Chapter 1: What Quantum Optimization Is

Optimization begins with a simple human activity: choosing.

You choose a route to work. A company chooses where to place warehouses. A hospital chooses a weekly staff schedule. A computer chooses how to allocate memory, bandwidth, and processor time. A scientist chooses model parameters that best fit data. In every case, there are many possible choices, and we want one that is best according to some rule.

That is the heart of optimization:

> Optimization is the search for the best solution among allowed possibilities.

The word “best” must be made precise. If we are planning a delivery route, “best” may mean shortest total distance. If we are investing money, “best” may mean high expected return with controlled risk. If we are scheduling workers, “best” may mean satisfying all legal and practical rules while minimizing overtime.

Quantum optimization asks whether ideas from quantum computing can help with this search.

Before we can understand that question, we need to understand what an optimization problem is, why many optimization problems become hard, and what “quantum” adds to the picture.

1.1 The basic shape of an optimization problem

An optimization problem has three main ingredients.

First, there are variables. A variable is a quantity we are allowed to choose.

For example, suppose a small café must decide how many muffins and sandwiches to prepare in the morning. Let

- x be the number of muffins,
- y be the number of sandwiches.

Here x and y are variables.

Second, there is an objective function. An objective function is a rule that gives a score to each possible choice. The score tells us how good or bad that choice is.

If the café earns \$2 profit per muffin and \$4 profit per sandwich, its profit might be

$$\text{profit}(x, y) = 2x + 4y.$$

If the café wants as much profit as possible, it wants to maximize this objective function.

Sometimes the objective is something we want to make small. A delivery company may want to minimize distance:

$$\text{distance(route)} = \text{total length of the route.}$$

When we make an objective large, we call it maximization. When we make it small, we call it minimization. These are not fundamentally different. Maximizing profit is the same as minimizing negative profit:

$$\max \text{profit} \quad \text{is equivalent to} \quad \min(-\text{profit}).$$

Third, there are often constraints. A constraint is a rule that a solution must obey.

The café may have only enough ingredients to make at most 100 items total:

$$x + y \leq 100.$$

It may also require that x and y be whole numbers:

$$x, y \in \{0, 1, 2, \ldots\}.$$

A choice that satisfies all constraints is called feasible. A choice that breaks at least one constraint is infeasible.

So an optimization problem can be described as:

> Find values of the variables that satisfy the constraints and give the best objective value.

This language will become much more precise in Chapter 2. For now, the key idea is enough: optimization is structured search.

1.2 A tiny example: choosing the best snack plan

Let us make the café example concrete.

Suppose the café can prepare muffins and sandwiches with the following rules:

- A muffin gives \$2 profit.
- A sandwich gives \$4 profit.
- The café can prepare at most 6 total items.
- Sandwiches require more staff time, so it can prepare at most 2 sandwiches.
- It must prepare whole numbers of each item.

The variables are

$$x = \text{number of muffins}, \qquad y = \text{number of sandwiches}.$$

The objective is

$$2x + 4y.$$

The constraints are

$$x + y \leq 6,$$

$$y \leq 2,$$

$$x, y \in \{0, 1, 2, \ldots\}.$$

A few possible choices are:

Muffins x	Sandwiches y	Feasible?	Profit 2x+4y
6	0	Yes	12
4	2	Yes	16
5	2	No, because x+y=7	18

Muffins x	Sandwiches y	Feasible?	Profit 2x+4y
0	2	Yes	8
3	3	No, because y>2	18

The best feasible choice is x=4, y=2, with profit \$16.

This problem is tiny, so we can solve it by checking every possible choice. But real optimization problems may contain thousands, millions, or more possible choices. In those cases, checking everything can become impossible in practice.

1.3 Why optimization becomes difficult

Some optimization problems are easy because their structure lets us solve them efficiently. Others are hard because the number of possible solutions grows extremely fast.

Consider a yes-or-no decision for each of n items. For example:

- include or exclude each investment,
- assign or do not assign each worker to a shift,
- place or do not place each sensor at a location,
- choose 0 or 1 for each binary variable.

If there are n binary decisions, there are

$$2^n$$

possible assignments.

For small n, this is manageable:

Number of binary variables n	Number of possible assignments 2^n
5	32
10	1,024
20	1,048,576
50	1,125,899,906,842,624

The growth is not merely “large.” It is exponential. Exponential growth means that adding one more variable multiplies the number of possibilities by a fixed factor. For binary variables, each added variable doubles the number of possibilities.

