

Hybrid optimization with QAOA

How can a short quantum circuit help us search for good solutions?

QUANTUM

Run a circuit.
Return measured bits.



CLASSICAL

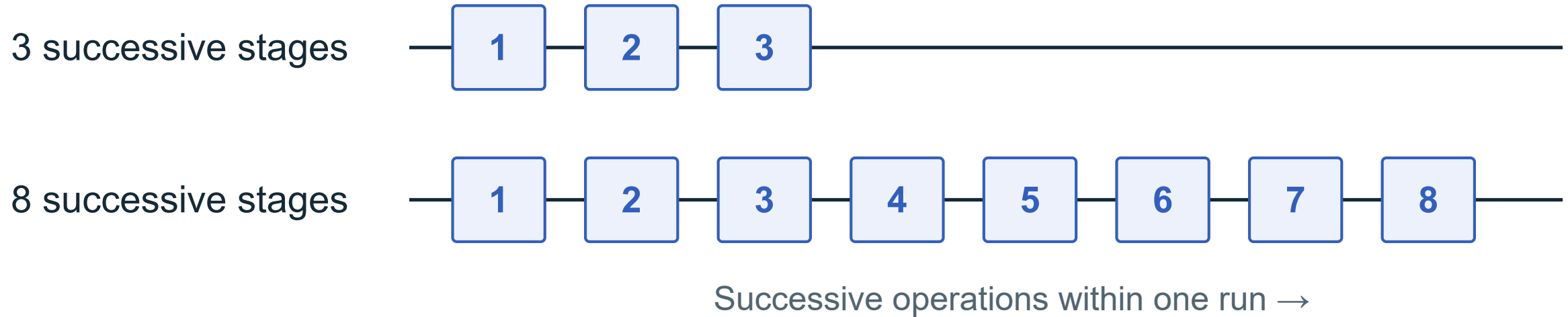
Assess the results.
Choose the next settings.

We can search for good solutions by combining short quantum circuits with classical feedback.

- 01 Classical feedback lets us search using circuits of limited depth.
 - 02 QAOA turns a scoring problem into an adjustable quantum circuit.
 - 03 Measured scores guide the search for better circuit settings.
-

Hardware imperfections limit reliable circuit depth.

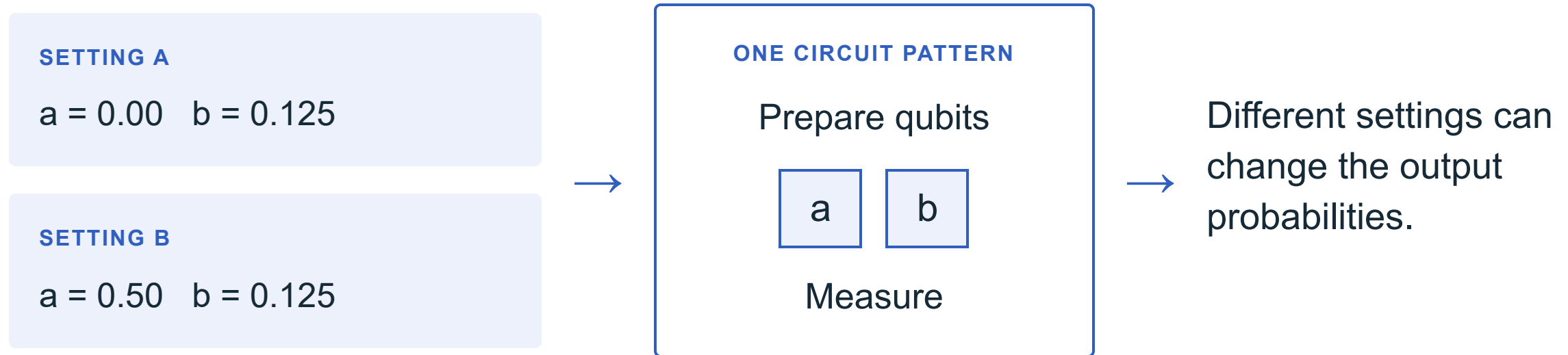
A gate can introduce an error. A longer sequence exposes the state to more operations and more elapsed time.



