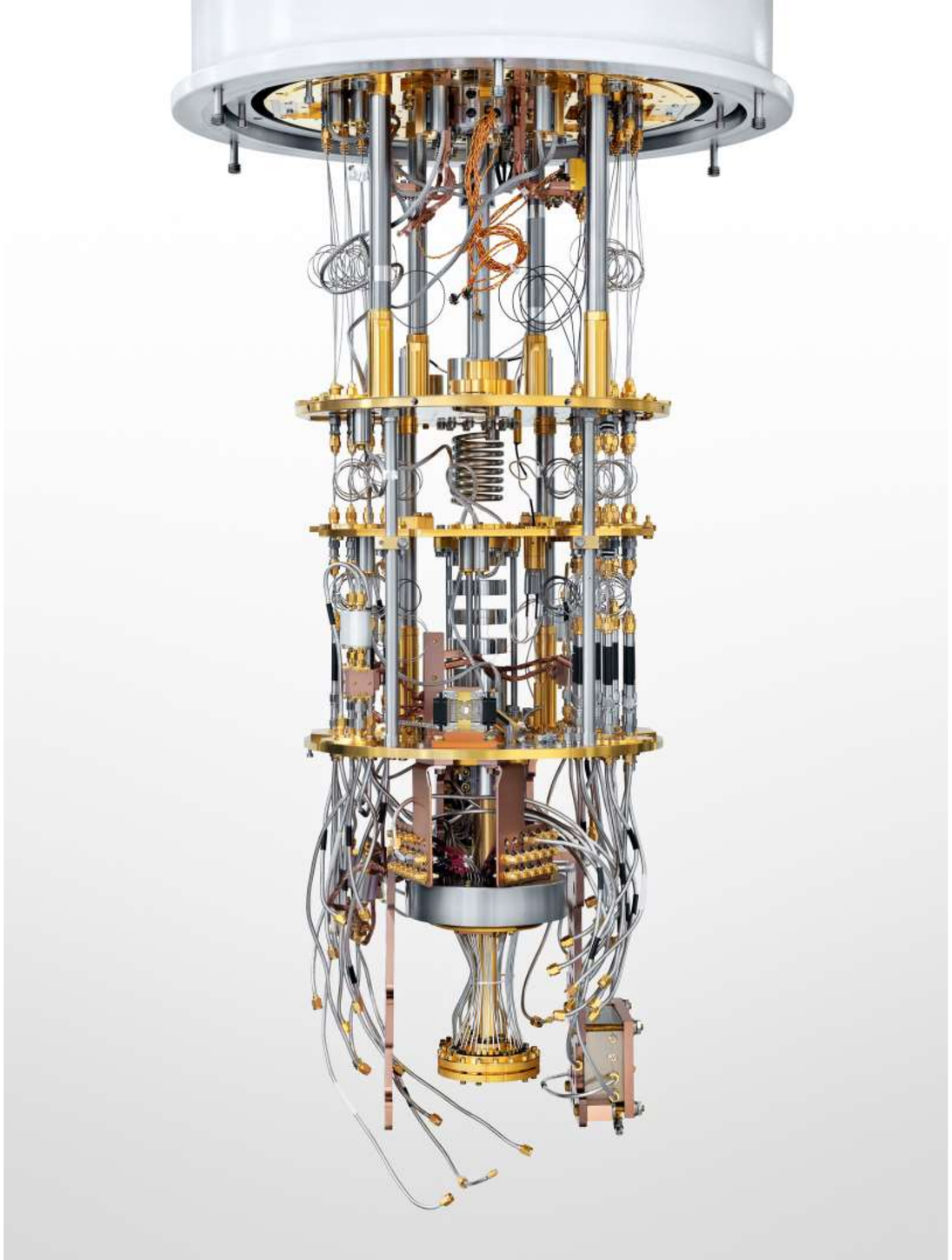
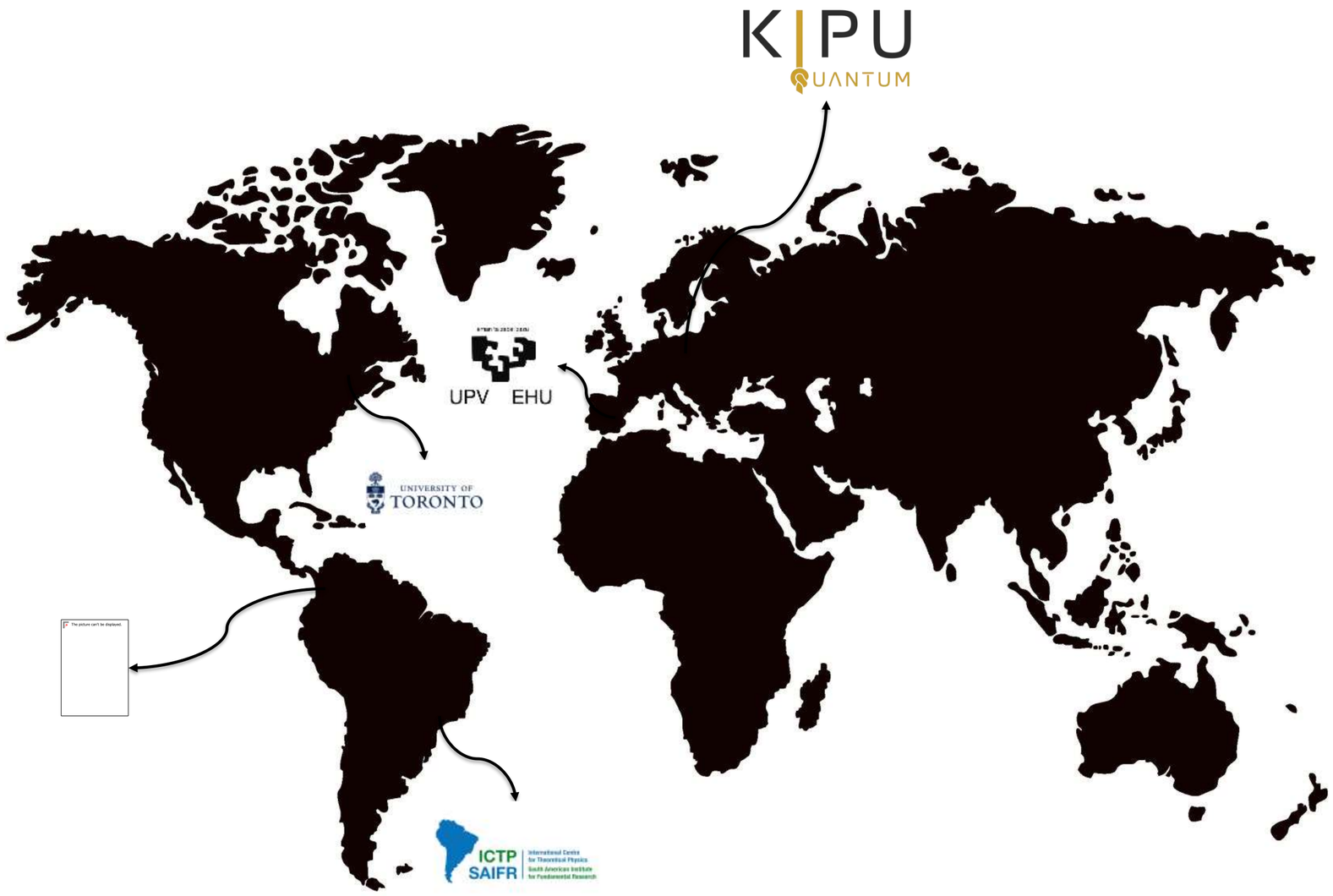
# Quantum Optimization Algorithms – Part 0

