Quantum Algorithms in Real-World Optimisation

Follow an optimisation application from an original decision to an independently verified answer.

Core 21 min lesson Dr. Charith Jayasekara

By the end, you'll be able to

- Encode a two-job scheduling objective.
- Verify transformed costs against original loads.
- Design an end-to-end comparison with relevant classical methods.

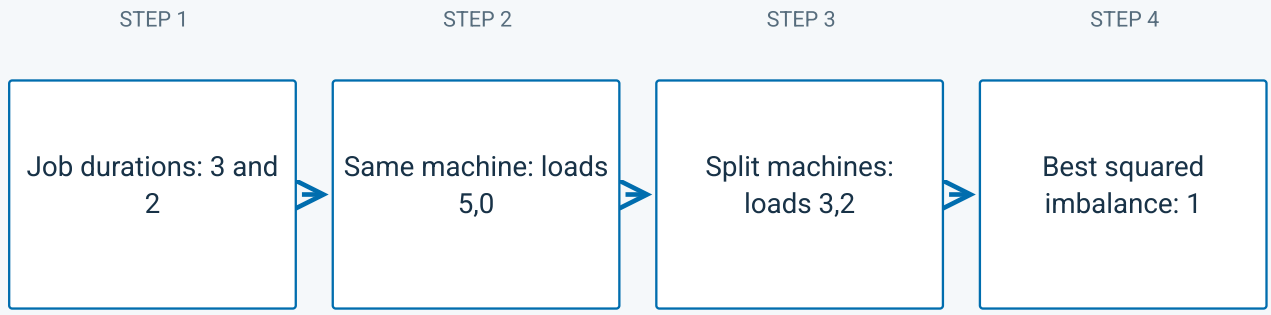
Begin with an application decision

A real optimisation project starts with a decision, constraints, data quality and an acceptable answer. A circuit is one possible solver component after those pieces are specified. The most useful first question is often whether a simpler model or classical method already meets the application's needs. Quantum experiments should test a concrete hypothesis about a defined workload.

Prerequisites: QUBO/Ising modelling, QAOA or annealing, and resource-aware complexity. This lesson develops a small scheduling example that is fully checkable and uses it to expose costs that a toy circuit can hide.

WORKED DIAGRAM · A34

Two-job load balancing



Costs are 25 for 00/11 and 1 for 01/10. The best makespan is 3; a zero imbalance is impossible for these durations.

The application setup

Specify the decision variables, hard constraints, objective, data assumptions and operational time budget. State whether the user needs an optimum, a bound, a robust feasible plan or an improved incumbent. A solution that saves a small

objective amount but takes longer than the planning window may be useless. A fast candidate that violates a safety or capacity constraint is also useless.

Keep an independent classical evaluator for the original application. Transformed energies, penalties and decoded bitstrings should be checked against that evaluator. This prevents an encoding mistake from being misreported as a solver improvement. Small known instances are especially valuable before large experiments become difficult to inspect.

Worked example: two jobs on two machines

There are jobs of durations three and two. Each must be assigned to machine zero or machine one. Let each binary variable be one when its job goes to machine one. Machine-one load is three x-zero plus two x-one. The other load is five minus that expression. Minimise the squared load difference.

The difference is five minus six x-zero minus four x-one. Squaring and using the binary-square identity gives

$$C = 25 - 24x_0 - 24x_1 + 48x_0x_1.$$

The assignments 00 and 11 put both jobs on the same machine and have cost 25. The assignments 01 and 10 split the jobs and have cost one. The optimum is not zero because durations three and two cannot be perfectly balanced. A solver returning cost one has solved this instance exactly; demanding zero would indicate a misunderstanding of the model.

The makespan is the larger load. It is five for unsplit assignments and three for split assignments. On this tiny instance, minimising squared imbalance also minimises makespan. In a larger application, verify that a convenient surrogate objective preserves the desired operational preference rather than assuming those metrics are always interchangeable.