Depth counts successive stages of operations in one circuit.

Gates on different qubits can share a stage. The usable depth depends on the device and the circuit.

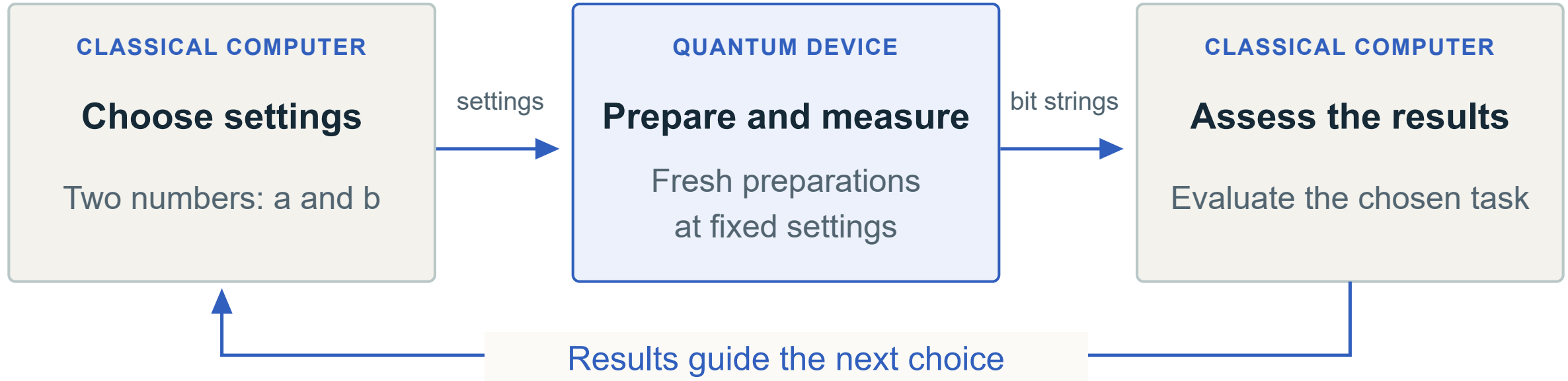
A fixed circuit pattern can have adjustable settings.



We change the circuit settings so that good candidate solutions occur more often when we measure.

a and b are settings we choose. The measured bit string describes a candidate solution.

Classical feedback chooses the next circuit settings.



The quantum device supplies samples. Classical computation guides further runs.

We search the settings to improve the sampled candidates. Each shot starts with a fresh preparation.

MaxCut asks us to split vertices into two groups.

This graph is small enough that we can check every answer. We use it to follow the method.

ONE GRAPH THROUGHOUT



Each vertex belongs to group **0** or group **1**.

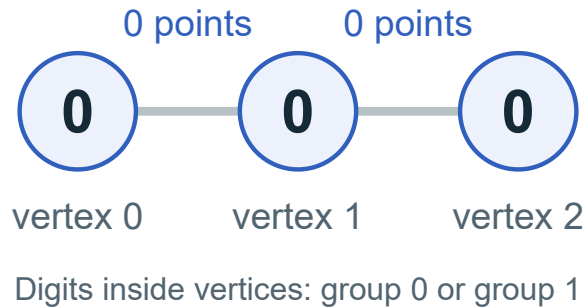
We earn one point for each edge joining different groups.

We want the largest score.

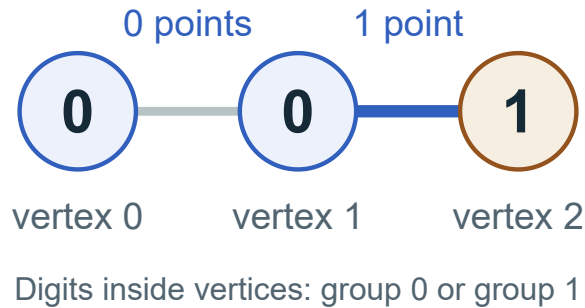
Candidate: $\mathbf{x_0x_1x_2}$ One bit per vertex, in the order 0, 1, 2.