Alejandro Gómez Cadavid

Kipu Quantum GmbH
3rd Quantum Computing School ICTP-SAIFR
São Paulo, 2026

# Introduction




KIPU
QUANTUM
UPV EHU
UNIVERSITY OF
TORONTO
The picture can't be displayed.
ICTP
SAIFR
International Centre
for Theoretical Physics
South American Institute
for Fundamental Research

KIPU QUANTUM

# We build quantum algorithms for current quantum computers

1 RESEARCH AND DEVELOPMENT

## Algorithms for industry's hardest problems

- Quantum-enhanced optimization
- Quantum-enhanced machine learning

2 PLATFORM

## The Kipu Quantum Hub

- Our algorithms as a service
- 22 quantum backends
- Deploy your own algorithms

# Create your free Kipu Quantum Hub account

GO TO

# hub.kipu-quantum.com

---

**1** **Open the link**
On your laptop or phone

**2** **Sign up for free**
Using your institutional email

# Optimization decides how industry spends time and money



## Logistics

Vehicle routing and delivery networks



## Scheduling

Shifts, production lines and flight crews



## Finance

Portfolio selection under risk limits



## Drug discovery

Protein folding and molecular docking



## Energy

Grid balancing and power dispatch



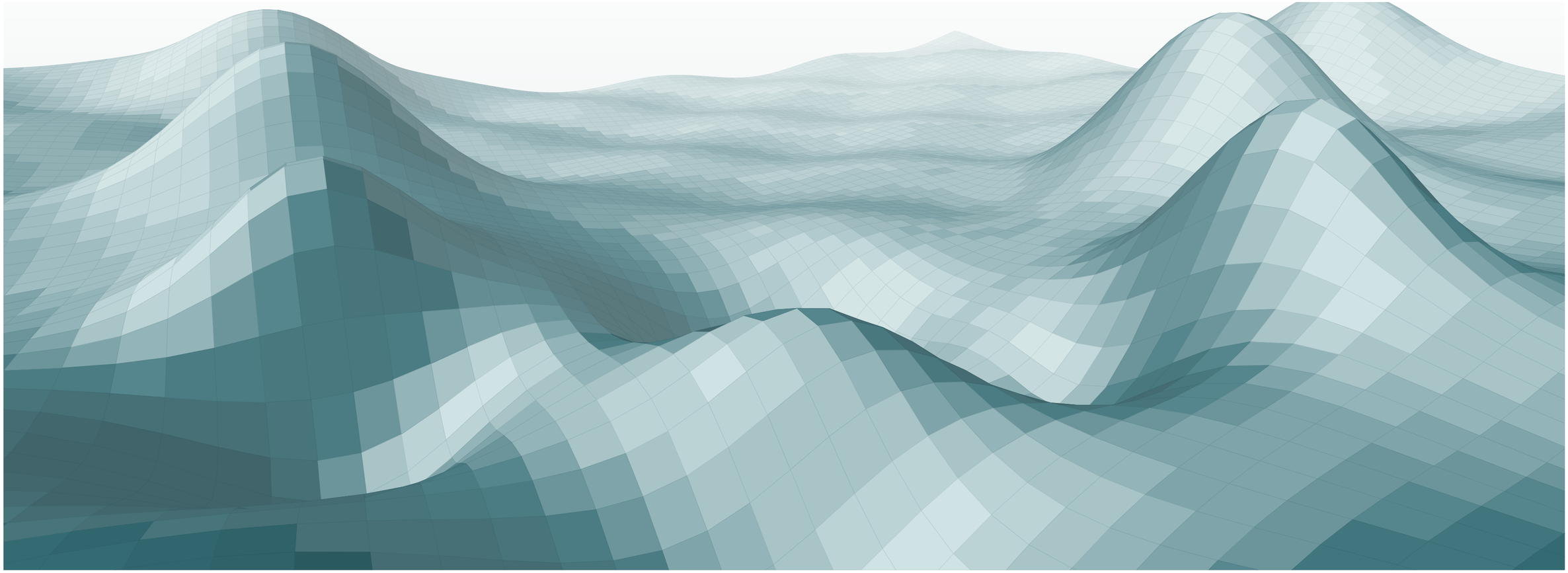
## Manufacturing

Chip layout, cutting and packing

Each one asks the same question: **which combination of choices is best?**

# Optimization is a search for the lowest point in an energy landscape

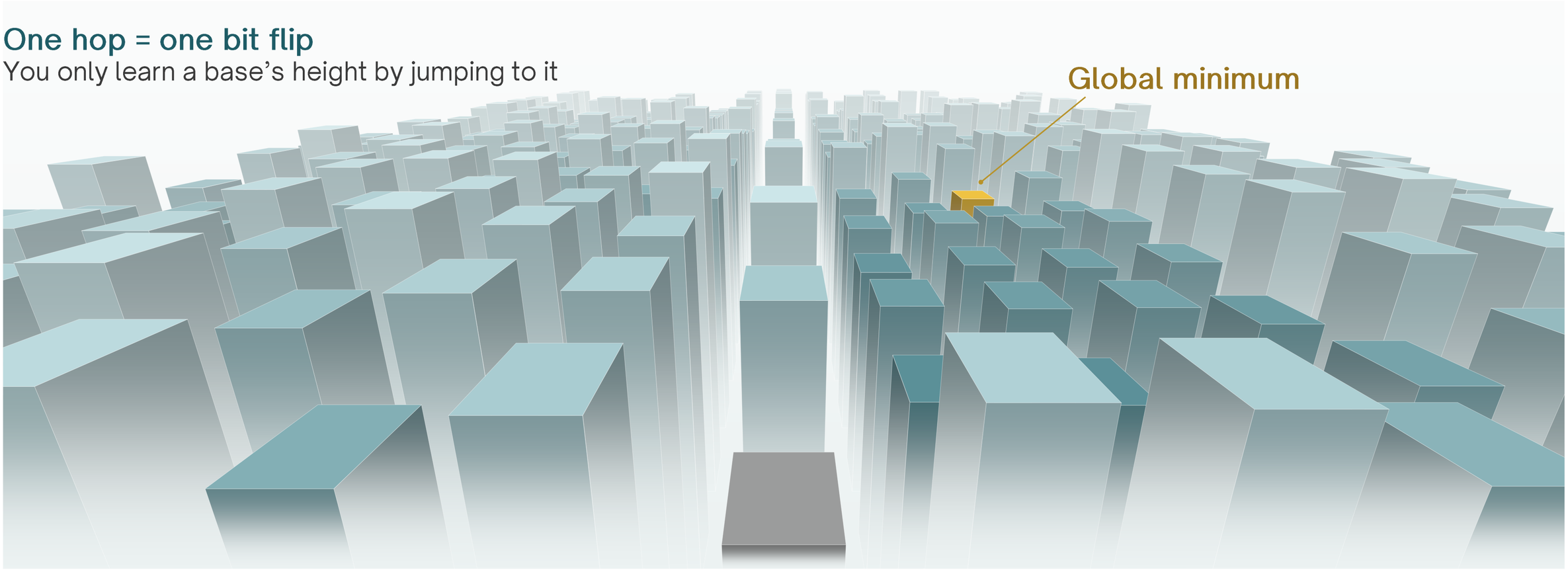
Every point is one possible solution; its height is the cost. An algorithm only sees the terrain around it.



**You are here, in a local minimum:** every nearby step goes uphill, so local search stops.

# In discrete problems there is no slope, only jumps between bases

Each base is one solution; its height is the cost. Neighbouring bases differ by a single bit flip.




**You are here, on the gray base:** every neighbouring base is higher, so no single hop improves the cost.

# Quantum gives optimization a new heuristic, with demonstrated speedups in specific cases

1

## A native fit for qubits

Solvers already take problems as QUBO/HUBO or Ising cost functions.

2

## Some speedups are empirically demonstrated

Exact methods like branch-and-bound scale exponentially on hard instances.

3

## A new heuristic to test

Heuristics are fast and may win on real instances but without a guarantee.

