salesman Problem (TSP)

The UPS delivery problem is a classic instance of the Traveling Salesman Problem (TSP), a benchmark in combinatorial optimization that grows factorially with the number of stops [5]. The TSP considers a list of n cities and a cost function $d(i, j)$ representing the travel cost between each pair of cities i and j. The objective is to determine the shortest possible closed route that begins and ends at the same city and visits every other city exactly once.
Although it appears simple, the TSP becomes computationally intractable as the number of locations increases. The number of possible routes scales as $(n-1)!/2$, quickly rendering brute-force search infeasible for even moderate values of n [6].
Formally, the TSP seeks a permutation π of the cities that minimizes the total tour cost:

$$\text{Cost}(\pi) = \sum_{i=0}^{n-2} d(\pi(i), \pi(i+1)) + d(\pi(n-1), \pi(0))$$

This expression captures the total length of a closed loop that visits all cities once.
Because of its high complexity, the TSP has motivated the development of a wide variety of solution strategies, from exact algorithms like dynamic programming and branch-and-bound [7], to heuristic methods such as nearest neighbor, simulated annealing, and genetic algorithms [8]. These approaches represent a trade-off between solution quality and computational cost.

4 Algorithm Performance Metrics

The performance of an algorithm is typically evaluated in terms of its computational complexity, expressed using Big O notation [6]. This metric describes how the number of operations required grows with input size and is crucial for predicting how well an algorithm scales. For instance, while an $O(n)$ algorithm grows linearly with problem size, an $O(n!)$ algorithm becomes impractical very quickly. Importantly, each computational step consumes energy, so higher complexity not only leads to slower execution but also to increased energy consumption. Thus, optimizing algorithmic efficiency is key not just for speed but also for reducing environmental impact in large-scale deployments.

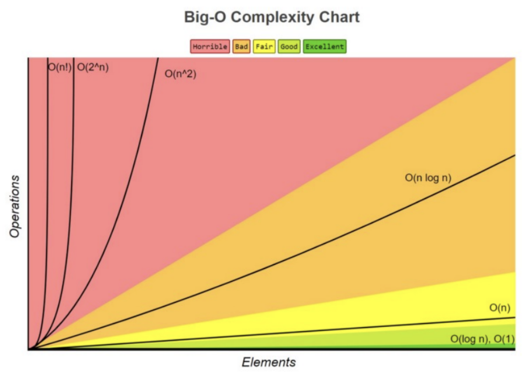

Figure 3: Big O complexity chart illustrating algorithm scalability and performance [9].

5 Classical vs Quantum Optimization

5.1 Classical Brute-Force Method

The classical brute force method for solving the Traveling Salesman Problem (TSP) involves exhaustively evaluating all possible permutations of city visits to determine the shortest possible route. For a graph of N cities, this means calculating $(N-1)!$ [1] potential paths, fixing the starting city. As illustrated in the Fig. 4, even for just 4 cities, the decision tree expands rapidly, underscoring the factorial growth in computational workload. Although brute force is guaranteed to find the optimal route, its factorial time complexity $O(n!)$ renders it impractical for larger instances [6]. On the upside, the memory usage is minimal, as only the current shortest path needs to be stored at any given time.

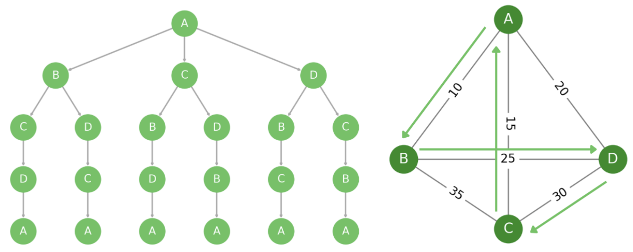

Figure 4: Brute Force TSP: Graph and Decision Tree

5.2 Classical Held-Karp (Dynamic Programming) Method

The Held-Karp algorithm is a classical dynamic programming approach to the Traveling Salesman Problem (TSP) that substantially improves over brute-force methods by avoiding redundant computations [7]. It works by solving smaller subproblems once and storing their results, which are then reused to build up solutions to larger sets. The core recurrence relation computes

[1] The factorial of a non-negative integer n, denoted $n!$, is the product of all positive integers less than or equal to n. For example, $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$. By convention, $0! = 1$.

the shortest path that ends at a specific city after visiting a subset of cities, progressively extending the path one node at a time. This reduces the computational complexity from $O(n!)$ to $O(n^2 \cdot 2^n)$, a significant improvement for moderate problem sizes. However, this efficiency gain comes at the cost of higher memory usage, as the algorithm requires caching of intermediate solutions. While still exponential, Held-Karp is tractable for larger instances than brute-force and reliably finds the optimal route.

