Quantum Optimisation and Classical Baselines

Compare optimisation methods using matched targets and complete resource accounting.

Core 22 min lesson Dr. Charith Jayasekara

By the end, you'll be able to

- Compute repetitions for a target success probability.
- Separate one-time setup from repeated cost.
- Design baselines and evaluation rules before selecting results.

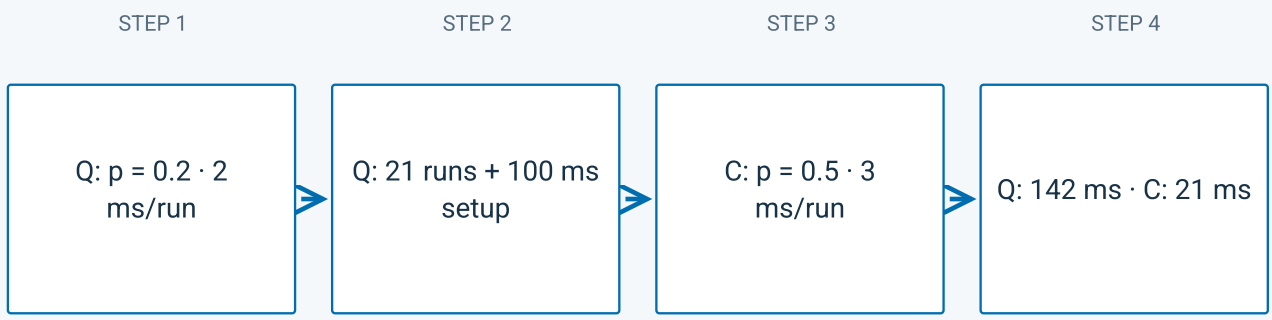
Compare solutions to the same problem

A quantum optimiser is useful only in relation to a defined task and an appropriate alternative. "Better" might mean a lower objective, a higher probability of feasibility, less time to a target, or a stronger proof of optimality. Those goals can rank methods differently. Begin by deciding what an application needs, then make every competing method meet the same requirement.

Prerequisites: complexity and resource models and the QAOA or annealing examples. This lesson is about experimental design rather than a new circuit. The primary speedup-methodology literature distinguishes several comparison questions and shows how limited datasets can mislead. Rønnow and colleagues.

WORKED DIAGRAM · A21

Time to 99% success



Independent repetitions need 21 attempts for Q and 7 for C at a 99% target. A shorter individual attempt does not imply a shorter task time.

State the comparison setup

Specify the instance family, instance sizes, target quality, success probability and permitted preprocessing. Define whether the timing includes model conversion, data loading, parameter training, programming, execution, readout and decoding. Equalising one resource while hiding another can reverse the conclusion. A circuit execution time alone is not an end-to-end application time.

Use appropriate classical algorithms. For a graph problem, candidates might include exact enumeration on small cases, branch-and-bound, specialised approximation algorithms, local search or strong mixed-integer solvers. A comparison against blind random search is informative only when random search is a meaningful baseline for the task. Give each method a reasonable tuning budget and report how it was selected.

Worked example: time to a target

Suppose method Q succeeds with probability 0.2 per independent attempt, each costing two milliseconds after a one-time setup of one hundred milliseconds. Method C succeeds with probability 0.5 per attempt, each costing three milliseconds and no setup. We want at least a 99-percent chance of one successful candidate.

For independent attempts with success probability p, choose the smallest integer R for which one minus one minus p raised to R is at least 0.99. Q needs 21 attempts, costing 42 milliseconds plus setup, or 142 milliseconds total. C needs seven attempts, costing 21 milliseconds. Q has a shorter individual attempt but loses on this target and accounting rule.

If the setup is reused across many identical tasks, its amortised cost changes. If attempts are correlated, the independent-repeat formula may be wrong. If success probabilities are estimated from few trials, their uncertainty affects the repetition estimate. Report those assumptions rather than presenting one rounded time as a universal property of the method.

Independent attempts convert per-run success into a confidence target and a complete time estimate.

$$R = \left\lceil \frac{\log(1-c)}{\log(1-p)} \right\rceil, \qquad T = T_{\text{setup}} + R\tau, \qquad c = 0.99: \quad T_Q = 100 + 21(2) = 142\ \text{ms}, \quad T_C = 7(3) = 21\ \text{ms}.$$

p is success probability per independent attempt, c the required probability of at least one success, and τ time per attempt. The formula assumes $0 < p, c < 1$ and includes the one-time setup.