# Your turn: find the optimization problems around you

5 MINUTES · DISCUSS IN PAIRS

## What needs to be optimized in your city, region or workplace?

---

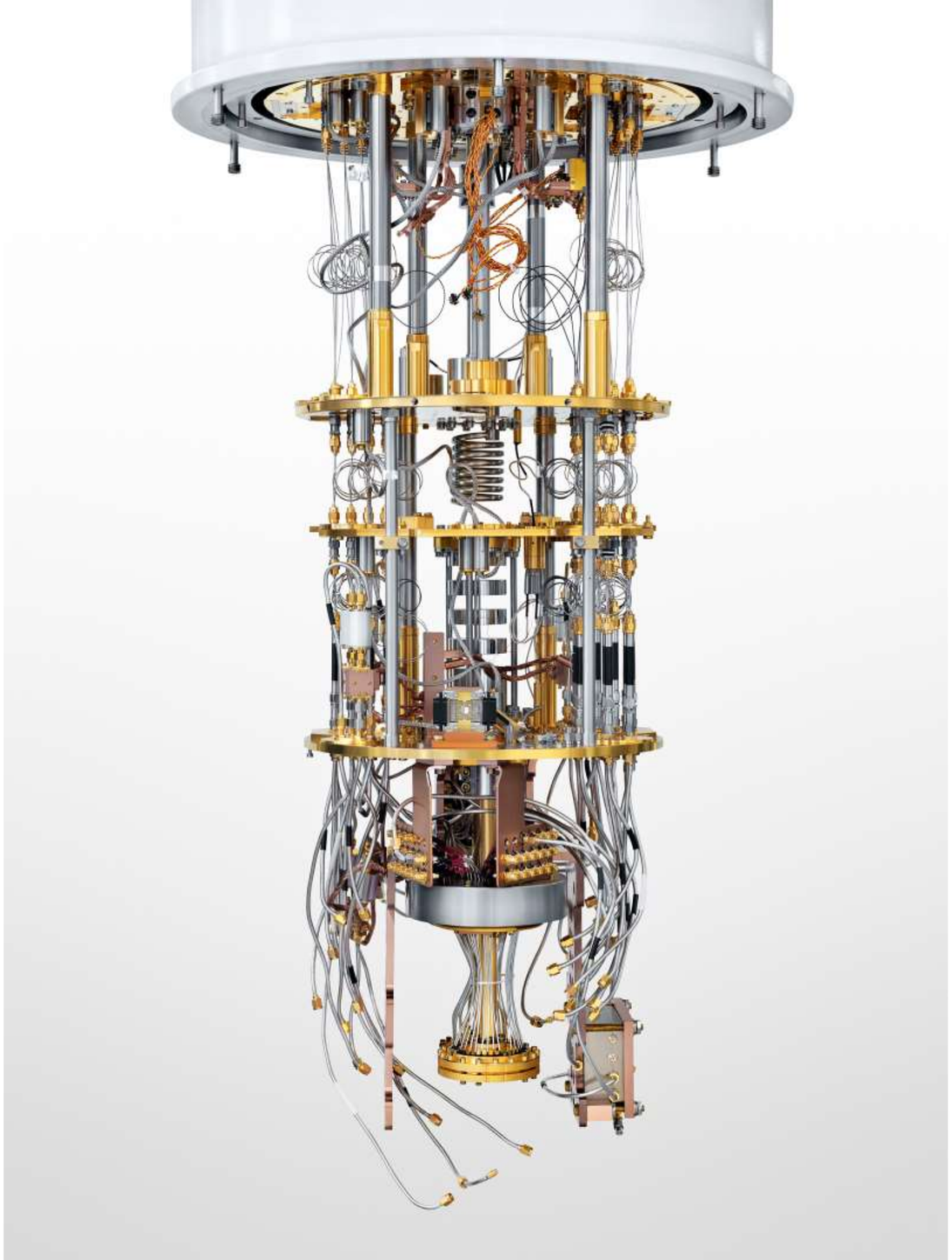
**1** **Pair up**
Turn to the person next to you

**2** **Share examples**
Traffic, transport, energy, water, health, farming

**3** **Pick one problem**
What can you choose, what does it cost, and what limits the options?

5:00

# End of Part 0

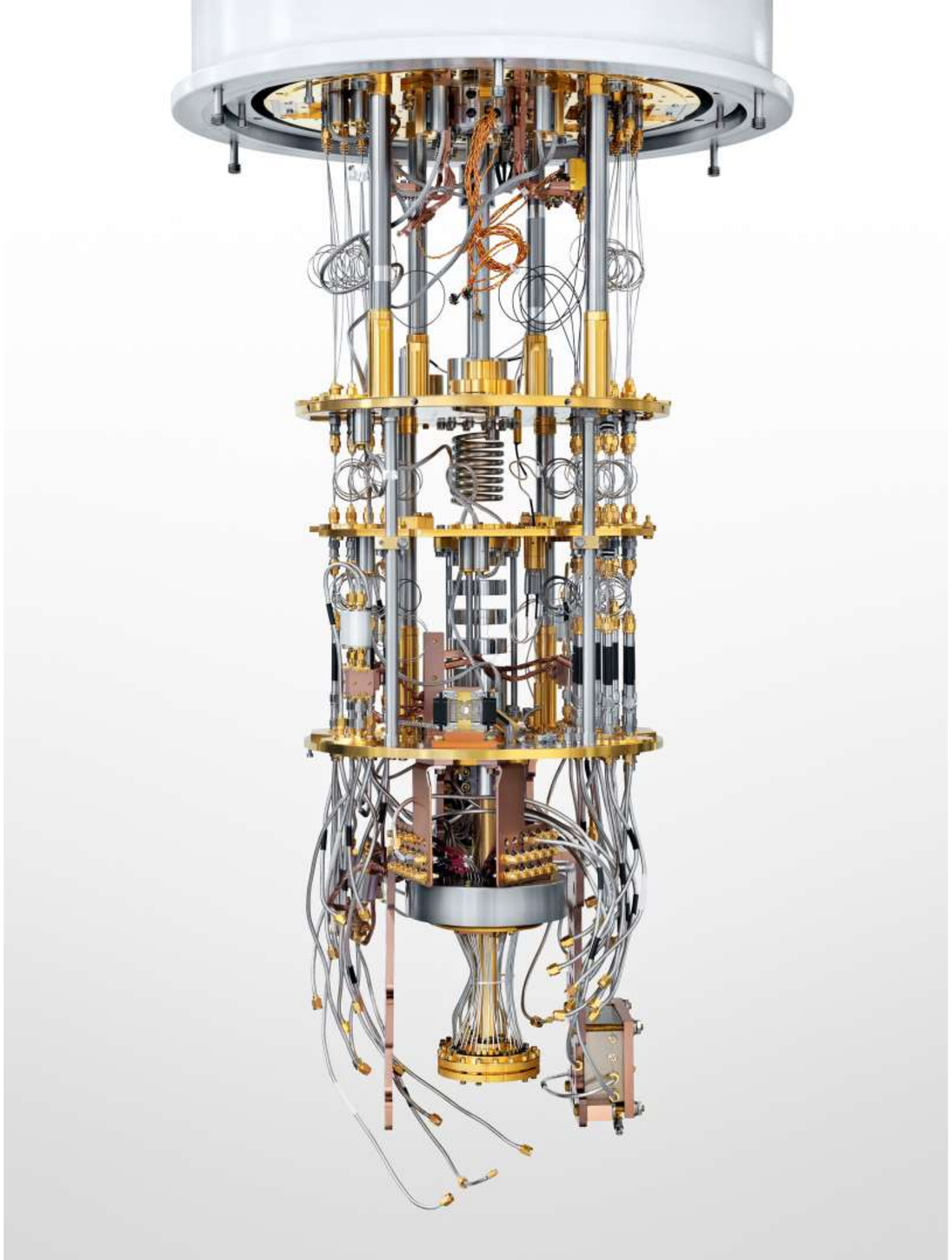


# Quantum Optimization Algorithms – Part I

Alejandro Gómez Cadavid

Kipu Quantum GmbH
3rd Quantum Computing School ICTP-SAIFR
São Paulo, 2026

# Optimization problems



# Optimization problems

Finding the value of a set of decision variables that minimizes an objective function while satisfying a given set of constraints.