5.3 Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical approach designed to tackle combinatorial optimization problems such as the Traveling Salesman Problem (TSP) [10]. It operates by encoding the problem into a quantum system via two central Hamiltonians: the cost Hamiltonian, which represents the objective function (e.g., the total distance of a tour), and the mixer Hamiltonian, which facilitates exploration of the solution space.

The cost Hamiltonian is defined as:

$$H_C = \sum_{(i,j)} w_{ij} \cdot \frac{1 - Z_i Z_j}{2},$$

where w_{ij} is the distance between cities i and j, and Z_i, Z_j are Pauli-Z operators encoding binary variables. This term penalizes paths that violate the desired optimization criteria by increasing the associated energy.

The mixer Hamiltonian is given by:

$$H_M = \sum_i X_i,$$

where X_i denotes the Pauli-X operator, which introduces transitions between different bitstring configurations, allowing the quantum system to explore alternative routes. The QAOA circuit initializes the system in a uniform superposition of all possible solutions. It then applies alternating layers of unitaries generated by the cost and mixer Hamiltonians:

$$|\psi(\vec{\gamma}, \vec{\beta})\rangle = \prod_{k=1}^{p} e^{-i\beta_k H_M} e^{-i\gamma_k H_C} |+\rangle^{\otimes n},$$

where p is the number of layers (or QAOA depth), and $\vec{\gamma}$, $\vec{\beta}$ are tunable parameters optimized using a classical algorithm to minimize the expected cost.

After optimization, the quantum state is measured, and the most frequently observed bitstring is interpreted as the best approximate solution. Although QAOA does not guarantee an optimal result, it offers a promising trade-off between solution quality and computational efficiency, especially for problem sizes where classical exact methods become infeasible [11, 12].

Cold vs. Hot Start

In the context of QAOA, the terms cold start and hot start refer to different strategies for initializing the variational parameters $(\vec{\gamma}, \vec{\beta})$ before classical optimization begins. A cold start initializes these parameters randomly, offering no prior knowledge or structure, which can lead to slower convergence and higher risk of getting trapped in local minima. In contrast, a hot start uses informed initial values, often derived from optimized parameters of a smaller instance, a lower-depth QAOA run, or classical heuristics. This warm initialization can significantly improve convergence speed and solution quality by guiding the optimizer toward more promising regions of the parameter space.

6 Time Complexity and Scaling

As optimization problems grow in size, the time it takes to solve them increases dramatically — a concept captured by time complexity. Classical algorithms for route optimization, like those solving the Traveling Salesman Problem, often scale as $O(N^2 \cdot 2^N)$, making them computationally expensive even for modest input sizes [7]. In contrast, quantum algorithms promise more favorable scaling, with some approaches theoretically achieving linear complexity $O(N)$ [11, 12].

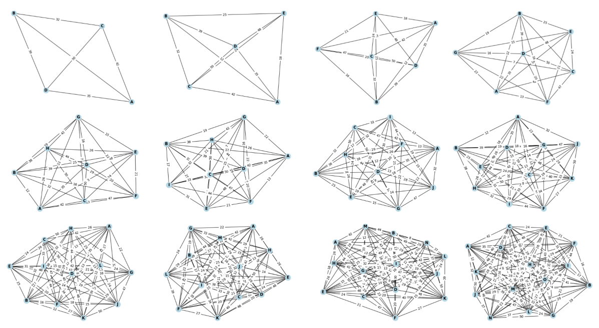

Figure 5: Growth of problem complexity with increasing network size.

7 Energy Consumption Analysis

Since each operation consumes energy, improving time complexity directly translates to reducing computational cost and environmental impact [12, 11].

To evaluate and compare the energy efficiency of the three TSP algorithms, we analyze the classical approaches by computing the product of their time-complexity and CPU power consumption [7], while the quantum algorithm is assessed by estimating the total number of quantum gates, their individual execution times, and the corresponding power requirements for each gate type [13].

7.1 Classical Brute-Force Method

Fixing the starting city, this approach evaluates all permutations of the remaining cities. Assuming a highly optimistic check time of $1\,\mu s = 10^{-6}\,s$ per route on a CPU consuming $200\,W = 0.2\,kW$, the total energy consumption in kilowatt-hours is given by

$$E_{\text{brute}}(n) = \frac{(n-1)! \cdot 1\,\mu s}{3600} \cdot 0.2\,\text{kWh} \tag{1}$$

Here, the division by 3600 converts seconds to hours. For example, evaluating all 11! routes for $n = 12$ cities yields an energy consumption of $\approx 7.98\,\text{kJ}$, showing explosive factorial scaling in both time and energy [6].

