Entropy Accounting for Quantum Circuit Compilation

How many qubits a question needs, and how much a run reveals

20 November 2026

General reader summary

A physicist wants to study how a row of 16 tiny magnets affect one another. Her program needs 64 qubits, the units of memory in a quantum computer. The machine she wants to use has 56. We found a way to express the same calculation using 32 qubits. That brings the experiment within the machine's memory limit and cuts the number of operations involving pairs of qubits by 68%. We tested the program in a simulation. An ordinary computer can also work out the result of this particular experiment, which gave us a way to check our answer.

Where did the saving come from? Think of recording one stop on a delivery route. You could give every possible city its own yes-or-no box and tick just one. Or you could write down the city's number. Both records tell you where to go. Our software can recognise this kind of waste in the instructions of an existing quantum program, even one written by another group. In an eight-city test, it cut the memory needed by 57% and proved that every valid route kept the same chance of appearing. A smaller program can still take more steps to run.

Fitting the calculation onto a machine is only half the problem. A quantum program can give a different answer each time it runs. Suppose you want the shortest round trip. Seeing the same tour in reverse tells you nothing new about which roads to take. The paper shows how to separate what an answer tells you about the best route from what it tells you about the direction in which the route was listed.

That distinction matters when deciding how to use a handful of answers. In 15 tests, the task was to work out the best way to assign facilities to sites from just twelve outputs. When the software interpreting them knew which answers the quantum program usually produced, training the program to make its answers more distinctive raised success from 64% to 74%. When the software knew only what each proposed arrangement would cost, the very same change lowered success from 44% to 27%. So the best way to train a quantum program depends on what the person or software reading its answers already knows.

Abstract

We report three results. First, a compiler recognises a one-hot program in a flat Pauli-rotation list or supported gate-level output, derives compact binary registers and, given a verified ordering of the recovered groups, certifies equivalence on the valid subspace without enumerating its basis. On the eight-city routing fixture the pass recovers the compact program from 1,309 ungrouped rotations and from Qiskit and TKET circuits synthesised from the one-hot input. An extension that preserves the order of individual rectangles also recovered compact programs from independently authored routing layers on nine and sixteen wires, yielding six and eight wires respectively. We checked every valid input at both sizes. With the ordered lowering rule frozen, we recovered the earlier eight-city examples at 21 logical wires and tested a second external source. Its three- and four-position generator lists gave six and eight logical wires. Gate-level acceptance also depended on the angle setting. With matched compilation, the four-position Tübingen layer fell from 528 to 296 CNOTs, while depth rose from 212 to 269. The Stanford generator also admitted six-position recovery, although a duplicate-merged one-hot implementation used fewer CNOTs there. The spin echo exposes a failure of count-ordered grouping when later merges change dependencies. Topologically ordered grouping recovered the correct echo on 32 of 64 source wires and, after fusion, used 140 rather than 432 native two-qubit rotations.

Second, under uniform hidden relabelling, a routing sample's entropy deficit separates exactly into information about the unoriented optimum and conditional information about orientation. The same

two terms exhaust the deficit in a reflection-symmetric quadratic-assignment test. For Max-Cut under vertex relabelling, an exact accounting also needs concentration across numbers of vertices on one side and information about permutations within the two sides. On the initial Max-Cut cohort, deficit training raised the mean deficit from 0.658 to 1.633 bits. Across that cohort and an independent check, it gave no consistent gain in twelve-sample calibrated recovery.

Third, the choice of training objective depends on the decoder. Across fifteen five-facility assignments, changing from cost to deficit training raised mean twelve-sample recovery from 64.3% to 73.8% using the exact reference output law and lowered it from 43.6% to 26.8% using a thermal model of costs. On 31 held-out Max-Cut graphs, we reused one angle pair per objective and supplied the decoder with each graph's exact reference law. Cost training led at one and three samples. Training for information about the hidden relabelling led at six and twelve, reaching 43.5% recovery at twelve against 30.4% for cost training. Replacing exact probabilities with 256 independent reference samples per graph gave 38.8% at the relabelling setting. At eight vertices the 256-reference-sample gain was 2.6 percentage points (paired 95% interval −1.1 to 6.4). On the larger cyclic-shift cut tasks, the cost-only reader at the cost setting outperformed the learned pair reader at either setting.

Supporting results apply support-size bounds, exact diagonal completion and basis-state propagation to resource accounting. On one routing instance the recovered encoding uses 21 of the original 49 data wires. A local-readability bound quantifies the extra width needed to recover positions separately. A benchmark audit certifies opportunities for reducing the final readout width on 39 of 103 analysed files. Under a specified scrambling model, information per attempted shot also gives a criterion for selecting depth.

Keywords Quantum compilation · entropy · invariant subspaces · Pauli rotations · permutation mixers · one-hot encodings · decompilation · error detection · qubit tapering · Hamiltonian simulation

Table of contents

General reader summary
Abstract
1 Input contract and scope
2 Entropy of a question
3 Closed word sets and injective sketches
4 Exact reduction of Pauli rotations
5 Certifying exchange operators and inferring registers
6 Deriving the label register from a one-hot description
7 Using what the parts know about each other
8 Preparing and lowering the accepted circuits
9 Reconstructing the routing benchmark
10 Transfer to an independently generated fermion model
11 Circuit controls and tests beyond routing
11.5 Spin-chain capacity test and group ordering
12 Interpretation and limits
Appendix A Reproduction protocol
Appendix B Six small reconstruction tests
References

1 Input contract and scope

Let n be the number of data wires. The input comprises a known bit string s, an initial-state convention, and an ordered list of pairs (H_j, t_j). Each H_j is a sparse real linear combination of Hermitian Pauli strings. The target evolution applies $\exp(-it_jH_j)$ in list order, with the earliest factor acting first. The initial state is either the basis state $|s\rangle$ or, for the permutation branch, the uniform positive-amplitude superposition over the orbit of s under the supplied exchanges. The latter convention is explicitly requested by the caller. The one-hot front end of Section 6 accepts the same two conventions.

We use little-endian wire indices and bit vectors over the binary field GF(2). Addition of bit vectors is XOR. For masks x, z, write h(x,z) for the integer number of wires on which both masks have a one. The binary dot product is $x\cdot z = h(x,z) \bmod 2$. Our convention fixes the sign of every occurrence of Y:

$$P(x,z) = i^{h(x,z)} X^x Z^z, \qquad P(x,z)|v\rangle = i^{h(x,z)} (-1)^{z\cdot v} |v + x\rangle. \tag{1}$$

A computational-basis measurement is followed by classical decoding when the logical register is smaller than the data register. Amplitudes may differ by a single global phase. This convention is sufficient for isolated state preparation, evolution and measurement. A controlled invocation of the entire circuit can turn a global phase into a relative phase and requires explicit phase bookkeeping.

Dispatch rule. For a basis input and one Pauli term per generator, apply the affine construction in Section 4. Otherwise inspect every generator that contains a nonzero flip mask. Require each such generator to equal a product of disjoint exchanges with unit coefficient. Require all remaining generators to be diagonal. Recover compatible registers and test the seed as in Section 5. If a test fails, the prototype returns "unsupported". A full-rank affine case uses the original wires and seed. This is a partial compiler with deterministic dispatch. It does not search all valid encodings for the cheapest one.

A one-hot front end (Section 6) can be placed before this dispatch. It accepts programs whose mixing generators act as exchanges of whole rows of wires on valid assignments only, rewrites them on binary label wires, and passes the result to the exchange test above. It is a separate module; the selector is unchanged. A flat pass, also in Section 6, accepts an ordered list of single Pauli rotations with no generator boundaries, rebuilds the generators, and feeds the front end. A lifting step turns a list of CNOTs and one-wire gates into such a list, and a hop pass handles one-hot rows whose moving terms are hops within a row.

Two further passes use the entropy accounting of Section 2. For an exchange program on a basis seed, the forest test of Section 7 is tried first; when it passes, a history register replaces the label register. For a label register, the cross-row sketch of Section 7 is an optional pass that removes one slot. Like the front end, these are separate modules, verified in Appendix A.

Relationship to earlier work. Symmetry tapering, permutation-preserving mixers, compact tour encodings, compilation from objective programs, and synthesis of Pauli networks each have prior literature [1–6]. So do encodings of fixed-particle-number sectors by classical codes [1, 11], the preparation of permutation states [12, 13], space-efficient and encoding-independent formulations of routing problems [14, 15], and the automated detection of invariant subspaces from an initial state [16]. The Lehmer code used in Section 7 is classical, and a register built on it reaches the floor for routing [18]. Three lines of work are closest to the material of Sections 6 to 9. Decompilers lift compiled circuits to Pauli rotations and recover layers, angles and the order of registers [22], and the cost of recovering phase data dispersed by fault-tolerant compilation has been bounded [23]; neither changes the encoding or the number of wires. The outputs of circuits of this family are known to be

constant on symmetry orbits [24] and, after one layer, to resemble thermal laws [25], and edge frequencies in pools of tours have been used to fix parts of a route, classically [21] and with quantum annealing [26]. Discarding runs that leave the valid subspace is an established mitigation for one-hot and particle-conserving circuits, with exact survival probabilities under depolarising noise [27, 28]. Qubit-reuse compilation reduces the number of physical qubits by measuring and resetting wires once their earlier computations finish [32]. Our width reductions exploit restrictions on which states the circuit can reach. Storing a permutation near its information content while keeping single entries readable is a studied problem of succinct data structures, with lower bounds on the surplus [35–37]; Section 7 states an exact small case of it for registers. Compiler passes that propagate known basis states or a bounded set of occupied words through a circuit [38, 39] are the analyses behind the audit of Section 11. Friedewald, Sichani and Shyu [40] optimise unspecified diagonal costs on reachable subspaces for QAOA placement. Section 9 applies this completion principle to the routing cost table.