$$\min_{x \in \chi} \; f(x)$$
$$s.t. \quad g_i(x) \leq 0, \qquad i = 1, \ldots, m$$
$$s.t. \quad h_j(x) = 0, \qquad j = 1, \ldots, p$$

| | |
|---|---|
| $\chi = \mathbb{R}^N$ | Continuous |
| $\chi = \mathbb{Z}^N or \{0,1\}^N$ | Integer / binary |
| $\chi = \mathbb{R}^{N_c} \times \mathbb{Z}^{N_d}$ | Mixed-integer |

**Problem class, by the shape of f(x)**

**LP** Linear program

$f(x) = c^T x$

Linear f, linear constraints

**QP** Quadratic program

$f(x) = \frac{1}{2} x^T Q x + c^T x$

Quadratic f, linear constraints

**NLP** Nonlinear program

$f(x)$ nonlinear

Nonlinear f, g or h

# Linear programs – example

Linear costs and capacity limits describe how goods, energy and data move through networks.

## The problem

### Plants
Each has limited stock: 40, 40 and 30 units.

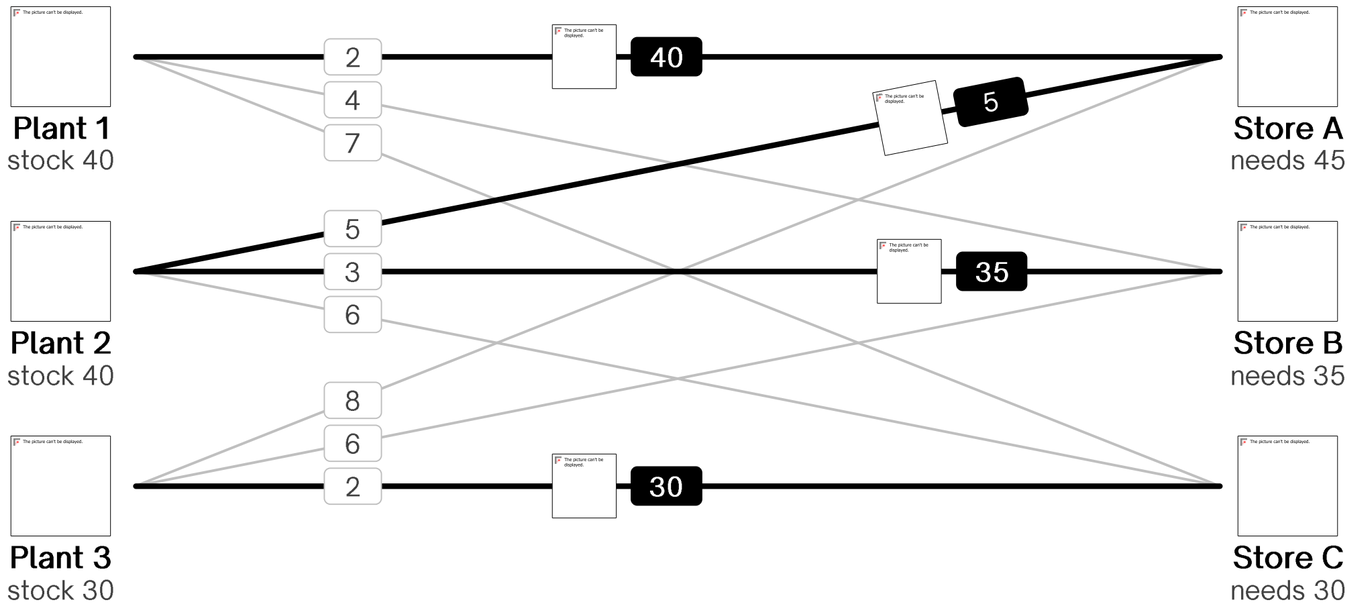
### Stores
Each needs a fixed amount: 45, 35 and 30 units.

### Roads
Every unit carried costs the price on its road.

**Goal**
Decide what each truck carries so every store is supplied at the lowest total cost.

**Best plan costs 270. Plant 1 runs out after 40 units, so Store A gets its last 5 from Plant 2.**

## Three plants, three stores, nine roads



# Writing it as a linear program

Each road becomes one variable; each plant and each store becomes one constraint.

## Example: transportation problem

$$\min \quad \sum_{i,j} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j} x_{ij} \le s_i \qquad \text{for each plant } i$$

$$\sum_{i} x_{ij} \ge d_j \qquad \text{for each store } j$$

$$x_{ij} \ge 0$$

| | |
|---|---|
| $x_{ij}$ | **decision: units sent from i to j** |
| $c_{ij}$ | cost per unit on that route |
| $s_i$ | capacity of plant i |
| $d_j$ | demand at store j |

**LP** Linear program
Variables: **continuous**
Objective: **linear**
Constraints: **linear**

## The same problem as a table

| Corner number: cost per unit | Store A | Store B | Store C | Supply $s_i$ |
|---|---|---|---|---|
| **Plant 1** | 2 **40** | 4 | 7 | 40 |
| **Plant 2** | 5 **5** | 3 **35** | 6 | 40 |
| **Plant 3** | 8 | 6 | 2 **30** | 30 |
| **Demand $d_j$** | 45 | 35 | 30 | |

Fill the cheapest routes first. Plant 1 is cheapest for Store A but holds only 40 units, so the last 5 come from Plant 2.

**Total cost:** $40 \cdot 2 + 5 \cdot 5 + 35 \cdot 3 + 30 \cdot 2 = 270$

# Quadratic programs – example

Quadratic objectives appear whenever the cost is a variance, an energy or a squared error.

## The problem

**Assets**
Each stock grows on average, but its value swings up and down.

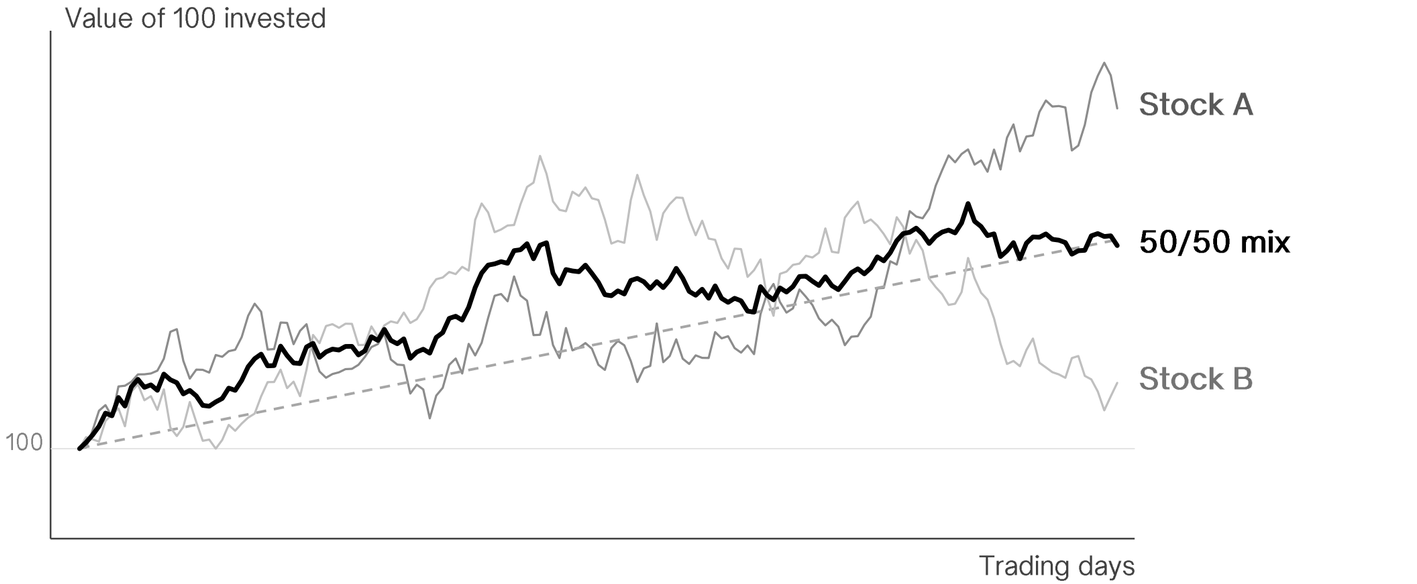
**Risk**
Risk is the size of those swings: the variance of the value.

