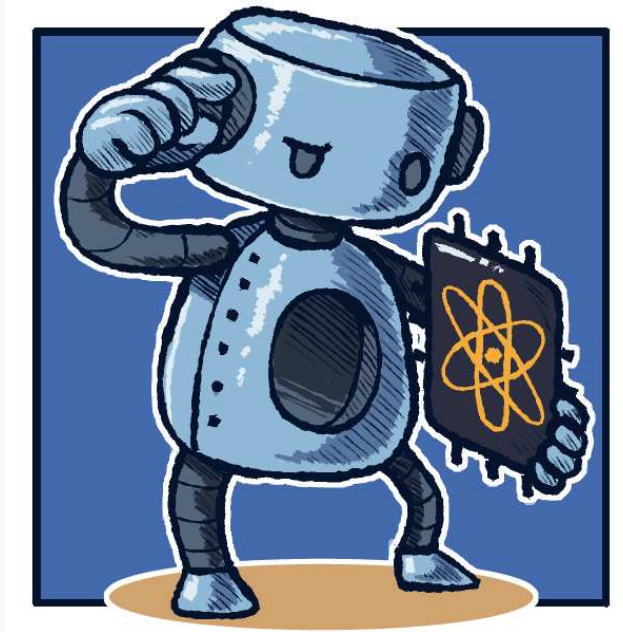


Quantum Oracle Engineering

Lesson 1: A different computer



A dice game

First to 100

you

0

THEM

0



roll

hold

Which move should we choose?

Pig, a dice game

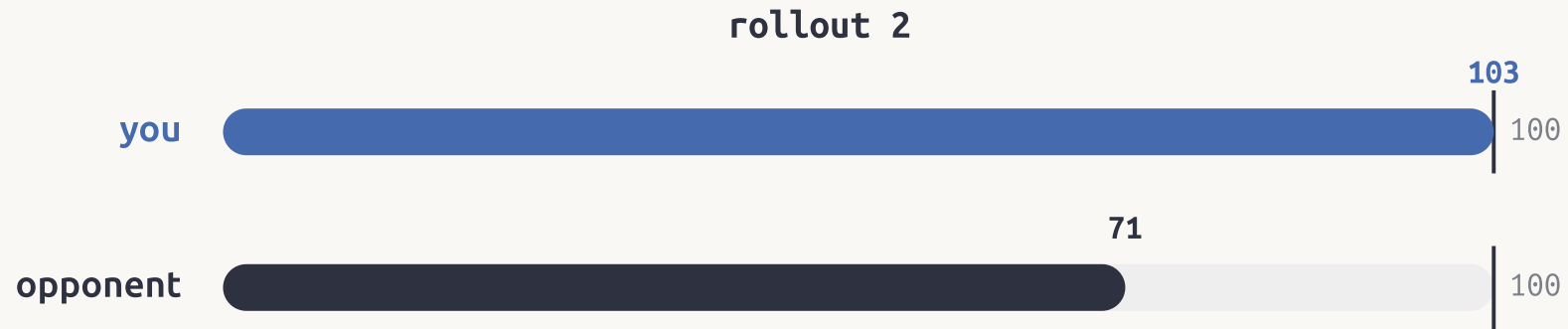
you	opponent	this turn
62	71	12
 +4	 ×	
a roll adds	a 1 wipes	a hold keeps it

roll again 

?

hold 

Play one possible future



you hold, +18

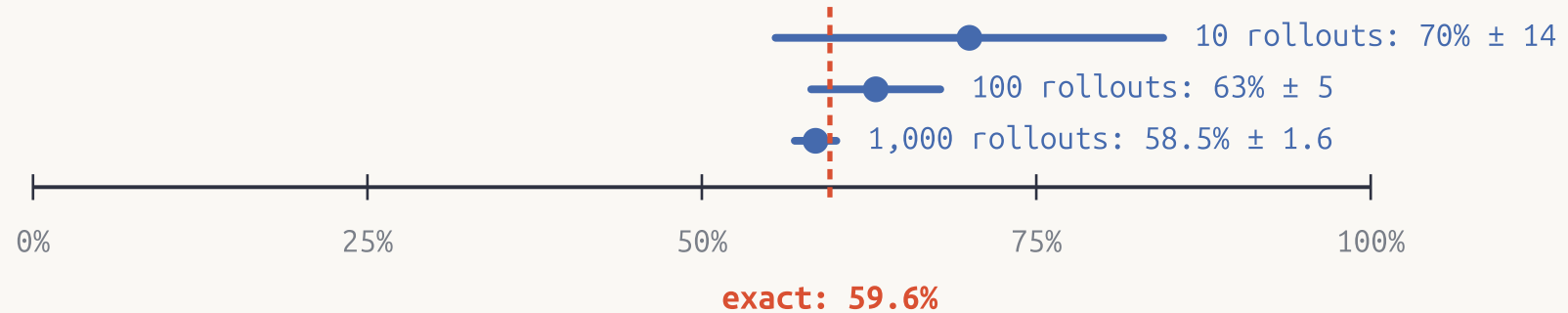
win = 1

Repeat, then count wins



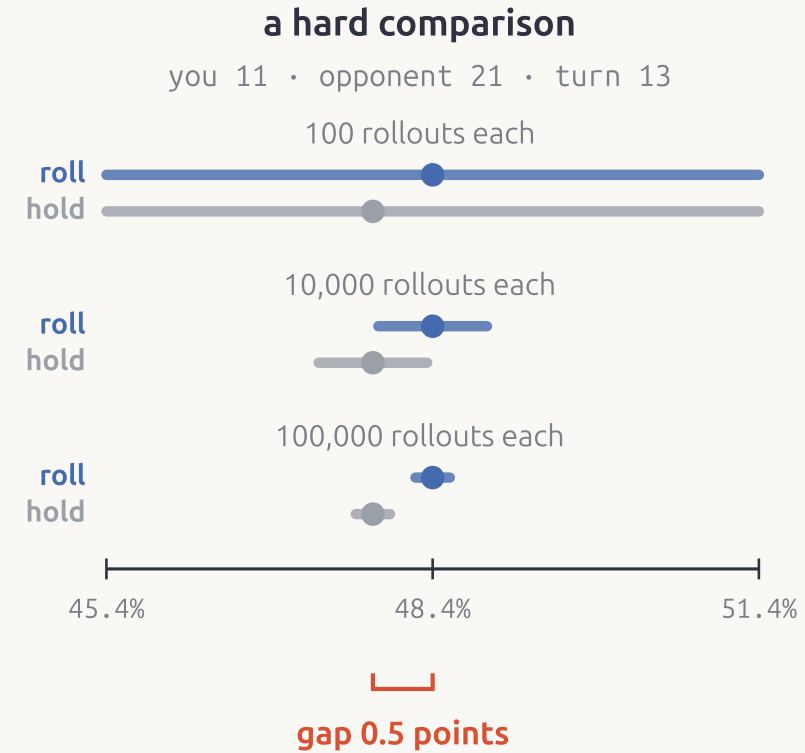
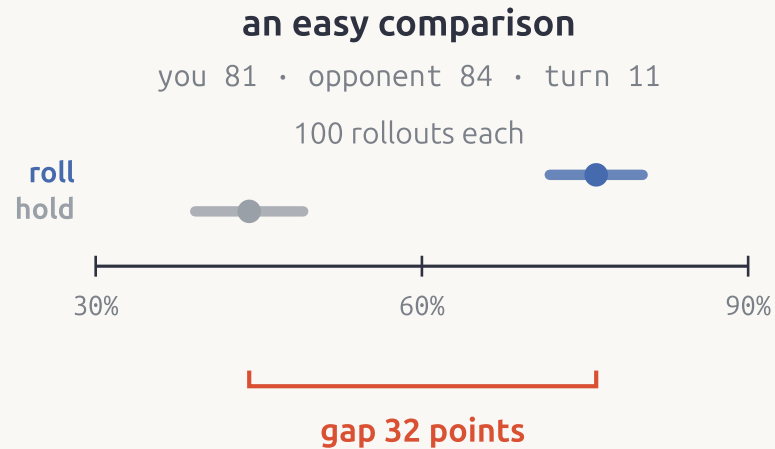
7 of 10

win rate of "roll", estimated



a Monte Carlo estimate

When is the answer clear enough?



Do we need the better move?

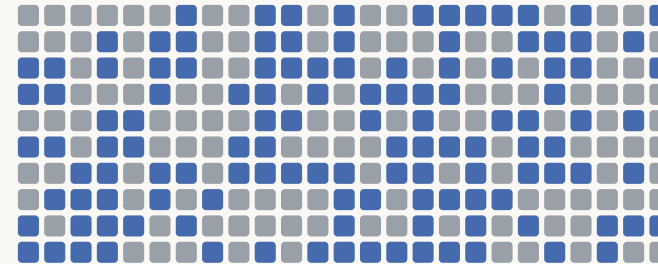
one decision



either move is fine

the worse pick costs half a point, once

a thousand decisions



505 to 495, over a thousand games

the gap begins to matter

ranking two strategies, tuning a player, a tournament

“Assume oracle access.”

As C is Hermitian, we can solve the equation $C\vec{y} = \begin{pmatrix} b \\ 0 \end{pmatrix}$ to obtain $y = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix}$. Applying this reduction if necessary, the rest of the Letter assumes that A is Hermitian.

We also need an efficient procedure to prepare $|b\rangle$. For example, if b_i and $\sum_{i=1}^n |b_i|^2$ are efficiently computable then we can use the procedure of Ref. [14] to prepare $|b\rangle$. Alternatively, our algorithm could be a subroutine in a larger quantum algorithm of which some other component is responsible for producing $|b\rangle$.

The next step is to decompose $|b\rangle$ in the eigenvector basis, using phase estimation [5–7]. Denote by $|u_j\rangle$ the eigenvectors of A (or equivalently of A^2), and by λ_j the corresponding eigenvalues. Let

$$y(\vec{x}) = \text{sign} \left(\sum_{j=1}^n \alpha_j k(\vec{x}_j, \vec{x}) + b \right), \quad (2)$$

Quantum machine learning with the kernel matrix. – In the quantum setting, assume that oracles for the training data that return quantum vectors $|\vec{x}_j\rangle = 1/\|\vec{x}_j\| \sum_{k=1}^N (\vec{x}_j)_k |k\rangle$, the norms $\|\vec{x}_j\|$, and the labels y_j are given. The quantum machine learning performance is relative to these oracles and

ing, where learning is achieved via interaction with a task environment. Here, we consider a special case of reinforcement learning, where the task environment allows quantum access. In addition, we impose certain “naturalness” conditions on the task environment, which rule out the kinds of oracle problems that are studied in quantum query complex-

a corresponding unitary quantum circuit of this form can be produced using standard reversible-computation techniques [5]. As is common in works based on quantum amplitude amplification and estimation [12], we also assume that we have the ability to execute the algorithm $\mathcal{A}^{-1}$, which is the inverse of the unitary part of $\mathcal{A}$. If we do have a description of $\mathcal{A}$ as a quantum circuit, this can be achieved simply by running the circuit backwards, replacing each gate with its inverse.

We first deal with the special case where the output of $\mathcal{A}$ is bounded between 0 and 1. Here a quantum algorithm for approximating $u := \mathbb{E}[v(\mathcal{A})]$ quadratically faster than is possible classically

5.1 The data structure

The input to the quantum procedure is a vector $x \in \mathbb{R}^n$ and a matrix $A \in \mathbb{R}^{m \times n}$. We assume that the input is stored in a classical data structure such that an algorithm that has quantum access to the data structure can create the quantum state $|x\rangle$ corresponding to the vector x and the quantum states $|A_i\rangle$ corresponding to each row A_i of the matrix A , in time $\text{polylog}(mn)$.

It is in fact possible to design a data structure for a matrix A that supports the efficient

$$\mathbb{E}_{g(\mathbf{x}) \sim p_{f,\mathbf{x}}} \|g(\mathbf{x}) - \nabla f(\mathbf{x})\|^2 \leq \sigma^2, \quad \forall \mathbf{x} \in \mathbb{R}^d.$$

In this paper, we further assume quantum access to a stochastic gradient oracle, or access to a quantum stochastic gradient oracle for brevity, that upon query returns a quantum superposition over the probability distribution $p_{f,\mathbf{x}}(\mathbf{v})$.

case, the best known upper bound on the number of queries required to create $|\psi\rangle_n$ is $O(2^n)$ [17]. However, approximations with polynomial complexity in n are possible for many distributions, e.g., log-concave distributions [18]. In the remainder of this section, we assume a given operator $\mathcal{R}$ such that $\mathcal{R} |0\rangle_n = |\psi\rangle_n$.

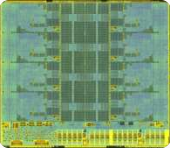
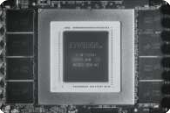
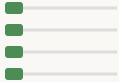
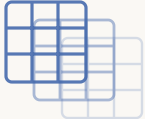
If we are interested in properties of $g(X)$, as discussed in the previous section, then, depending on g , we can use basic arithmetic operations to construct the operator C . Numerous quantum algorithms exist for arithmetic

noisy hardware of the near future.

Another question that has only briefly been addressed in this paper is the loading of considered random distributions or stochastic processes. For auto-correlated processes this can be rather costly and needs to be further investigated. Techniques known from classical Monte Carlo, such as importance sampling [42], might be employed here as well to improve the results or reduce the

quantum transition oracle, which leverages quantum parallelism, amplitude amplification, and entanglement. This powerful concept has been utilized in foundational algorithms such as Grover’s search (Grover, 1996). In this work, we assume access to a quantum transition oracle and a quantum initial state oracle, which serve as quantum analogs for sampling from the classical environment and the initial state distribution, respectively. Using these quantum oracles, we aim to demonstrate the possibility that quantum computation can provide in the

Three cost models

DEVICE		STRENGTH	WORKLOADS				
	CPU	 latency	 branching  queries				
	GPU	 throughput	 training  graphics				
	QPU	 precision  simulation  factoring  rollouts  expectations					

Mismatch costs

CPU

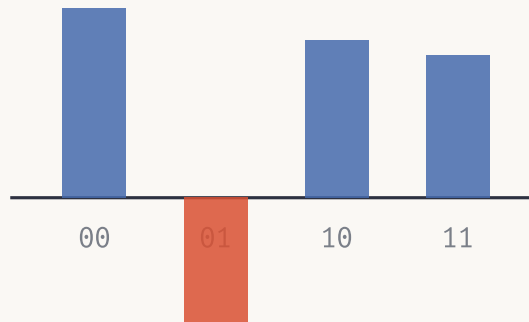


CPU + GPU



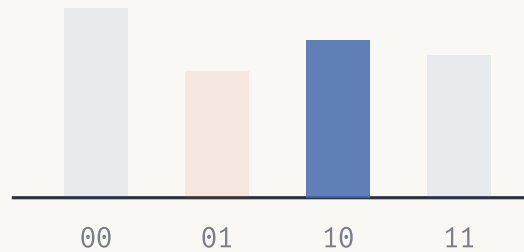
What changes on a quantum computer?

amplitudes



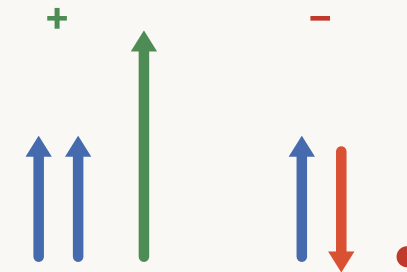
signed, and not yet odds

measure



one outcome: a rollout again

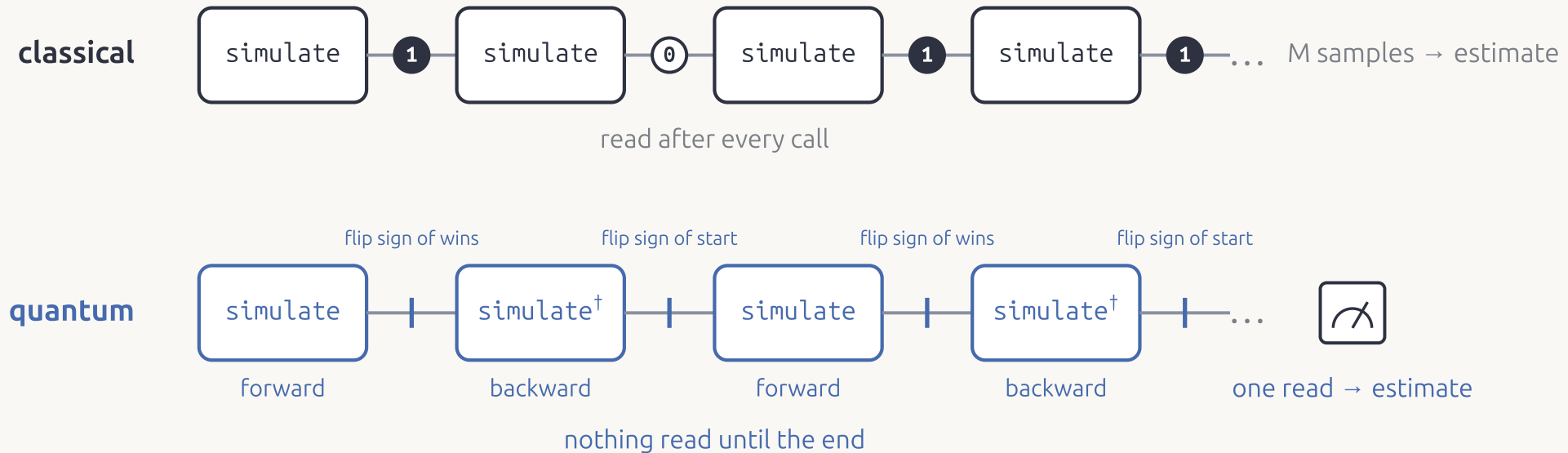
interfere first



agree: add · disagree: cancel

the odds are reshaped before anyone measures

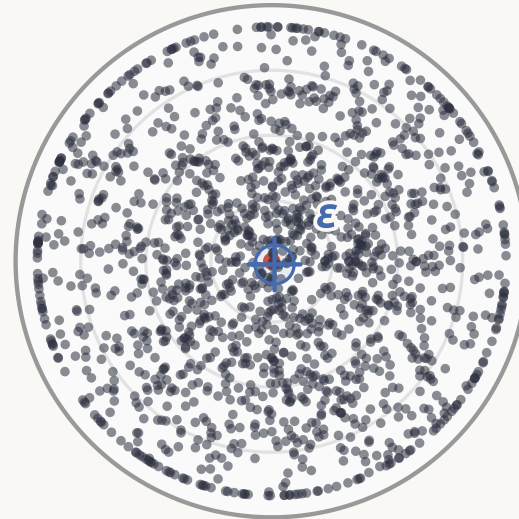
The simulator is the oracle



The classical rate

$$\underbrace{\varepsilon_{\text{classical}}(M) \sim \frac{1}{\sqrt{M}}}_{\text{error after } M \text{ samples}} \implies \underbrace{M_{\text{classical}}(g) \sim \frac{1}{g^2}}_{\text{samples to resolve gap } g}$$