The compiler contribution is a sufficient test for re-encoding a redundant register on fewer wires. For the one-layer routing family it starts from a flat Pauli list or from gates admitting the lifting of Proposition 16. The layout and equivalence arguments apply under the stated input conditions. The SDK gate lists come from independently implemented synthesis passes applied to our source instances. Proposition 23 gives the information carried about a hidden relabelling and its optimum, and a necessary sample bound for certain recovery. The measured decoder success is a separate finite-sample result. Proposition 20 bounds the wires of any register in which several positions can be read separately. The training and decoding results of Section 9 are empirical, on maps of eight to ten cities in exact simulation. The register redundancy also bounds fault detection and, under the specified scrambling model, determines the information yield together with the ideal law and fault-free probability.

2 Entropy of a question

Three measures of information organise the paper. The support size determines how many distinct words an exact register must distinguish. Shannon entropy weights each measured word by its probability. Mutual information requires a further choice: which unknown quantity the measurement is supposed to reveal. We give that quantity and its prior explicitly in Section 9.

Capacity. The *capacity* of a set of words is the base-two logarithm of the number of words in it. The capacity of the question is the capacity of the smallest set of words on which the state is supported throughout the circuit. Section 3 calls a set that the generators cannot leave closed. Capacity is counted in bits and it bounds wires.

Proposition 1 (floor). An exact encoding of a set O of words, in the sense of Section 3, uses at least $\log_2|O|$ wires.

Proof. An encoding is an injective map from O to r-bit words, and there are 2^r of those. ■

Parts and what they know about each other. Split the wires into parts A_1, …, A_k, and write O_A for the set of patterns that part A shows as the word ranges over O. Every word of O is a choice of one pattern from each part, so $|O|$ is at most the product of the $|O_A|$. The difference

$$K = \Sigma_A \log_2|O_A| - \log_2|O| \geq 0$$

is the capacity that the parts hold in common: what they know about each other. It is zero exactly when every combination of patterns occurs. Call an encoding *part-local* if the pattern of each part can be decoded from wires assigned to that part alone.

Proposition 2 (local floor). A part-local encoding uses at least $\Sigma_A \lceil\log_2|O_A|\rceil$ wires. An encoding with fewer than $\Sigma_A \log_2|O_A|$ wires decodes some part from wires assigned to other parts, and no encoding saves more than K bits in that way.

Proof. The wires assigned to part A must distinguish its $|O_A|$ patterns. The last statement is Proposition 1. ■

For the routing question with m cities to place, the one-hot wires taken singly have a capacity of one bit each, m^2 in all. The rows taken singly can each show m patterns, $m \log_2 m$ in all. The whole has $\log_2 m!$. At eight cities these are 49, 19.65 and 12.30 bits. The front end of Section 6 uses what the wires of one row know about each other. Section 7 uses what the rows know about each other.

Information shared by rows. Under a uniform random permutation the first row has Shannon entropy $\log_2 m$. Given the first j rows, the next row is uniform over m − j labels. The chain rule [17] gives the joint entropy as the sum of $\log_2(m - j)$ over j from 0 to m − 1, which is $\log_2 m!$. Each row alone has entropy $\log_2 m$, so the total correlation of the rows, the sum of their entropies minus the joint entropy, equals $m \log_2 m - \log_2 m!$. For seven rows this is 7.35 bits, and here it coincides with K. The last row has zero conditional entropy given the other six, which is why the cross-row register of Section 7 can omit its three wires.

Proposition 3 (capacity and depth). Let the state before a step be supported on a set W of words. A diagonal step leaves the support inside W. A step $\exp(-i\theta S)$, where S permutes words and $S^2 = I$, leaves it inside the union of W and S(W). Hence capacity grows by at most one bit for each such step, and after t of them from a basis word it is at most the smaller of t and the capacity of the orbit.

Proof. $\exp(-i\theta S) = \cos(\theta)\, I - i \sin(\theta)\, S$. ■

Measurement entropy and deficit. For the decoded outcome X with distribution P, the *measurement entropy* is $H(X) = -\Sigma_x P(x) \log_2 P(x)$, with zero terms taken as zero. For the uniform distribution U on O, the relative entropy is $D(P||U) = \Sigma_x P(x) \log_2[P(x)/U(x)] = \log_2|O| - H(X)$. We call it the *entropy deficit*. It measures how concentrated the outcomes are in the measurement basis. Permuting the probabilities among the words preserves it and can change the chance of success, so it is not by itself a measure of progress towards an answer. An isolated circuit state is pure and keeps zero von Neumann entropy throughout; the entropies reported here concern classical measurement outcomes.

Computing them without listing words. The certificates of Sections 4 to 6 give these numbers in closed form. An affine space of rank r has capacity r. A register of m slots holding m distinct labels, with a connected exchange graph, has capacity $\log_2 m!$, and each slot alone has capacity $\log_2 m$. A one-hot row of m wires has capacity $\log_2 m$. The number of two-way steps is read from the generator list. Only the measurement entropy and the mutual information need amplitudes, and in this paper they are computed by the verifier.

Capacity counts words, which is the right measure for an exact register. The Shannon entropy of a single part under the uniform distribution on O can be smaller, since a one-hot wire among seven is set one time in seven, but that number bounds compression on average and not an exact encoding. For general supports the count difference K and the Shannon total correlation can differ.

3 Closed word sets and injective sketches

The reductions of Sections 4 to 7 are instances of one statement. Call a set O of n-bit words *closed* for the program if every generator H_j maps the span of $\{|v\rangle : v \in O\}$ into itself.

Lemma 4 (closed sets). If O is closed, its span is invariant under every $\exp(-it_jH_j)$. If the initial state lies in that span, so does the state after every step.

Proof. The span is finite-dimensional and each generator maps it to itself, so every partial sum of the exponential series does too, and so does the limit. ■

A convenient sufficient condition uses the form of the terms. Call an operator *word-moving* if it sends every basis word to a multiple, possibly zero, of one basis word. Pauli strings are word-moving by (1). So are diagonal operators and wire permutations. If a generator is a combination of word-moving terms, each of which sends every word of O to zero or to a multiple of a word of O, then it maps the span of O into itself. The grouping matters: the mixing generators of Section 6 preserve their closed set although their individual Pauli strings do not.

The intersection of closed sets is closed, so the seed has a smallest closed set, its orbit. Any closed superset of the orbit serves equally in what follows. The compiler never lists a closed set. It needs a description from which it can decide which local words a certificate must probe.

Lemma 5 (sketch criterion). Let L be an $r \times n$ binary matrix, c a fixed r-bit word, and $\lambda(v) = Lv + c$. Write $O \oplus O$ for the set of sums $u + v$ with u and v in O. Then λ is injective on O exactly when

$$\ker L \cap (O \oplus O) = \{0\}. \tag{2}$$

Proof. $\lambda(u) = \lambda(v)$ is the statement $L(u + v) = 0$. ■

When (2) holds we call λ a *sketch* of O and r its width. More generally, an *encoding* of O is any injective map λ from O to r-bit words; a sketch is an encoding given by a matrix. The *code space* C is the span of the states $|\lambda(v)\rangle$ with v in O. Define $V|\lambda(v)\rangle = |v\rangle$ for v in O, and $V = 0$ on basis states outside $\lambda(O)$. Then V is an isometry from C onto the span of O.

Theorem 6 (transfer). Let O be closed, let λ be an encoding of O, and let H be a generator. Let $\hat{H}$ be any Hermitian operator on r wires that maps C into itself and satisfies $\langle\lambda(u)|\hat{H}|\lambda(v)\rangle = \langle u|H|v\rangle$ for all u, v in O. Then, on C,

$$H\,V = V\,\hat{H}, \qquad \exp(-itH)\,V = V\exp(-it\hat{H}). \tag{3}$$

Consequently, for an ordered list of generators whose transferred generators are chosen in this way,

$$U_{\text{physical}}\,V = V\,U_{\text{logical}} \quad \text{on } C. \tag{4}$$

For every initial state $|\psi\rangle$ in the span of O, preparing $V^\dagger|\psi\rangle$ on r wires, running the logical circuit, measuring y and returning $\lambda^{-1}(y)$ reproduces the physical output law.

