Chapter 1: Why Quantum Optimization Matters

Optimization is one of the hidden engines of modern civilization.

When a delivery company chooses routes, it is solving an optimization problem. When a bank chooses a portfolio, it is solving an optimization problem. When a power-grid operator decides which generators should be active tomorrow, it is solving an optimization problem. When a machine-learning model adjusts millions or billions of parameters to reduce prediction error, it is solving an optimization problem.

At the simplest level, an optimization problem asks:

> Among many possible choices, which choice is best according to a rule?

The rule that tells us how good a choice is called an objective function. If we want the cheapest route, the objective function may be total travel cost. If we want the safest investment portfolio, the objective function may measure financial risk. If we want the most accurate machine-learning model, the objective function may measure prediction error.

But real optimization almost always includes limits. These limits are called constraints. A truck has a maximum load. A factory machine can only process one job at a time. A hospital schedule must give workers enough rest. A portfolio may not be allowed to invest more than a certain fraction of money in one asset.

So a practical optimization problem is usually a combination of three things:

1. Decision variables — the choices we are allowed to control.
2. Objective function — the quantity we want to minimize or maximize.
3. Constraints — the rules that a valid solution must satisfy.

For example, suppose a small delivery company has five packages and two trucks. The decision variables may describe which truck carries each package and in what order. The objective function may be total driving distance. The constraints may say that each package must be delivered exactly once and no truck may carry more than its capacity.

This sounds simple. The difficulty is that the number of possible choices can grow extremely quickly.

The explosion of possibilities

Imagine a problem with only 30 yes-or-no decisions. Each decision has two possibilities: yes or no, 1 or 0, selected or not selected. The total number of possible assignments is

$$2^{30} = 1{,}073{,}741{,}824.$$

That is more than one billion possible solutions from just 30 binary decisions.

This kind of problem is called a combinatorial optimization problem. “Combinatorial” means that the solution is built from combinations of discrete choices: choose this asset or not, assign this worker to that shift or not, put this job before that job or not.

A famous example is the traveling salesperson problem. A salesperson must visit a list of cities and return to the starting city while minimizing total travel distance. For a small number of cities, one can try all possible orders. But the number of possible tours grows roughly like a factorial, meaning it becomes enormous very quickly. The traveling salesperson problem has become a central benchmark in combinatorial optimization and has motivated decades of algorithmic research (Applegate et al., 2006).

Many important combinatorial problems belong to families known to be computationally difficult. In theoretical computer science, NP-hard problems are problems at least as hard as the hardest problems in the class NP, under standard reductions. Informally, NP-hardness tells us that we should not expect a single efficient algorithm that solves every instance of the problem exactly and quickly, unless major assumptions in complexity theory turn out to be wrong. Richard Karp’s 1972 work showed that many natural combinatorial problems are linked by reductions, helping establish NP-completeness as a central idea in complexity theory (Karp, 1972).

This does not mean NP-hard problems are impossible to solve. It means we must be careful. Real-world solvers often use structure, approximation, heuristics, decomposition, and domain knowledge. A logistics company does not need a proof that every possible route problem can be solved perfectly. It needs a good route by tomorrow morning.

That practical attitude is important for quantum optimization too.

Why optimization appears everywhere

Optimization appears whenever resources are limited and choices matter.

In logistics, companies optimize vehicle routes, warehouse locations, package assignments, inventory levels, and delivery schedules. A route that is only slightly better can save fuel, time, and labor across thousands of deliveries.

In finance, portfolio construction asks how to allocate money among assets. Harry Markowitz's mean-variance portfolio theory formalized a famous version of this problem: balance expected return against risk, often represented by variance or covariance of returns (Markowitz, 1952). A simplified portfolio problem might ask: choose exactly 10 assets from 100 possible assets, maximizing expected return while keeping risk below a limit.

In energy systems, operators optimize power generation, transmission, storage, and demand response. A grid must satisfy demand while respecting physical limits, generator constraints, and reliability requirements. As renewable energy grows, uncertainty in wind and solar output makes optimization even more important.

In manufacturing, optimization appears in job-shop scheduling, supply-chain planning, machine assignment, maintenance planning, and quality control. A factory may need to decide which job runs on which machine and in what order, while reducing delays and avoiding bottlenecks.

In machine learning, training a model usually means minimizing a loss function. A loss function measures how wrong the model is. For example, if a model predicts house prices, the loss may measure the difference between predicted prices and real prices. Training adjusts the model's parameters to reduce that loss.