Separate quality from feasibility

A low transformed energy is useful only if its decoded candidate meets the original constraints. Report feasibility rate and objective quality separately. If a penalty model permits infeasible low-energy states, a solver can appear strong by optimising the wrong target. Independent verification should evaluate the application constraints directly.

For approximate solutions, define a gap or approximation measure. If the optimum is known, a relative gap can be useful, but it behaves badly when objective values are zero or change under constant shifts. A domain-specific absolute tolerance may be better. State the exact formula and sign convention so another reader can reproduce the ranking.

Prevent selection bias

Choose an instance-generation process before looking at results. Report distributions across instances, not only the examples where one method wins. Keep tuning instances separate from evaluation instances where possible. Random seeds, graph structures and coefficient precision can change difficulty for both classical and quantum methods.

Report failed or timed-out runs with a consistent rule. Removing hard cases from one method’s average can create an artificial advantage. Median, quantiles and success curves often explain a heavy-tailed distribution better than a single mean. Include uncertainty from repeated trials and enough instance detail for reproduction.

Understand different kinds of advantage

A finite-size empirical advantage means one implementation beat specified alternatives on specified cases. An asymptotic advantage concerns scaling with input size under a resource model. A query advantage counts abstract oracle access. A physical advantage includes implementation costs. None of these statements automatically implies all the others.

Classical algorithms improve, and quantum hardware changes. Date a benchmark and describe the versions used. A past result remains evidence about that experiment even if a later method changes the best-known comparison. Avoid converting one benchmark into a timeless statement that a whole category of problems is solved faster quantumly.

Step-by-step benchmark plan

First freeze the task and instance distribution. Establish exact answers or trustworthy bounds for small cases. Select and tune baselines on a separate budget. Run all methods under the declared accounting convention. Verify outputs independently, estimate success and quality distributions, and calculate time-to-target only under justified statistical assumptions. Finally publish enough parameters and raw summaries for another investigator to challenge the result.

For a learning exercise, use a family of small MaxCut graphs. Compare uniform sampling, a simple greedy local search and an ideal QAOA parameter search. Count every objective evaluation in the QAOA training loop. The goal is to learn what a fair comparison requires, not to manufacture a speedup from a simulator's update speed.

A second target changes the comparison

Change the worked success target from 99 percent to 50 percent. Q now needs four independent attempts, since three attempts succeed with probability only 0.488 while four give 0.5904. Its total with setup is 108 milliseconds. C needs one attempt and three milliseconds. The target probability is therefore an essential benchmark parameter rather than an optional reporting detail.

For a stream of many tasks reusing the same setup, compute both total and amortised costs. Do not amortise a setup across hypothetical future uses if the actual application performs only one task. Conversely, charging reusable preprocessing repeatedly can unfairly penalise a method designed for repeated queries.

The success probability must also refer to the same answer quality. If Q's 0.2 rate means exact optimum but C's 0.5 rate means merely feasible, the numerical comparison is invalid. Write the target predicate first, then estimate both methods against it.

Exercises and worked answers

Exercise 1. If Q's setup is excluded, does it beat C in the worked example at the 99-percent target?

<details>
<summary>Check the worked answer</summary>

No. Q still requires 42 milliseconds of attempts, compared with C's 21. Setup exclusion changes the numbers but does not change this ranking. The exercise also shows why a shorter attempt is insufficient evidence.
</details>

Exercise 2. A report chooses the best of one hundred tuned quantum runs but compares it with one untuned classical run. What must change?

<details>
<summary>Check the worked answer</summary>

Give both methods a declared comparable tuning policy, include tuning cost when relevant and evaluate selected configurations on fresh instances or trials. Report the distribution, not merely the best observed candidate. Otherwise selection effort is silently counted as free for one side.
</details>

Check your understanding

PAUSE & THINK

Which method is faster at the stated 99% target including setup?

- [ ] Q, because each run is shorter
- [ ] C, at 21 ms versus Q's 142 ms
- [ ] They have equal total time

Answer: C, at 21 ms versus Q's 142 ms. The success target requires different repetition counts. Include the setup and all attempts, not just one run.

Sources

- Rønnow and colleagues