This is why brute force search often fails. Brute force means checking every possible solution. Brute force is simple and reliable for tiny problems, but it becomes useless when the search space is enormous.

A search space is the set of all possible candidate solutions. If a problem has 50 binary variables, its search space has 2^{50} candidates. If it has 500 binary variables, the number is so large that simply listing all candidates is beyond ordinary computation.

This does not mean every problem with many variables is hard. Some large problems have special structure that can be exploited. But many important optimization problems are difficult because no known method can solve all large instances efficiently. The theory of NP-completeness and NP-hardness formalized this difficulty for many combinatorial problems, including classic problems such as satisfiability and traveling salesperson variants (Karp, 1972; Garey and Johnson, 1979).

A combinatorial optimization problem is an optimization problem where the solution is assembled from discrete choices, often choices like yes/no, orderings, assignments, or subsets. “Combinatorial” comes from “combinations”: we are choosing combinations of items or decisions.

Examples include:

- selecting a subset of projects under a budget,
- assigning jobs to machines,
- choosing a route through cities,
- coloring a graph so neighboring nodes have different colors,
- placing facilities to serve customers.

These problems matter because they appear in logistics, manufacturing, finance, medicine, energy systems, chip design, machine learning, and scientific computing.

1.4 Local best versus global best

A common difficulty in optimization is that a solution can look good nearby but still not be the best overall.

A global optimum is the best solution in the entire feasible region. The feasible region is the set of all feasible solutions.

A local optimum is a solution that is better than nearby alternatives, even if some faraway solution is better.

Imagine hiking in fog on a mountain landscape. If your goal is to find the lowest valley, you may walk downhill until every nearby step goes upward. You have found a local minimum. But another valley far away may be lower. Without seeing the whole landscape, you may not know.

Optimization often uses the metaphor of an energy landscape. In this metaphor:

- each possible solution is a point in the landscape,
- the objective value is the height or energy,
- minimizing the objective means finding a low point,
- the global minimum is the lowest point,
- local minima are valleys that are low nearby but not lowest overall.

This energy landscape picture is especially important for quantum optimization, because many quantum methods represent optimization problems as physical systems whose low-energy states correspond to good solutions.

We will study this carefully in Chapter 8.

1.5 What “quantum” means at a first glance

The word quantum refers to the rules used by modern physics to describe very small systems, such as atoms, electrons, photons, and certain engineered devices. Quantum theory is not just a theory of tiny objects; it is a mathematical framework for how physical systems store information, change over time, and produce measurement outcomes.

Classical computing uses bits. A bit is a unit of information that has value 0 or 1.

Quantum computing uses quantum bits, or qubits. A qubit is not merely a bit that is sometimes uncertain. A qubit is described by a quantum state, which can involve complex-number amplitudes. These amplitudes determine measurement probabilities, but they can also interfere with each other before measurement. This interference is one reason quantum computation differs from ordinary randomized computation. Standard textbook treatments of qubits, amplitudes, measurement, and quantum circuits can be found in Nielsen and Chuang (2010).

Do not worry if that sounds unfamiliar. Chapters 3 through 7 will build the needed mathematics and quantum concepts from scratch.

For now, the important point is this:

> Quantum computing is a model of computation based on the mathematical rules of quantum mechanics.

This model is not science fiction. It grew from serious questions about whether ordinary computers can efficiently simulate quantum physics. Feynman argued that simulating quantum systems may require computational resources that grow very rapidly on classical machines and suggested using quantum systems themselves for such simulation (Feynman, 1982). Deutsch later formulated a model of a universal quantum computer, showing that quantum mechanics could be used as a general theory of computation (Deutsch, 1985).

Quantum optimization is one branch of this larger field.

1.6 What quantum optimization tries to do

Quantum optimization studies ways to use quantum systems or quantum algorithms to help solve optimization problems.

A careful definition is:

> Quantum optimization is the study and practice of formulating optimization problems so that quantum computational processes can search for, sample, or improve candidate solutions.

This definition has several important words.

Formulating means translating a real problem into mathematical form. For example, “schedule nurses fairly” must become variables, constraints, and an objective function.

Quantum computational processes include quantum circuits, quantum annealing devices, and other quantum models.

Search means exploring possible solutions.

Sample means produce candidate solutions according to some probability distribution. Quantum devices often return samples rather than a single guaranteed best answer.

Improve means use a quantum method as one part of a larger algorithm that gradually finds better solutions.

Notice what the definition does not say. It does not say that quantum optimization always gives the best answer. It does not say that quantum computers automatically beat classical computers. It does not say that hard problems become easy.