In drug discovery and materials science, researchers search for molecules or materials with desired properties. This can involve optimizing molecular structure, energy, binding behavior, or reaction pathways. The search space is often huge.

These examples differ in details, but they share the same skeleton:

> Find the best allowed choice among many possible choices.

That skeleton is the reason quantum optimization is interesting. If a new computing model can help search, sample, or evaluate complicated solution spaces, it could matter in many industries.

But the word "could" is important.

Quantum optimization is promising, not magical.

What quantum computing changes

A classical bit is either 0 or 1. A qubit, the basic unit of quantum information, can be in a quantum state described by amplitudes for 0 and 1. These amplitudes are not ordinary probabilities. They are complex numbers whose magnitudes determine measurement probabilities through the Born rule, which we will study carefully in Chapter 4.

A quantum computer does not simply try all possible solutions in parallel and then read out the best one. That is a common misunderstanding. If we measure a quantum state, we get one outcome according to probabilities. The power of quantum algorithms comes from more subtle effects: superposition, interference, and entanglement.

- Superposition means a quantum state can involve amplitudes for many classical states at once.
- Interference means amplitudes can combine positively or negatively, increasing the chance of some outcomes and decreasing the chance of others.
- Entanglement means the state of a multi-qubit system can contain correlations that cannot be explained as independent states of each qubit.

For optimization, the hope is to design quantum procedures that increase the probability of measuring high-quality solutions.

For example, suppose each bitstring represents a possible delivery plan. A bitstring is a sequence such as

$$101001.$$

Each bit may represent a yes-or-no decision. A quantum algorithm may prepare a state whose measurement produces different bitstrings with different probabilities. If the algorithm is successful, good bitstrings appear more often than bad ones.

This is why many quantum optimization methods are sampling methods. They do not always output a single guaranteed-best solution. Instead, they produce samples, and we examine the best samples found.

This sampling view is especially important for variational quantum algorithms.

The variational idea

The word variational means that we choose from a family of possible trial solutions and adjust parameters to improve the result.

A simple classical example is fitting a line to data. Suppose we want a line

$$y = ax + b$$

that predicts data well. The parameters are a and b. We try values, measure the error, and adjust a and b to reduce the error. This is a variational process: we vary the parameters to optimize an objective.

A variational quantum algorithm, or VQA, uses the same broad idea, but the trial solution is prepared by a parameterized quantum circuit.

A parameterized quantum circuit is a quantum circuit containing adjustable numbers, usually angles. For example, one gate may rotate a qubit by an angle θ. The classical computer chooses θ, the quantum computer runs the circuit and measures the result, and the classical computer uses the measurement data to choose a better θ.

The basic loop is:

1. Choose circuit parameters.
2. Run the quantum circuit many times.
3. Measure outcomes.
4. Estimate the objective value.
5. Use a classical optimizer to update the parameters.
6. Repeat.

This is called a hybrid quantum-classical algorithm because the quantum and classical computers work together.

Two foundational examples are the Variational Quantum Eigensolver and the Quantum Approximate Optimization Algorithm. The Variational Quantum Eigensolver, or VQE, was introduced for estimating ground-state energies of quantum systems, such as molecules (Peruzzo et al., 2014). The Quantum Approximate Optimization Algorithm, or QAOA, was introduced as a variational method for combinatorial optimization problems (Farhi et al., 2014).

This book focuses especially on how these ideas connect to optimization.

Why the noisy intermediate-scale quantum era matters

Current quantum computers are powerful scientific instruments, but they are not yet large, fully error-corrected machines.

John Preskill introduced the term NISQ, meaning noisy intermediate-scale quantum, to describe quantum devices with roughly tens to hundreds or more physical qubits that are not protected by full fault-tolerant error correction (Preskill, 2018). “Noisy” means that gates, measurements, and stored quantum states are imperfect. “Intermediate-scale” means the devices are too large to simulate trivially in all cases, but not yet large enough or clean enough to run the most demanding fault-tolerant quantum algorithms.

VQAs became important because they seem compatible with this era. They can use relatively short quantum circuits, and they allow a classical computer to guide the search. A long, fully error-corrected quantum computation may be impossible on near-term hardware, but a shorter circuit measured many times may be feasible.

This does not mean VQAs automatically work well. They face serious challenges: noise, limited qubit connectivity, measurement cost, difficult parameter landscapes, and barren plateaus. A barren plateau is a situation where the optimization landscape becomes so flat that gradients are extremely small, making training difficult. These issues are now central topics in VQA research, and broad reviews emphasize both the promise and the limitations of the field (Cerezo et al., 2021).