The score counts edges whose endpoint bits differ.

CANDIDATE 000

Score **0**

CANDIDATE 001

Score **1**

CANDIDATE 010

Score **2**

010 cuts both edges. Its score is 2, the best possible for this graph.

101 describes the same partition with the group labels exchanged.

Uniform sampling gives each of the eight strings probability $\frac{1}{8}$.

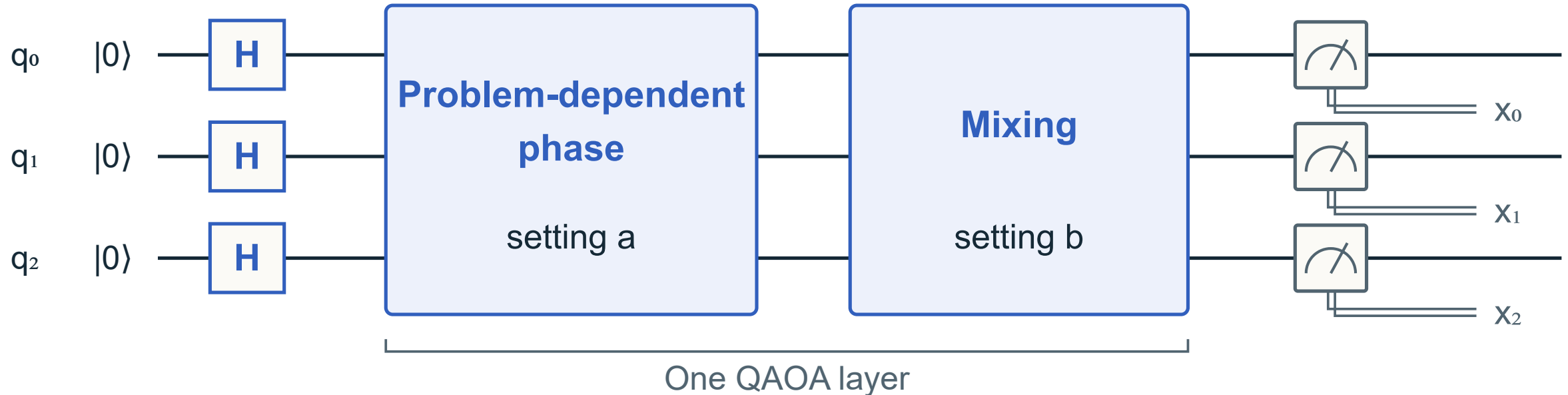
Score	Candidate strings $x_0x_1x_2$	Probability of this score
0	000, 111	$2/8 = 25\%$
1	001, 011, 100, 110	$4/8 = 50\%$
2	010, 101	$2/8 = 25\%$

Uniform sampling finds a score-2 candidate on one quarter of the runs, on average.

This gives us a starting distribution to compare with other circuit settings.

QAOA alternates problem phases and mixing.

Quantum Approximate Optimization Algorithm · One qubit per vertex.



PROBLEM-DEPENDENT PHASE

a sets the phase change per score point.

MIXING

b controls how amplitudes combine afterward.

H on every qubit prepares equal amplitudes. Each block contains gates; this example uses one layer.

Does the phase block change the immediate probabilities?

At setting B, $a = 0.50$ adds a phase of $-\pi/2$ per score point.

Score	Phase added	Amplitude factor
0	0	1
1	$-\pi/2$	$-i$
2	$-\pi$	-1

MEASURE BEFORE MIXING

H preparation gives each candidate probability $1/8$.

What is it after these phase changes?