A more honest expectation is:

> Quantum optimization explores whether quantum effects can provide useful computational advantages for some optimization tasks, some problem structures, or some hardware regimes.

This is a scientific question, not a slogan.

1.7 Three main routes: annealing, gate-based algorithms, and hybrids

At a high level, modern quantum optimization has three major routes:

1. Quantum annealing
2. Gate-based quantum optimization
3. Hybrid quantum-classical optimization

These routes overlap. They are not isolated kingdoms. But separating them at first helps us build a clear map.

Quantum annealing

Annealing is a word borrowed from materials processing. In ordinary physical annealing, a material is heated and then slowly cooled so that its atoms can settle into a lower-energy, more orderly arrangement.

In simulated annealing, a classical optimization algorithm imitates this idea. It allows occasional uphill moves early in the search, then gradually becomes more selective. This can help escape some local minima.

Quantum annealing uses a related idea, but with quantum physics. The optimization problem is encoded into an energy function. The machine begins with a quantum system that is easy to prepare. Then the system is gradually changed so that its final low-energy states correspond to good solutions of the optimization problem.

One important mathematical model of quantum annealing uses a transverse-field Ising model, where quantum fluctuations are introduced through terms that can flip binary-like variables. Kadowaki and Nishimori studied quantum annealing in this setting and compared its behavior with classical simulated annealing (Kadowaki and Nishimori, 1998).

A simple image may help.

Suppose an optimization problem is like a landscape with many valleys. Classical local search may get trapped in one valley. Quantum annealing tries to use quantum dynamics to explore the landscape differently. One often mentioned quantum effect is tunneling, where a quantum system can pass through an energy barrier in a way that has no exact classical particle analogue. This does not guarantee success, but it suggests a different search mechanism.

In practice, quantum annealing is often used to produce many candidate solutions. The user then inspects the samples, chooses the best one found, or combines the samples with classical post-processing.

Quantum annealing will be studied in detail in Chapter 13.

Gate-based quantum optimization

The second route is gate-based quantum computing.

A classical computer applies logical operations to bits. A gate-based quantum computer applies quantum gates to qubits. A quantum gate is a controlled transformation of a quantum state. A sequence of gates is called a quantum circuit.

In gate-based quantum optimization, we build a quantum circuit whose behavior is connected to an optimization problem. We run the circuit, measure the qubits, and obtain candidate solutions. Then we adjust the circuit or interpret the results to seek better solutions.

One famous gate-based optimization algorithm is the Quantum Approximate Optimization Algorithm, usually called QAOA. QAOA uses alternating quantum operations related to the objective function and to a mixing process, with adjustable parameters chosen by a classical optimizer (Farhi, Goldstone, and Gutmann, 2014). We will build QAOA step by step in Chapter 15.

For now, imagine a problem where each candidate solution is a bit string such as

$$010110.$$

A QAOA circuit creates a quantum state spread over many bit strings, modifies the state using operations related to the problem's objective function, and then measures. Measurement returns ordinary bit strings. Some bit strings may appear more often than others. The hope is that good solutions appear with useful probability.

This is not the same as checking all solutions in parallel and reading out the best one. That popular description is misleading. Quantum measurement does not reveal all amplitudes at once. A quantum algorithm must carefully arrange interference so that good answers become more likely and bad answers become less likely.

Gate-based quantum optimization will be introduced in Chapter 14.

Hybrid quantum-classical methods

The third route is hybrid quantum-classical optimization.

A hybrid method uses both a quantum processor and a classical computer. This is important because current quantum devices are limited in size, precision, and reliability. Preskill described the current technological period as the Noisy Intermediate-Scale Quantum era, or NISQ era: quantum devices with enough qubits to be scientifically interesting, but still affected by noise and not yet protected by full quantum error correction (Preskill, 2018).

In a hybrid optimization loop, the quantum computer may prepare and measure quantum states, while the classical computer updates parameters, handles data, checks constraints, or improves candidate solutions.

A typical hybrid loop looks like this:

1. The classical computer chooses parameter values.
2. The quantum processor runs a circuit or annealing process using those values.
3. Measurements produce samples.
4. The classical computer estimates solution quality.
5. A classical optimizer chooses new parameters.
6. The loop repeats.

QAOA is usually used this way on near-term hardware. Many variational quantum algorithms have the same structure: a parameterized quantum circuit is trained or tuned using classical feedback. We will return to this family in Chapter 16.