$\varepsilon \div 2$



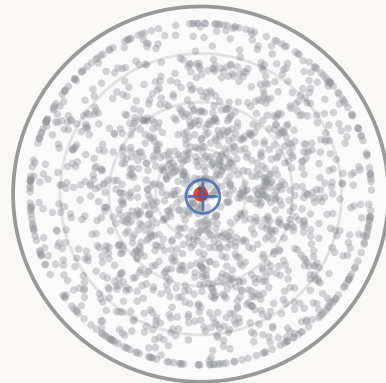
M × 4

M **1,600**

The quantum rate

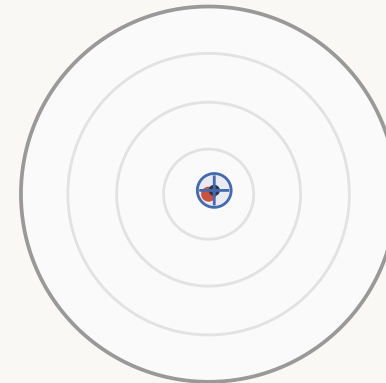
$$\underbrace{\varepsilon_{\text{quantum}}(M) \sim \frac{1}{M}}_{\text{error after } M \text{ queries}} \implies \underbrace{M_{\text{quantum}}(g) \sim \frac{1}{g}}_{\text{queries to resolve gap } g}$$

classical



1,600 darts

quantum



one estimated win rate

0.05: 400 → 20

10^{-3} : $10^6 \rightarrow 10^3$

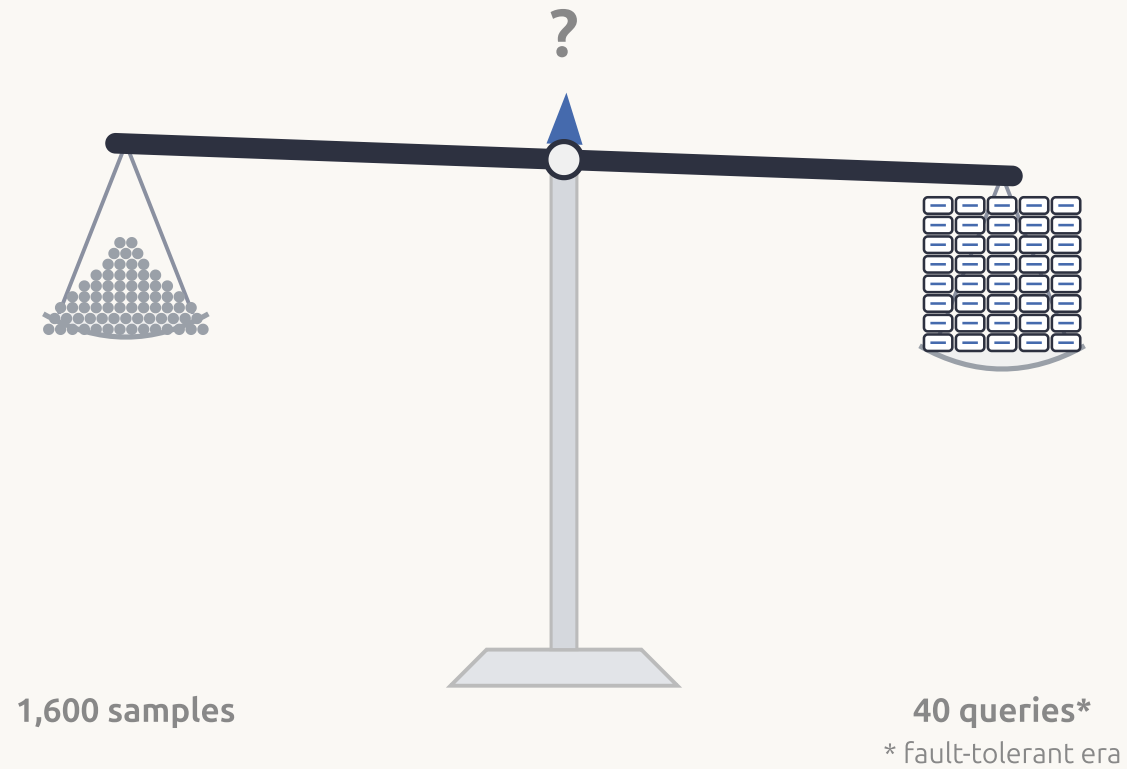
10^{-5} : $10^{10} \rightarrow 10^5$

40

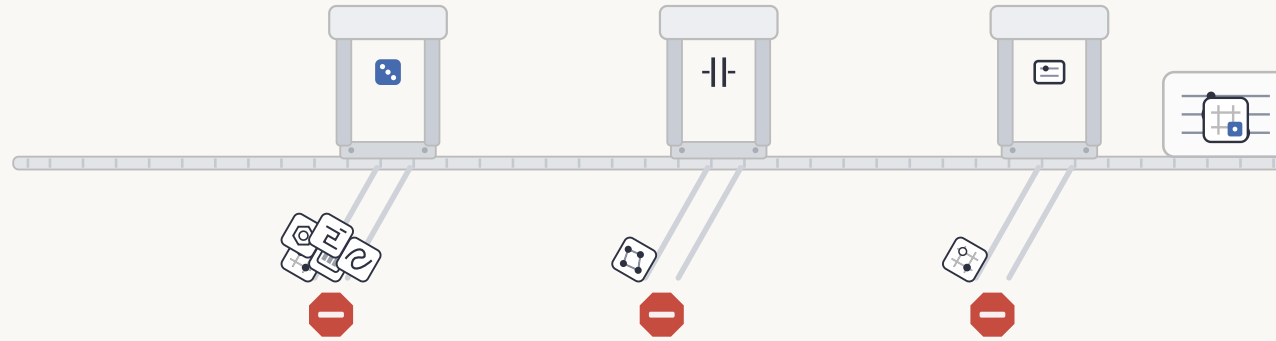
queries

The mechanism

The fine print

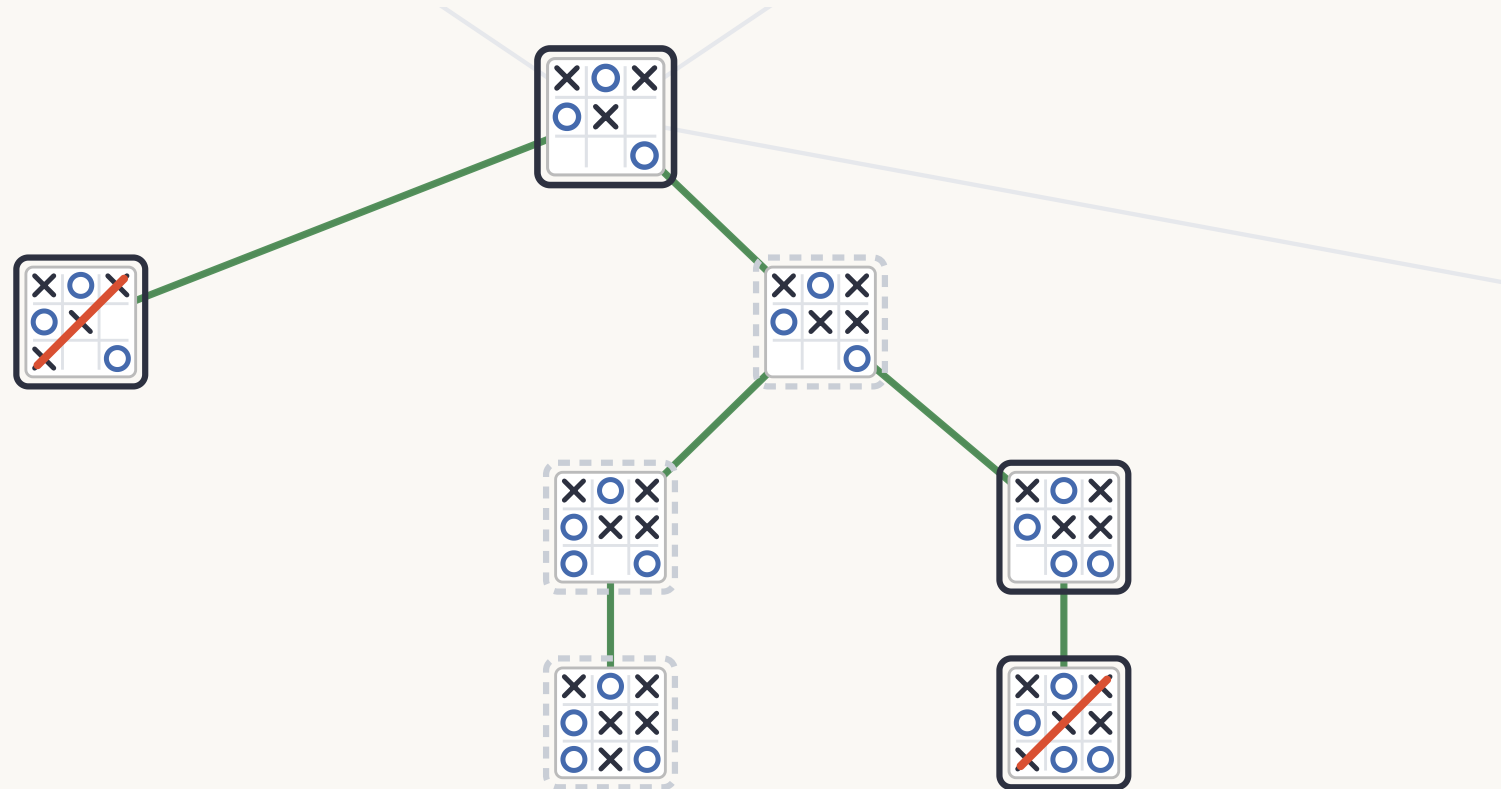


Three questions



- does the best classical method still sample?
- is precision the bottleneck?
- is the oracle worth building?

A deterministic world



Pig, solved

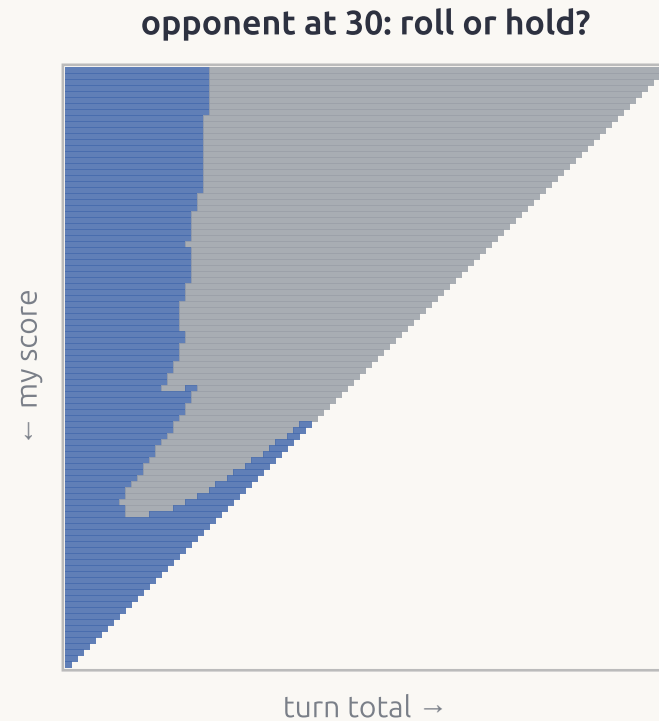
■ roll
■ hold

505,000

positions, one table

no rollouts

Neller and Presser, 2004



Whose dice?

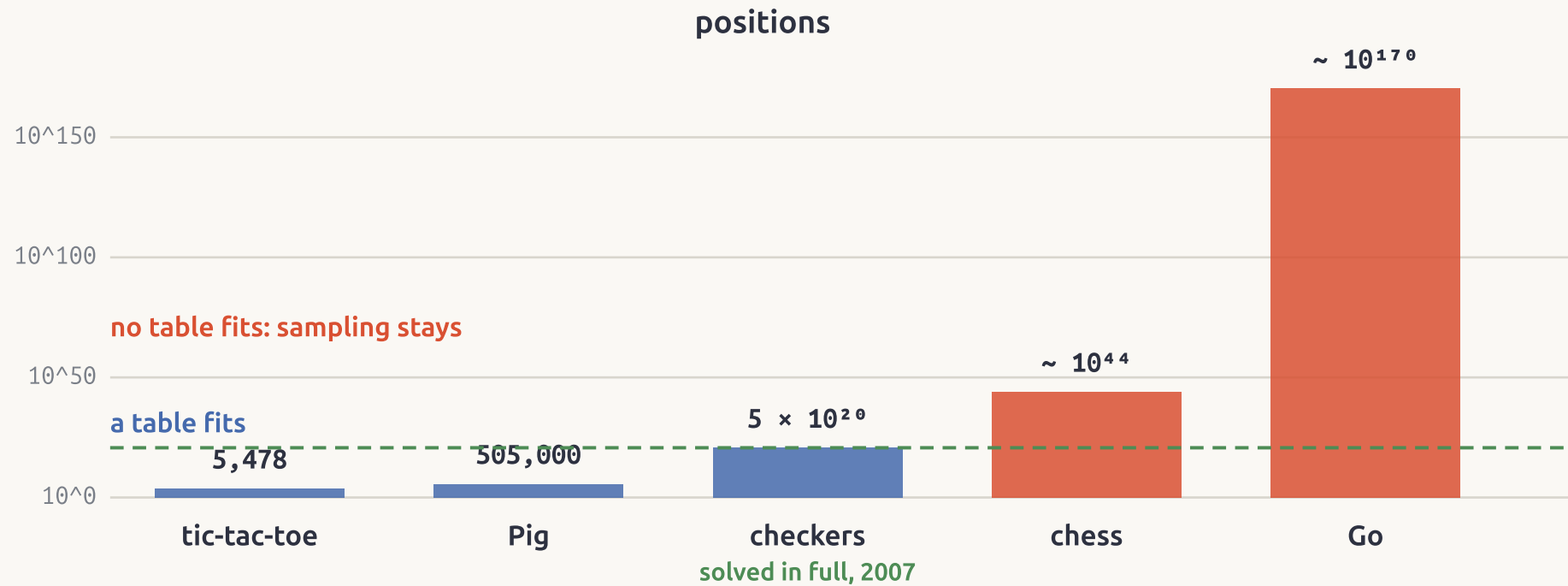


in the task



in the solver

When sampling stays



The diagnostic

after the strongest classical reformulation



does sampling still dominate?