The key point is balanced:

> VQAs are important because they are among the most practical quantum algorithm frameworks for current hardware, not because they already solve every hard optimization problem better than classical computers.

Mapping optimization problems to quantum form

To use many quantum optimization techniques, we often translate a problem into a mathematical form involving binary variables.

A binary variable is a variable that can take only two values, usually 0 or 1. For example:

$$x_i = \text{begincases } 1, \text{ if asset } i \text{ is selected, ; } 0, \text{ if asset } i \text{ is not selected. endcases}$$

A common optimization format is QUBO, which stands for quadratic unconstrained binary optimization. “Quadratic” means the objective includes terms involving one variable, such as x_i, and pairs of variables, such as $x_i\,x_j$. “Unconstrained” means constraints are usually included by adding penalty terms to the objective. “Binary” means the variables are 0 or 1.

A simple QUBO objective may look like

$$C(x) = -3x_1 - 2x_2 + 5x_1x_2.$$

If we minimize C(x), the first two terms reward selecting items 1 and 2, while the $5x_1x_2$ term penalizes selecting both together.

Another closely related form is the Ising model, originally from physics. In the Ising form, variables usually take values -1 or +1 instead of 0 or 1. Many optimization problems can be written as Ising models or QUBO problems, and Lucas surveyed Ising formulations for many NP problems (Lucas, 2014).

This matters because quantum devices naturally manipulate physical systems. If an optimization problem can be written as an energy function, then solving the problem resembles finding a low-energy state. In quantum optimization, a cost function can be encoded into a Hamiltonian.

A Hamiltonian is an operator that represents energy in quantum mechanics. In optimization, we often design a Hamiltonian whose lowest-energy states correspond to the best solutions of the original problem. The best solution becomes the ground state, meaning the state with minimum energy.

For example, in a graph problem such as MaxCut, each bitstring can represent a way to divide graph vertices into two groups. The cost function rewards edges crossing between the groups. QAOA encodes this cost into a quantum operation and tries to produce bitstrings with large cut values.

We will study this carefully in later chapters. For now, the important idea is:

> Quantum optimization often begins by translating a real decision problem into a binary mathematical model, then into a quantum cost structure.

That translation is not just a technical detail. It can determine whether the quantum method is practical or useless.

Where quantum optimization might help

Quantum optimization may be useful when a problem has a large, structured search space and when a quantum device can represent or sample useful candidate solutions in a way that complements classical computation.

One possible area is combinatorial optimization, especially problems that can be expressed naturally as QUBO or Ising models. Examples include MaxCut, certain scheduling problems, portfolio selection models, graph coloring, maximum independent set, and simplified routing formulations. These problems are attractive because binary variables map naturally to qubits.

Another possible area is hybrid decomposition. Large real-world problems may be broken into smaller subproblems. A classical computer manages the full problem, while a quantum routine is tested on selected subproblems. For example, a supply-chain optimizer might use classical methods for global planning but call a quantum optimizer for a difficult binary assignment subproblem. This is not guaranteed to outperform classical methods, but it is a realistic way quantum optimization may enter workflows.

A third area is sampling diverse high-quality solutions. In many real settings, one does not need only the single best solution. A planner may want several good schedules because some workers may become unavailable. An investor may want several portfolios with similar risk-return profiles. A drug researcher may want several candidate molecules. Quantum algorithms that produce probability distributions over solutions could be valuable if those distributions contain useful diversity.

A fourth area is quantum chemistry and materials, where the system being simulated is itself quantum mechanical. VQE originally became influential because estimating molecular ground-state energies is a natural quantum problem (Peruzzo et al., 2014). Although this is not always “optimization” in the logistics sense, it uses variational optimization and can support real applications in chemistry and materials science.

A fifth area is learning-assisted optimization. Classical machine learning may help choose quantum circuit parameters, predict good initial points, or identify problem structure. Conversely, quantum sampling may become a component inside larger classical optimization systems. This hybrid direction is one of the practical themes of modern VQA research (Cerezo et al., 2021).

These are possible areas of value. They are not promises of automatic advantage.

Where quantum optimization currently does not help much

It is equally important to understand where quantum optimization is not currently the right tool.

First, quantum optimization does not remove the difficulty of NP-hardness. There is no known quantum algorithm that efficiently solves all NP-hard optimization problems exactly. Quantum algorithms can sometimes offer speedups for specific tasks, but general hard optimization remains hard. For unstructured search, Grover's algorithm gives a quadratic speedup, not an exponential one (Grover, 1996). A quadratic speedup can be important, but it does not make enormous search spaces disappear.