7.2 Classical Held-Karp (Dynamic Programming) Method

The Held-Karp algorithm improves this significantly with a time complexity of $O(n^2 \cdot 2^n)$ [7]. If we assume each basic operation (such as a table lookup or cost update) takes $5\,\text{ns} = 5 \times 10^{-9}\,\text{s}$, the total energy becomes

$$E_{\text{HK}}(n) = \frac{n^2 \cdot 2^n \cdot 5\,\text{ns}}{3600} \cdot 0.2\,\text{kWh} \tag{2}$$

Although still exponential, this grows far slower than the brute-force method. For instance, for $n = 12$, Held-Karp only consumes about 0.59 J, several orders of magnitude below brute-force.

7.3 Quantum Approximate Optimization Algorithm (QAOA)

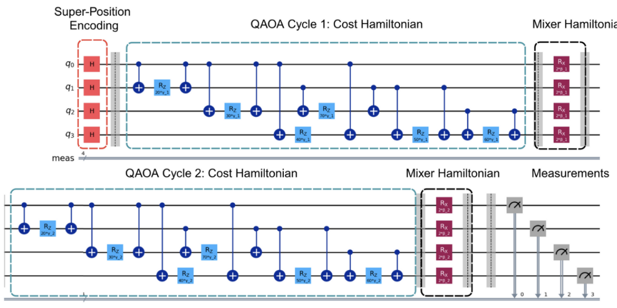

Figure 6: Circuit representation of a two layer QAOA for 4 cities

We begin by analyzing its quantum circuit structure. Fig. 6 shows a complete circuit with two full cycles, each consisting of alternating applications of cost Hamiltonians and mixer Hamiltonians [10].
From the diagram, we identify the number of quantum gates used per cycle:

4 Hadamard (H) gates

4 Rotation around the x-axis (Rx)

4 Measurement gates

6 Rotation around the z-axis (Rz)

12 Controlled-NOT (CNOT) gates

These gate counts serve as the basis for estimating execution time and power consumption. This can easily be generalized to n nodes (cities) and p cycles (layers).

Gate Type	Count
H gates	n
R_x gates	n
R_z gates	$\frac{1}{2}n(n-1)$
CNOT gates	$n(n-1)$
Measurement gates	n

Table 1: Gate counts per cycle for QAOA with n cities

To estimate the energy consumption of p QAOA cycles, we begin by grouping the quantum gates into three categories: single-qubit gates, two-qubit gates, and measurement operations. The total number of single-qubit gates, including Hadamard, R_x, and R_z gates, is given by

$$N_{1q} = p \cdot \left(2n + \frac{n(n-1)}{2}\right) \tag{3}$$

The number of two-qubit gates, which are primarily CNOTs, is

$$N_{2q} = p \cdot n(n-1) \tag{4}$$

while the number of measurement operations is

$$N_{\text{meas}} = n \tag{5}$$

since they only occur once at the end of the algorithm. Each of these gate classes is associated with a specific execution time sourced from the IBM Sherbrooke device [13]:

$$T_{1q} = 5.69 \times 10^{-8}\,\text{s}, \tag{6}$$

$$T_{2q} = 5.33 \times 10^{-7}\,\text{s}, \tag{7}$$

$$T_{\text{meas}} = 1.216 \times 10^{-6}\,\text{s} \tag{8}$$

By multiplying the number of gates in each category by their respective durations and summing the results, we obtain the total runtime for p cycles:

$$T_{run} = \left(\frac{T_{1q} \cdot N_{1q} + T_{2q} \cdot N_{2q} + T_{\text{meas}} \cdot N_{\text{meas}}}{3600}\right) \tag{9}$$

Finally, multiplying this runtime by the power draw of the quantum processor (in kilowatts)

$$P_{\text{Q}} = 25\,\text{kW},$$

yields an estimate of the energy required to execute the QAOA algorithm [12].

$$E_{\text{hot}} = T_{run} \cdot \text{runs} \cdot P_{\text{Q}} \tag{10}$$

$$E_{\text{cold}} = T_{run} \cdot (\text{steps} \cdot \text{shots} \cdot \text{repeats}) \cdot P_{\text{Q}} \tag{11}$$

This method provides a realistic, hardware-aware evaluation of quantum energy consumption grounded in circuit-level details.