**Mixing**
Stocks that tend to move in opposite directions partly cancel out.

**Goal**
Split the money so the portfolio earns at least $R$ with the smallest swings.

**Variance squares the swings and pairs up assets, so risk is quadratic in the shares.**

## Two swinging stocks, one steady mix




The mix still swings, but far less than either stock (dashed line: its expected growth).

# Writing it as a quadratic program

Each share is one variable. Risk adds up every pair of assets, so it is quadratic in the shares.

## Example: Markowitz portfolio selection

$$\min \quad \frac{1}{2}\sum_i \sum_j \sigma_{ij} w_i w_j$$ risk: every pair of assets

$$\text{s.t.} \quad \sum_i \mu_i w_i \geq R$$ expected return at least $R$

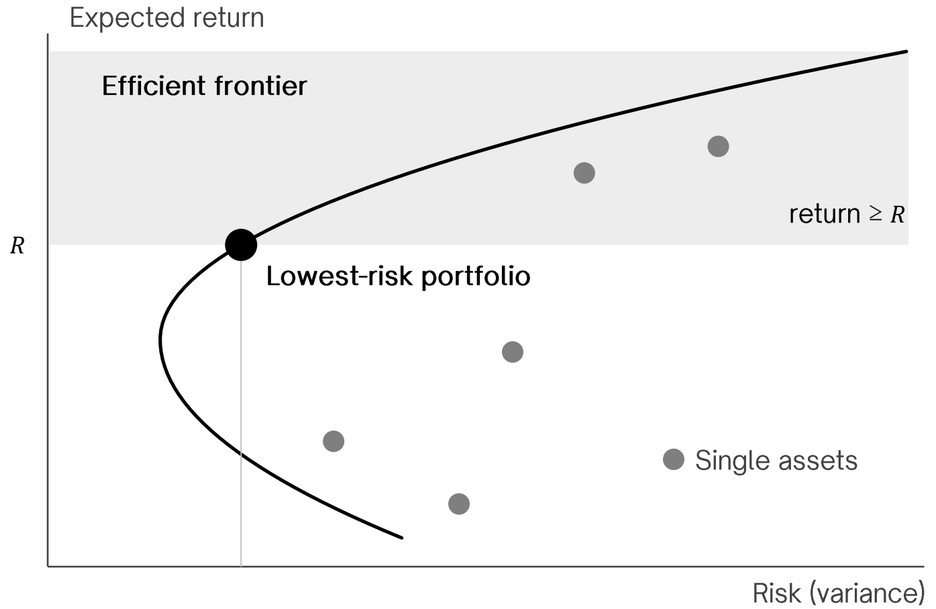
$$\sum_i w_i = 1$$ all money invested

$$w_i \geq 0$$ no short selling

| | |
|---|---|
| $w_i$ | **decision: share in asset i** |
| $\sigma_{ij}$ | covariance of assets i and j |
| $\mu_i$ | expected return of asset i |
| $R$ | minimum acceptable return |

**QP** Quadratic program
Variables: **continuous**
Objective: **quadratic**
Constraints: **linear**

## Goal: lowest risk for a target return




Each dot is one asset; mixing assets traces the frontier of best portfolios.

# Nonlinear programs – example

Products and powers of variables make a model nonlinear, as in shape design, physics and neural networks.

## The problem

**Volume**
Every can must hold the same X mL of soda.

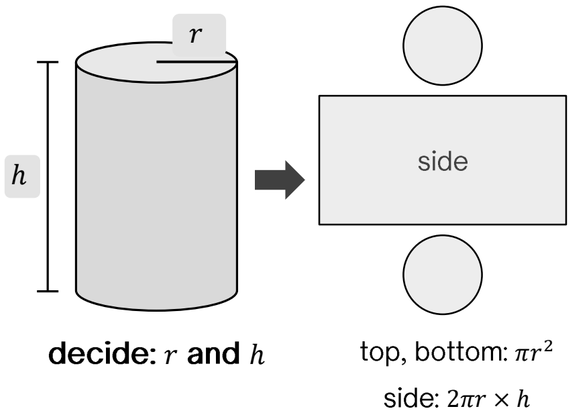
**Metal**
The cost is the sheet metal: a top, a bottom and one wrapped side.

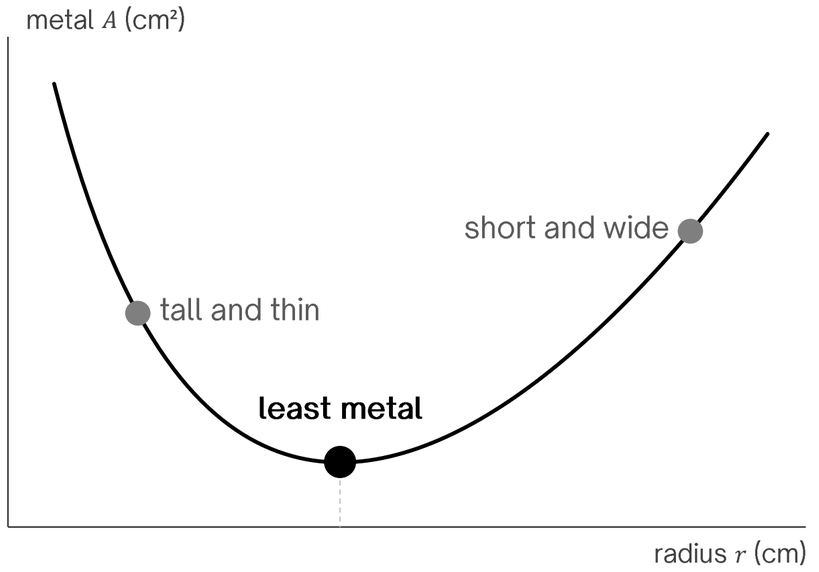
**Trade-off**
A taller can needs a bigger side, while a wider one needs bigger top and bottom.

**Goal**
Pick the **radius and height** that hold X mL with the least metal.

**Area and volume multiply and square the dimensions, so the problem is nonlinear.**

## Unroll the can, then trade height for width



# Writing it as a nonlinear program

Radius and height are the variables; area and volume multiply them, so the model is nonlinear.