Proof. Take v in O. Closure gives $H|v\rangle = \Sigma_u \langle u|H|v\rangle\,|u\rangle$ with the sum over u in O. Because $\hat{H}$ maps C into itself, $\hat{H}|\lambda(v)\rangle = \Sigma_u \langle\lambda(u)|\hat{H}|\lambda(v)\rangle\,|\lambda(u)\rangle$ with the same range of u, and applying V gives the same vector. This is the first identity in (3). Powers follow by induction, since $\hat{H}$ keeps C, and the exponential follows by summing the series. Composing the factors in list order gives (4). The measurement statement holds because V sends $|\lambda(v)\rangle$ to $|v\rangle$ and λ is injective on O. ■

The theorem leaves two things open, and they are where the work lies. First, $\hat{H}$ is fixed only on the code space; elsewhere the compiler may choose whatever extension is cheapest to implement. Second,

the theorem does not say how to find O, λ and $\hat{H}$ without enumeration. Three transfer rules make it effective.

Translations. If every term is a Pauli string and O is the affine space s + span{x_j}, then O $\oplus$ O is that span, (2) asks for a left inverse of a basis matrix, the code space is the whole logical register, and each transferred term is a signed Pauli string. This is Theorem 7.

Wire permutations. If a term acts on O as a permutation ρ of wires, and L carries it to a permutation τ of sketch wires, Lρ = τL, then the transferred term is τ. Theorem 11 uses this with rows of wires exchanged as blocks. Section 5 is the case in which the sketch is the identity.

Diagonal terms. A diagonal term with eigenvalue d(v) transfers to the diagonal function $y \mapsto d(\lambda^{-1}(y))$ on the code space, with any values elsewhere. In general it is no longer a single Pauli string. Its Walsh expansion gives the phase tables of Section 9.

Three sizes of sketch. Criterion (2) is a statement about a linear code: the kernel of L must contain no nonzero difference of two reachable words. For an affine space of dimension r the differences fill a subspace, and the smallest width is r. For a row of m wires that always holds exactly one set wire, the differences are the words with two set wires. Criterion (2) then asks only that the columns of L be distinct, so $\lceil \log_2 m \rceil$ wires suffice. For words with exactly N set wires among k, the differences have at most 2N set wires, and (2) asks for the parity-check matrix of a code of minimum distance greater than 2N. Such codes exist with about $N \log_2 k$ rows. This third case is the known use of classical codes for fixed particle number [1, 11]. We checked the criterion for two set wires among fifteen with an eight-row check matrix (Appendix A) and implement no lowering for it. The first two cases are implemented and are the subject of Sections 4 to 6.

4 Exact reduction of Pauli rotations

This is the translation rule with every sign made explicit.

Theorem 7. Suppose every generator is a single Pauli string with its real coefficient absorbed into its rotation angle. Let B be an $n \times r$ binary matrix whose columns form a basis of the span of all flip masks x_j. Define the isometry V by $V|y\rangle = |s + By\rangle$. Each physical rotation admits an explicitly computable rotation on r qubits that intertwines with V. Thus the compressed circuit on the all-zero logical input yields the physical amplitudes after decoding.

Proof. Independence of the columns makes $y \mapsto s + By$ injective, so the images of the logical basis are orthonormal. For a given P(x,z), solve $Bu = x$ and set $v = B^T z$. Then $z \cdot By = v \cdot y$. Applying (1) to the image of a logical basis state and comparing coefficients gives

$$P(x,z)\, V = \sigma\, V\, P(u,v), \qquad \sigma = (-1)^{z \cdot s}\, i^{h(x,z) - h(u,v)}. \tag{5}$$

The exponent difference is even: $x \cdot z = (Bu) \cdot z = u \cdot v$. Hence σ is real and belongs to $\{+1, -1\}$. Equation (5) proves invariance and fixes the logical rotation angle, including for strings containing Y. Taking powers in (5), expanding the exponential, and composing in order proves (4), with the code space equal to the whole logical register.

Measuring y and returning $s + By$ therefore reproduces the physical output law. The affine space is an invariant envelope. At particular angles or circuit depths, the state can occupy a proper subset. The argument gives no minimality claim for all conceivable encodings.

Relation to tapering. The $n - r$ wires that disappear are accounted for by the strings Z^c with c orthogonal to every flip mask. Each such string commutes with every generator and has eigenvalue $(-1)^{c \cdot s}$ on the input. Theorem 7 is therefore symmetry tapering [1] restricted to symmetries that are diagonal in the computational basis, in the sector fixed by the basis input, with the signs of an ordered rotation sequence made explicit. Symmetries that are not diagonal are outside its reach. The Ising control of Section 11 shows the consequence.

How to implement it. Scan flip masks in input order. Maintain a binary echelon table keyed by the highest remaining set bit, recording the coordinates of each pivot in the selected original columns. Eliminate a candidate by XOR with its matching pivots. Append the original mask as a column only when a nonzero residual remains. A second pass gives u for each term. Set bit k of v from the parity of z intersected with column k. Evaluate the exponent in (5) modulo four, then multiply the angle by the resulting sign. This takes polynomial work in the number of input terms and wires. The compiler never constructs the 2^n basis states. If $r = 0$, diagonal evolution of the specified basis state contributes only phase.

Order of operations. Theorem 7 translates rotations in their given order. If the flat front end first collects rotations into groups, their final order must respect every dependency between noncommuting source rotations. Proposition 13 gives the condition. Section 11.5 compares count-ordered and topologically ordered grouping on a spin echo.

Worked sign check. Take $s = 01$ and $B = (1,1)^T$. The labels $y = 0, 1$ decode to 01, 10. The string YY has $u = 1$, $v = 0$, $h(x,z) = 2$ and $z \cdot s = 1$. Thus YY acts as $+X$ on this sector. For $s = 00$, it acts as $-X$. Dropping either sign in (5) would change one of these rotations.

5 Certifying exchange operators and inferring registers

This section is the wire-permutation rule with a certificate at the level of operators. No wires are removed. The result is a register structure that the lowering of Section 8 can use.

Lemma 8. For distinct wires a, b, their exchange is

$$S_{ab} = (I + X_aX_b + Y_aY_b + Z_aZ_b) / 2. \tag{6}$$

Act on 00 and 11: the XX and YY contributions cancel, and I + ZZ returns twice the original state. On 01 and 10 the diagonal terms cancel, while XX + YY gives twice the swapped state. This proves (6) on a basis. For w disjoint pairs, the product S has 4^w distinct Pauli strings, each with coefficient 2^{-w}. Disjoint supports ensure that local factors introduce no additional multiplication signs.

Certificate. Apply the candidate generator H symbolically, using (1), to every basis word e_j with a single one. Each result must be one single-one word with coefficient +1. Read these outputs as proposed wire images and require their map to be an involution. Its nontrivial cycles are disjoint pairs. Regenerate their entire Pauli expansion using (6) and compare it with the supplied coefficient dictionary. Equality proves H = S on every input because Pauli strings are a linearly independent operator basis. The final equality is essential: adding a projector onto the state in which two specified wires are both one can leave every single-one probe unchanged. The equality test rejects that modification.

Register test. Give each wire a signature consisting of the indices of certified exchanges that move it. Wires with identical signatures form candidate slots. For each exchange require exactly two touched slots, complete coverage of both slots, and bijective pairing across them. Choose a bit ordering in the first slot and propagate that order through the pairings. Reject inconsistent orders on cycles and disconnected slots. Finally try common bit-significance orders until the decoded seed labels are exactly 0, …, m − 1 in some order, with slot width $w = \lceil \log_2 m \rceil$. These are sufficient tests. For example, an isolated two-slot pair can give its two ends identical signatures, and the implementation can decline a real symmetry.

Theorem 9. Under these tests every certified operation exchanges complete labels of aligned slots. The group they generate acts transitively on all m! orderings of the seed labels. Diagonal phases and the exchange rotations preserve the span of those orderings.

Proof. Alignment identifies bit k at either end of each edge. The product in (6) exchanges matched bits, so it exchanges the two labels. Edge transpositions of a connected graph generate all slot permutations: on a path, move one endpoint towards the other, swap at the final edge and undo the preceding moves in reverse order. This effects the endpoint transposition. Arbitrary transpositions generate the symmetric group. Distinct seed labels give m! orderings. Since $S^2 = I$, $\exp(-itS) = \cos(t)\, I - i \sin(t)\, S$, a linear combination of two operations preserving this span. A diagonal operator preserves it termwise. Transitivity concerns the generated group, not the support reached by a single finite sweep. ■

6 Deriving the label register from a one-hot description

The compact routing program of Section 9 stores each position of a tour as a binary label. The textbook description [2] instead uses one wire for every pairing of a city with a position: wire (a, p) is set when city a is visited at position p. With m cities to place this takes m^2 wires where $m\lceil \log_2 m \rceil$ suffice. This section derives the compact program from the one-hot program by Theorem 6. The certificate inspects m^2 local words for each mixing generator and enumerates no tours.

