Introduction

Optimization begins with a very ordinary question:

Among many possible choices, which one is best?

That question appears everywhere. A delivery company wants the shortest set of routes. A hospital wants to schedule nurses without overworking anyone. An investor wants a portfolio that balances expected return and risk. A factory wants to assign jobs to machines so that work finishes quickly. A computer network wants to send data while avoiding congestion. In each case, there are choices, rules, and a way to judge whether one answer is better than another.

This book is about what happens when we bring quantum computing into that search for better answers.

You do not need to know quantum mechanics before starting. We will build the subject from the ground up. The goal is not to make quantum optimization sound mysterious. The goal is to make it precise, usable, and honest.

Quantum optimization is not a magic button that automatically solves every difficult problem. Many important optimization problems are computationally hard in a deep mathematical sense; the theory of NP-completeness and NP-hardness explains why some problems appear to resist efficient solution as their size grows (Karp, 1972; Garey and Johnson, 1979). Quantum computers change the rules of computation, but they do not erase all difficulty. For many optimization tasks, especially general NP-hard problems, no known quantum algorithm is proven to solve every instance efficiently.

That honesty is important. It protects us from hype. It also lets us see the real opportunity.

Quantum optimization is a developing field that asks:

Can quantum physical systems, quantum algorithms, or quantum-inspired models help us search difficult spaces of solutions more effectively than ordinary methods in some useful cases?

Sometimes the answer may be yes. Sometimes the answer may be no. Often the correct answer is: we need careful modeling, good baselines, and honest experiments.

This book will teach you how to think that way.

The basic shape of an optimization problem

Before we use the word “quantum,” let us understand “optimization.”

An optimization problem has three basic parts.

First, there are variables. A variable is a quantity we are allowed to choose. For example, suppose a café must decide how many blueberry muffins and chocolate muffins to bake tomorrow. Let

$$x = \text{number of blueberry muffins}$$

and

$$y = \text{number of chocolate muffins.}$$

Here, x and y are variables.

Second, there is an objective function. This is the score we want to make as small or as large as possible. If the café wants to maximize profit, the objective function might be

$$\text{profit}(x, y) = 2x + 3y,$$

where each blueberry muffin gives 2 units of profit and each chocolate muffin gives 3 units.

Third, there may be constraints. A constraint is a rule that a solution must obey. Maybe the café has enough ingredients to bake at most 100 muffins:

$$x + y \leq 100.$$

Maybe it must bake at least 20 blueberry muffins:

$$x \geq 20.$$

A solution that obeys all constraints is called feasible. A solution that breaks at least one rule is infeasible.

So an optimization problem asks:

> Among all feasible choices of the variables, which choice gives the best objective value?

This simple structure becomes difficult when the number of variables is large, when the constraints are complicated, or when the variables are discrete. A discrete variable can take separated values, such as 0 or 1, instead of any value along a continuous line. Many real decisions are discrete: choose this route or not; assign this worker or not; include this asset or not.

A particularly important kind of discrete variable is a binary variable, which can take only two values:

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

Binary variables are powerful because they represent yes-or-no decisions. For example:

$$x_i = \begin{cases} 1, & \text{if task } i \text{ is selected,} \\ 0, & \text{if task } i \text{ is not selected.} \end{cases}$$

Much of quantum optimization begins by rewriting problems in terms of binary variables.

Why hard optimization matters

Some optimization problems are easy. If you want the smallest number in a short list, you can simply check every number. If the list is sorted, it is even easier.

But many practical optimization problems grow explosively. Imagine assigning 30 jobs to 30 time slots. The number of possible assignments can become enormous. If you tried every possibility one by one, you might run out of time long before finding the best answer.

This is where the language of computational complexity becomes important. Computational complexity studies how the resources needed to solve a problem grow as the problem size grows. The resource might be time, memory, number of arithmetic operations, or number of calls to a subroutine.

A problem is not considered hard merely because one small example is annoying. It is considered hard when the required effort appears to grow very rapidly as the input size increases. Many famous combinatorial optimization problems are connected to NP-hardness, meaning that an efficient general-purpose method for solving them exactly would imply efficient methods for a very large family of difficult problems (Karp, 1972; Garey and Johnson, 1979).

A combinatorial optimization problem is an optimization problem where the solution is assembled from discrete choices. Examples include:

- choosing which cities to visit in which order,
- choosing which edges of a graph to cut,
- assigning workers to shifts,
- selecting items under a budget,
- arranging jobs on machines.

A graph is a collection of objects called vertices connected by edges. Graphs are one of the most common languages for optimization. For example, cities can be vertices and roads can be edges. People can be vertices and friendships can be edges. Computers can be vertices and network links can be edges.