7.4 Comparison

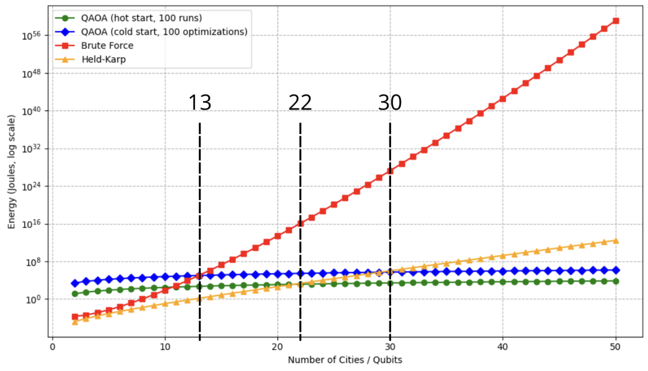

Figure 7: Comparison of Energy Consumption of the above discussed Algorithms

Fig. 7 presents the energy consumption of the four studied TSP algorithms: Brute Force, Held-Karp, QAOA with hot start, and QAOA with cold start, plotted over increasing input size n, based on the closed-form energy scaling expressions derived in the previous section [7, 10, 12, 11].
The y-axis is plotted logarithmically in joules, revealing exponential and factorial trends with clarity. As shown, the Brute Force method displays a steep factorial increase, with energy demands surpassing 10^{16} J at around 22 nodes and rapidly diverging thereafter [6]. Held-Karp, while asymptotically more efficient ($O(n^2 2^n)$), still grows exponentially and exceeds the energy usage of both quantum approaches beyond approximately 30 nodes.
In contrast, both quantum variants exhibit flat or near-constant energy behavior over the input range. The cold-started QAOA line sits consistently above the hot-started variant due to repeated classical optimization overhead [10], but both remain orders of magnitude more efficient than their classical counterparts for moderate to large n [12].
Vertical markers at $n = 13$, 22, and 30 highlight key crossover points: at $n \approx 13$, Brute Force becomes more energy-intensive than cold QAOA; at $n \approx 22$, the hot-started quantum variant outperforms Brute Force and Held-Karp; and by $n = 30$, the classical methods are no longer competitive.
These findings underscore a critical insight: even though today's quantum devices are limited in size and fidelity, their energy efficiency will potentially outpace classical algorithms for problem sizes well within reach of near-term quantum processors [11]. This makes energy an important secondary axis of comparison, especially for large-scale or resource-constrained applications, and strengthens the case for hybrid quantum optimization pipelines.

8 Alignment with the United Nations Sustainable Development Goals (SDGs)

These findings highlight QAOA's potential as an energy-efficient alternative to classical optimization for large problem sizes [11, 12]. With realistic gate times and hardware models [13], quantum computing offers tangible energy advantages in domains such as logistics, where energy costs directly affect emissions and infrastructure demands [4].

SDG 9: Industry, Innovation, and Infrastructure

The findings of this analysis support SDG 9, which emphasizes fostering resilient infrastructure, promoting inclusive and sustainable industrialization, and encouraging innovation [14]. By rigorously comparing the energy demands of classical and quantum optimization algorithms [7, 10], this work contributes to a data-driven discourse on how emerging quantum technologies could reshape industrial computation.
The ability of QAOA to deliver meaningful energy savings, especially for large-scale combinatorial problems, demonstrates that quantum hardware can be not only a theoretical advance, but a practical driver of energy-aware computational infrastructure. Incorporating execution times from real-world hardware such as the IBM Sherbrooke quantum processor [13] grounds this analysis in realistic assumptions, pushing beyond abstract theory to provide actionable insights into hardware-level energy performance. This aligns with SDG 9's call for sustainable innovation and provides a roadmap for industries seeking to optimize operations under tighter energy and environmental constraints.

SDG 11: Sustainable Cities and Communities

SDG 11 calls for the development of sustainable cities and human settlements [14]. One of the primary applications of optimization algorithms, both classical and quantum, is in the design and management of urban systems, especially transportation and logistics networks [4]. Route planning, waste collection, emergency services deployment, and public transit scheduling are all optimization problems that grow exponentially more complex as cities expand.
As demonstrated in the energy scaling plots [12], classical methods become prohibitively energy-intensive be-

yond moderate input sizes, potentially making their real-time use unsustainable in large-scale applications. In contrast, the energy consumption of QAOA remains nearly constant across increasing problem sizes [11]. This robustness positions quantum algorithms as promising tools for powering the smart cities of the future—ones that require real-time, low-power, high-efficiency decision-making tools to stay sustainable. Thus, this work directly contributes to the knowledge base required to build sustainable, algorithmically optimized urban environments.