yes

remaining expectation



keep going



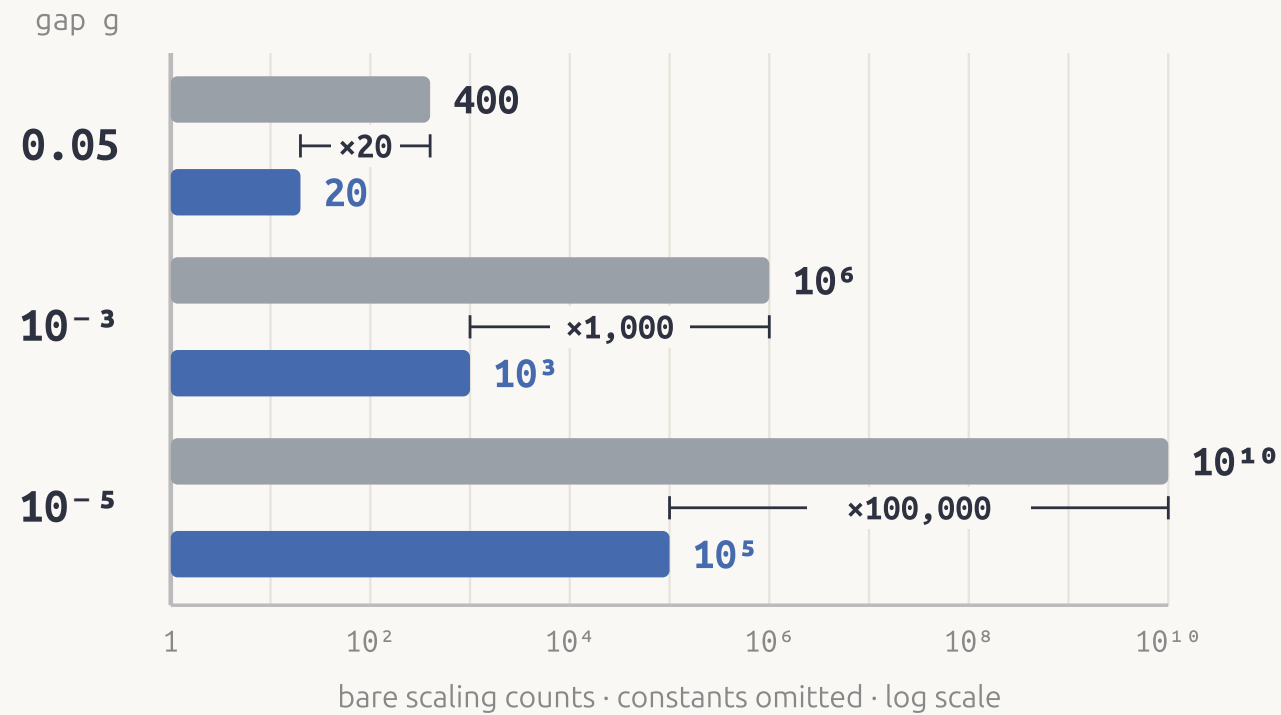
no

classical structure removes it

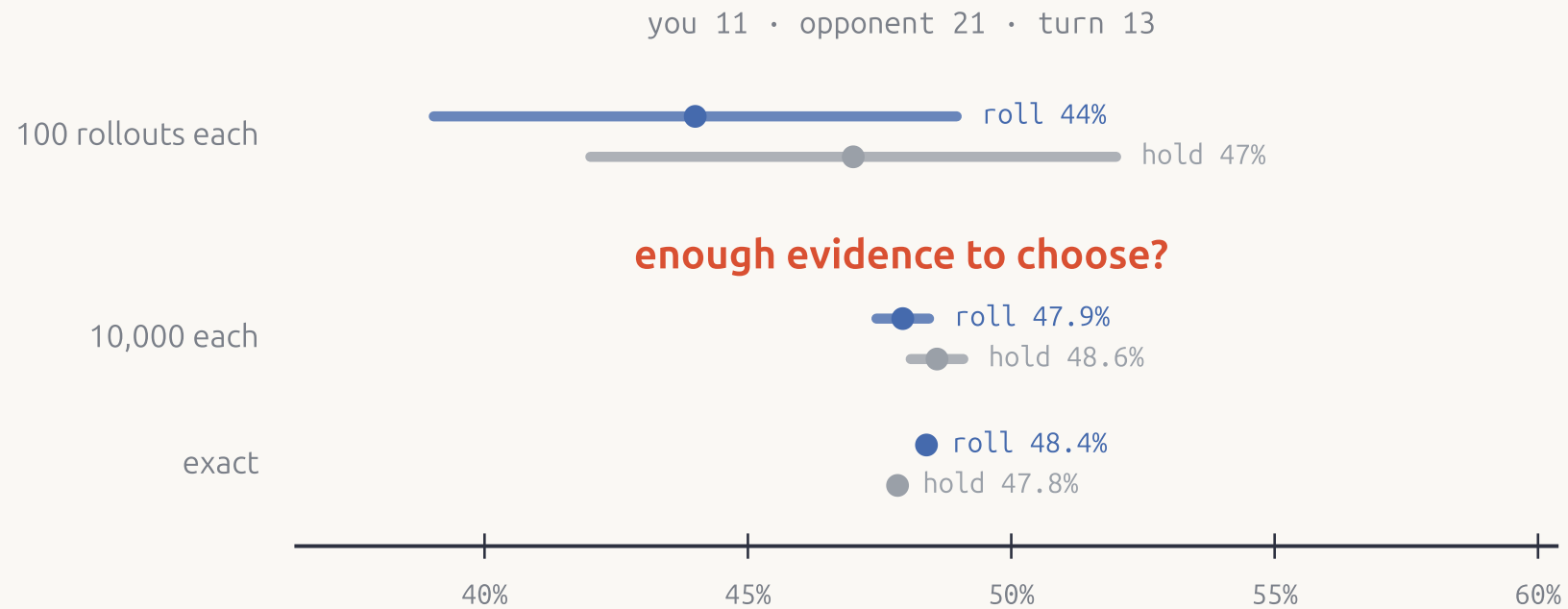


stop here

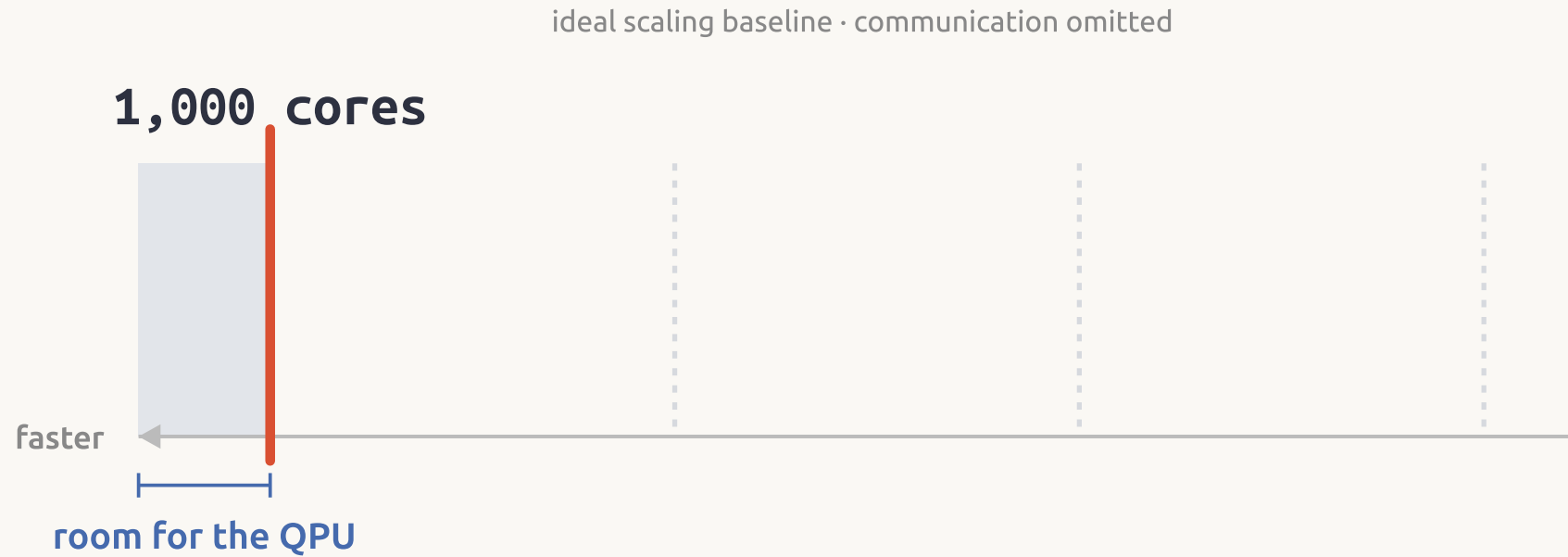
The cost, itemized



Enough evidence to choose?



The classical counterpunch



The second diagnostic

once sampling is the last resort standing



does the gap drive the cost?



yes

tiny gap, samples pile up



keep going



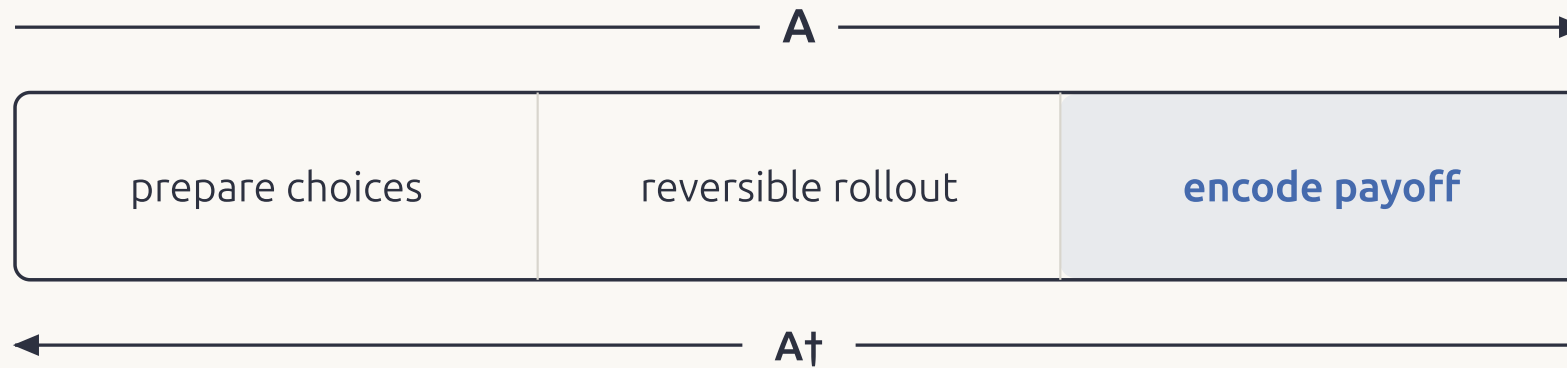
no

wide gap, rollouts stay few

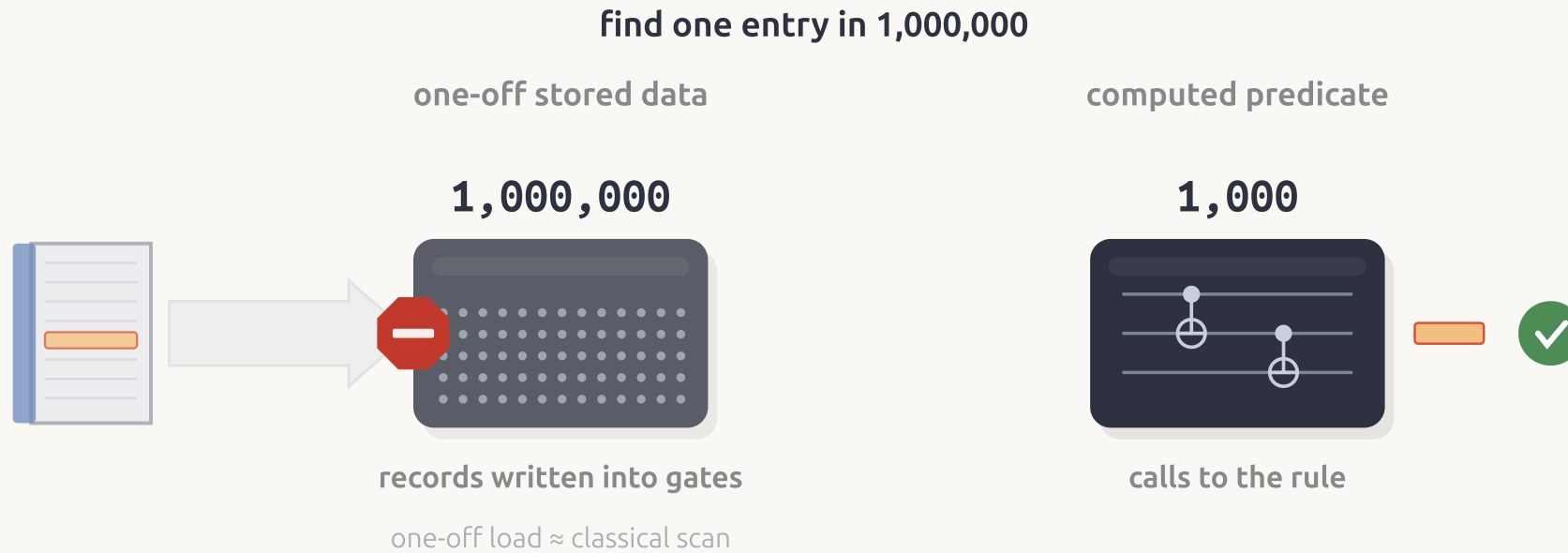


stop here

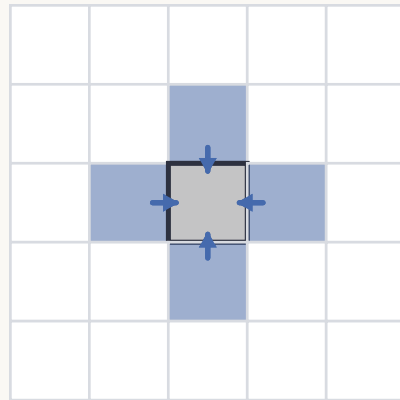
The rollout query



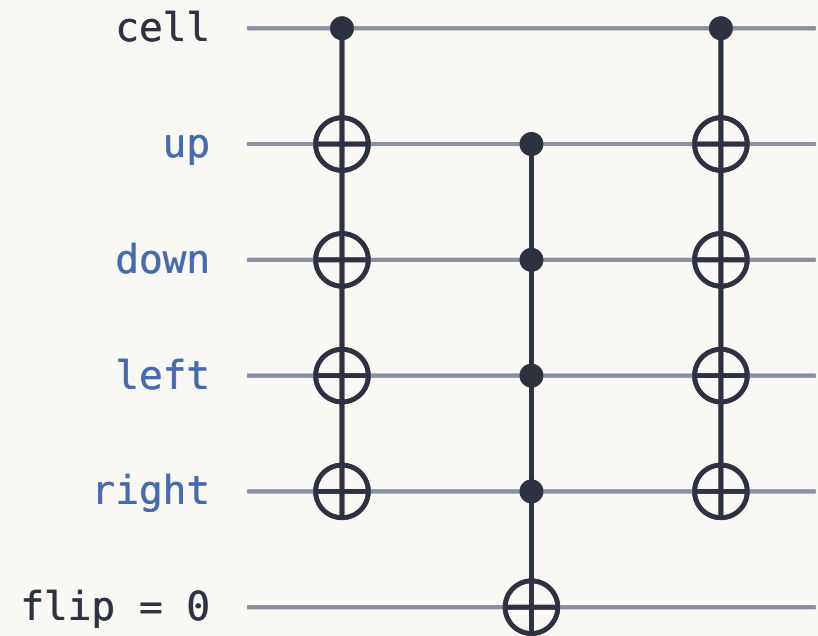
An oracle can be heavy



Rules that compile

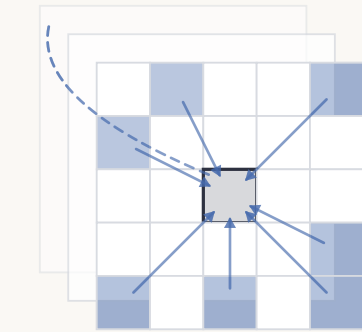


reads the cell and its four neighbours

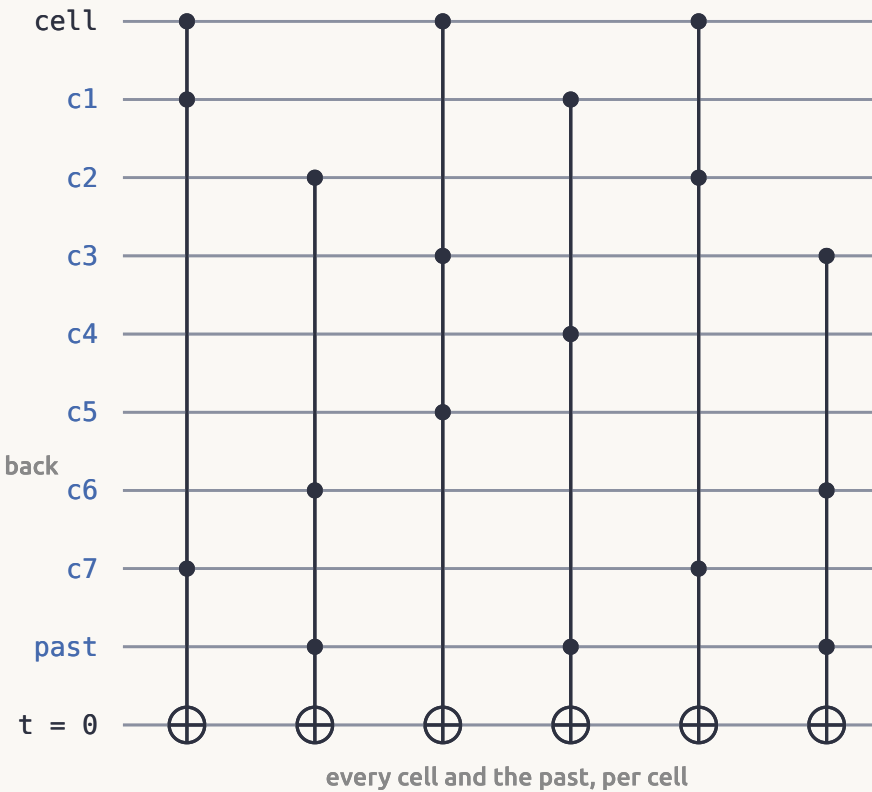


5 reads per cell, at any board size

Rules that tangle



the whole board, and one board back



The third diagnostic

with a tiny gap and sampling to close it



is one coherent query cheap?