Second, current quantum hardware is noisy. Noise can corrupt the state prepared by a circuit, distort measurement results, and make objective estimates unreliable. If a variational algorithm requires many circuit evaluations, noise and sampling cost can become major bottlenecks.

Third, classical optimization algorithms are very strong. Industrial solvers for linear programming, mixed-integer programming, constraint programming, local search, simulated annealing, and problem-specific heuristics have been developed over many decades. A quantum method must compete not with a weak brute-force search, but with highly optimized classical software and expert modeling.

Fourth, data loading can be a problem. If an application requires loading a huge classical dataset into a quantum computer, the loading cost may destroy any potential speedup. A quantum algorithm is not useful simply because the problem is large; the structure of the input and output matters.

Fifth, small demonstration problems can be misleading. Solving a tiny portfolio problem with five assets or a small MaxCut graph is useful for learning, but it does not prove practical advantage. Honest benchmarking must compare against strong classical baselines, include all relevant costs, and study scaling as problem size grows.

This book will return to these warnings many times. They are not meant to discourage you. They are meant to help you think like a serious practitioner.

A small example: portfolio selection

Let us make the discussion concrete.

Suppose you have five possible assets:

$$A_1, A_2, A_3, A_4, A_5.$$

You want to choose exactly two. Define binary variables:

$$x_i = \begin{cases} 1, & \text{if asset } A_i \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases}$$

The constraint “choose exactly two” becomes

$$x_1 + x_2 + x_3 + x_4 + x_5 = 2.$$

The objective may include expected return and risk. A simplified form might be:

$$\text{minimize } \lambda \cdot \text{risk}(x) - \text{return}(x),$$

where λ controls how much we care about risk. If λ is large, we strongly avoid risky portfolios. If λ is small, we care more about return.

To use QAOA, we would usually convert this into a QUBO. The constraint can be added as a penalty:

$$P(x) = M(x_1+x_2+x_3+x_4+x_5-2)^2,$$

where M is a penalty weight. If a portfolio selects exactly two assets, the penalty is zero. If it selects too many or too few, the penalty is positive.

The total QUBO objective becomes something like

$$C(x) = \lambda \cdot \text{risk}(x) - \text{return}(x) + P(x).$$

Now each bitstring corresponds to a portfolio. For example:

$$11000$$

means select A_1 and A_2. The quantum algorithm samples bitstrings. We evaluate them and look for portfolios with low objective values that satisfy the constraint.

This example is small, but it shows the workflow:

1. Start with a real decision problem.
2. Choose binary variables.

3. Write the objective and constraints.
4. Convert constraints into penalties if needed.
5. Encode the objective into a quantum-friendly form.
6. Run a variational quantum algorithm.
7. Compare the result with classical methods.

Chapter 21 will build a fuller portfolio case study.

A small example: routing

Now consider routing.

Suppose three delivery locations must be visited by one vehicle. A possible route is:

$$\text{Depot} \rightarrow B \rightarrow A \rightarrow C \rightarrow \text{Depot}.$$

The objective is total distance. The constraints say each location must be visited exactly once.

For three locations, the problem is tiny. But for 50, 100, or 1,000 locations, the number of possible routes becomes enormous. Practical routing also includes time windows, driver shifts, traffic, vehicle capacities, fuel costs, and customer priorities.

A quantum optimization approach would not simply “put all routes into a quantum computer” and instantly receive the best one. Instead, researchers must decide how to encode the route, how to enforce constraints, how many qubits are needed, how deep the circuit becomes, and how results compare with classical vehicle-routing methods.

This is the central pattern of real quantum optimization:

> The hard part is not only running a quantum algorithm. The hard part is modeling the real problem so the quantum algorithm has a meaningful role.

Chapter 22 will return to routing and scheduling in more detail.

Why “advantage” is a careful word

People often ask: “Can quantum computers solve optimization problems faster?”

The honest answer is: sometimes in theory for some tasks, maybe in practice for some future applications, but not generally today.

A quantum advantage means a quantum method performs better than classical methods under a clearly defined comparison. But “better” can mean different things:

- lower runtime,
- better solution quality,
- lower energy use,
- better scaling with problem size,
- better sampling diversity,
- ability to solve a problem classical methods cannot handle.

A claim of advantage must say which meaning is intended.