SDG 12: Responsible Consumption and Production

Finally, the most direct and urgent impact of energy-aware computation is its contribution to SDG 12, which urges the reduction of greenhouse gas emissions and the advancement of climate-resilient solutions [14]. High-performance computing, especially when applied to large-scale optimization tasks, contributes significantly to energy consumption and associated emissions [1, 2], particularly when operated continuously in data centers powered by non-renewable sources.
The exponential energy demands of classical methods, as quantitatively shown in this study [7, 6], underscore the unsustainable trajectory of relying solely on traditional computing for growing logistical and operational challenges. By contrast, QAOA offers a more energy-scalable approach [11, 12], efficiently opening the door to cleaner, more responsible computing.
If quantum algorithms can substitute classical methods in large-scale deployments, especially for recurring optimization tasks, the cumulative energy savings over time could translate to a measurable reduction in carbon emissions. This analysis, therefore, offers a clear argument for considering quantum computing not only as a technological innovation but as a climate-conscious strategic investment.

9 Final Discussion

This report presented a detailed, hardware-aware comparison of energy consumption between classical and quantum approaches to solving the Traveling Salesman Problem (TSP). By deriving explicit formulas for energy use based on problem size n, and grounding these in realistic execution times from IBM's Sherbrooke quantum processor [13], we were able to model and visualize the scaling behavior of brute-force, Held-Karp, and QAOA (both hot and cold start) approaches.
The findings indicate a clear crossover point beyond which classical algorithms become prohibitively energy-intensive, while quantum approaches maintain near-constant energy consumption [11]. This positions quantum computing as a compelling energy-efficient alternative for large-scale combinatorial optimization.
The results suggest that, for sufficiently large problem instances, quantum optimization has the potential to outperform classical methods not only in time complexity but also in energy efficiency. This insight is particularly relevant in application domains like logistics [4], where the computational cost of optimization directly translates into environmental and economic impact.
However, these results should be interpreted with care. While the IBM Sherbrooke data accounts for many physical overheads [13], the QAOA simulations still assume idealized conditions at the algorithmic level, neglecting noise, decoherence, and readout errors that are common in NISQ devices [10]. Additionally, the energy consumption of cold-start QAOA depends heavily on optimizer performance and the number of classical iterations, which can vary depending on the problem landscape.
Despite these limitations, the analysis remains valuable in outlining the theoretical energy trends and highlighting the potential for quantum advantage in energy-constrained computing tasks. While current quantum hardware is still in the noisy intermediate-scale quantum (NISQ) era and lacks the error correction capabilities required for reliable large-scale computation, rapid progress in hardware fidelity, qubit connectivity, and compiler optimizations suggest that practical applications of quantum optimization are within reach. The results of this report contribute to the broader discussion on sustainable computation and underline the importance of continuing to evaluate emerging technologies through both technical and environmental lenses.

References

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010," https://www.analyticspress.com/datacenters.html, 2011, analytics Press.

[2] International Energy Agency, "Data centres and data transmission networks," https://www.iea.org/reports/data-centres-and-data-transmission-networks, 2022, iEA Report.

[3] S. Saifi, "The real environmental cost of every google search (with actual calculations)," *Medium*, July 2025, accessed: 2025-08-14. [Online]. Available: https://medium.com/@sohail_saifi/the-real-environmental-cost-of-every-google-search-with-actual-calculations-6b72e2dd7f3c

[4] United Parcel Service, "Orion: On-road integrated optimization and navigation," 2016, technical Overview.

[5] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley, 1985.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.

[7] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, no. 1, pp. 196–210, 1962.

[8] GeeksforGeeks, "Travelling salesman problem using dynamic programming," https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/, n.d.

[9] K. Sahu, "Big-o complexity chart," https://www.researchhub.com/post/1559/big-o-complexity-chart, February 2024, researchHub Post.

[10] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014. [Online]. Available: https://arxiv.org/abs/1411.4028

[11] Y. Nakajima and A. Auffèves, "Energy-consumption advantage of quantum computation," *PRX Energy*, vol. 4, no. 2, p. 023008, 2024. [Online]. Available: https://journals.aps.org/prxenergy/abstract/10.1103/PRXEnergy.4.023008

[12] D. Jaschke and S. Montangero, "Is quantum computing green? an estimate for an energy-efficiency quantum advantage," *arXiv preprint arXiv:2205.12092*, 2022. [Online]. Available: https://arxiv.org/abs/2205.12092

[13] IBM Quantum, "Qiskit runtime backend properties," 2025, iBM Sherbrooke, Accessed via Qiskit API.

[14] United Nations, "Transforming our world: the 2030 agenda for sustainable development," https://sdgs.un.org/goals, 2015, united Nations General Assembly.