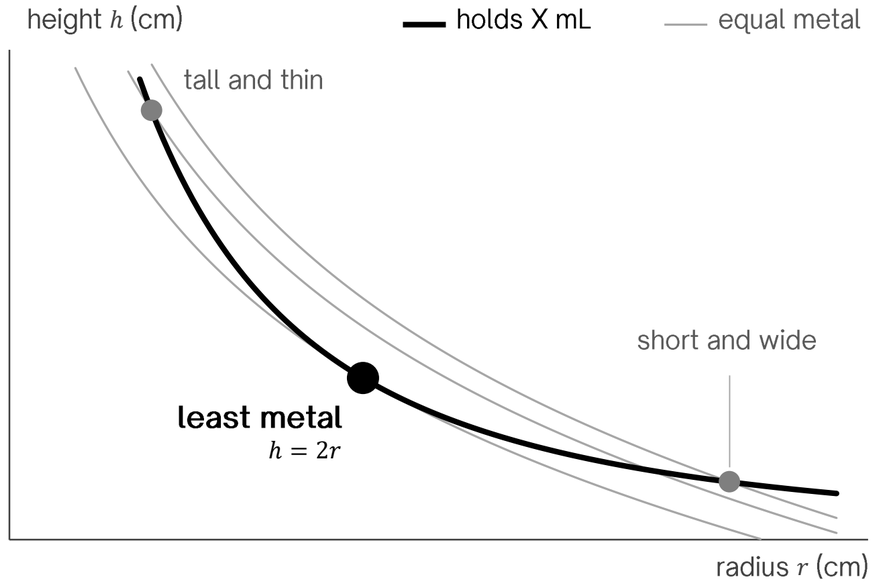
### Example: designing a soda can

| | | |
|---|---|---|
| min | $A = 2\pi r^2 + 2\pi rh$ | metal: top, bottom, side |
| s.t. | $\pi r^2 h = V$ | holds the required volume |
| | $r > 0, \quad h > 0$ | real dimensions |

| | |
|---|---|
| $r$ | **decision: radius of the can** |
| $h$ | **decision: height of the can** |
| $V$ | required volume |
| $A$ | metal used (surface area) |

**NLP** Nonlinear program
Variables: **continuous**
Objective: **nonlinear**
Constraints: **nonlinear**

### Goal: the feasible can on the lowest metal curve



# Binary programs – example

When every decision is yes or no, each variable is a bit. Hold an asset or not, open a site or not.

## The problem

**Assets**
Six candidates; each is held or not: a bit.

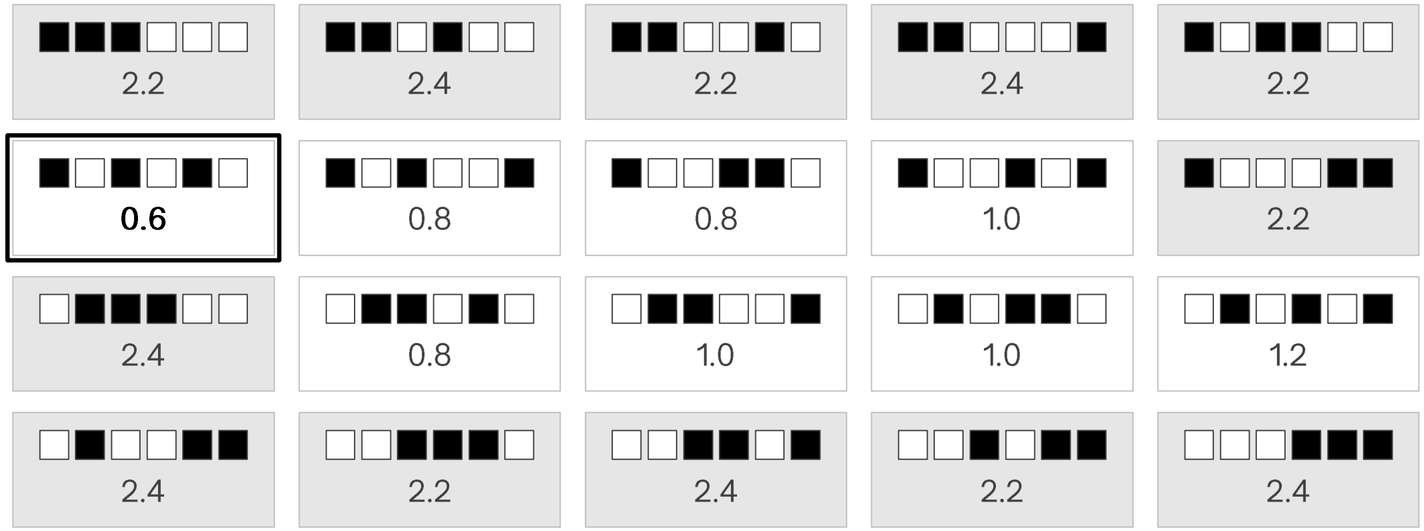
**Pairs**
Two assets that move together add risk when both are held.

**Choices**
Hold exactly 3 of 6: 20 portfolios. Hold 25 of 50: over $10^{14}$.

**Goal**
Hold the 3 assets with the best return and least shared risk.

**Every decision is 0 or 1, so you compare combinations instead of sliding along a curve.**

## Twenty ways to hold three of six assets



# Writing it as a binary program

Each asset becomes one bit; shared risk links pairs of bits, so the objective is quadratic.

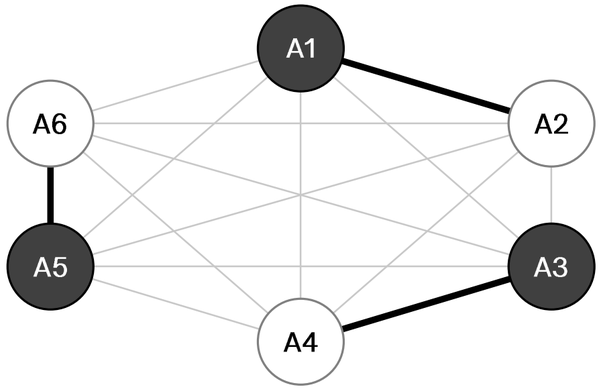
## Example: picking $K$ assets

| | | |
|---|---|---|
| min | $qx^T\Sigma x - \mu^T x$ | risk minus return |
| s.t. | $\mathbf{1}^T x = K$ | hold exactly $K$ assets |
| | $x_i \in \{0,1\}$ | each asset in or out |

| | |
|---|---|
| $x_i$ | **decision: hold asset $i$?** |
| $\Sigma, \mu$ | risk and return of the assets |
| $q$ | weight given to risk |
| $K$ | number of assets to hold |

**BQP** Binary QP
Variables: **binary**
Objective: **quadratic**
Constraints: **linear**

## Goal: high return, low shared risk



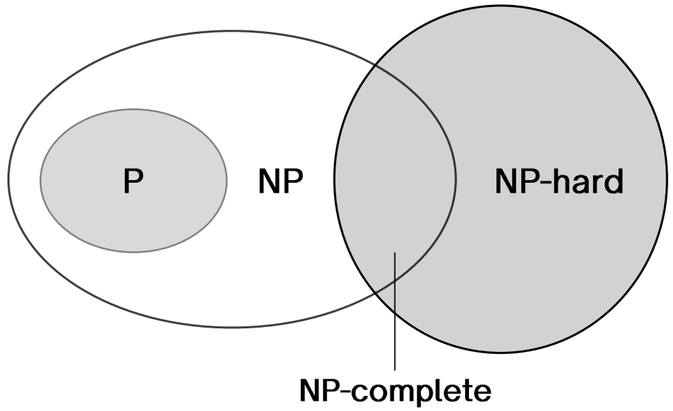

$x =$

| 1 | 0 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|
| A1 | A2 | A3 | A4 | A5 | A6 |

# About difficulty

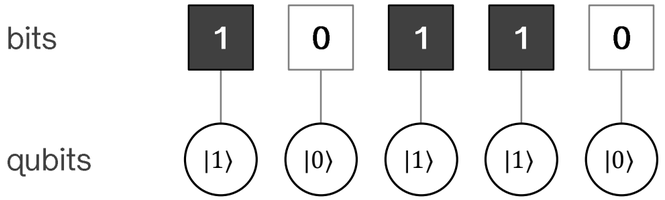
Each extra bit doubles the number of candidate solutions, and each bit maps onto one qubit.

## Classes




Checking an answer against a target cost vs finding the best answer.