One-hot input. The rows are the positions p = 0, …, m − 1, each with m wires. A valid word has exactly one set wire in every row, and no city is set in two rows. The cost generator is diagonal: a real combination of products of occupation numbers $(I - Z)/2$ on at most two rows. For two rows p and q the mixing generator is the sum of partial swaps [2]

$$M_{pq} = \Sigma_{a<b} \left(\sigma^{+}_{a,p}\, \sigma^{+}_{b,q}\, \sigma^{-}_{a,q}\, \sigma^{-}_{b,p} + \text{h.c.} \right), \qquad \sigma^{\pm} = (X \mp iY) / 2. \tag{7}$$

Each summand is supplied as its eight Pauli strings, with coefficients +1/8 or −1/8. The summand for (a, b) exchanges cities a and b between the two rows when both are present, and returns zero otherwise. As an operator M_{pq} is not the wire exchange of the two rows: the exchange fixes the all-zero word and M_{pq} annihilates it. The certificate of Section 5 therefore rejects M_{pq}, correctly.

Lemma 10 (local certificate). Let H and H′ act as the identity outside a wire set A, and let F be a set of words. If $H|a\rangle = H'|a\rangle$ for every word a on A that is the restriction of some word of F, then H and H′ agree on the span of F.

Proof. Split a word v of F into its part on A and the rest. Then $H|v\rangle$ is H applied to the part on A, tensored with the rest, and likewise for H′. ■

Row recovery. Give each wire the set of mixing generators that act on it, where a generator acts on a wire if one of its Pauli strings is not the identity there. Wires with identical signatures form candidate rows. Each mixing generator must cover exactly two rows, and all rows must have the same size. This is the register test of Section 5 applied to supports.

Probes. For a mixing generator on rows P and Q, apply it symbolically, using (1), to each of the m^2 words with one set wire in P and one set wire in Q. Each image must be zero, or the same word, or a single word with one set wire in each row and coefficient +1. From the images that move, read a pairing between the wires of P and Q: the set wire that leaves P reappears at its partner in Q. Require the pairing to be a well-defined involutive bijection. Require every word whose two set wires are not partners to map, with coefficient +1, to the word in which both have moved to their partners. A word whose two set wires are partners places one city in both rows. It lies outside the valid family and is not constrained.

Pairings propagate along the graph of rows and group the wires into classes, one class for each city. Require every class to contain exactly one wire from each row, which rejects inconsistent cycles and disconnected rows. Finally require the seed to have exactly one set wire in each row, all in different classes.

Theorem 11 (one-hot rows to labels). Suppose the tests pass with q rows and m classes, and put $w = \lceil \log_2 m \rceil$. Number the classes 0, …, m − 1. Let F be the family of words with one set wire in each row and all set wires in different classes. Then:

(i) F is closed and contains the seed.

(ii) The linear map L that sends the wire of class a in row p to the binary word of a on the w label wires of slot p satisfies (2) on F. It is a sketch of width qw.

(iii) Each mixing generator transfers to the exchange (6) of the two label slots.

(iv) Each diagonal generator transfers to a table on the label wires of the rows it touches, in which a label outside 0, …, m − 1 is read as an empty row.

The transferred program satisfies (4), and its exchange generators pass the certificate of Section 5.

Proof. (i) and (iii). The restriction of F to rows P and Q consists of the probed words whose set wires are not partners. On these the probes show that the generator equals the exchange of the two rows along the pairing. By Lemma 10 the two agree on the span of F. A row exchange permutes the rows of a word and so maps F to F, and diagonal generators fix every word, so F is closed. The map L carries the row exchange to the exchange of slots p and q bit by bit. This is the wire-permutation rule of Section 3, and the label exchange maps the code space to itself because it permutes labels.

(ii) Two different words of F differ in some row, where their sum is a word with two set wires of different classes. L sends it to the sum of two different binary words, which is nonzero. Hence no nonzero element of F $\oplus$ F lies in the kernel.

(iv) On a word of F with label a in row p, the operator Z on the wire of class b in row p has eigenvalue −1 if a = b and +1 otherwise. A product of such factors is a function of the labels of the rows it touches. Theorem 6 leaves its values on other labels free. Reading such a label as a row with no set wire makes every occupation number vanish, which is the zero padding used in Section 9. A wire outside every row is never moved, and a Z factor there contributes the constant sign fixed by the seed. ■

Lemma 12 (the one-hot Pauli product is the same calculation). On the span of F, the partial swaps in (7) for one pair of rows commute with one another, and each is a sum of eight commuting Pauli strings. Hence $\exp(-i\beta M_{pq})$ equals, on that span, the ordered product of the Pauli rotations of its strings, taken one partial swap after another.

Proof. On the restriction of F to the two rows, the summand for (a, b) acts within the two words that hold a and b, and annihilates the others. Different summands act on orthogonal two-dimensional blocks, so on the span their products vanish in either order. The eight strings of one summand share their four wires, carry an even number of Y factors, and so differ pairwise by X against Y on an even number of wires. They commute. ■

Lemma 12 matters for the comparison in Section 9: a general-purpose compiler given the one-hot Pauli strings in this order implements, on valid words, the same evolution as the label circuit.

Cost and scope. A mixing generator has $4m(m - 1)$ Pauli strings and is probed on m^2 words. A diagonal term that touches k rows gives a table with 2^{kw} entries. No tour is enumerated. At twelve cities, with 121 one-hot wires, 5,622 input strings and 1,210 probes, the front end ran in about one second including construction of its input.

The certificate is a statement on F only. A term that acts solely on invalid words, such as a projector onto two set wires in one row, passes unnoticed and leaves the output unchanged. Under the input contract this is the correct behaviour. It is also the opposite of the operator certificate of Section 5, which must reject such a term. The front end guarantees nothing for seeds outside F, and it declines them. Like the register test, the signature step cannot separate two rows joined by a single generator.

Recognition from a flat list of rotations. Everything above assumes that the caller marks which Pauli strings form one generator. A circuit does not arrive that way. The textbook routing circuit, written

out, is an ordered list of single Pauli rotations: the cost strings, then eight strings for every partial swap. The pass described here rebuilds the generators from that list and requires the emitted groups to respect the original dependencies for the equality proof to apply. Partitioning Pauli rotations into commuting sets is standard in compilers [6]; the pass uses it to recover word-moving terms and then a grid.

Proposition 13 (regrouping). Read the list as a dependency order: rotation i precedes rotation j when i comes first and the two do not commute, or when a chain of such pairs leads from i to j. Label every rotation by its flip mask and by the parity of its number of Y factors. Take the labels in a fixed order. For each, gather a set of its rotations into one group whenever no rotation outside the set lies between two of its members in the dependency order; the gathered set then counts as one element, and later tests are made on the reduced order. Return the groups in an order that extends the reduced order, with ties broken by the contents of the groups. Then (i) the rotations of one group commute with each other; (ii) the product of the groups in order, each taken as the exponential of the sum of its strings weighted by their angles, equals the product of the original list as an operator; (iii) every group is a word-moving term; (iv) two lists that differ by exchanges of commuting neighbours give the same sequence of groups; (v) if some order in the commutation class of the list writes it as a sequence of blocks, each made of rotations with one label, and no label is used by two blocks, then the groups are exactly those blocks.

Proof. Two strings with the same flip mask x commute exactly when $x \cdot z$ takes the same value on both, and $x \cdot z$ is the parity of the number of Y factors. This gives (i). Strings with one flip mask all send $|v\rangle$ to a multiple of $|v + x\rangle$, and so does their sum, which gives (iii). A set with no outside rotation between two of its members can be made contiguous in an order that respects every dependency, and contracting it leaves a partial order. The returned order therefore respects every dependency of the list and has each group contiguous. Two orders of the same rotations that agree on every non-commuting pair are connected by exchanges of commuting neighbours [30], and commuting exponentials can be exchanged and merged, which gives (ii). Exchanges of commuting neighbours do not change the dependency order, and every choice in the procedure depends only on that order and on the rotations themselves; this is (iv). For (v), in the block order every outside rotation comes before all members of a block or after all of them, so it cannot follow one member and precede another in the dependency order. The same holds for every subset of a block, since members of a block have no dependency on each other, and it remains true when other blocks have been contracted. Each block is therefore gathered whole. ■

Statement (iv) makes the pass a function of the commutation class and not of the list. Statement (v) covers the routing circuit and every list equivalent to it, since each partial swap and the cost block have labels of their own. A list of several layers repeats labels. Statements (i) to (iv) still hold for it, but (v) does not apply, and the family test of Proposition 14 is written for one layer.

Ordering contracted groups. Proposition 13(ii) requires the emitted order to respect the dependencies after all groups have formed. The count-ordered rule ranks groups by an ancestor count saved at formation. Later contractions can make those counts stale. The topologically ordered rule sorts the final dependency graph, using group content to break ties among groups ready to emit. Section 11.5 tests both rules.

Classifying a term. Apply the term symbolically to every word of its own wires. For a partial swap these are sixteen words. The term is a *rectangle move* if its flip mask has four wires, it acts on no others, and it is nonzero on exactly two words: a word with two set wires and its complement on the four, with the same real coefficient. The two set wires before the move form one diagonal of the

rectangle, and the two after it form the other. The four pairs that join a wire of one diagonal to a wire of the other are its sides.