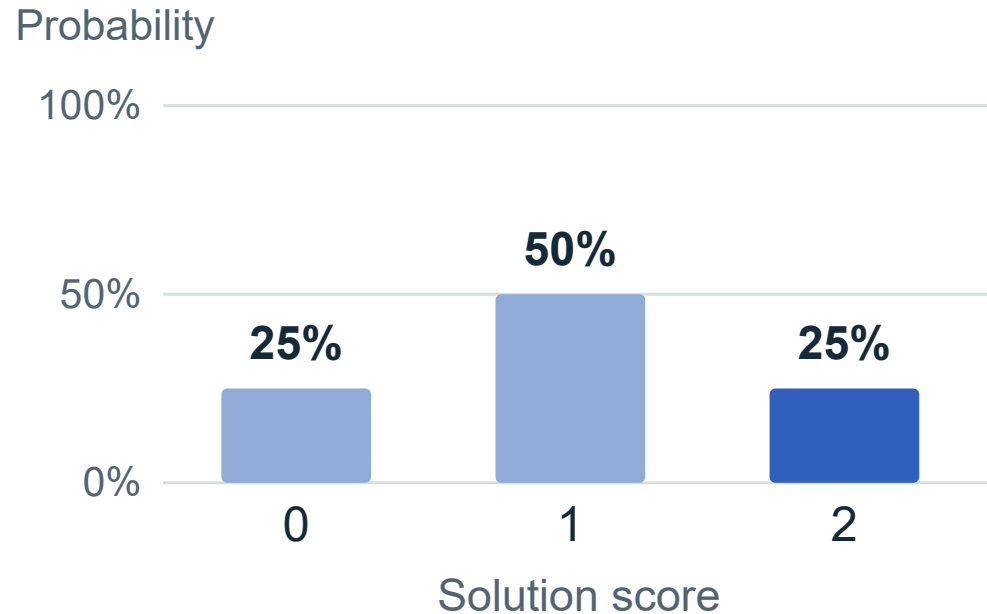
Still $1/8$. All three phase factors have magnitude 1.

Mixing then allows the phases to affect the final probabilities.

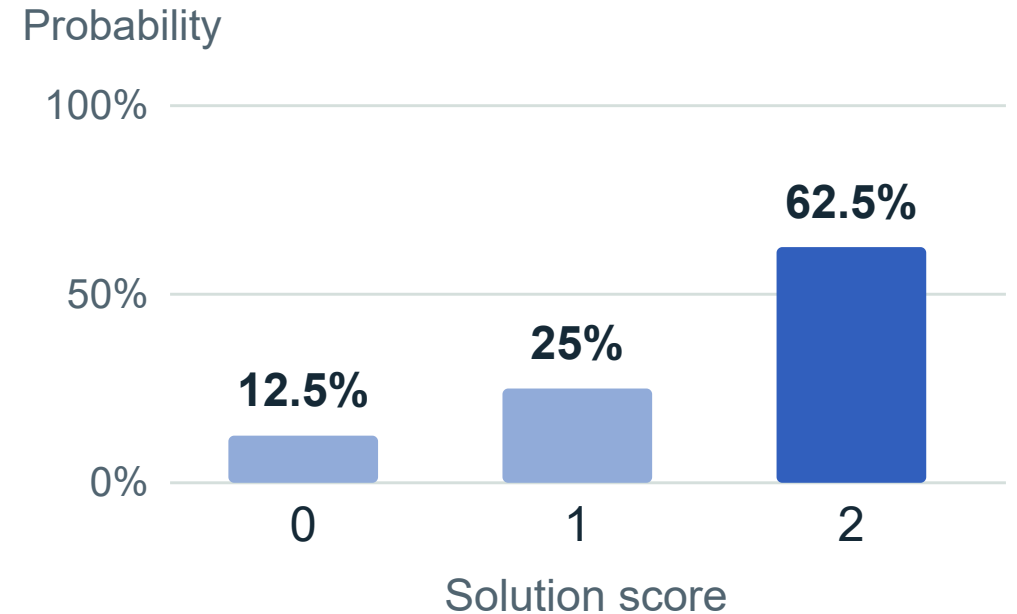
Changing the settings changes the sampled scores.

b controls mixing. At $b = 0$ there is no mixing; here both choices use $b = 0.125$.

SETTING A · UNIFORM BASELINE · $a = 0.00$



SETTING B · $a = 0.50$



At B, mixing turns the score-dependent phases into more score-2 outcomes.

Ideal probabilities · The score influences interference; the settings determine whether the result is useful.