One graph problem we will meet later is Max-Cut. In Max-Cut, we divide the vertices of a graph into two groups. An edge is “cut” if its endpoints land in different groups. The goal is to maximize the number, or total weight, of cut edges. This sounds simple, but the general problem is computationally difficult; Max-Cut is one of the classic problems connected to NP-completeness through its decision version (Karp, 1972).

Here is a small example. Suppose three people, A, B, and C, are connected in a triangle: A knows B, B knows C, and C knows A. We want to split them into two groups so that as many friendships as possible go across the split. Put A in group 0 and B, C in group 1. Then edges A-B and A-C are cut, but B-C is not. We cut 2 out of 3 edges. For a triangle, that is optimal.

For a graph with millions of vertices, the same idea becomes much harder.

What “quantum” adds

Quantum computing is based on the mathematical structure of quantum mechanics, the physical theory used to describe atoms, photons, electrons, and other microscopic systems. In ordinary digital computing, the basic unit of information is the bit, which has value 0 or 1. In quantum computing, the basic unit is the qubit, whose state is described by complex-valued amplitudes; measurement converts those amplitudes into ordinary probabilistic outcomes according to the rules of quantum mechanics (Nielsen and Chuang, 2010).

That sentence contains several new ideas, so let us slow down.

A state is a mathematical description of what a system is like. For an ordinary bit, the state is simple: it is either 0 or 1.

A qubit can be in a state written as

$$\alpha|0\rangle + \beta|1\rangle.$$

The symbols |0⟩ and |1⟩ represent the two basic states of a qubit. The numbers α and β are called amplitudes. They can be complex numbers, meaning they may involve the imaginary unit i, where $i^2 = -1$.

When we measure the qubit in the standard basis, we do not see the amplitudes directly. Instead, we get outcome 0 with probability $|\alpha|^2$, and outcome 1 with probability $|\beta|^2$, assuming the state is normalized so that

$$|\alpha|^2 + |\beta|^2 = 1.$$

For example, if

$$\alpha = \frac{1}{\sqrt{2}}, \qquad \beta = \frac{1}{\sqrt{2}},$$

then measurement gives 0 with probability 1/2 and 1 with probability 1/2.

This may look like an ordinary random coin flip, but it is not the same thing. Before measurement, a quantum state can interfere with itself through its amplitudes. Interference means that amplitudes can combine in ways that increase the probability of some outcomes and decrease the probability of others. Quantum algorithms are designed to arrange this interference so that useful answers become more likely.

A quantum computer is therefore not just a faster classical computer. It is a device that transforms quantum states and then measures them. The art of quantum algorithm design is to choose transformations that make good measurement outcomes more likely.

The optimization viewpoint: energy as a score

Quantum optimization often uses a powerful translation:

> An optimization score can be treated like an energy.

In physics, a Hamiltonian is a mathematical object that represents the energy of a system. In quantum mechanics, Hamiltonians are represented by special matrices called Hermitian matrices, whose eigenvalues correspond to possible energy values (Nielsen and Chuang, 2010). We will explain matrices, eigenvalues, and Hermitian matrices carefully later. For now, think of a Hamiltonian as an energy rule.

If we can encode an optimization problem into an energy rule, then a good solution becomes a low-energy state. The very best solution corresponds to the ground state, which means the state of minimum energy.

Here is a simple example using binary variables. Suppose we want two binary variables x_1 and x_2 to have different values. That is, we prefer

$$x_1 \neq x_2.$$

We can define an energy penalty:

$$E(x_1, x_2) = (1 - x_1 - x_2)^2.$$

Check the four cases:

x_1	x_2	$E(x_1,x_2)$
0	0	1
0	1	0
1	0	0
1	1	1

The lowest energy is 0, and it occurs exactly when the variables are different. So minimizing this energy solves the tiny optimization problem.

This idea scales. Many optimization problems can be written in forms such as QUBO, meaning Quadratic Unconstrained Binary Optimization, or as Ising models, which use variables that take values -1 and +1. Ising formulations are widely used because they connect naturally to physics-based models of interacting spins, and many NP problems can be expressed in Ising form with suitable constructions (Lucas, 2014).

The word spin here comes from physics, but in optimization we can first treat it as a two-valued variable. A spin variable s has value

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

A binary variable x ∈ \0,1\ can be converted into a spin variable by

$$s = 2x - 1.$$

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

This simple conversion is one of the bridges between ordinary optimization models and quantum optimization hardware or algorithms.

Three major routes through quantum optimization

Quantum optimization is not one method. It is a family of related approaches. This book will guide you through three major routes.

The first route is quantum annealing. Annealing is inspired by the physical idea that systems can settle into low-energy configurations. In quantum annealing, a system begins with a Hamiltonian whose low-energy state is easy to prepare. Then the Hamiltonian is gradually changed toward one whose low-energy state encodes the optimization problem. Transverse-field Ising models are central in standard presentations of quantum annealing, including the work of Kadowaki and Nishimori (1998). Practical quantum annealing also involves hardware constraints, embedding, sampling, and careful interpretation of results.