Proposition 14 (grid recovery). Colour the sides of all rectangle moves with two colours so that opposite sides of every rectangle agree and adjacent sides differ. Suppose that (a) such a colouring exists and the sides form one connected family; (b) the connected sets of wires under each colour, called lines, are such that every line of one colour meets every line of the other in exactly one wire; (c) for one colour, the slot lines, the rectangle moves on each pair of slot lines that occurs are present for every pair of lines of the other colour, with one common angle β, and can be gathered into a block by passing only groups they commute with; (d) the seed has exactly one set wire on every line. Let F be the set of words with one set wire on every line. Then F is closed, the moves of one block commute on the span of F, and their product is exp(−iβM) there, where M is their sum divided by β. The grouped program so obtained equals the flat list on the span of F, and Theorem 11 applies to it.

Proof. A rectangle move takes one set wire off each of two slot lines and each of two lines of the other colour and puts one back on each, so it maps a word of F to a word of F or to zero. Diagonal groups fix every word. Hence F is closed. On the restriction of F to two slot lines, the move for a given pair of the other lines is nonzero only on the two words that hold that pair, so different moves of a block act on orthogonal two-dimensional subspaces and commute on the span, as in Lemma 12. Their sum is the generator (7) written on the recovered grid. Gathering exchanges only commuting groups, so with Proposition 13 the grouped program equals the list on the span of F. ■

The pass is not told which strings form a partial swap, which partial swaps form a mixer, where the rows are, or which family of lines holds the positions. It tries both directions and keeps the one whose blocks are complete, and it then runs the local certificate of this section on each block. An order that agrees with the original only on valid words, such as a permutation of the partial swaps of one position pair, yields the same program, because by Lemma 12 those orders agree on F.

Hop circuits. A term whose flip mask has two wires, and which is nonzero exactly on the words with one of the two set, with one magnitude, is a *hop*. Hops conserve the number of set wires on each connected component of the graph they form. The words that share the seed's count on every component are therefore a closed set, and its capacity is a sum of logarithms of binomial coefficients. This gives the capacity of a fermionic step from its flat list. No register is built for that case, where a hop carries a sign that depends on other wires.

When every hop acts on its two wires alone, with one real coefficient, and the seed has one set wire in every component, the components are one-hot rows of independent variables. This is the usual description of graph colouring, with one wire for each colour of each vertex [14].

Proposition 15 (hop rows to labels). Let the moving terms of a regrouped list be hops of this kind, let the rows be the components of the hop graph, and let the seed have one set wire in every row. Number the wires of row r from 0 to $k_r - 1$ and give the row a label register of $w_r = \lceil \log_2 k_r \rceil$ wires. Then the words with one set wire in every row form a closed set F of capacity $\Sigma_r \log_2 k_r$; the map from a word to its row labels is linear and satisfies (2) on F; a hop with coefficient c between the wires numbered a and b becomes $c(|a\rangle\langle b| + |b\rangle\langle a|)$ on the label register of its row, which, writing w for w_r, is a sum of 2^{w-1} commuting Pauli strings with flip mask $a \oplus b$; and a diagonal string becomes a table on the label registers of the rows it touches. The label program equals the list on the span of F.

Proof. A hop moves the set wire of one row to another wire of that row or gives zero, and diagonal terms fix words, so F is closed. Two words of F differ in the label of some row, which is (2). On F a hop is nonzero exactly on the two words that hold a or b in its row, and it exchanges them with coefficient

c. By (1) the operator $c(|a\rangle\langle b| + |b\rangle\langle a|)$ is the sum of the strings with flip mask $a \oplus b$ and $z\cdot(a \oplus b) = 0$, each with coefficient $\pm c/2^{w-1}$, and strings with one flip mask and one value of $x\cdot z$ commute. The diagonal rule is that of Section 3, and Theorem 6 gives the equality. ■

Recognition from a compiled circuit. A compiled circuit is a list of CNOTs and one-wire gates, and its rotations are no longer visible. Lifting such a circuit to a sequence of Pauli rotations followed by a Clifford frame is standard [22]. Two points need care for the present use. A one-wire gate between two CNOTs may be a product of Clifford gates and a rotation, merged by the compiler. And the angle of a rotation is then fixed only up to quarter turns, because a quarter turn about a Pauli axis is itself a Clifford gate.

Proposition 16 (lifting). Let a circuit consist of CNOTs and one-wire gates, each of which is a Clifford gate C or a product $C\exp(-i\theta W/2)$ with W a Pauli axis. Read the gates in order, keeping the product Φ of the Clifford gates read so far. For a gate with a rotation, append $\exp(-i(\theta/2)\,\Phi^\dagger W \Phi)$ to a list, with θ taken in $(-\pi/4, \pi/4]$, and then multiply Φ by C. Then (i) the circuit equals the final frame Φ applied after the listed rotations; (ii) for any listed rotation $\exp(-i\varphi P)$ and integer k, replacing it by $\exp(-i(\varphi - k\pi/4)P)$, replacing each later axis Q by $E^\dagger Q E$, where $E = \exp(-ik\pi/4\ P)$, and replacing Φ by ΦE, preserves (i); (iii) if such replacements bring the frame to a relabelling of wires, the list equals the circuit up to that relabelling.

Proof. For a rotation R read when the frame is Φ, $R\Phi = \Phi(\Phi^\dagger R \Phi)$, and $\Phi^\dagger R \Phi$ is the rotation by the same angle about $\Phi^\dagger W \Phi$, which is a Pauli string with a sign because Φ is Clifford. Induction on the gates gives (i). For (ii), $\exp(-i\varphi P) = \exp(-i(\varphi - k\pi/4)P)\ E$, and E is Clifford; using $R_Q E = E(E^\dagger R_Q E)$ for each later rotation R_Q conjugates its axis by $E^\dagger$ on the left and E on the right and leaves E next to the frame. (iii) is (i) after (ii). ■

The smallest angle is the right choice for a rotation $\exp(-i\varphi P)$ with $|\varphi| < \pi/8$, which covers the moving strings of a partial swap whenever $|\beta| < \pi$. A cost rotation can be larger. Its surplus quarter turns then appear at the end as a diagonal Clifford frame. Such a frame is a product of quarter turns about Z strings on one and two wires. The pass reads these off the frame and returns each to a diagonal rotation with the same string, by (ii); if no such rotation exists, or the frame is still not a relabelling of wires, the circuit is declined. Diagonal angles recovered in this way are fixed modulo π, which changes the cost on every valid word by a multiple of 2π and a constant, and so changes nothing but a global phase.

7 Using what the parts know about each other

Sections 4 to 6 compress each part on its own: each pair of modes, each slot, each row. This section uses knowledge shared between parts, in three steps of increasing cost. All three are exact.

History register. Take a slot register with distinct seed labels and certified exchanges $S_1, \ldots, S_t$ applied in list order with angles $\theta_1, \ldots, \theta_t$ from the basis seed. Suppose every diagonal generator precedes the first exchange. By Proposition 3 the capacity is at most t bits, whatever the size of the register. Draw one edge between the two slots of each exchange.

Theorem 17 (history register). If the edges form a forest, the 2^t orderings obtained by applying subsets of the exchanges in list order are distinct, and the final state is, up to the global phase of the diagonal generators,

$$\Sigma_h \, a_1(h_1) \cdots a_t(h_t) \, |\pi_h(s)\rangle, \qquad a_b(0) = \cos\theta_b, \quad a_b(1) = -i \sin\theta_b, \tag{8}$$

where h runs over the t-bit strings and $\pi_h(s)$ is the seed after the exchanges with $h_b = 1$. The map from $\pi_h(s)$ to h is an encoding of the reached words on t wires. On those wires the circuit is one rotation $\exp(-i\theta_b X)$ on each wire from the all-zero input, with no two-qubit gate. Measuring h and applying the selected exchanges to the seed classically reproduces the physical output law.

Proof. Expanding every factor as $\cos(\theta_b)\, I - i \sin(\theta_b)\, S_b$ gives (8). For distinctness, fix h. The chosen edges form a sub-forest. The exchanges on the edges of one tree, each used once and in any order, compose to a single cycle through all the slots of the tree, and exchanges on different trees act on different slots. So the cycles of the composed permutation are the trees of the sub-forest. An edge of the forest was chosen exactly when its two ends lie in one cycle, because an edge that was not chosen and joined two slots of one tree would close a loop. Hence the permutation determines h, and distinct seed labels make the words distinct. The rotation $\exp(-i\theta X)$ sends the zero state to $\cos(\theta)|0\rangle - i \sin(\theta)|1\rangle$, which is the factor a_b. ■

The forest test reads only the generator list. It fails when a pair of slots is used twice or the edges close a loop, and then different subsets can reach the same word and interfere; two controls in Appendix A show this. If a diagonal generator follows an exchange, the t wires still suffice by Proposition 3, but the phase becomes a table on the history wires, and that case is not implemented. From the uniform start the capacity is already that of the orbit, and the theorem gives nothing.