Hybrid methods are not a compromise in a negative sense. They are often the natural way to use quantum devices. Classical computers are excellent at many tasks: bookkeeping, arithmetic, data preparation, local improvement, and statistical analysis. Quantum processors may be useful for specific subroutines. A good algorithm uses each tool where it is strongest.

1.8 Encoding: turning a problem into energy

Most quantum optimization methods require an important translation step: we must encode the optimization problem into a form the quantum method can use.

A common target form is an Ising model or a QUBO.

A QUBO is a Quadratic Unconstrained Binary Optimization problem. Let us unpack that phrase.

Binary means each variable is 0 or 1.

Unconstrained means the problem is written without explicit constraints. If there are rules, they are usually built into the objective function using penalties.

Quadratic means the objective function can contain terms involving one variable or two variables multiplied together, but not products of three or more variables.

A simple QUBO might look like

$$\min_{x_1,x_2\in\{0,1\}} (3x_1 - 2x_2 + 5x_1x_2)\,.$$

The term $3x_1$ is linear. The term $-2x_2$ is also linear. The term $5x_1x_2$ is quadratic because it multiplies two variables.

An Ising model usually uses spin variables

$$s_i \in \{-1, +1\}.$$

These variables come from physics, where spins can be idealized as pointing in one of two directions. In optimization, we can use them simply as two-valued variables.

Binary variables and spin variables can be converted into each other. One common conversion is

$$s = 2x - 1.$$

If x=0, then s=-1. If x=1, then s=+1.

Many NP optimization problems can be formulated as Ising models, often by adding penalty terms that make constraint-violating assignments energetically unfavorable; Lucas collected Ising formulations for many such problems (Lucas, 2014). This does not mean those problems become easy. It means they can be expressed in a form suitable for certain physics-inspired and quantum optimization methods.

Here is a tiny encoding example.

Suppose we want two binary variables to be different. That is, we prefer

$$x_1 \neq x_2.$$

The good assignments are:

- $x_1=0$, $x_2=1$,
- $x_1=1$, $x_2=0$.

The bad assignments are:

- $x_1=0$, $x_2=0$,
- $x_1=1$, $x_2=1$.

The expression

$$(x_1 - x_2)^2$$

is 0 when the variables are equal and 1 when they are different. So if we want them to be different, we might maximize $(x_1\text{-}x_2)^2$, or equivalently minimize

$$-(x_1 - x_2)^2.$$

Because $x_i^2 = x_i$ for binary variables, this can be rewritten as

$$-(x_1 + x_2 - 2x_1x_2).$$

That is a quadratic binary objective. We have turned a simple preference into a QUBO-like expression.

This style of translation is central to quantum optimization. We will study it carefully in Chapter 9.

1.9 What quantum optimization is not

Because quantum computing is exciting, it is easy to misunderstand what it promises. Let us remove several false ideas early.

Quantum optimization is not magic. A quantum computer does not try every solution and simply print the best one. Quantum states can involve amplitudes over many possibilities, but measurement gives limited classical information. Algorithms must be designed so that measurement is likely to reveal useful answers.

Quantum optimization is not guaranteed superiority. For many practical optimization problems, classical methods are extraordinarily strong. Mixed-integer programming, constraint programming, local search, simulated annealing, tabu search, branch and bound, evolutionary methods, and specialized heuristics can solve many real instances very well. A quantum method must be compared against serious classical baselines, not against naive brute force.

Quantum optimization is not only about exact answers. Many hard optimization problems are approached through approximation, heuristics, sampling, or finding “good enough” solutions within time limits. This is true in classical optimization and also in quantum optimization.

Quantum optimization is not one algorithm. It is a field containing models, encodings, algorithms, hardware, benchmarks, and open research questions.

Finally, quantum optimization is not separate from classical thinking. To use quantum optimization well, you must understand classical optimization. That is why Chapter 2 comes before the quantum chapters.

1.10 A first example: Max-Cut as an optimization problem

One of the most common teaching examples in quantum optimization is Max-Cut.

A graph is a collection of nodes connected by edges. For example, nodes might represent computers, people, cities, or abstract objects. An edge indicates a relationship between two nodes.

A cut divides the nodes into two groups. An edge is “cut” if its endpoints are placed in different groups.

The Max-Cut problem asks:

> How can we divide the nodes into two groups so that as many edges as possible go between the groups?

Here is a small graph:

- Nodes: A, B, C
- Edges: (A,B), (B,C), (A,C)

This is a triangle.