yes

local rules, nothing loaded



worth costing out



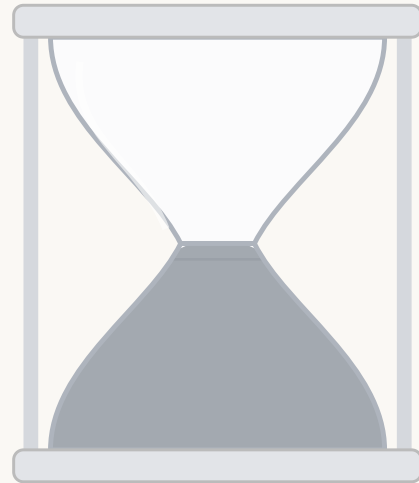
no

tangled rules or loaded data

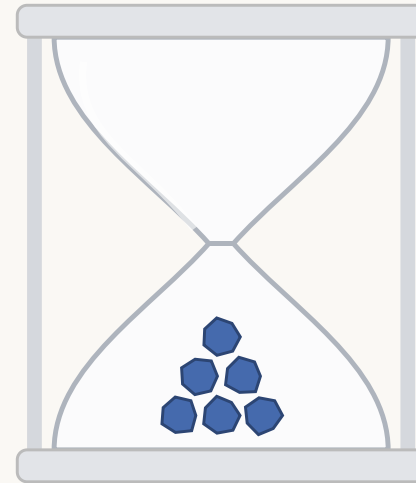


stop here

Winning queries, losing the clock



classical · samples



QPU · one query lane

Which machine finishes first?

$$\underbrace{T_{\text{CPU}} \sim \frac{t_{\text{roll}}}{P} \frac{1}{g^2}}_{\text{ideal } P\text{-way classical baseline}} \quad \underbrace{T_{\text{QPU}} \sim C t_{\text{oracle}} \frac{1}{g}}_{\text{one coherent query lane}}$$

classical



QPU · one lane



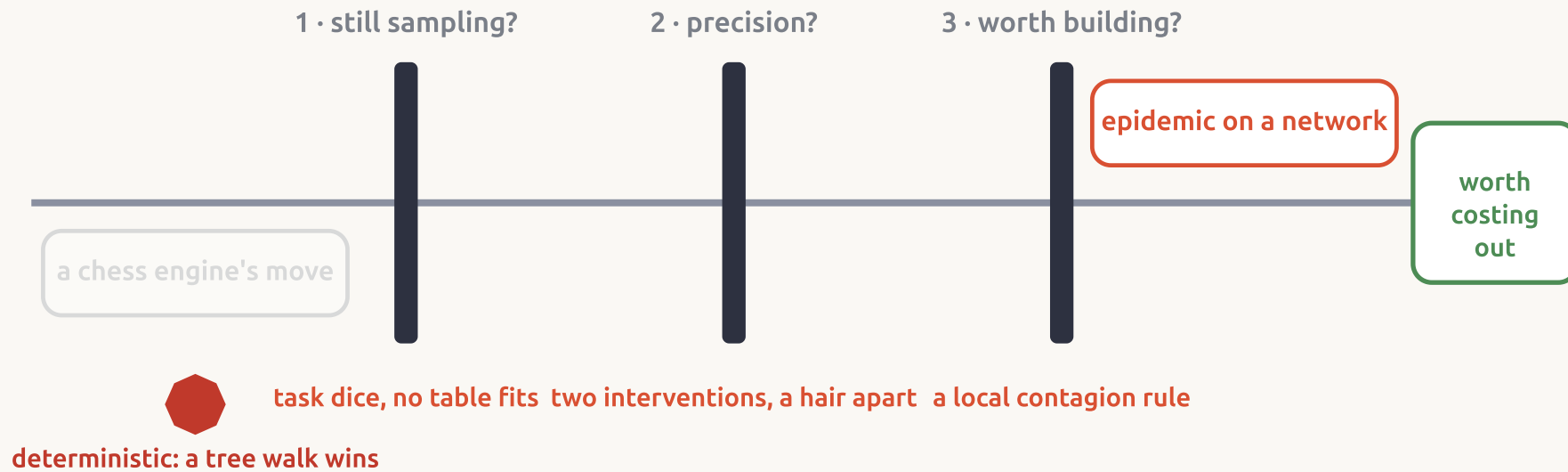
Break-even

$$\underbrace{t_{\text{oracle}}}_{\text{one query}} < \frac{\overbrace{t_{\text{roll}}}_{\text{one rollout}}}{\underbrace{C}_{\text{overhead}} \cdot \underbrace{P}_{\text{cores}} \cdot \underbrace{g}_{\text{gap}}}$$

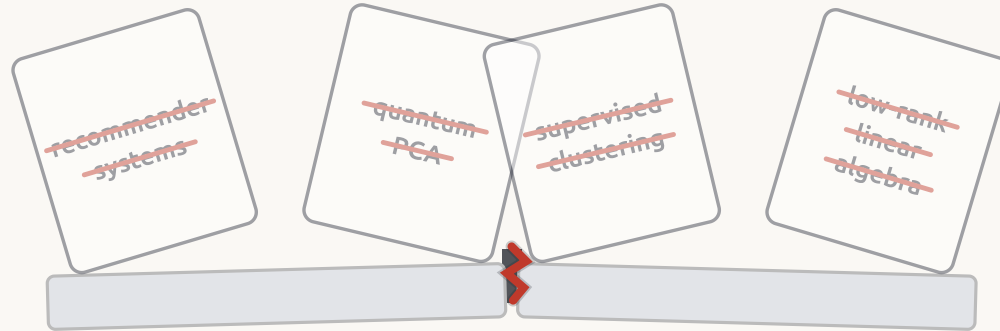
A hundred times faster

$$\underbrace{g = 0.01}_{\text{gap}} \quad \underbrace{P = 1000}_{\text{cores}} \quad \underbrace{C = 10}_{\text{overhead}} \quad \implies \quad t_{\text{oracle}} < \frac{t_{\text{roll}}}{100}$$

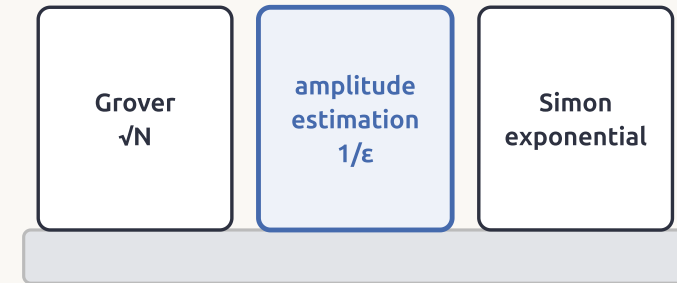
Apply the three questions



The dominoes



2018 · Tang → 2020 · general framework



A different computer

1

choose

the problem

2

prove

the opening

3-7

build

the oracle

8

trust

the pieces

9-11

locate

the quantum

12

judge

the next claim

Continue to Lesson 2

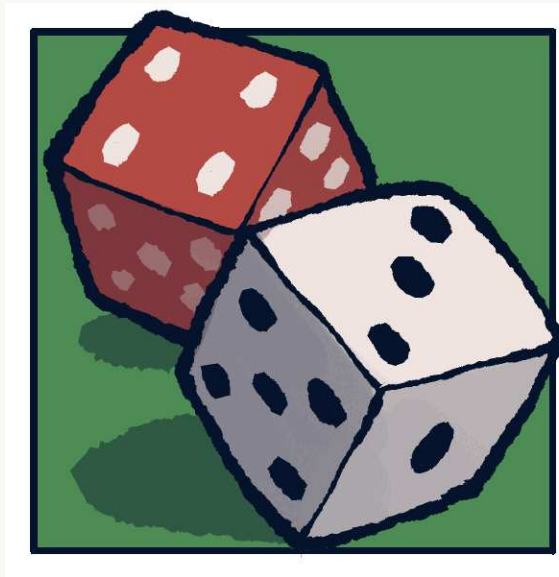


shukla.io/quantum-oracle-engineering

Subscribe on the course page for each new lesson

Quantum Oracle Engineering

Lesson 2: The Monte Carlo speedup

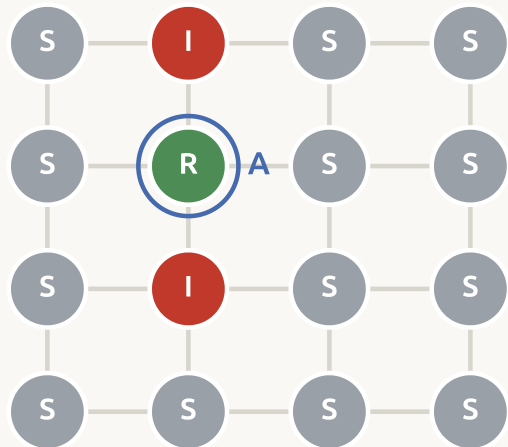


“Monte Carlo gets a quadratic quantum speedup.”

Which intervention should we choose?

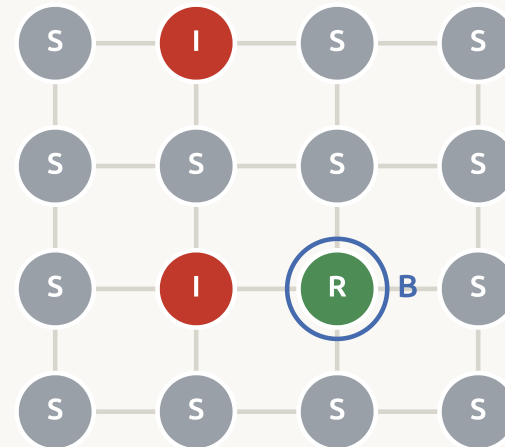
Goal: at most 2 infected after 6 steps.

Vaccinate A first



2 infected

Vaccinate B first



2 infected

S susceptible

I infected

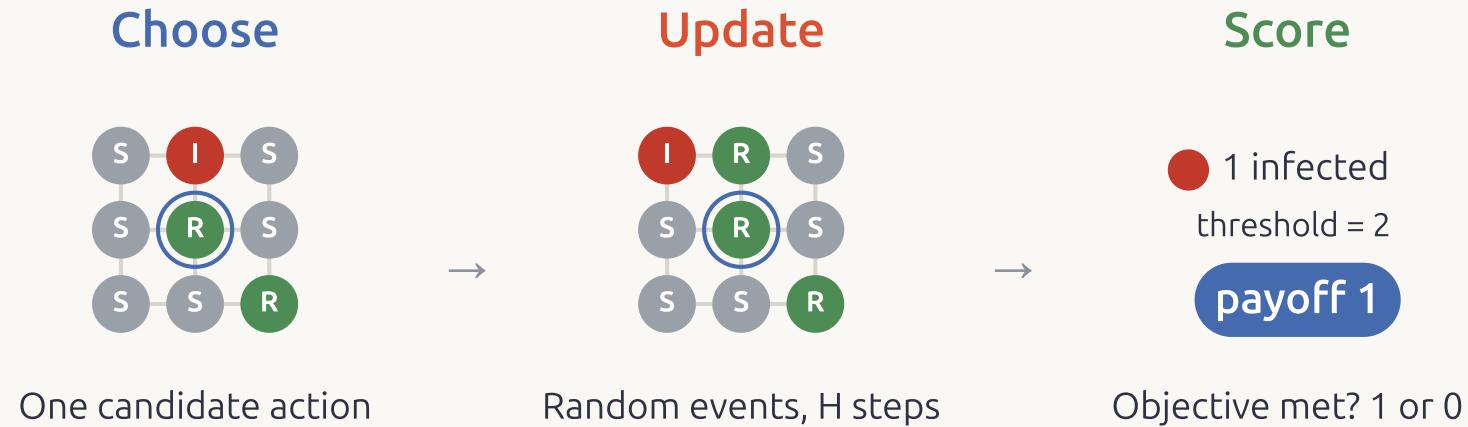
R recovered / vaccinated

Next step

New future

Step 0 / 6

What does the simulator need?



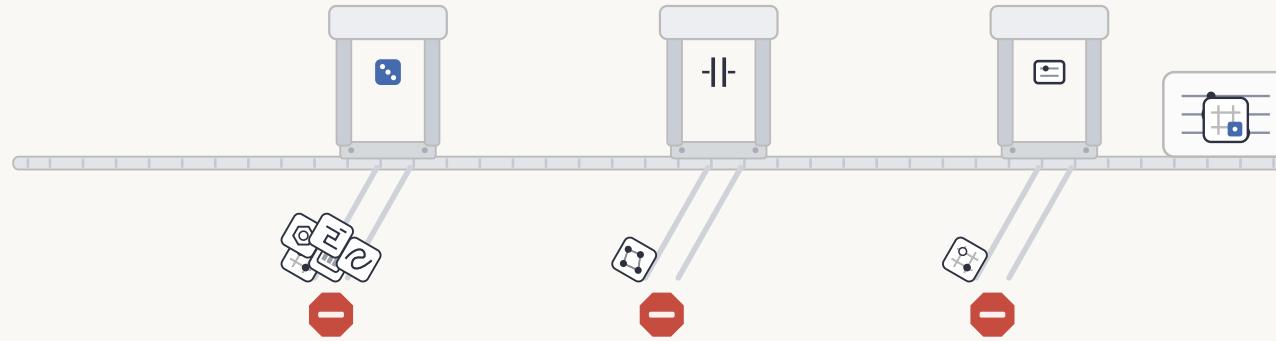
1 0 1 1 0 → average = 3/5

Same starting state + action. A fresh future each time.

[Back](#) 4 / 4 · Repeat and average [Complete](#)

One simulated future is a rollout. Later actions follow a fixed policy.

Three questions, no free passes



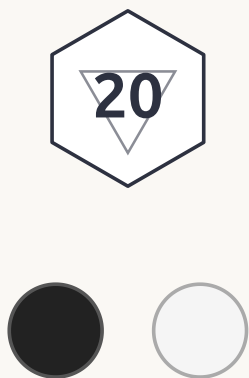
- still sampling?
- precision?
- worth building?

We'll build it first in Sway

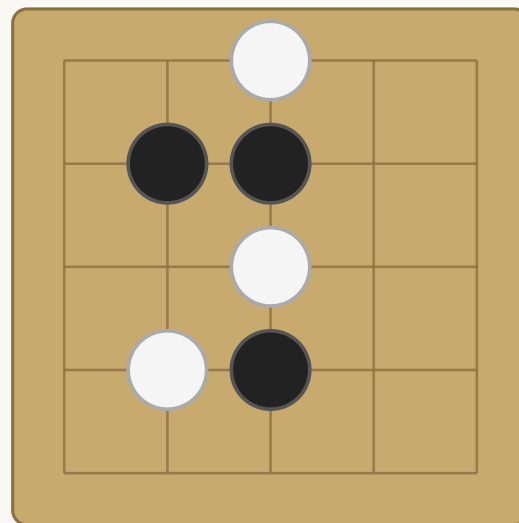


One event and illustrative final counts shown. Repeat updates to H before scoring.