The operational loads show what the squared-imbalance objective means.

$$L_1 = 3x_0 + 2x_1, \qquad L_0 = 5 - L_1, \qquad C = (L_0 - L_1)^2, \qquad T_{\mathrm{makespan}} = \max(L_0, L_1).$$

$x_i = 1$ assigns job i to machine one. A split assignment has loads three and two, cost one and makespan three; the unsplit case has cost twenty-five and makespan five. These are different quantities even when their minimisers coincide on this instance.

Translate only after testing the model

Enumerate the four assignments and compare original loads, squared difference and QUBO cost. Convert to Ising form if the chosen solver needs it, then verify the same energy ordering. Decide whether the algorithm optimises expectation, samples low energies or searches for a certificate. Choose a measurement and decoding rule consistent with that task description.

For this example, uniform sampling succeeds half the time and a deterministic classical rule immediately splits the jobs. A quantum implementation is a learning exercise. Scaling the number of jobs does not automatically produce an advantage: specialised scheduling heuristics and exact methods exploit structure that blind search ignores.

A complete experimental workflow

Choose an instance family reflecting the application rather than arbitrary convenient coefficients. Establish classical reference answers or bounds where possible. Select a quantum method and include model conversion, embedding or circuit compilation, training, sampling and decoding costs. Evaluate candidate feasibility and quality with the original application evaluator.

Use held-out instances for final comparisons after tuning. Report time-to-target or quality-under-budget distributions, not just the best sample. Record random seeds, parameter schedules, stopping criteria and failures. A business-facing objective should be translated back into meaningful units, such as delay minutes or energy cost, rather than only an unexplained Ising energy.

Data and constraints can dominate

Real schedules include machine availability, dependencies, setup times, uncertain durations and changing requests. Encoding all of them can add variables and higher-order interactions. Reducing those interactions to quadratic form may require auxiliaries, and hardware connectivity may add further overhead. Report the final logical and physical sizes separately.

Uncertainty creates another modelling decision. An optimum for one forecast may be fragile under small input changes. Robust or stochastic optimisation changes the objective and possibly the candidate representation. A quantum algorithm does not remove the need to choose and justify that model. Check performance under realistic perturbations rather than evaluating only a fixed clean dataset.

Compare against relevant alternatives

For scheduling, a classical baseline should include problem-aware heuristics or solvers, not only random guessing. Match the allowed planning time and target quality. If quantum parameters are trained across many related instances, decide how that training cost is amortised and give classical learning or tuning methods a comparable opportunity.

A query theorem for unstructured search does not establish an application speedup for a structured scheduling problem. QAOA is a concrete candidate method with instance-dependent behaviour; its primary formulation motivates cost-and-mixer circuits, not a guarantee of universal practical superiority. Farhi, Goldstone and Gutmann. Fair comparison methodology is discussed in Rønnow and colleagues.

What counts as a useful result?

A negative result can be informative: perhaps loading dominates, the encoding is too large, a classical heuristic already meets the target, or noise erases a small improvement. Documenting that result helps refine the research question. A useful project need not end with a quantum advantage claim; it should end with evidence about the defined application hypothesis.

Before scaling, establish a small end-to-end example with no hidden steps. Students should be able to calculate its objective, inspect its encoding and verify the returned schedule. That makes later resource and performance arguments concrete rather than relying on the prestige of an algorithm name.

Exercises and worked answers

Exercise 1. Change the job durations to three and three. What are the best squared imbalance and makespan?

Check the worked answer

Splitting the jobs gives equal loads three and three, squared imbalance zero and makespan three. Putting both on one machine gives imbalance squared 36 and makespan six. The optimum changes because the input data changed.

Exercise 2. A solver returns a low penalty energy but assigns one job to no machine after decoding. Has it solved the scheduling task?

Check the worked answer

No. The original assignment constraint is violated. Inspect the model, penalty weight and decoding rule. Solver success on a transformed objective is meaningful only when it corresponds to a valid application solution under the declared task description.

Check your understanding

PAUSE & THINK

What is the minimum squared load imbalance for jobs of durations 3 and 2?

- [ ] 0
- [ ] 1
- [ ] 25

Answer: 1. Splitting gives load difference 1, whose square is 1. Both jobs on one machine give difference 5 and squared cost 25.

Sources

- Farhi, Goldstone and Gutmann
- Rønnow and colleagues