Cross-row check. On the label register of Theorem 11 with m slots holding the labels $0, \ldots, m-1$, the bitwise XOR of all the slots is the same word $c = 0 \oplus 1 \oplus \ldots \oplus (m-1)$ on every valid word. This is knowledge shared by all the rows, and it is linear.

Proposition 18 (cross-row check). The projection that forgets the last slot satisfies (2) on the valid words. It is a sketch of width $(m-1)w$. Under it, exchanges and tables on stored slots are unchanged. The exchange of the last stored slot with the implied slot becomes the involution

$$l_{m-2} \mapsto l_{m-2} \oplus \tau, \qquad \tau = c \oplus l_0 \oplus \ldots \oplus l_{m-3}, \tag{9}$$

and a table that involves the implied slot becomes a function of the word $c \oplus l_0 \oplus \ldots \oplus l_{m-2}$.

Proof. Two valid words that agree on the first $m-1$ slots agree on the last, because the last is c plus the XOR of the others. So no nonzero difference of two valid words is supported on the last slot alone, which is (2). After the exchange, slot $m-2$ holds the old implied label $c \oplus l_0 \oplus \ldots \oplus l_{m-2}$, which is $l_{m-2} \oplus \tau$. The statement about tables is the diagonal rule of Section 3. ■

Lowering the cross-row register. Two blocks change. For the tables, XOR slots 1, …, m − 2 into slot 0 and apply X where c has a one. Slot 0 then holds the implied label. Evaluate the tables of the last two edges on slots m − 2 and 0 by the phase rule of Section 8, and undo the XOR network. This adds 2(m − 2)w CNOTs. For the exchange (9), apply Hadamard gates to slot m − 2. In that basis the involution is diagonal with sign $(-1)^{q}$, where q is the parity of the bitwise AND of slot m − 2 with τ. For each bit, gather the bit of τ with a ladder of m − 3 CNOTs, write the AND into a clean auxiliary wire with a relative-phase Toffoli, apply diag(1, $e^{2i\beta}$) to the auxiliary, and undo. The compute phases cancel as in Section 11. The block costs 2w(m − 3) + 6w = 2wm CNOTs, against 16w for the exchange block of Section 8.

The uniform state can be prepared on the label register as in Section 8 and the last slot then cleared by (m − 1)w CNOTs and released, since it equals c plus the XOR of the others. The saving in wires therefore applies to the evolution. A preparation native to the shorter register was not built.

Conditional register. The remaining shared knowledge is not linear. Let position k store the rank of its city among the cities not used at positions 0, …, k − 1. This is the Lehmer code of the ordering. Position k then takes m − k values whatever the earlier positions hold, and the last position takes one value and needs no wire.

Proposition 19 (conditional register). The Lehmer code is an encoding of the m! orderings on Σ_k $\lceil \log_2(m-k) \rceil$ wires, with k from 0 to m − 2. An exchange of positions j and j + 1 changes only the digits (a, b) at those positions, to (b + 1, a) if $b \geq a$ and to (b, a − 1) otherwise. A table on the cities at positions j and j + 1 becomes a function of the digits at positions 0, …, j + 1.

Proof. The code is a bijection between orderings and digit strings. The cities before position j, and the set of cities left after position j + 1, are unchanged by the exchange, so only two digits move; the rule is the effect of removing the same two cities from the unused list in the other order. The city at a position is determined by its digit together with all earlier digits. ■

By Theorem 6 the register supports an exact circuit. Its width is 3, 5, 8, 11 and 14 wires for four to eight cities, against floors of 3, 5, 7, 10 and 13. Packing the digits into a single integer reaches the floor exactly, and such a register has been used for variational routing [18]; we keep one field for each digit so that an exchange touches two fields. The price is in the last sentence of the proposition: the cost formula is no longer a sum of tables on neighbouring slots. Section 9 measures it.

What local reading costs. The label register lets the city at each position be read from that position's own wires. The conditional register does not: a digit means nothing without the digits before it. Say that a position is *readable* from a set of wires if, on valid words, the city at that position is a function of those wires alone. The next statement prices readability in wires.

Proposition 20 (readable positions cost wires). Let an exact register of r wires hold the m! orderings, and suppose q positions are readable from q sets of wires, no two of which share a wire. Then

$$r \geq \log_2 m! + K_q, \qquad K_q = q \log_2 m - \log_2(m!/(m-q)!),$$

where K_q is what the cities at q positions know about each other when the ordering is uniform. If every position is readable from at most t wires and no wire is read by more than u positions, then q can be taken to be at least $m/(1 + t(u-1))$.

Proof. Give the valid words the uniform law. The total correlation of the r wires, the sum of their entropies minus their joint entropy, is at most $r - \log_2 m!$. Total correlation is the relative entropy between a joint law and the product of its marginals, so it does not grow when wires are grouped into

blocks, when blocks are discarded, or when each block is replaced by a function of itself [17]. Group the wires into the q reading sets, discard the rest, and replace each set by the city it determines. Each city is uniform on m values and the q cities together are uniform on the m!/(m − q)! tuples of distinct cities, so their total correlation is K_q, and $K_q \le r - \log_2 m!$. For the last statement, call two positions in conflict when their reading sets share a wire. A position is in conflict with at most t(u − 1) others, so choosing positions greedily leaves at least m/(1 + t(u − 1)) of them with disjoint sets. ■

The proposition interpolates between the two floors of Section 2. With q = 1 it is the floor of Proposition 1, and with q = m it is the local floor m $\log_2$ m of Proposition 2. In between it charges K_q, the knowledge that q separately readable positions cannot share. Since $K_2 = \log_2(m/(m - 1))$, a register within that margin of the floor, 0.22 bits at eight cities, has no two positions readable from separate wires: every two reading sets overlap. A counting argument with the inequality of arithmetic and geometric means gives the same bound without entropies.

A family that meets the bound. Store the first a positions as labels and every later position as its rank among the cities not yet used, the last position being implied. With a = 0 this is the conditional register and with a = m the label register. The first a positions are readable from their own wires, so q = a, and the width is a⌈$\log_2$ m⌉ plus ⌈$\log_2(m - k)$⌉ for each later position k. When m is a power of two the family meets the bound with equality for the larger a: at m = 4 for a = 2, 3 and 4, and at m = 8 for a = 6, 7 and 8. Otherwise it exceeds the bound by the rounding of each field to whole wires, by at most 1.7 wires at m = 7. Section 9 costs the family.

Relation to data structures. Storing a permutation in close to $\log_2 m!$ bits while answering queries for single entries is a studied problem of succinct data structures [35, 36]. A recent lower bound shows that when each entry must be answered from a few cells fixed in advance, with overlaps allowed, the store needs polynomially many bits above $\log_2 m!$ [37]. That is the asymptotic and far more general form of the statement here. Proposition 20 is elementary, holds with exact constants at the sizes of this paper, and covers disjoint reading sets only. Encodings that interpolate between a compact and a local register for routing were constructed in [15]. The proposition bounds every such interpolation from below.

Redundancy. An exact register of r wires for a set O of words has redundancy $R = r - \log_2|O|$. When every wire takes both values on O, this is the quantity K of Section 2 with the single wires as parts: what the wires know about each other. Compression removes it. It is also what makes a faulty run visible, because a word outside the code can only come from a fault. Discarding such runs is established practice for one-hot and particle-conserving circuits [27, 28]. The two statements below say how much detection a register carries and how much of it is useful.

Proposition 21 (redundancy and detection). (i) A uniformly random word of the register is valid with probability 2^{-R}. (ii) If every pattern of at most t flipped wires turns each valid word into an invalid one, then $R \ge t$. (iii) For the routing registers on $m \ge 3$ positions, the smallest number of flipped wires that turns one valid word into another is 4 for the one-hot register, 2 for the label register, and 1 for the cross-row and conditional registers.

Proof. (i) counts words. (ii) is the Singleton bound [29]: if all patterns of t flips are detected, deleting any t wires leaves the valid words distinct, so there are at most 2^{r-t} of them. For (iii), two permutation matrices differ in at least two rows and two one-hot rows differ in two wires. Two orderings differ in at least two slots, and exchanging the labels 0 and 1 changes one wire in each. In the cross-row register, replacing a stored label by the label of the dropped slot gives another valid word, and the two labels can differ in one wire. In the conditional register every digit string in range is valid. ■

The first statement reads K as detection: each wire that compression removes doubles the chance that a scrambled run passes for a valid one. The second prices certain detection at one bit of redundancy for each flip. A register at the floor detects nothing with certainty.

Yield under faults. Let a run be free of faults with probability F and suppose, as the simplest model, that every other run returns a uniformly random word. Valid outcomes are kept. Write P for the ideal law on O and U for the uniform law, and let $x = (1 - F)2^{-R}$ be the chance that a run is faulty and kept. Call the *yield* Y the kept fraction times the deficit of the kept outcomes. In the relabelling experiment of Section 9 the yield is the information that one shot, kept or not, carries about the hidden names, since scrambled words carry none.