One shot returns one candidate bit string.



ILLUSTRATIVE RECORDS FROM FRESH SHOTS

010 001 101 000 ...

We keep the settings fixed while collecting a batch of samples.

Strings use the labelled vertex order $x_0x_1x_2$. These are separate circuit executions.

Repeated shots give counts for each score.

Candidate score	Number of results
0	10
1	20
2	50

$10 + 20 + 50 = 80$ shots.

ONE BATCH AT SETTING B

$a = 0.50$

$b = 0.125$

80 fresh runs

Every returned string is scored with the same graph.

The counts tell us how often each score occurred in this batch.

Illustrative counts, chosen to match the ideal proportions. Actual finite samples can differ.

The average score is total score divided by shots.

Score	Count	Contribution to the total score
0	10	$0 \times 10 = 0$
1	20	$1 \times 20 = 20$
2	50	$2 \times 50 = 100$

$$\text{Average score} = \frac{0 + 20 + 100}{80} = 1.50$$

This estimates the expected score at setting B.

The average evaluates a circuit setting across many samples.

Which tested setting has the highest average score?

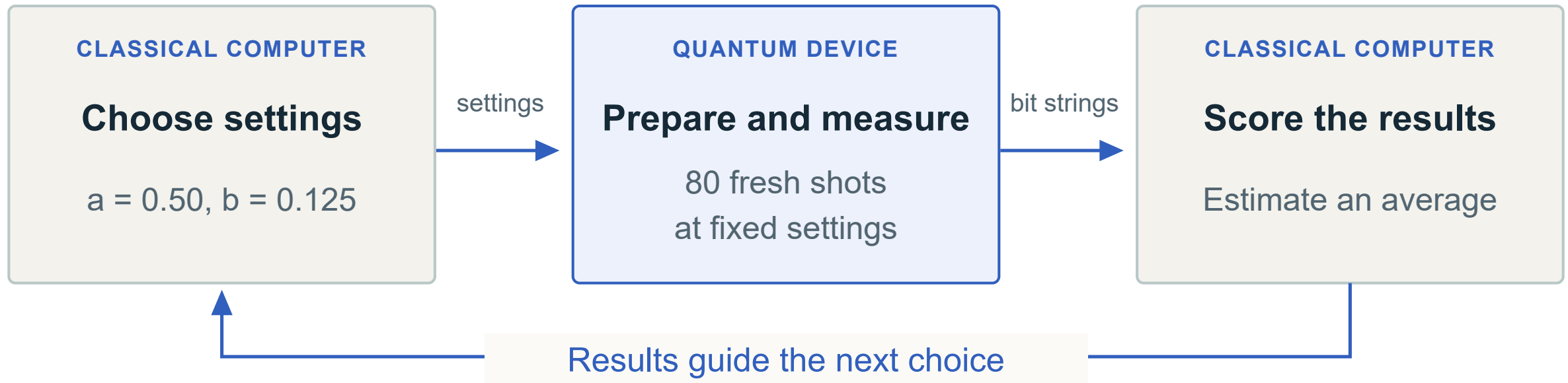
Setting	Score 0 count	Score 1 count	Score 2 count	Average score
A	20	40	20	1.00
B	10	20	50	1.50
C	50	20	10	0.50

B has the highest observed average.

C performs worse than A. A change of settings need not improve the result.

80 fresh shots per setting · Illustrative counts.
C uses $a = 0.50$, $b = 0.375$; the graph and one-layer circuit pattern stay fixed.

The classical search uses scores to propose further settings.



We retain the best observed setting while exploring further choices.

One option is a coarse grid of settings, followed by a closer search around the best observed pair.

The output is a candidate, with its own score.

A POSSIBLE RESULT AFTER TUNING

USED TO COMPARE SETTINGS

Observed average at B

1.50

One number summarizing a batch of samples.