## Native mapping to quantum




$$H = \sum_{i,j} J_{ij} Z_i Z_j + \sum_i h_i Z_i$$

# Your turn: classify the problem you picked last time

5 MINUTES · SAME PAIRS · AI-USAGE IS ENCOURAGED

# Is your problem an LP, a QP, an NLP? What about the decision variables?

---

**1** **Write it down**

Decision variables, objective and constraints, in words first

**2** **Formalize it (AI allowed)**

Ask for min $f(x)$ s.t. constraints, then check every term.

**3** **Label it**

Variables: continuous, integer or binary? Cost and constraints: linear, quadratic or nonlinear?

5:00

**Try:** “Write my problem as min f(x) subject to constraints, and state the type of each variable.”

# What standard quantum optimizers need

## Binary variables

**Why**
Each qubit is read out as 0 or 1, so every variable must be a bit.

**How we get there**
Start binary, as in the portfolio, or encode integers with several bits.

## No constraints

**Why**
The qubits explore all $2^n$ states and the hardware cannot forbid any of them.

**How we get there**
Move each constraint into the cost as a penalty.

## Polynomial cost

**Why**
The qubit energy is a sum of products of Z operators.

**How we get there**
Degree 2 gives a QUBO; higher degree gives a HUBO.



# Equality constraints as penalties

The ideal penalty forbids every infeasible state.

## The ideal penalty: an indicator

Keep the cost, and add a term that rules out every infeasible state:

$$\min \; f(x) + I(x)$$

| | |
|---|---|
| $I(x) = 0$ | if x satisfies the constraint |
| $I(x) = +\infty$ | otherwise |

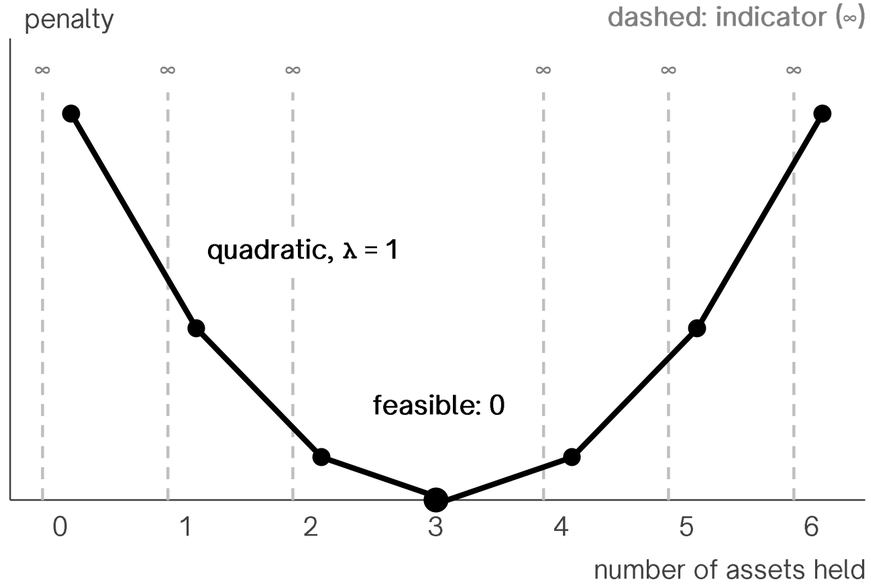
**Exact, but not a polynomial in the bits, so it cannot be written as a QUBO, for example.**

## A quadratic stand-in

$$P(x) = \lambda \left( \sum_i x_i - K \right)^2$$

Zero on every feasible state and growing with the violation. With $\lambda$ large enough it behaves like the indicator, and it is quadratic.

## Penalty against assets held, K = 3



# Inequalities need slack variables

A slack variable fills the gap up to the limit, so every feasible state can still reach zero penalty.

## A general inequality

$$\sum_i x_i \le K$$

integer weights $w_i$, limit $K$

**Squaring the gap to K would also punish every state below the limit.**

### Add a slack $y$ that fills the gap

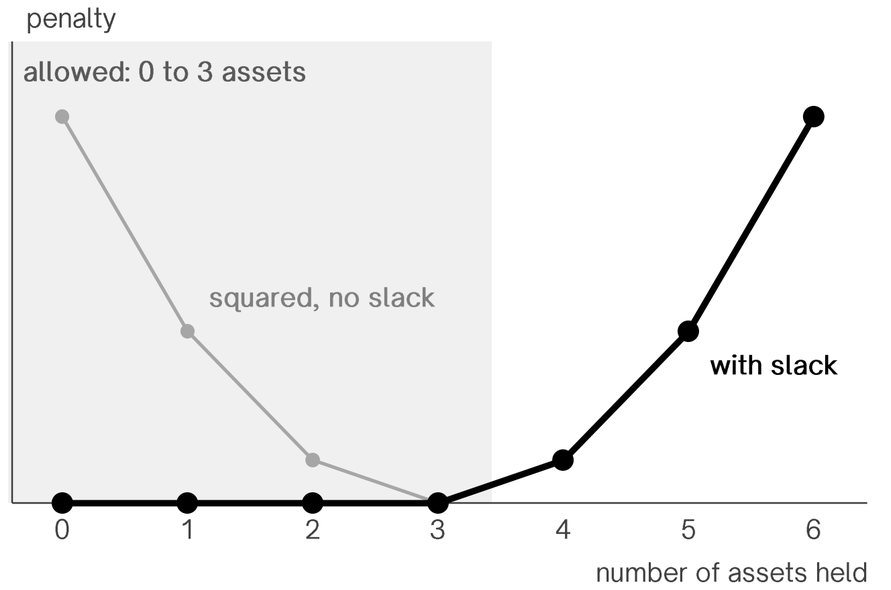
$$\sum_i x_i + y = K, \qquad 0 \le y \le K$$

$$y = \sum_k 2^k y_k$$

with $\lceil \log_2(K+1) \rceil$ slack bits

$$P(x,y) = \lambda \left( \sum_i x_i + \sum_k 2^k y_k - K \right)^2$$

## Example: at most 3 assets

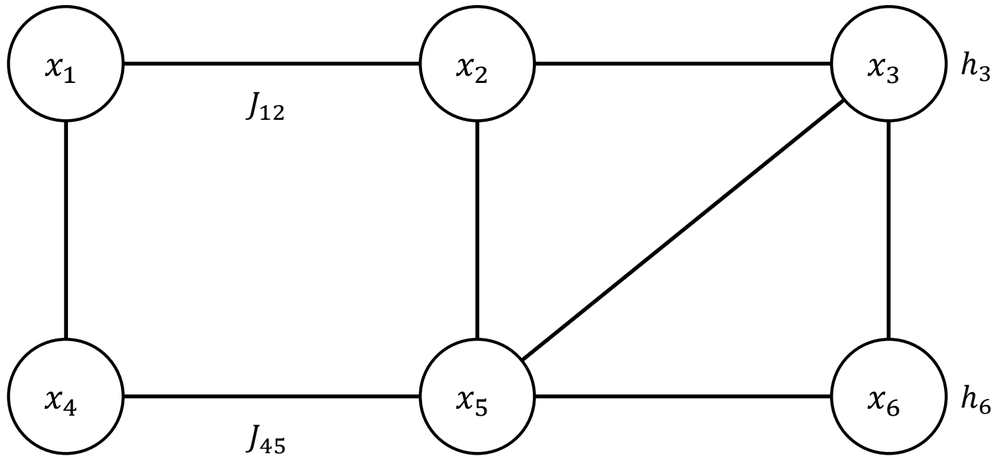


# QUBO and HUBO: costs on graphs

A QUBO couples bits in pairs (a graph); a HUBO also couples three or more (a hypergraph).