The template for most problems

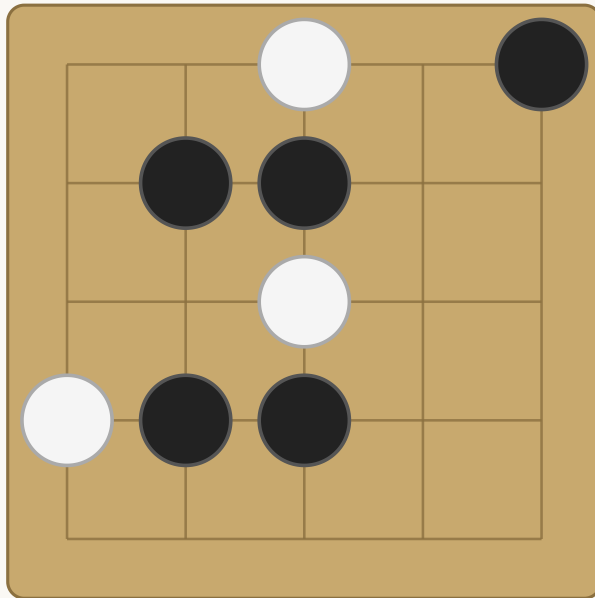


Sway



5x5 • Black to move • 3 rounds left

Black, White, then the board

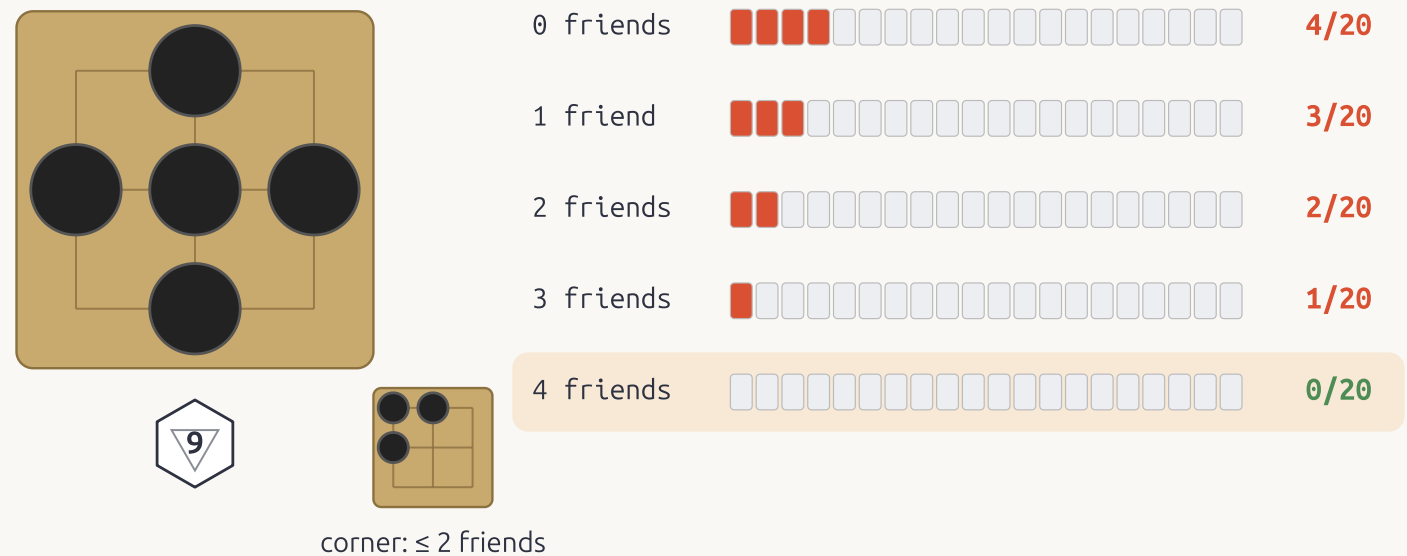


round 2 of 3

- 1 Black places
- 2 White places
- 3 **Sway!**

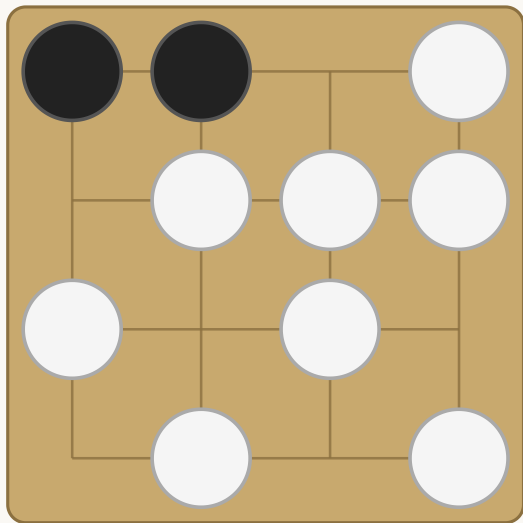
the flips land together; on to round 2

Friends make a stone harder to sway



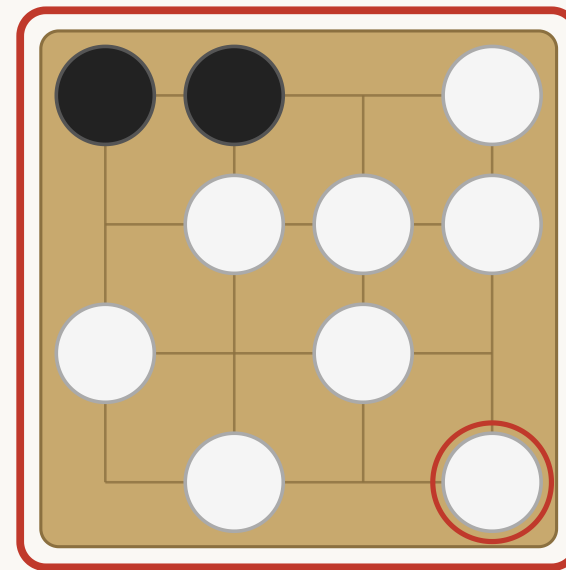
$$\Pr[\text{flip}] = \frac{4 - c}{20}$$

Everyone decides from the old board



synchronous

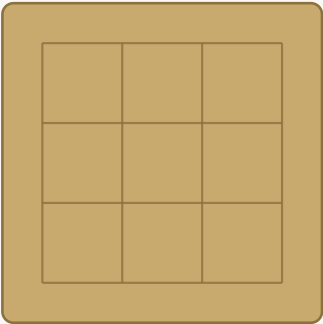
same dice



sequential

How do we score a rollout?

Black ahead



●●●●●●

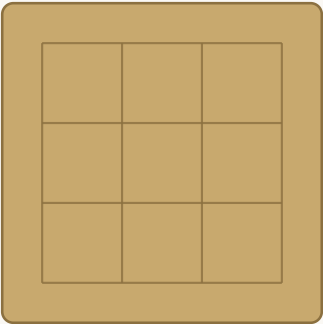
○●○●○●

6

4

payoff 1

White ahead



●●●●

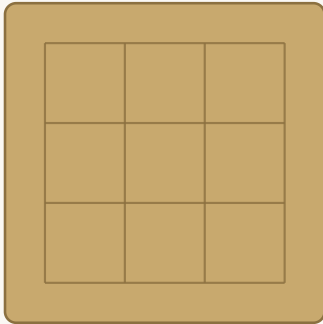
○●○●○●○●

4

6

payoff 0

A tie



●●●●●

○●○●○●○●

5

5

payoff 0

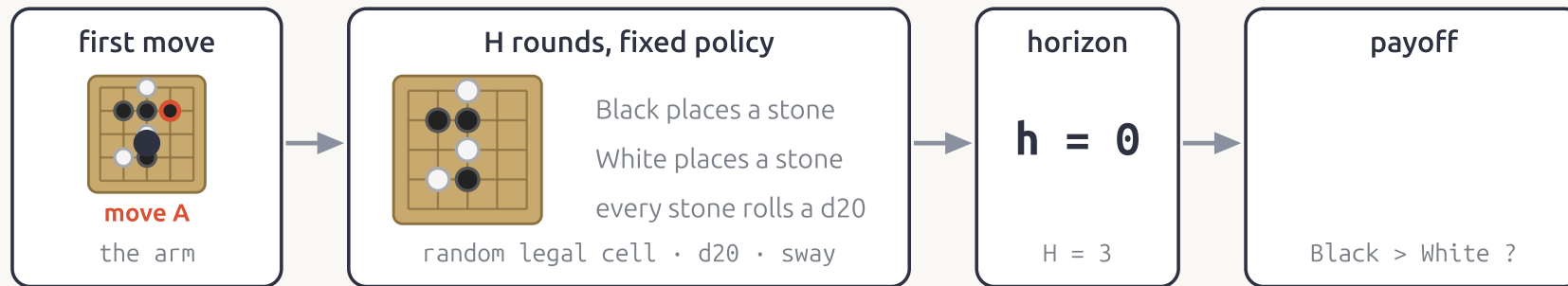
Back

4 / 4 · Reveal the tie

Complete

Our objective: win probability. Winning by 1 or 10 both return 1.

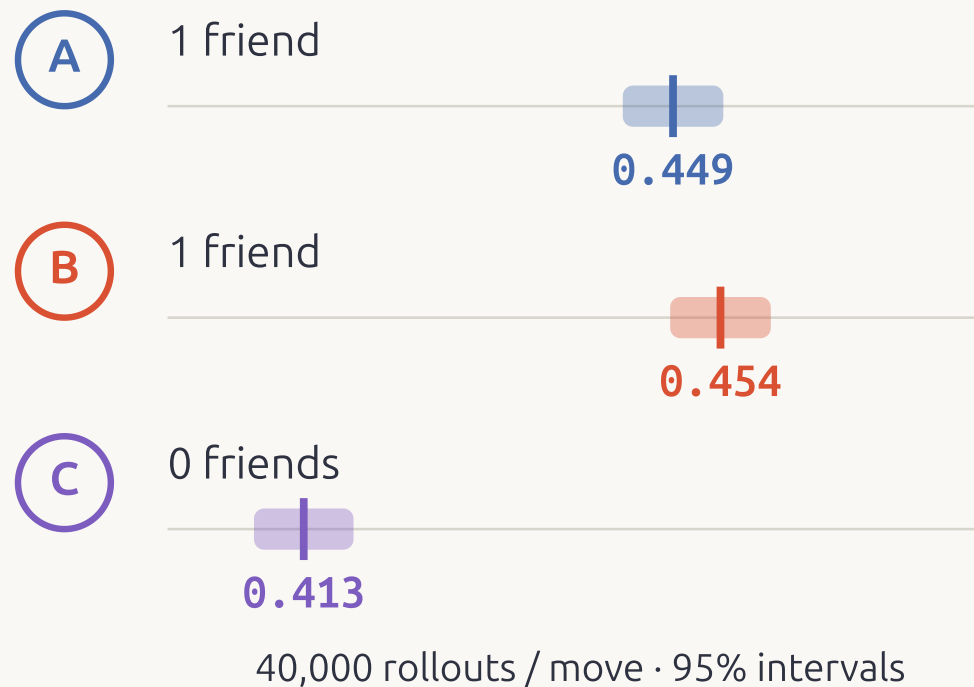
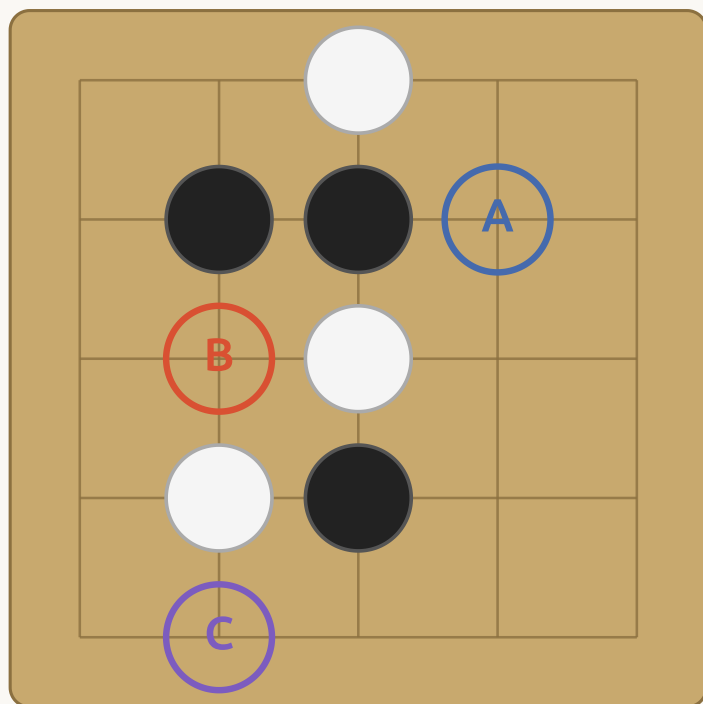
Follow one Sway rollout



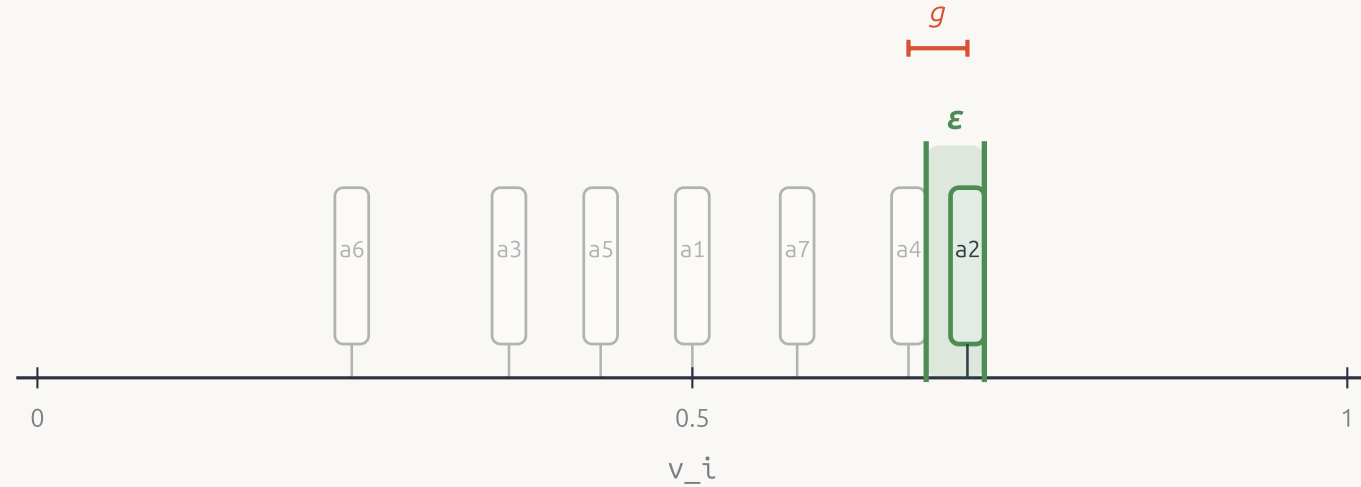
running average: –

$$v_i = \Pr \left[\text{Black outnumber White at } H \mid \text{first move } i, \text{ fixed policy} \right]$$

Where would you play?

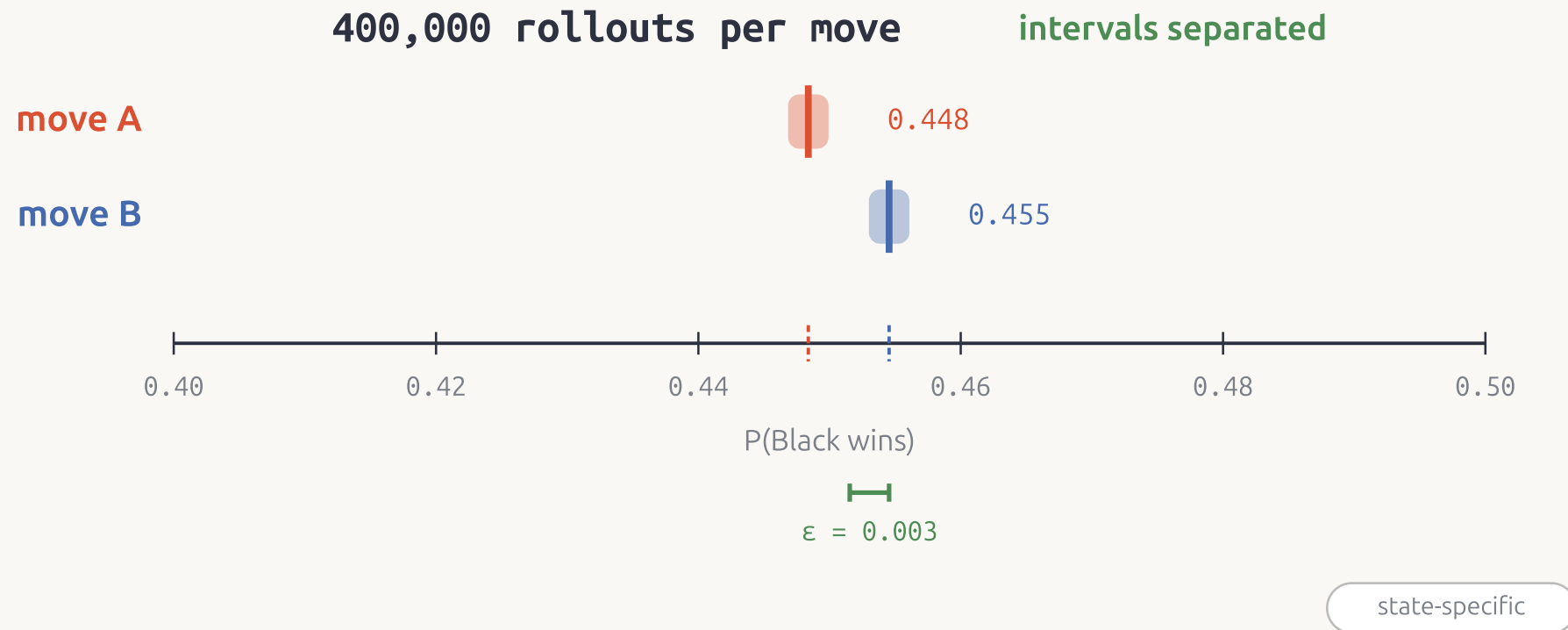


How good does the chosen move need to be?