Proposition 22 (useful redundancy). In this model the kept outcomes follow the law $\lambda P + (1 - \lambda)U$ with

$$\lambda = F / (F + x) = 1 / (1 + 2^{-\Delta}), \qquad \Delta = R - R_u, \qquad R_u = \log_2((1 - F)/F),$$

and $Y = (F/\lambda)\, D(\lambda P + (1 - \lambda)U \,\|\, U)$. Hence (i) $Y \le F\, D(P\|U)$, with equality in the limit of large R; (ii) at fixed F the yield does not fall when R rises, and $\partial Y/\partial x = -D(U \,\|\, \lambda P + (1 - \lambda)U)$; (iii) for R well below R_u, $Y \approx F\, 2^{\Delta}\, \chi^2/(2 \ln 2)$, where $\chi^2 = \Sigma_o\, (P(o) - U(o))^2/U(o)$.

Proof. A kept outcome is free of faults with probability $F/(F + x)$, and the kept fraction is $F + x = F/\lambda$. (i) is the convexity of the divergence in its first argument. For (ii) write $Y = \Sigma_o\, \mu_o \log_2(\mu_o/\nu_o)$ with $\mu = FP + xU$ and $\nu = (F + x)U$. Its derivative in x is $\Sigma_o\, U(o) \log_2(\mu_o/\nu_o)$, the other terms cancelling, and x falls as R rises. (iii) is the second-order expansion of the divergence about U, with $\lambda \approx 2^{\Delta}$. ■

The yield depends on redundancy only through the excess Δ. We call R_u the *useful redundancy*: a register with a few bits more loses almost nothing to undetected faults, and a register with fewer loses a factor of two in yield for every bit it is short. Redundancy beyond that is paid for in wires and, when it lengthens the circuit, in F. With G two-qubit gates that each fail with probability ε, $F = (1 - \varepsilon)^G$. Real faults are not scrambles, and a fault that changes only phases leaves the word valid. Section 9 tests the formula against faults at the level of gates.

8 Preparing and lowering the accepted circuits

Uniform preparation. Start with slots containing 0, …, m − 1. Suppose the first j slots already hold each ordering of 0, …, j − 1 with equal positive amplitude. Introduce label j in slot j, prepare a coherent uniform choice of k ∈ {0, …, j} and swap slots k, j. Each resulting ordering has one predecessor: locate j, undo that exchange and delete j. The mapping from predecessor and choice to output is bijective, so all (j + 1)! outputs have amplitude $1/\sqrt{((j + 1)!)}$. The location of j recovers the choice, allowing the choice register to be cleared coherently. Induction proves uniformity. A reordered valid seed has the same uniform permutation orbit. The procedure is a quantised Fisher–Yates shuffle [13].

The implementation encodes the choice as a single excitation on j + 1 wires. With h possible positions left, leave amplitude $1/\sqrt{h}$ at the current position and transfer $\sqrt{((h - 1)/h)}$ to the next. A controlled $R_y(2 \arccos(1/\sqrt{h}))$ on the empty next wire followed by a CNOT from next to current implements the split. For j a power of two, its unique highest bit across slots can hold this excitation directly. Other steps clear each choice flag by testing only the set bits of j: among integers at most j, only j contains all those set bits. Bits above the active label width need no swap. These details affect the gate counts.

Exchange exponentiation. With a clean auxiliary wire let W = (H ⊗ I) controlled(S) (H ⊗ I), where H here denotes the Hadamard gate. For an eigenstate of S with eigenvalue λ = ±1, W writes (1 − λ)/2 to the auxiliary wire. Apply diag(1, e^{2it}) there and undo W. The auxiliary returns to zero; the data acquire $e^{it}e^{-it\lambda}$. Thus the block implements exp(−itS) up to a known global phase. Each controlled register exchange uses w controlled wire swaps. The chosen decomposition costs two CNOTs and one exact Toffoli per wire swap. Charging six CNOTs per Toffoli, the complete block costs 16w CNOTs.

Diagonal phases. A term cZ^z contributes $\exp[-itc(-1)^{z \cdot b}]$ to word b. Compute the parity z·b into a participating target wire, apply diag(1, e^{2itc}), and undo the parity network. This differs from the desired phase by the same factor e^{itc} on every word. Combine terms sharing the highest set bit as pivot. Order their remaining masks by inverse binary Gray rank, toggle only the bits where successive masks differ, and restore the target at the end. The zero mask contributes only global phase. Section 4 uses the usual single-string parity ladder after local basis changes.

9 Reconstructing the routing benchmark

Use n cities including a fixed depot and m = n − 1 data slots. For each tested size seed NumPy's default generator with 9018 + n. Draw an n × n integer array uniformly from 1 through 100, keep its strict upper triangle and add the transpose. This gives a symmetric complete cost table D; the experiment imposes no triangle inequality. Label the depot 0 in D and city a + 1 by a in a slot. For ordering p prepend and append the depot and sum its n edge costs to obtain L(p).

Prepare the uniform permutation state. First apply the phase $\exp[-0.02iL(p)]$. Then apply rotations $\exp(-i\beta S_{j,j+1})$ on successive adjacent slot pairs from left to right, with $\beta = -0.5$. Build the supplied phase generator from local distance tables padded by zero for invalid codes. A table f on k bits has coefficient $2^{-k}\,\Sigma_b\, f(b)(-1)^{a\cdot b}$ on mask Z^a. Add all edge contributions with identical masks. Each supplied exchange generator is expanded using (6). The selector receives these ordered groups, wire masks, seed and preparation convention. It receives no slot map.

Independent amplitude calculation. Enumerate tours only in the verifier. Set $A_p = e^{-0.02iL(p)}/\sqrt{(m!)}$. For edge j, update every entry simultaneously using $A'_p = \cos(\beta)\, A_p - i \sin(\beta)\, A_q$, where $q = S_j p$ is the ordering p with slots j and j + 1 exchanged. Compare the decoded circuit amplitude vector with this reference after aligning global phase by their inner product. Check the overlap magnitude and support, including clean workspace. On 13 original maps (three each at four through seven cities and one at eight), the maximum absolute discrepancy was below 3.8×10^{-16}. Five additional tests scrambled the physical wire order. Splitting all exchange generators into individual Pauli terms prevented the permutation branch from accepting the uniform-input instance.

Cities	Shared preparation	Selector total	Qiskit best tested	TKET best tested
4	25	115	107	**85**
5	106	240	220	**205**
6	182	**604**	852	809
7	353	**879**	1,215	1,108
8	557	**1,187**	1,587	1,495

Table 1. Whole-circuit logical CNOT counts. At eight cities, m = 7 and w = 3. The total decomposes as 557 preparation, 342 phase and 6 × 48 = 288 mixing CNOTs. The saving over TKET is (1495 − 1187)/1495 = 20.6%. Four further eight-city maps (seeds 401–404) gave 1,187 each, versus 1,481–1,495 for greedy TKET. At twelve cities the selector counted 4,951 and Qiskit Rustiq 8,435; this is a resource count only.

Starting from the one-hot description. We also wrote the same instances in the one-hot form of Section 6: m^2 wires, the diagonal cost built from products of occupation numbers with the depot terms, and one generator (7) for each pair of neighbouring positions, all as Pauli dictionaries with exact rational coefficients. The front end received these ordered groups, the seed and the preparation convention. It received no row map and no city labels.

For 4, 5, 6, 7, 8 and 12 cities the output of the front end equals the compact program specified at the start of this section, built independently from that specification: the same number of wires, the same exchange dictionaries, and the same phase coefficient on every mask, compared as exact rationals. The input of the selector is thereby reproduced, and the circuits and counts of Table 1 apply to the one-hot starting point without change.

Three further checks do not rely on that comparison. First, for every valid word at four to eight cities, up to 5,040 words, each one-hot generator acts on the word as its transferred generator acts on the sketch of the word. This holds with the canonical wire order and with scrambled wires and seeds. Second, at four and five cities we evolved the full one-hot register of 9 and 16 wires with no restriction to valid words, from the uniform state and from a basis seed. No weight appeared outside the valid words. The amplitudes matched the tour recurrence above within 8.1×10^{-16} and the separately evolved label register within 1.3×10^{-15}. Third, five controls behaved as intended. A flipped coefficient sign, mixers split into single Pauli strings, a seed with two cities at one position and a seed repeating a city were declined. An added term acting only on invalid words was accepted with unchanged output. The split list is the input of the flat pass, reported next, which accepts it.

Cities	Wires, one-hot → label	One-hot, Qiskit best tested	One-hot, TKET best tested	Label, Qiskit	Label, TKET	Label, selector
4	9 → 6	104	108	82	**60**	90
5	16 → 8	306	364	114	**99**	134
6	25 → 15	672	869	670	627	**422**
7	36 → 18	1,250	1,727	862	755	**526**
8	49 → 21	2,088	3,102	1,030	938	**630**

Table 2. Logical CNOT counts for the evolution only, with the same calculation written two ways. The one-hot columns compile the one-hot Pauli strings directly, in the order of Lemma 12: Qiskit with default and Rustiq synthesis, TKET with PauliSimp and GreedyPauliSimp, settings as in Appendix A. Default synthesis and GreedyPauliSimp won at every size. The label columns are Table 1 with the shared preparation removed. By Lemma 12 every column implements the same evolution on valid words. Preparation is excluded because the one-hot preparation of a uniform state over permutation matrices [12] was not compiled here. The selector column uses one clean auxiliary wire. At eight cities the label circuit uses 28 fewer data wires and 630/2088 = 30.2% of the best one-hot count.