If we put A in group 0 and B,C in group 1, then:

- edge (A,B) is cut,
- edge (A,C) is cut,
- edge (B,C) is not cut.

So the cut size is 2.

For a triangle, the maximum cut size is 2. We cannot cut all 3 edges, because if A differs from B, and B differs from C, then A must be the same as C when there are only two groups.

Max-Cut is useful for learning because it has a clean binary structure. Each node receives a binary label:

$$x \in \{0,1\}.$$

An edge (i,j) is cut when

$$x_i \neq x_j.$$

As we saw earlier, this condition can be expressed mathematically using binary variables. That makes Max-Cut a natural example for QUBO, Ising models, quantum annealing, and QAOA. Max-Cut is also computationally important: Karp's classic work included related graph and satisfiability problems in the development of NP-completeness theory, and Max-Cut is a standard NP-hard optimization problem in the broader complexity literature (Karp, 1972; Garey and Johnson, 1979).

We will return to Max-Cut many times because it is simple enough to understand but rich enough to show real algorithmic ideas.

1.11 The promise and the discipline

Quantum optimization has a real promise: it gives us new computational models, new physical processes, and new algorithmic tools for exploring difficult solution spaces.

But the discipline is just as important as the promise.

A serious quantum optimization workflow must ask:

- Is the problem formulated correctly?
- Are the variables and constraints clear?
- Is the encoding faithful?
- Are penalty terms chosen carefully?
- What classical baseline is being used?
- How is solution quality measured?
- How much time and hardware access are required?
- How does performance scale as the problem grows?
- Are the results statistically meaningful?
- Is there evidence of a quantum advantage, or only evidence that a quantum device produced samples?

These questions may seem strict, but they make the field stronger. Quantum optimization is not only about building circuits or sending problems to quantum hardware. It is about modeling, algorithms, implementation, and scientific evaluation.

This book follows that path.

First, we will learn classical optimization language. Then we will build the linear algebra, probability, and quantum mechanics needed for quantum computing. After that, we will connect optimization to Hamiltonians, Ising models, and QUBO. Then we will study quantum annealing, gate-based algorithms, QAOA, variational methods, Grover search, quantum walks, noise, modeling, software, benchmarking, and case studies.

The path is long, but each step has a purpose.

1.12 Chapter summary

Optimization is the search for the best feasible solution according to an objective function. Some optimization problems are easy, but many important combinatorial problems become difficult because their search spaces grow exponentially and because no known algorithm solves all large instances efficiently.

Quantum optimization studies how quantum systems and quantum algorithms can help search, sample, or improve solutions to optimization problems. It includes quantum annealing, gate-based methods such as QAOA, and hybrid quantum-classical workflows.

The central translation step is encoding: turning a practical problem into a mathematical form, often an Ising model or QUBO, that a quantum method can process. This translation is powerful, but it must be done carefully.

The most important lesson of this chapter is balance. Quantum optimization is neither magic nor empty hype. It is a serious research and engineering field built at the intersection of optimization, quantum information, computer science, and physics.

In the next chapter, we begin with the classical foundation: objective functions, constraints, feasible regions, local and global optima, convexity, nonconvexity, and the basic reasons combinatorial optimization can be hard.

References

Aharonov, D., van Dam, W., Kempe, J., Landau, Z., Lloyd, S., and Regev, O. (2008). Adiabatic quantum computation is equivalent to standard quantum computation. SIAM Review, 50(4), 755–787.

Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 400(1818), 97–117.

Farhi, E., Goldstone, J., and Gutmann, S. (2014). A Quantum Approximate Optimization Algorithm. arXiv:1411.4028.

Feynman, R. P. (1982). Simulating physics with computers. International Journal of Theoretical Physics, 21, 467–488.

Garey, M. R., and Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman.

Kadowaki, T., and Nishimori, H. (1998). Quantum annealing in the transverse Ising model. Physical Review E, 58(5), 5355–5363.

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

Lucas, A. (2014). Ising formulations of many NP problems. Frontiers in Physics, 2, Article 5.

Nielsen, M. A., and Chuang, I. L. (2010). Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press.

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. Quantum, 2, Article 79.

Document information

Chapter 1: What Quantum Optimization Is

Project	Quantum Optimization from First Principles
Document	Document 1.5
Author	amnanoor
Verifier	Not verified
Downloaded	July 08, 2026 10:38 KST
Status	Working
Document link	https://theorytrace.com/projects/quantum-optimization-from-first-principles/documents-/chapter-1-what-quantum-optimization-is/