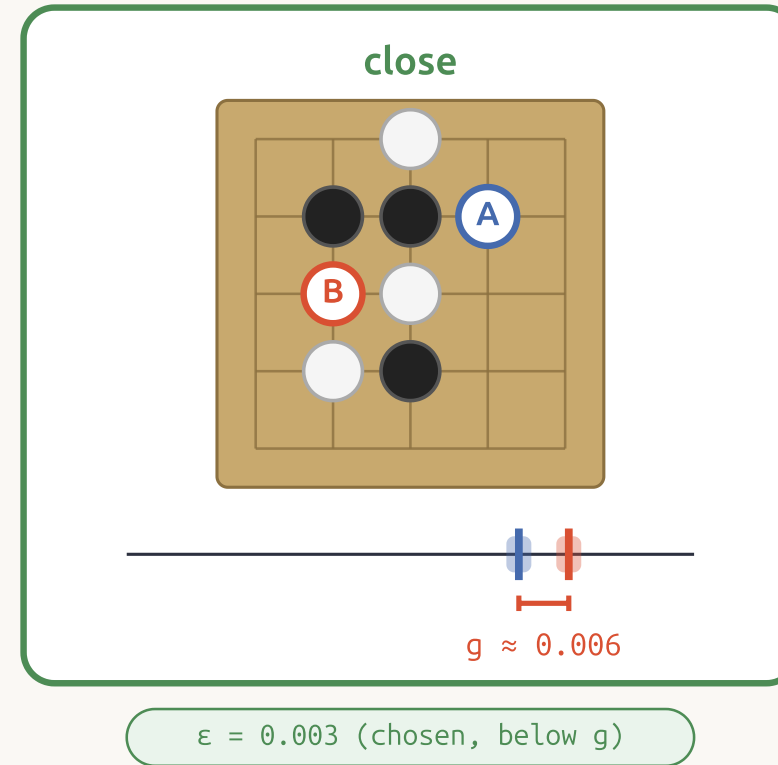
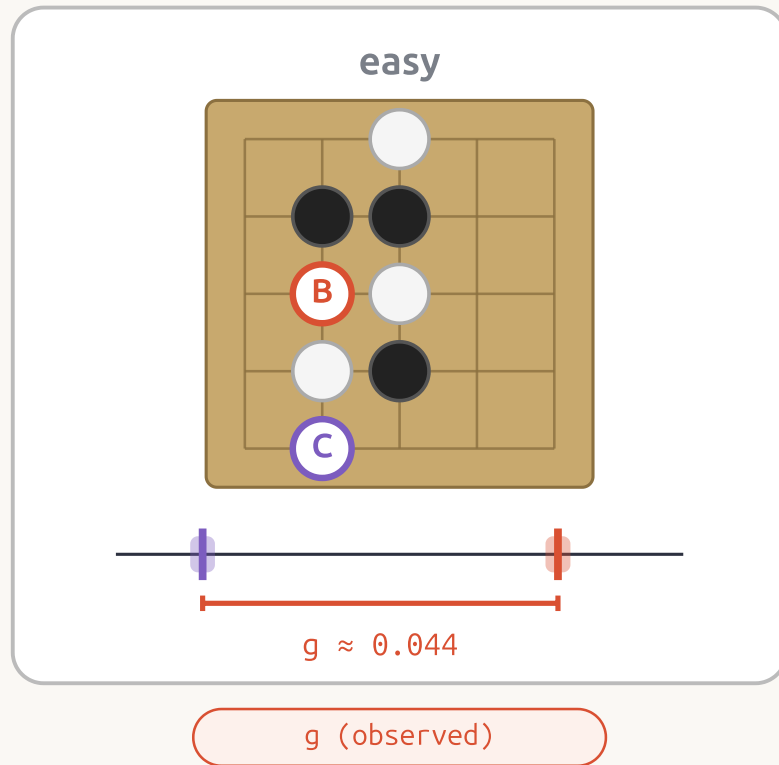


$$\Pr \left[v_{\hat{a}} \geq \max_j v_j - \varepsilon \right] \geq \frac{2}{3}$$

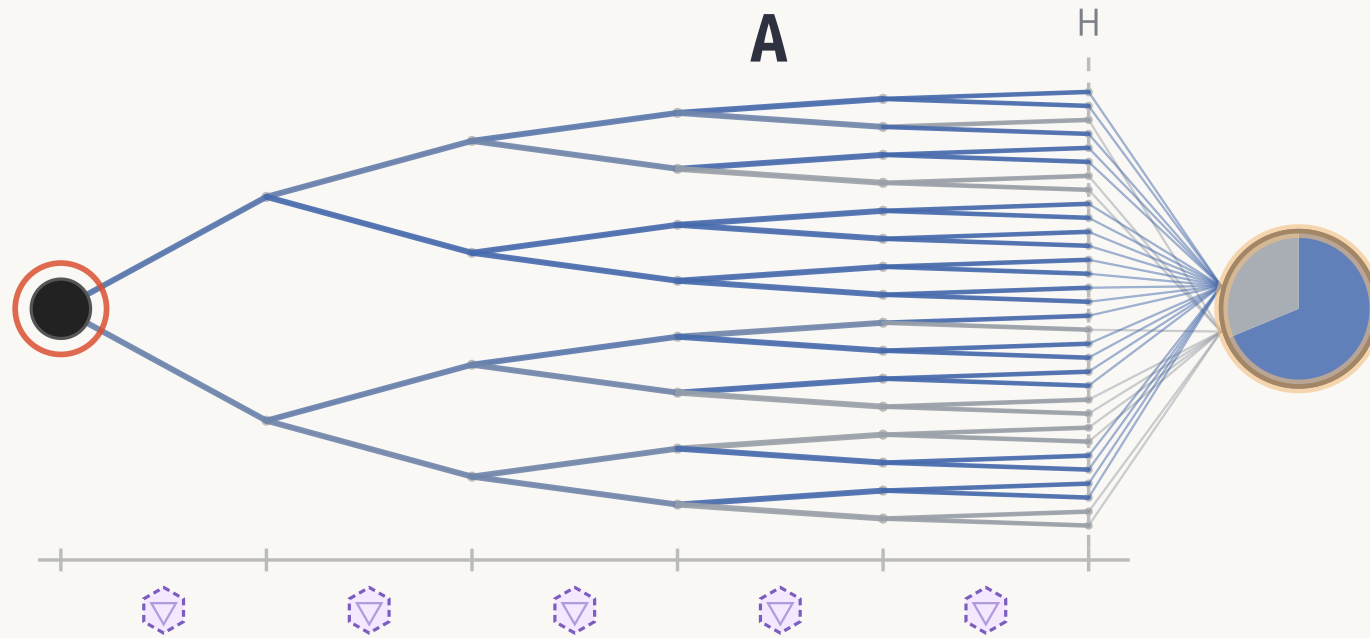
Close calls can be designed and measured



Close moves need more evidence

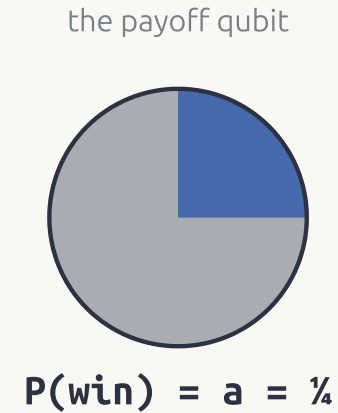
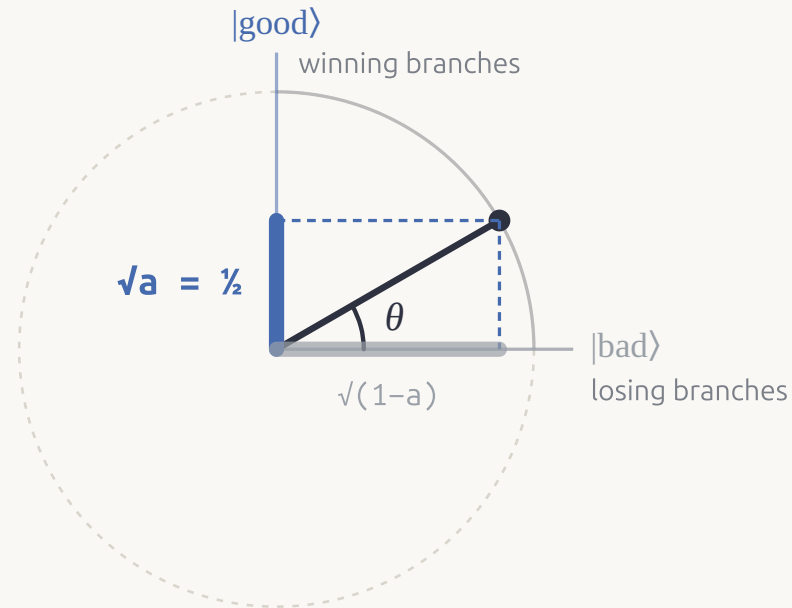


A coherent rollout keeps every branch



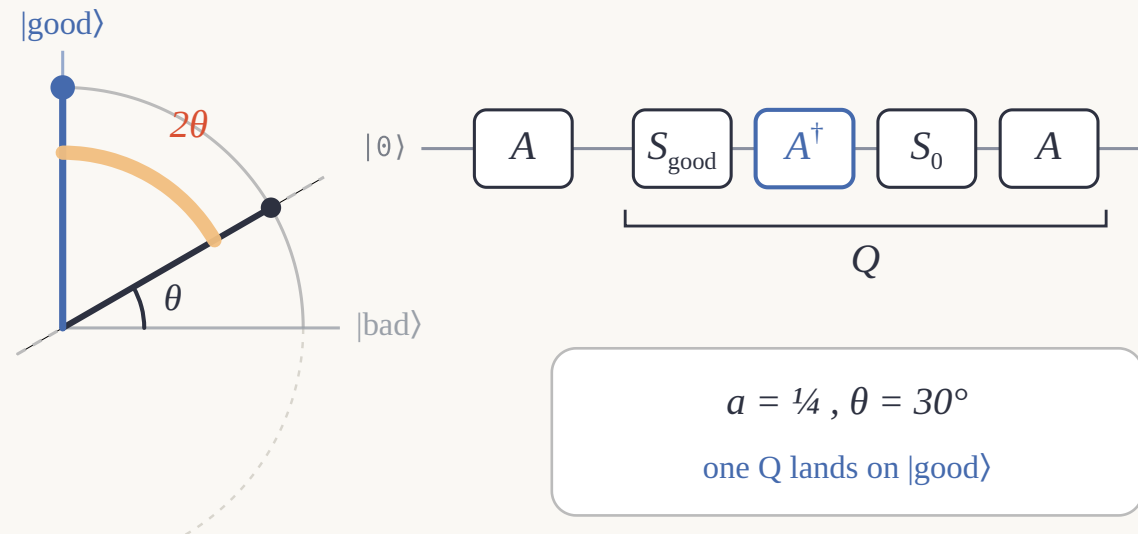
$$A_i |0\rangle = \sqrt{1 - a_i} |\psi_{i,0}\rangle |0\rangle + \sqrt{a_i} |\psi_{i,1}\rangle |1\rangle \quad \text{and } A_i^\dagger \text{ on request}$$

A probability becomes an angle



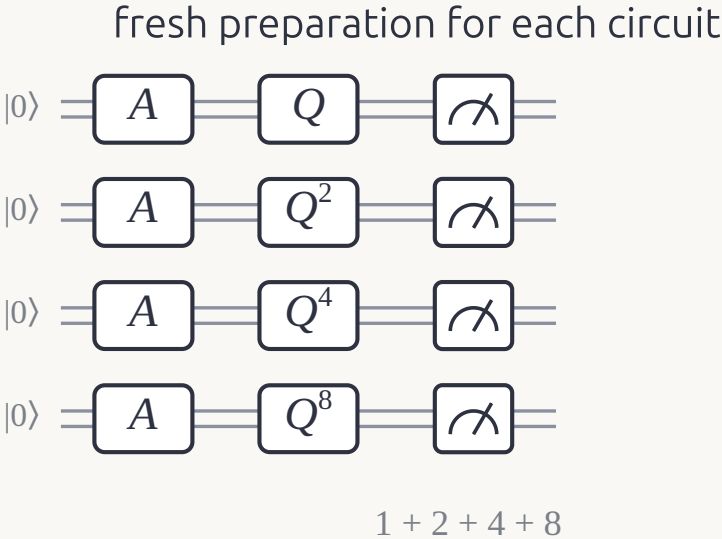
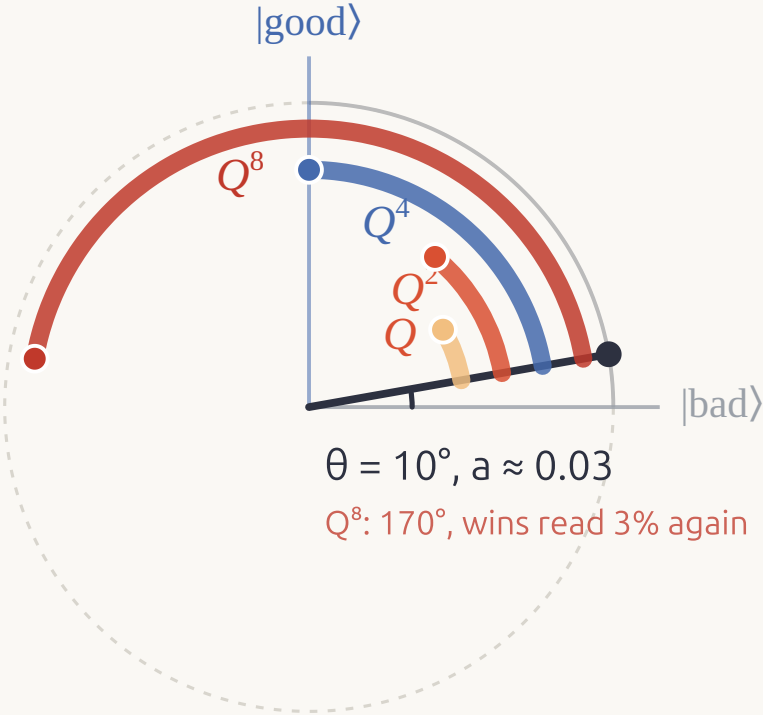
$$a = \sin^2 \theta$$

Two reflections make a rotation


[Back](#)
4 / 4 · Read the rotation: 2θ
[Complete](#)

$$Q = -A S_0 A^\dagger S_{\text{good}} = \text{rotation by } 2\theta$$

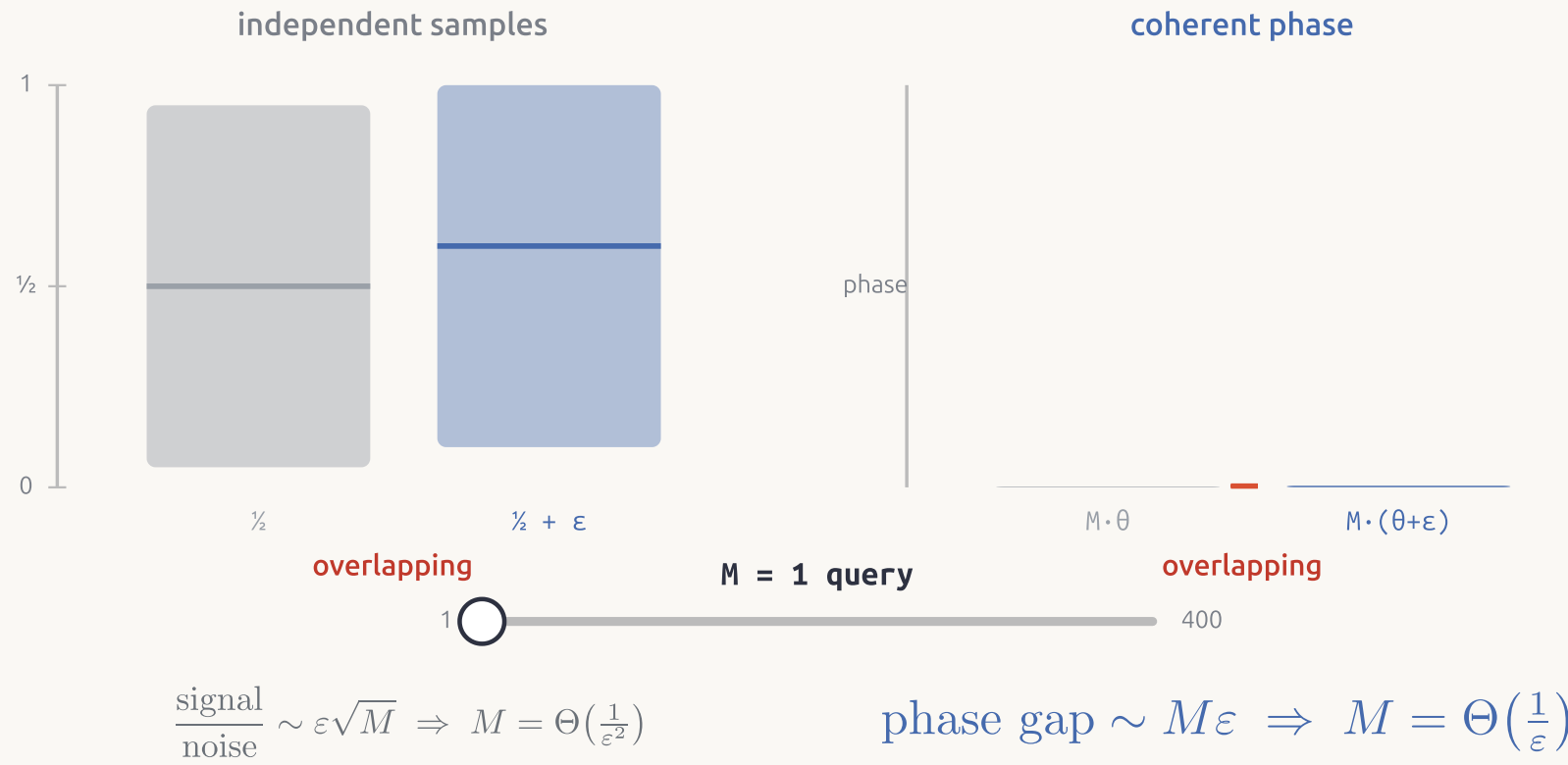
Every turn of Q adds 2θ



15

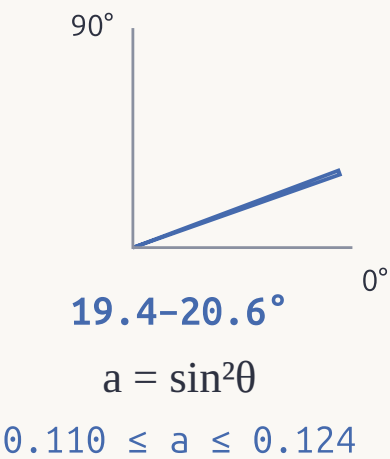
turns of Q · one A and one $A^\dagger$ each

Nearby probabilities become easier to distinguish



Run, measure, refine

Possible original angles



Run

Measure

Narrow

	$\rightarrow$	1 0 0 0 0 0 0 0 60 / 512 ones	
	$\rightarrow$	1 1 1 1 0 1 1 0 384 / 512 ones	
	$\rightarrow$	0 0 1 0 1 0 0 1 212 / 512 ones	

Interval width $< 2\epsilon$: stop here for $\epsilon = 0.01$.