For optimization, this is especially tricky because many practical algorithms are heuristics. A heuristic is a method designed to work well in practice without guaranteeing the best solution in every case. Classical heuristics are often excellent. A quantum heuristic must be compared against strong classical heuristics, not against exhaustive search.

For example, if QAOA finds a good MaxCut solution on a small graph, that is interesting. But to claim practical advantage, we must ask:

- How large was the graph?
- What classical solvers were used for comparison?
- How many quantum circuit evaluations were needed?
- How many measurement shots were used?
- Was hardware noise included?
- Did the method scale better as problem size increased?
- Did it solve instances that matter outside the laboratory?

These questions are not negative. They are scientific. Chapter 18 will focus on honest benchmarking.

The role of VQAs in the bigger quantum landscape

Variational quantum algorithms are not the only quantum algorithms.

Some quantum algorithms are designed for future fault-tolerant quantum computers. A fault-tolerant quantum computer uses quantum error correction to protect computations from noise. Such machines may one day run long algorithms reliably, but they require many physical qubits and sophisticated error correction.

VQAs aim at a nearer-term possibility. They ask whether noisy devices can still do useful work when paired with classical optimization. This is why they became central in the NISQ era (Preskill, 2018; Cerezo et al., 2021).

The strengths of VQAs are:

- They can use relatively shallow circuits.
- They fit naturally with optimization problems.
- They can be adapted to hardware constraints.
- They allow classical computers to guide quantum experiments.
- They provide a flexible framework for both physics and combinatorial problems.

Their weaknesses are:

- They may require many measurements.
- Their optimization landscapes can be difficult.
- Noise can hide useful signals.
- Performance can depend strongly on circuit design.
- Classical competitors are powerful.
- General quantum advantage is not established.

So VQAs are neither a miracle solution nor a dead end. They are a serious research framework at the boundary between quantum physics, computer science, and applied optimization.

What you should remember from this chapter

Optimization matters because it is everywhere. It appears in logistics, finance, energy, manufacturing, machine learning, chemistry, and many other fields. Many real optimization problems are difficult because the number of possible solutions grows extremely fast.

Quantum optimization matters because quantum computers offer a different way to represent and sample from large spaces of possibilities. Variational quantum algorithms are especially important because they combine quantum circuits with classical optimization, making them suitable for current noisy intermediate-scale quantum devices.

But quantum optimization is not magic. It does not automatically solve NP-hard problems. It does not automatically outperform classical solvers. Its practical value depends on modeling, hardware quality, circuit design, measurement cost, noise, and fair benchmarking.

The right mindset is:

> Learn the mathematics. Understand the quantum mechanics. Respect classical algorithms. Test quantum methods honestly.

That mindset will guide the rest of this book.

In the next chapter, we build the mathematical foundations needed to describe optimization problems precisely: vectors, matrices, objective functions, constraints, gradients, convexity, and the difference between continuous and discrete search spaces.

References

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). The Traveling Salesman Problem: A Computational Study. Princeton University Press.

Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S. C., Endo, S., Fujii, K., McClean, J. R., Mitarai, K., Yuan, X., Cincio, L., & Coles, P. J. (2021). Variational quantum algorithms. Nature Reviews Physics, 3, 625–644. https://doi.org/10.1038/s42254-021-00348-9

Farhi, E., Goldstone, J., & Gutmann, S. (2014). A Quantum Approximate Optimization Algorithm. arXiv:1411.4028. https://arxiv.org/abs/1411.4028

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. Proceedings of the 28th Annual ACM Symposium on Theory of Computing, 212–219. https://doi.org/10.1145/237814.237866

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, & J. D. Bohlinger (Eds.), Complexity of Computer Computations (pp. 85–103). Plenum Press.

Lucas, A. (2014). Ising formulations of many NP problems. Frontiers in Physics, 2, Article 5. https://doi.org/10.3389/fphy.2014.00005

Markowitz, H. (1952). Portfolio Selection. The Journal of Finance, 7(1), 77–91. https://doi.org/10.2307/2975974

Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., & O’Brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. Nature Communications, 5, Article 4213. https://doi.org/10.1038/ncomms5213

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. Quantum, 2, Article 79. https://doi.org/10.22331/q-2018-08-06-79

Document information

Chapter 1: Why Quantum Optimization Matters

Project	Variational Quantum Algorithms for Optimization
Document	Document 1.5
Author	phone
Verifier	Not verified
Downloaded	July 04, 2026 18:10 KST
Status	Working
Document link	https://www.theorytrace.com/projects/variational-quantum-algorithms-for-optimization/-documents/chapter-1-why-quantum-optimization-matters/