## QUBO: pairs only

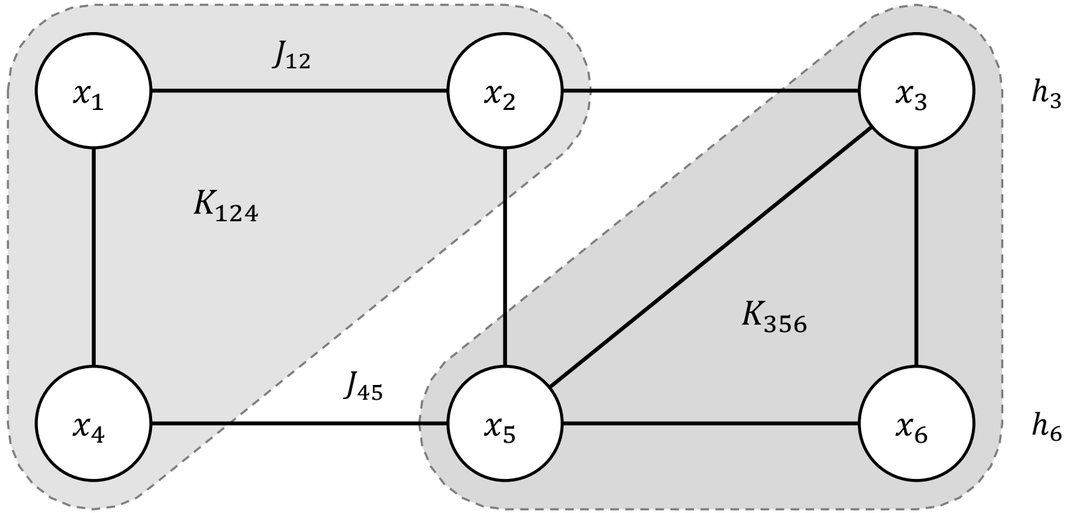
$$f(x) = \sum_i h_i x_i + \sum_{i<j} J_{ij} x_i x_j$$




Node: bit $x_i$, weight $h_i$. Edge: pair term, weight $J_{ij}$.

## HUBO: three or more bits

$$f(x) = \sum_i h_i x_i + \sum_{i<j} J_{ij} x_i x_j + \sum_{i<j<k} K_{ijk} x_i x_j x_k + \cdots$$




Hyperedge: a term on three or more bits, weight $K_{ijk}$.

**On qubits, every edge becomes a $Z_i Z_j$ interaction and every hyperedge a $Z_i Z_j Z_k$ interaction.**

# Minimizing f(x) is finding a ground state

Write the cost as the energy of spins, then turn each spin into a qubit operator.

## From bits to a Hamiltonian

Cost $f(x) = \sum_i h_i x_i + \sum_{i<j} J_{ij} x_i x_j + \cdots$

$x_i = \frac{1-s_i}{2}$ with spin $s_i = \pm 1$

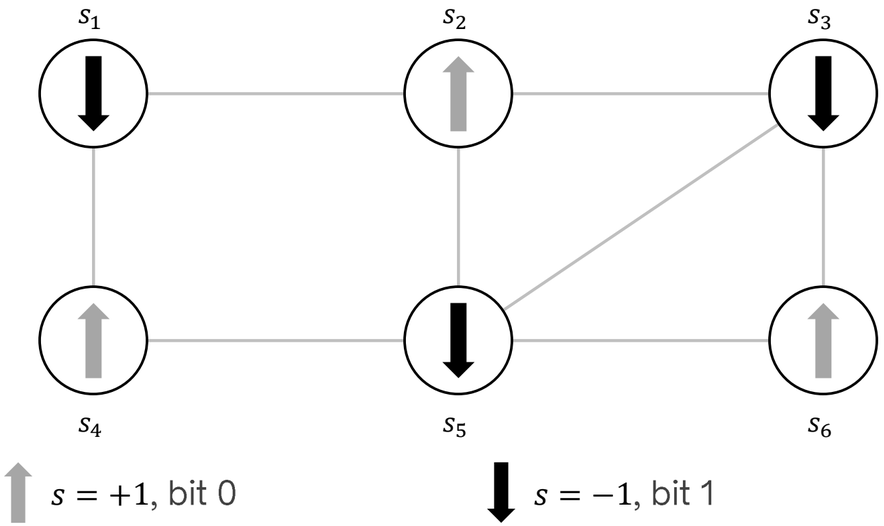
Spin energy $E(s) = \sum_i a_i s_i + \sum_{i<j} b_{ij} s_i s_j + \cdots$

$s_i \to Z_i$ since $Z_i|0\rangle = +|0\rangle$, $Z_i|1\rangle = -|1\rangle$

Hamiltonian $H = \sum_i a_i Z_i + \sum_{i<j} b_{ij} Z_i Z_j + \cdots$

$H|x\rangle = f(x)|x\rangle$
**Ground state = optimal bit string, with energy min f.**

## Spins settle into the lowest energy




**Ground state** $|101010\rangle \to x^* = (1,0,1,0,1,0)$

Each bit is one decision: $x_i^* = 1$ means take option i.

# Summary

We turned a constrained optimization problem into a Hamiltonian whose ground state is the answer.

**1 Constrained problem**

$\min\ f(x)$ s.t. $g(x) \leq K$

Decision variables, cost and constraints: LP, QP, NLP.

**2 Unconstrained QUBO / HUBO**

$\min\ f(x) + \lambda P(x)$

Binary bits; constraints become penalties, with slack for inequalities.

**3 Hamiltonian**

$H|x\rangle = f(x)|x\rangle$

Substitute $x_i = \frac{1-s_i}{2}$, then $s_i \to Z_i$. Ground state = optimal $x^*$.

**Problem**
LP, QP, NLP, BQP

**Bits + penalties**
remove constraints

**QUBO / HUBO**
unconstrained cost

**Hamiltonian**
bits to spins

**Ground state**
= best answer

Suggested reading
A. Lucas, **"Ising formulations of many NP problems"**, Frontiers in Physics 2, 5 (2014). doi:10.3389/fphy.2014.00005

**Next: how do we prepare that ground state on a quantum computer?**

# QOBLIB: benchmarking quantum optimization

Ten problem classes that are hard for classical solvers, with instances and best-known solutions, so quantum and classical methods can be compared fairly.



nature › nature computational science › resources › article

Resource | Open access | Published: 23 June 2026

**The Quantum Optimization Benchmarking Library**

Thorsten Koch, David E. Bernal Neira, Ying Chen, Giorgio Cortiana, Daniel J. Egger, Raoul Heese, Narendra N. Hegade, Alejandro Gomez Cadavid, Rhea Huang, Toshinari Itoko, Thomas Kleinert, Pedro Maciel Xavier, Naeimeh Mohseni, Jhon A. Montanez-Barrera, Koji Nakano, Giacomo Nannicini, Corey O'Meara, Justin Pauckert, Manuel Proissl, Anurag Ramesh, Maximilian Schicker, Noriaki Shimada, Mitsuharu Takeori, Víctor Valls, David Van Bulck, Stefan Woerner & Christa Zoufal — Show fewer authors

*Nature Computational Science* **6**, 653–671 (2026) | Cite this article

The QOBLIB paper, Nature Computational Science (2026)
Instances and code: **github.com/ZIB-AOPT/QOBLIB**

Source: T. Koch, S. Woerner et al., "The Quantum Optimization Benchmarking Library", Nature Computational Science 6, 653–671 (2026); preprint arXiv:2504.03832.

# Final task: can your problem run on a quantum optimizer?

5 MINUTES · SAME PAIRS · AI-USAGE IS ENCOURAGED

## Does your problem admit an Ising formulation, and at what cost?

---

**1** **Make it binary**

How many bits per variable? Integers and continuous values need encoding

**2** **Remove constraints**

A penalty for each constraint, slack bits for inequalities, then check every term.

**3** **Count the overhead**

Total qubits, couplings, highest interaction order (QUBO or HUBO)

5:00

**Try:** “Turn my problem into a QUBO or HUBO, list penalties and slack bits, and count qubits.”

# End of Part I