Back

9 / 9 · Narrow the interval

Complete

Fresh preparation for every shot. Measure only after the coherent sequence.

Illustrative batches · 512 shots each · [Iterative Quantum Amplitude Estimation](#)

The access contract

ACCESS CONTRACT

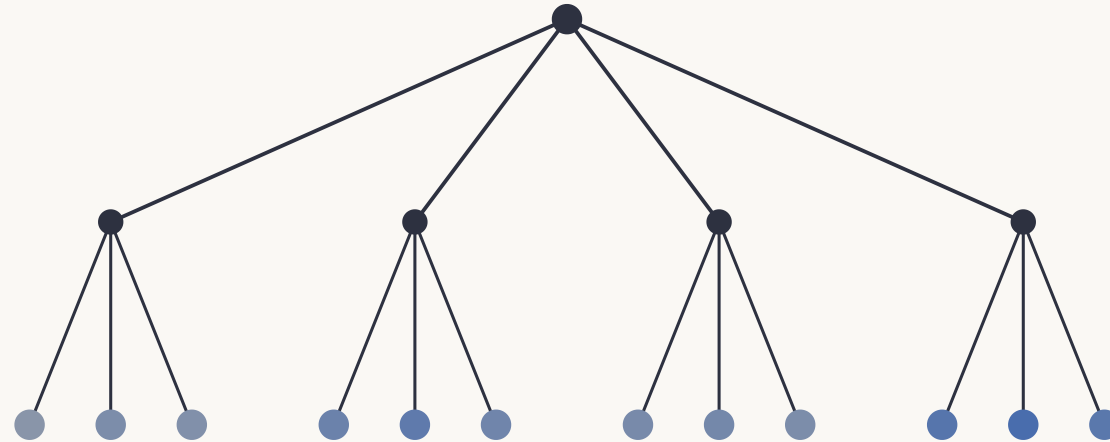
- ✓ bounded payoff
- ✓ coherent A
- ✓ inverse $A^\dagger$
- ✓ no measurement, no fresh randomness inside

ordinary sampling endpoint



- ✓ bounded payoff
- ✗ coherent A
- ✗ inverse $A^\dagger$
- ✗ measures, resamples

Go: compare against the best classical method



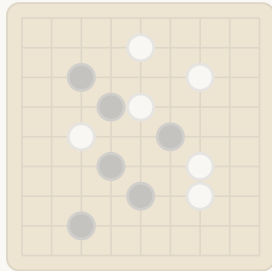
value network

positions are scored; nothing is rolled

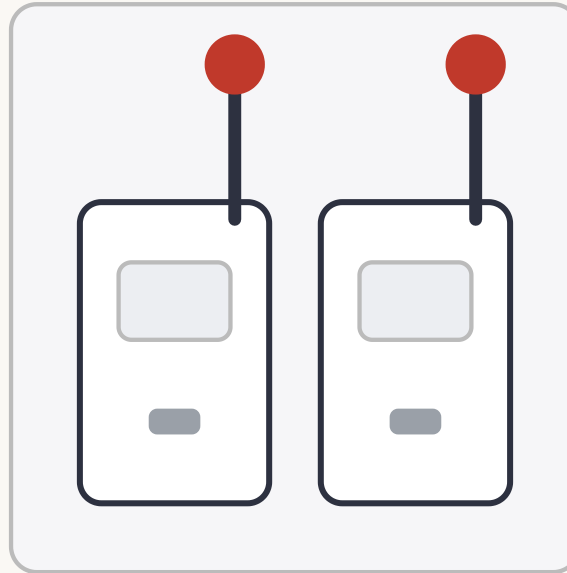
the tree never changes

Check the access and count the work

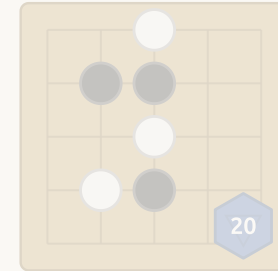
Two arms with nearly equal rewards



Go



two-arm bandit



close-calls game

The sampled bandit lacks an inverse

ACCESS CONTRACT

- ✓ bounded payoff
- ✓ coherent A
- ✓ inverse $A^\dagger$
- ✓ no measurement, no fresh randomness inside

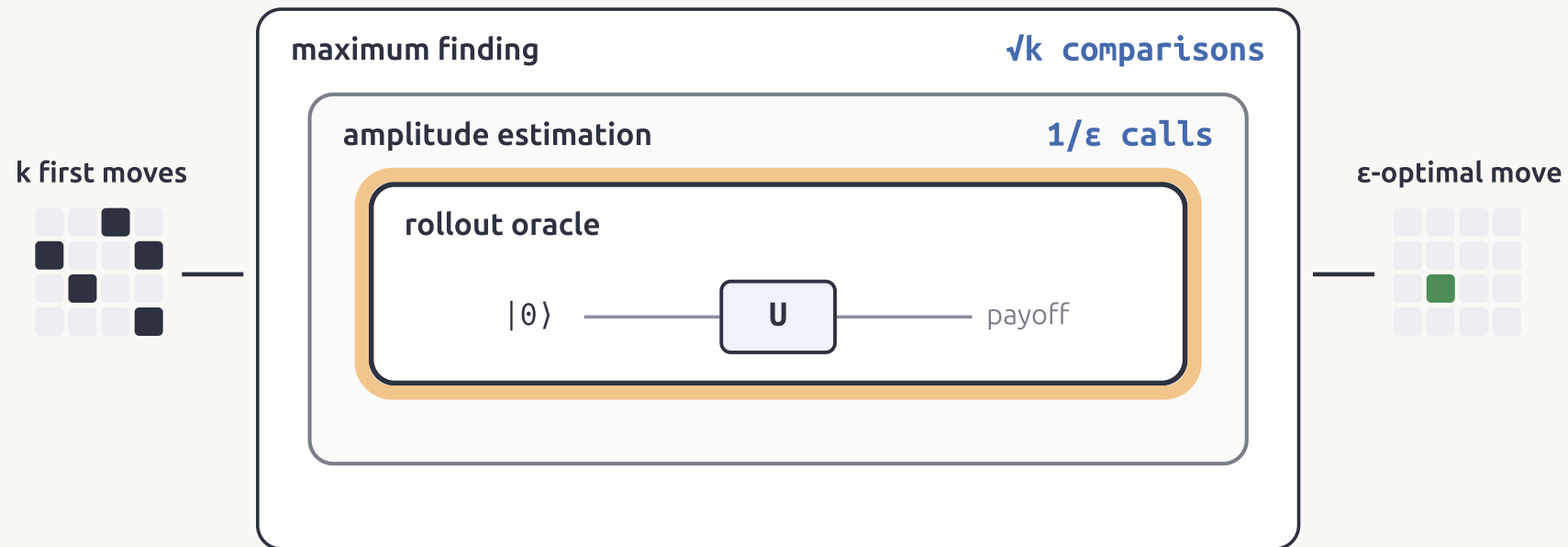
MISSING $A^\dagger$

ordinary sampling endpoint

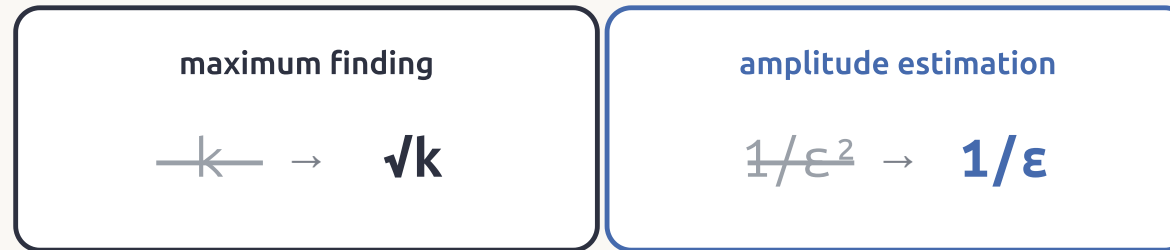


- ✓ bounded payoff
- ✗ coherent A
- ✗ inverse $A^\dagger$
- ✗ measures, resamples

Choosing among k actions



The quantum composition



$$\tilde{O}(\sqrt{k} / \epsilon)$$

$$\Omega\left(\frac{k}{\epsilon^2}\right) \text{ rollout pulls} \quad \text{versus} \quad \tilde{O}\left(\frac{\sqrt{k}}{\epsilon}\right) \text{ calls to } A, A^\dagger$$

Check the classical alternatives

Does sampling remain expensive after the classical shortcuts?

SMALL STATE SPACE

Enumerate outcomes or reuse repeated states. Check sampled answers against exact values.

LARGER STATE SPACE

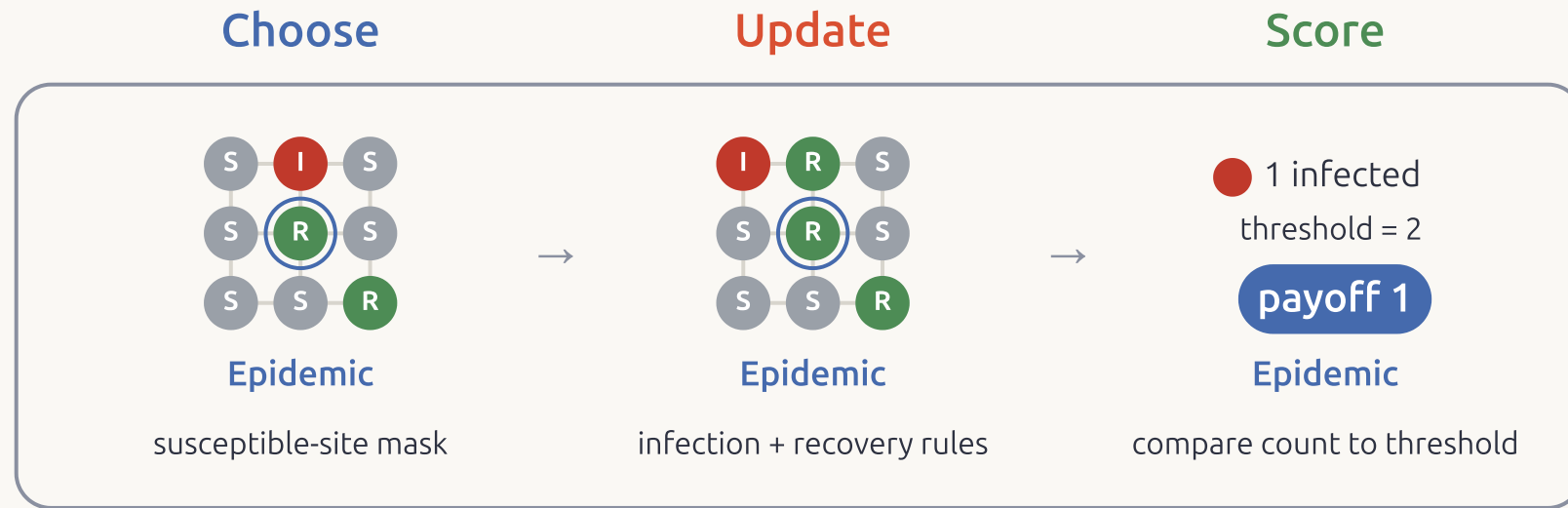
Try structure, approximations, and ways to reduce variance. Count their cost and error.

REMAINING UNCERTAINTY

Measure whether sampling still dominates at the precision the decision needs.

Randomness and a large tree alone don't establish a sampling lower bound.
The appendix gives a bound for a specific access model and hard family.

What carries over to the epidemic model?



Shared reversible circuit

randomness registers · reversible updates · cleared scratch

[Back](#)

4 / 4 · Swap the score

[Complete](#)

The structure carries over. State encoding and gate costs depend on the model.

[Course paper · two implementations](#)

Reuse the construction

Two models, checked

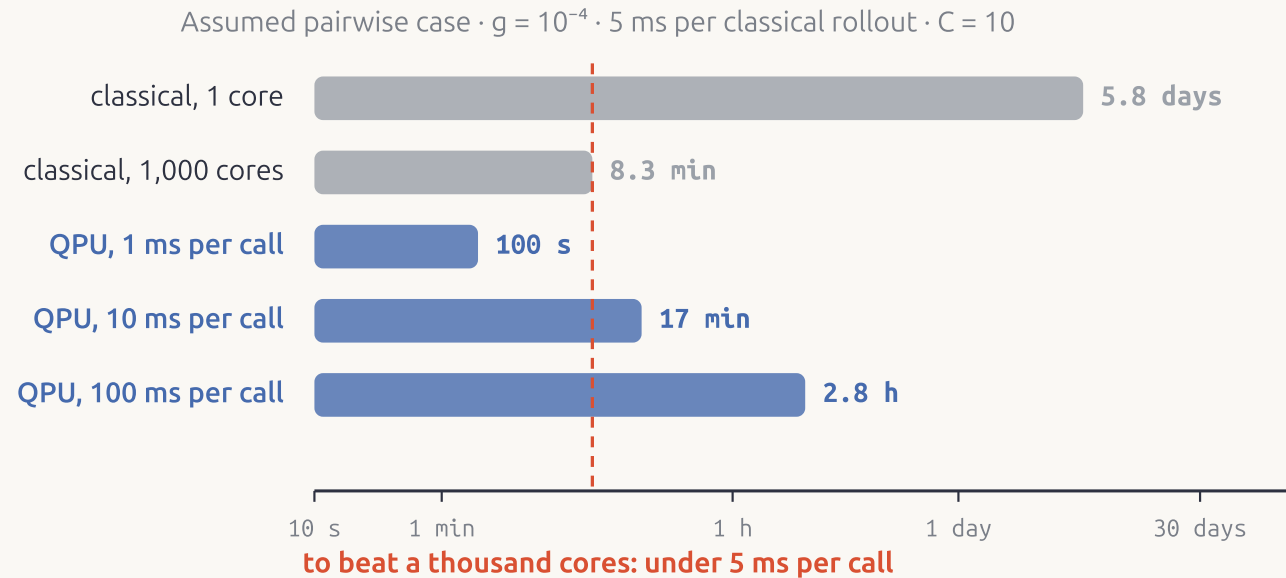
Two small oracle implementations have been checked.

3×3 · 2 rounds	Sway	Epidemic model
Qubits	169	146
Gates	9,768	6,472
Sampled payoff	0.281 ± 0.028	0.883 ± 0.020
Exact payoff	0.271	0.891

Tested branches agree with the classical simulations. Aggregate estimates agree with exact enumeration.

[Course paper, Table I](#) · published validation instances, separate from the live illustration · gate counts before decomposition and error-correction overhead

What would one close comparison cost?