Digits inside vertices: group 0 or group 1

010

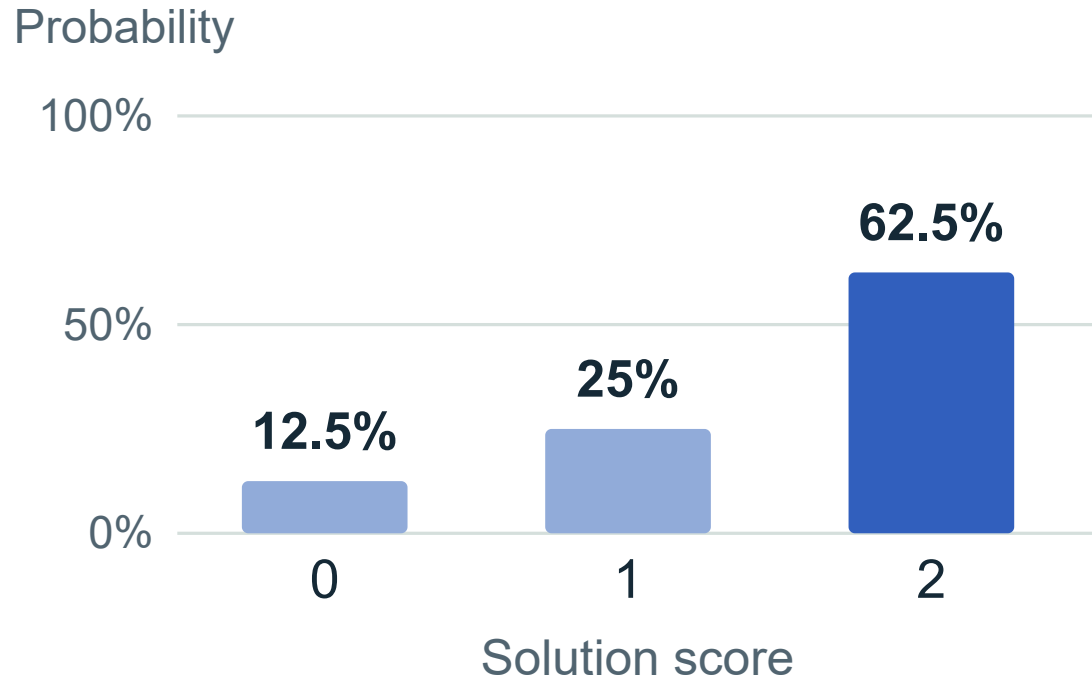
Candidate score: 2

We sample with the chosen settings, score the candidates and keep a good result.

The average guides tuning. The bit string describes the solution.

Does the better setting always give an optimal cut?

SETTING B · IDEAL PROBABILITIES



No. A score of 0 or 1 still has total probability **37.5%**.

Every returned candidate can be scored classically.

For this tiny graph we can verify the optimum by inspection. In general, a good candidate does not certify global optimality.

Short circuits can still require many executions.

SETTING A

80

fresh circuit runs

SETTING B

80

fresh circuit runs

SETTING C

80

fresh circuit runs

3 settings × 80 shots

240 circuit executions

Depth describes one run. Shots and trials determine how often we run it.

More layers add settings and circuit depth. Noise limits how many layers we can use reliably.

A useful comparison includes solution quality and total work.

OUR GRAPH IS EASY TO SOLVE DIRECTLY



Digits inside vertices: group 0 or group 1

Assign alternating groups.
Both edges cross: score **2**.

FOR THE SAME PROBLEM

How good are the solutions?

How much quantum and classical work produces them?

How does that compare with a suitable classical method?

This example demonstrates the workflow. It does not establish a speedup.

We can search for good solutions by combining short quantum circuits with classical feedback.

- 01 Classical feedback lets us search using circuits of limited depth.

- 02 QAOA turns a scoring problem into an adjustable quantum circuit.

- 03 Measured scores guide the search for better circuit settings.

Which setting would we retain from these results?

Setting	Score 0 count	Score 1 count	Score 2 count
D	20	20	40
E	40	20	20

- 01 What is the average score for each setting?
- 02 Which setting has the better observed average?
- 03 Would one more shot necessarily return a score-2 candidate?

80 fresh shots per setting · Supplied illustrative sample data.
Average score = total score / number of shots.