The second route is adiabatic quantum computing. The word adiabatic refers to a slow change. Very roughly, the adiabatic idea says that if a quantum system starts in the ground state of one Hamiltonian and the Hamiltonian changes slowly enough under suitable conditions, the system can remain close to the ground state of the changing Hamiltonian. Adiabatic quantum computation was developed as a model of computation by Farhi, Goldstone, Gutmann, and Sipser (2000). In optimization, the hope is to end near the ground state of a Hamiltonian that encodes the answer.

The third route is gate-based quantum optimization. This is closer to the circuit model of quantum computing. A quantum circuit is a sequence of quantum gates, where each gate transforms qubits. One of the most important gate-based optimization algorithms is the Quantum Approximate Optimization Algorithm, usually called QAOA, introduced by Farhi, Goldstone, and Gutmann (2014). QAOA uses a parameterized quantum circuit, measures candidate solutions, and uses a classical optimizer to adjust the parameters.

That last sentence introduces the phrase hybrid quantum-classical algorithm. Hybrid means that part of the work is done by a quantum processor and part by an ordinary classical computer. This is especially important today because present quantum devices are noisy and limited in scale. Preskill described the current era as the NISQ era, meaning “Noisy Intermediate-Scale Quantum,” where devices have enough qubits to be scientifically interesting but are not yet large, fully error-corrected quantum computers (Preskill, 2018).

In a hybrid workflow, the quantum device may generate samples or estimate an expected cost, while the classical computer updates parameters, checks constraints, stores data, and compares results. This is not a weakness of the field. It is one of its main practical forms.

What this book will not pretend

Because quantum optimization is exciting, it is easy to overstate what is known.

This book will not claim that quantum computers already solve all large optimization problems better than classical computers. They do not.

It will not claim that every QUBO model is a good model. A bad formulation can make a problem much harder than necessary.

It will not claim that a quantum sample is automatically meaningful. Samples must be interpreted statistically, compared with baselines, and tested across problem sizes.

It will not hide classical optimization. In fact, you will learn classical baselines because they are essential. If a quantum method is tested only against a weak classical method, the comparison teaches very little. Good science requires strong baselines, clear metrics, reproducible experiments, and careful uncertainty estimates.

This book will also not treat mathematics as decoration. The mathematics is the language that keeps us honest. But we will introduce it slowly. Linear algebra, probability, measurement, Hamiltonians, tensor products, and circuits will all be built from first principles.

A small first example: choosing between four answers

Let us end this introduction with a tiny optimization example.

Suppose you have two yes-or-no decisions:

- $x_1 = 1$ means choose project 1.
- $x_2 = 1$ means choose project 2.

There are four possible solutions:

Solution	Meaning
00	choose neither project
01	choose only project 2
10	choose only project 1
11	choose both projects

Now suppose the cost function is

$$C(x_1, x_2) = 3x_1 + 2x_2 - 4x_1x_2.$$

If we want to minimize cost, compute all four values:

x_1x_2	$C(x_1,x_2)$
00	0
01	2
10	3
11	1

The best solution is 00, with cost 0.

This example is so small that checking all answers is easy. But it contains the same structure as larger problems:

1. represent decisions with binary variables,
2. define a cost function,
3. search for low-cost solutions,
4. compare candidate answers.

Quantum optimization asks whether quantum systems or quantum algorithms can help with step 3, especially when the number of possible answers becomes enormous.

With two binary variables, there are

$$2^2 = 4$$

possible assignments.

With ten binary variables, there are

$$2^{10} = 1024.$$

With one hundred binary variables, there are

$$2^{100},$$

which is far larger than the number of seconds in the age of the universe. This does not mean every 100-variable problem is impossible; structure matters. But it shows why clever search methods matter.

Quantum methods are one family of clever search methods. To understand them, we need to learn the language of optimization, the mathematics of quantum states, and the practical discipline of implementation and benchmarking.

That is the path of this book.

We begin gently: what quantum optimization is, what problems it tries to solve, and why the promise is both fascinating and limited.

References

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

Farhi, E., Goldstone, J., Gutmann, S., and Sipser, M. (2000). Quantum Computation by Adiabatic Evolution. arXiv:quant-ph/0001106.

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. https://doi.org/10.1103/PhysRevE.58.5355

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher (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

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. https://doi.org/10.22331/q-2018-08-06-79

Document information

Introduction

Project	Quantum Optimization from First Principles
Document	Document 1.4
Author	amnanoor
Verifier	Not verified
Downloaded	July 08, 2026 10:28 KST
Status	Working
Document link	https://theorytrace.com/projects/quantum-optimization-from-first-principles/documents-/introduction/