Assumed timings · fixed pair of actions · full k-action selection adds search cost

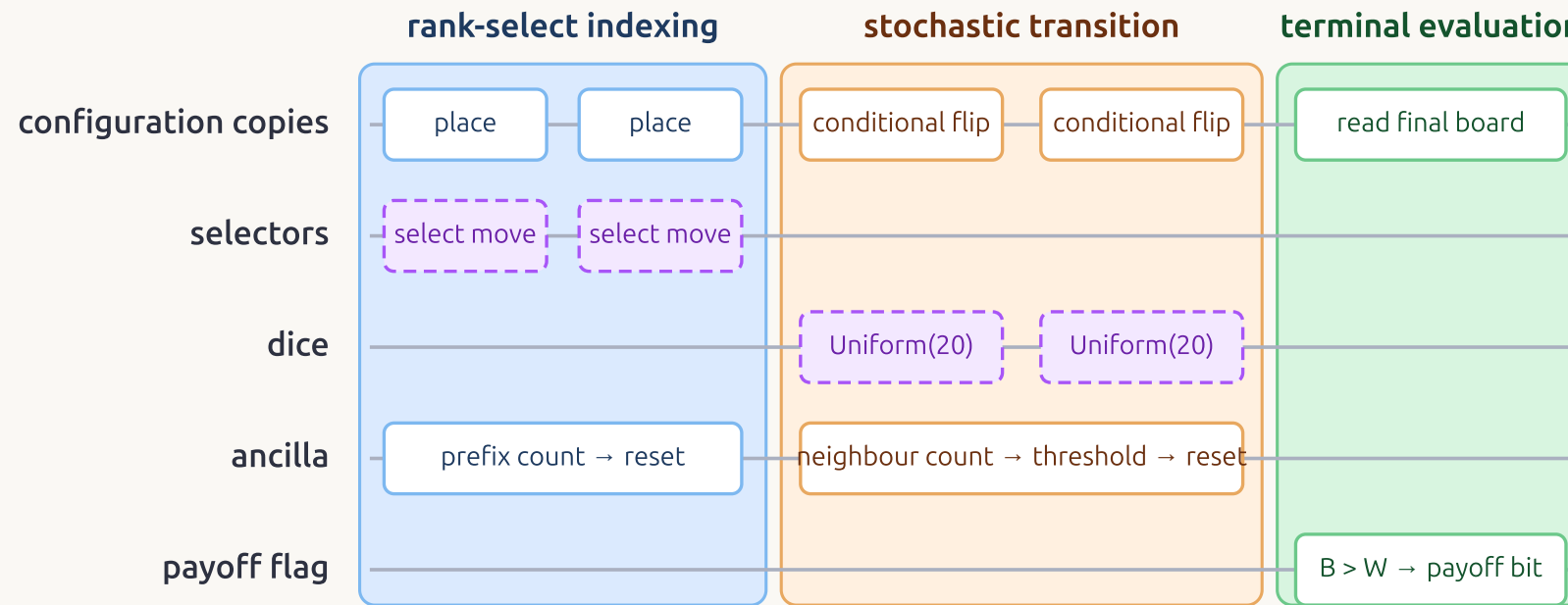
$$t_{\text{oracle}} < \frac{t_{\text{roll}}}{C \cdot P \cdot g}$$

What we can reuse

Example	What we learned	Next step
Go	Rollouts are one classical approach.	Compare against stronger classical methods.
Sampled bandit	Close arms need many samples; the interface lacks $A^\dagger$.	Coherent access would need a different implementation.
Sway + epidemic	The shared oracle structure has two implementations.	Build the components and count their costs.

We have a reusable construction to study. A runtime advantage still depends on the instance and hardware costs.

Zoom into one query



Continue to Lesson 3

Next: build one rollout circuit.

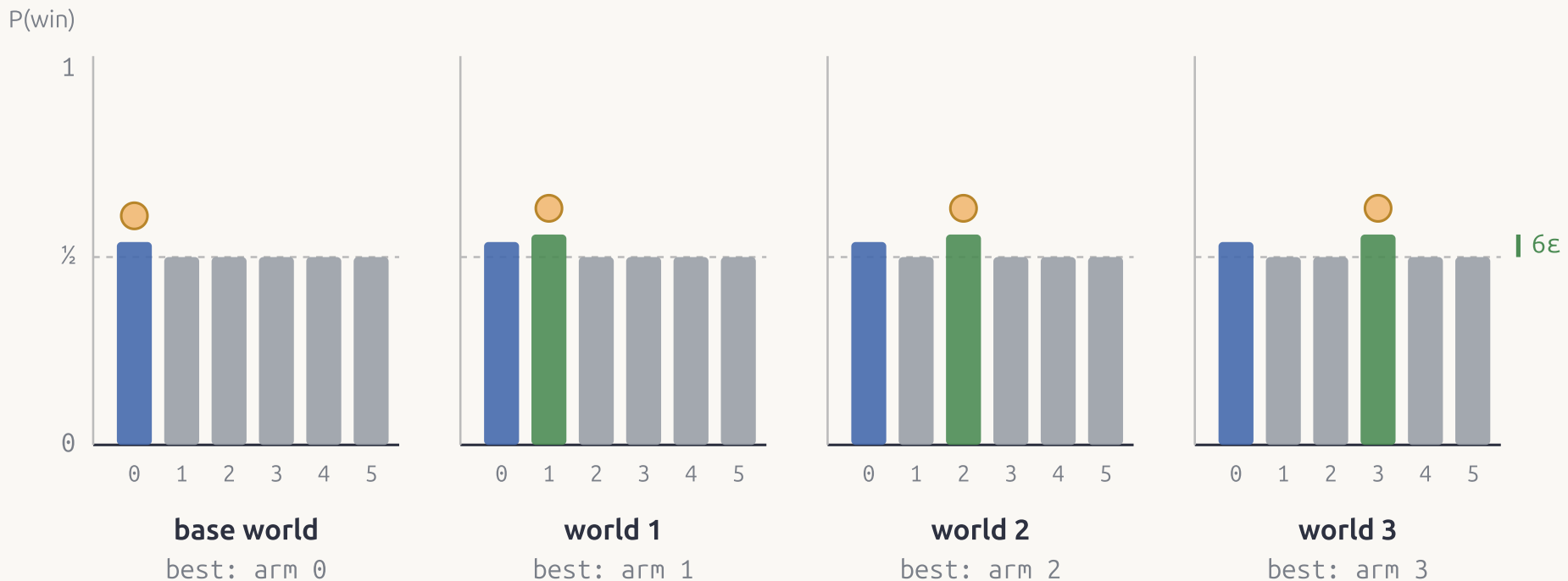
We'll implement the board, legal-move selection, random choices, and payoff in Lesson 3.

[Continue to Lesson 3](#)

[Optional proof and readout details](#)

[Course paper](#) · [Oracle implementation and proofs](#)

A family of almost-identical worlds



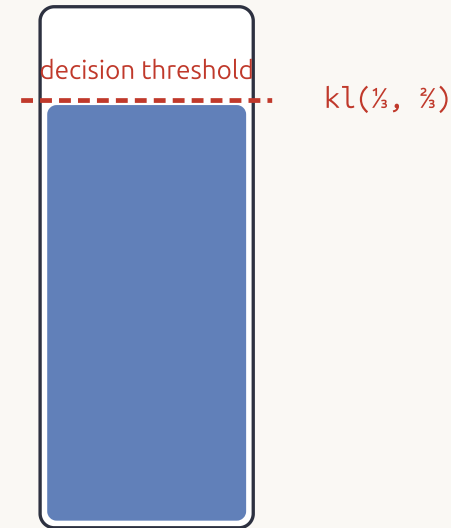
Optional: the sampling lower bound

One pull reveals only ε^2 of evidence



pulls: 24

$\approx 1/\varepsilon^2$ of them



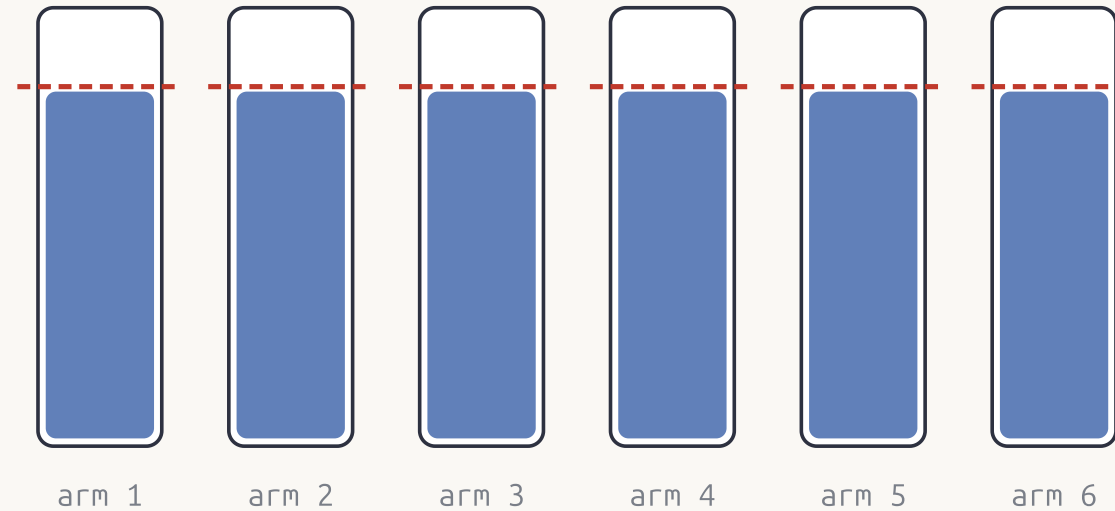
+ $\theta(\varepsilon^2)$ per pull

$$kl\left(\frac{1}{2}, \frac{1}{2} + 6\varepsilon\right) \leq 96\varepsilon^2$$

$$kl\left(\frac{1}{3}, \frac{2}{3}\right) = \frac{\ln 2}{3}$$

$$\mathbb{E}[N_j] \geq \frac{\ln 2}{288\varepsilon^2}$$

Every contender must be ruled out

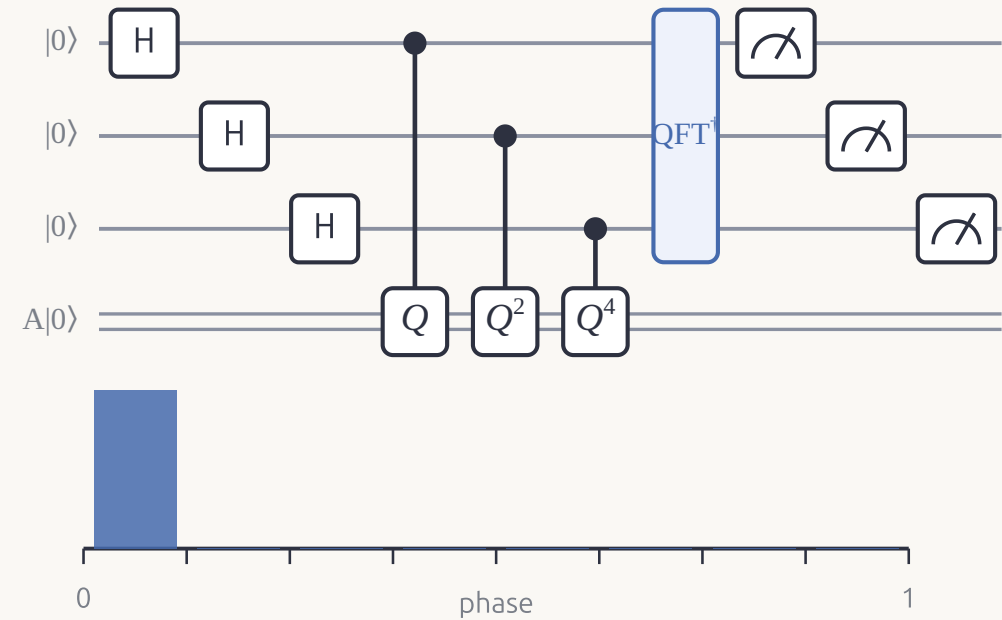
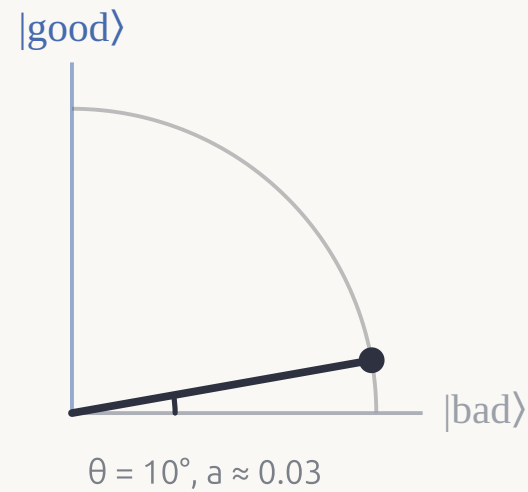


$$(k-1) \cdot \Omega(1/\epsilon^2) = \Omega(k/\epsilon^2)$$

What the theorem does, and does not, say

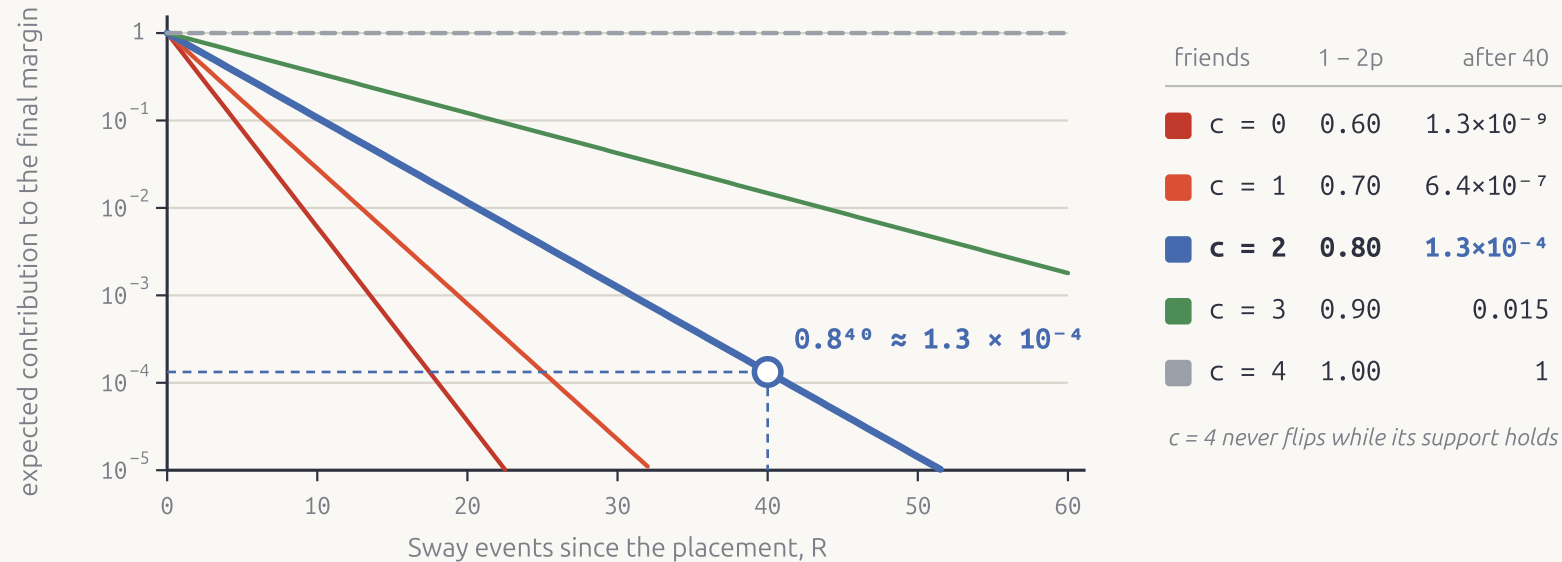
PROVEN	$\tilde{O}(\sqrt{k}/\epsilon)$ upper bound · $\Omega(k/\epsilon^2)$ lower bound on the hard family
PROVEN, ROBUSTNESS	the hard instance recurs across an exponential family (bounded influence)
ASSUMED	coherent access: $A, A^\dagger$, no measurement inside
NOT INCLUDED	unrestricted classical structure · FT overhead · wall clock

The Fourier-based readout



read θ , then $a = \sin^2\theta$

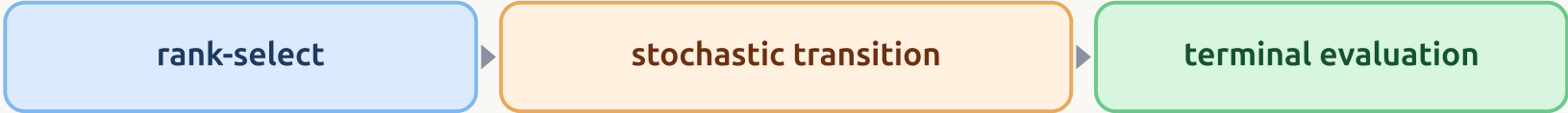
A simple model of fading influence



Fixed-support model: expected color margin after 40 events. Win-rate gaps need separate measurement.

$$\mathbb{E}[\text{contribution after } R \text{ events}] = (1 - 2p)^R$$

Larger compiled circuits



instance	Sway		epidemic (SIR)	
	qubits	gates	qubits	gates
validated				
3×3, H=2	169	9,768	146	6,472
5×5, H=3	667	55,597	—	—
compiled only				
5×5, H=5	916	76,720	767	56,602
10×10, H=5	3,363	481,201	2,893	280,230
10×10, H=10	6,503	793,901	5,463	558,995
20×20, H=10	25,189	6,072,641	21,409	2,592,183

pre-decomposition, before FT overhead

[Back to Lesson 3 and course links](#)