At four and five cities the gate lists returned by both SDKs for the one-hot input, run from a basis seed, matched the tour recurrence within 4.6×10^{-15} after removal of a global phase. For larger sizes the one-hot entries are counts.

The table separates two effects. The change of representation, from one-hot rows to labels, removes 28 wires at eight cities and by itself brings the best SDK count from 2,088 to 938. The exchange-aware lowering of Section 8 then brings 938 to 630. At four and five cities the first effect is present and the second is absent: the SDKs compile the small label program better than the selector does.

Starting from a flat list. For 4, 5, 6, 7, 8 and 12 cities the routing circuit was written as an ordered list of 69, 196, 425, 786, 1,309 and 5,621 single rotations, with exact rational angles and no boundaries. The pass regrouped them into 7, 19, 41, 76, 127 and 551 terms, recovered the grid, and returned a compact program equal to the one specified at the start of this section in every coefficient and every step time, apart from the identity term, which is a global phase. The twelve-city list took about four seconds.

At four to eight cities, ten lists at each size were reordered by exchanging commuting neighbours at random until more than nine positions in ten had changed. All fifty were recovered, with the same output. Five more at each size were drawn at random among all orders that keep every non-commuting pair in its original order; their regrouped lists were identical to that of the written order,

as Proposition 13(iv) requires. Lists in which the partial swaps of each position pair were permuted, an order that agrees with the original only on valid words, were recovered too. At four and five cities a reordered list was applied rotation by rotation to the full one-hot register. It matched the tour recurrence within 2.2×10^{-15} after removal of a global phase, with a norm of at most 1.2×10^{-16} outside the valid words. Six controls were declined: a partial swap with one string removed, with one sign flipped, removed entirely, or at twice the angle; a cost string that does not commute, placed inside a position pair; and a seed with two cities at one position.

Starting from compiled circuits. The circuits behind the one-hot columns of Table 2 were lifted by Proposition 16: for each of four to eight cities, the gate lists returned by Qiskit with default and Rustiq synthesis and by TKET with PauliSimp and GreedyPauliSimp, in their one-wire-gate and CNOT bases. At eight cities they have 2,088, 2,262, 4,288 and 3,102 CNOTs on 49 wires. In all twenty cases lifting returned as many rotations as the source list, 1,309 at eight cities, with the same moving strings. Between 9 quarter turns at four cities and 121 at eight were returned from the frame to diagonal rotations. The GreedyPauliSimp circuits end in a relabelling of wires, which the frame recovered. The flat pass then produced the label program, on 21 wires at eight cities: mixing generators and step times equal to the reference, and a cost phase equal to γ times the tour length on every tour within 7×10^{-11}, modulo 2π and a constant. At four cities each lifted list was also compared with its gate list as an operator on a random nine-wire state, and the difference was at most 2.2×10^{-15}.

Five controls were run on the five-city Qiskit circuit. An inserted pair of identical CNOTs changed nothing. A removed CNOT, and a one-wire gate replaced by a random unitary, were declined. An added eighth turn about Z, and a small rotation with its angle altered, were accepted and returned a cost function that differs from the reference, as it should. The kits received the one-hot strings of this paper as their input. What the test shows is that the structure survives their synthesis, reordering and merging of one-wire gates, and can be read back from the gates alone.

Two further families. Quadratic assignment places m facilities at m sites with a cost that couples every pair of sites, and on the one-hot grid its mixer can exchange any two sites. Flat lists for three to six facilities, of 99 to 2,226 rotations on 9 to 36 wires with all pairs of sites exchanged in turn, were shuffled as above and went through the routing pass unchanged. They gave 6 to 18 label wires, the exchanges written directly on labels, in an equivalent order, and a cost equal to the assignment cost on every assignment in exact arithmetic. Section 3.2 of [40] constructs the same type of compact mixer for placement: each binary register holds a site label, and a partial swap exchanges two complete registers while preserving valid permutations. Their placement model also allows vacancy tokens. Here the input is a shuffled list of one-hot rotations, from which the compiler recognises the exchanges, constructs the label registers and certifies the assignment costs.

For graph colouring, six flat lists with hops between neighbouring colours or between all pairs of colours went through the hop pass of Proposition 15 and gave one binary label for each vertex: 8 label wires for 12 one-hot wires with three colours on four vertices, 8 for 16 with four colours on four vertices, and 18 for 30 with five colours on six vertices. Where the one-hot register has at most 16 wires, both programs were run on a random superposition of all colourings and agreed within 3.2×10^{-16} in every amplitude. Five controls were declined. Compression saves wires here and not always gates. After GreedyPauliSimp the label circuits had 16 to 56 CNOTs where the one-hot circuits had 37 to 102 when the number of colours is two or four, and 66 to 747 against 53 to 179 when it is three or five.

Capacity ledger. Table 3 applies Section 2 to the routing question from the uniform start and lists the width of four exact registers.

Cities	Capacity (bits)	Rows singly (bits)	One-hot wires	Label wires	Cross-row wires	Conditional wires
4	2.58	4.75	9	6	4	3
5	4.58	8.00	16	8	6	5
6	6.91	11.61	25	15	12	8
7	9.49	15.51	36	18	15	11
8	12.30	19.65	49	21	18	14

Table 3. Capacity of the routing question and the widths of the registers of Sections 6 and 7. The label register is part-local for rows, so by Proposition 2 it cannot go below the row total, rounded up slot by slot. The cross-row and conditional registers decode one row from the others.

Cities	Label, evolution	Cross-row, evolution	Label, phase only	Conditional, phase only
4	90	**76**	26	**6**
5	134	**128**	38	**28**
6	**422**	426	230	**224**
7	**526**	542	**286**	1,728
8	**630**	658	**342**	13,632

Table 4. Logical CNOT counts under the lowering rules of Sections 7 and 8. The evolution columns use one clean auxiliary wire on both sides. Exchanges among stored slots are charged at the 16w block, and the last exchange of the cross-row register uses the 2wm block of Section 7. For the conditional register only the cost formula was lowered, by expanding it directly on the register; its exchanges were not compiled. At eight cities that formula has 13,302 phase terms, against 343 on the label register.

The cross-row program was run from the uniform start at every size at the level of tours, with its phase network executed gate by gate. It matched the tour recurrence within 5.8×10^{-16}, and the in-place workspace returned to its input on every word. The new exchange block was checked as a unitary on all data words with a clean auxiliary at every size, within 2.2×10^{-16}.

Three wires cost 28 CNOTs at eight cities: a register one seventh narrower for 4% more gates. At four and five cities the shorter register is also the cheaper one. The last four wires are a different matter. At eight cities the cost formula alone needs 13,632 CNOTs on the conditional register, more than twenty times the whole label evolution. Capacity is a floor on wires and not a target: at eight cities the two cheap registers sit eight and five wires above it.

Wires against readable positions. Table 5 costs the family of Section 7 at eight cities, from the conditional register in its first row to the label register in its last.

Labels a	Wires	Readable positions q	Bound, bits	Largest reading set, wires	Cost-phase rotations	Cost-phase CNOTs
0 or 1	14	1	12.30	14	13,245	13,614
2	14	2	12.52	14	13,860	13,900
3	14	3	13.01	14	14,277	14,336
4	15	4	13.81	15	28,665	28,672
5	16	5	15.04	16	65,370	65,534
6	18	6	16.84	18	261,783	262,142
7	21	7	19.65	3	343	**342**

Table 5. Registers that store the first a positions as labels and the rest as ranks, at eight cities, m = 7. The bound is that of Proposition 20. The cost formula is expanded table by table on the reading sets, with zero on unused codes, and lowered by the phase network of Section 8. In every row it reproduces the tour length exactly on all 5,040 orderings. The first row is the conditional register; it differs from the 13,632 CNOTs of Table 4 only in the values given to unused codes.

The wires follow the bound row by row, to within rounding. The gates do not trade against them. Each added label makes one more position readable and costs wires, but the cost formula does not shrink. It grows from 13,614 CNOTs to 262,142, and falls to 342 only when the last rank is gone. The reason is the last clause of Proposition 19. A table on two neighbouring positions expands into about as many rotations as there are words on the union of their reading sets, and a rank must be read together with everything stored before it, labels included. The widest reading set therefore grows with the register until no rank is left. On this family the trade is a cliff and not a slope: the intermediate registers pay the wires of local reading and get none of its benefit. A cheap register between 14 and 21 wires would need a small reading set at every position, and the lower bound of [37] says that, asymptotically, a register near the floor cannot have them.

The exchange rule of Proposition 19 was checked again on every ordering and every neighbouring pair for m = 4 to 7. The bound was tested by exhaustive search for reading sets on 1,200 encodings of the 24 orderings of m = 4 into five to eight wires, 900 of them random and 300 linear images of the label register. None had more readable positions than the bound allows, the smallest margin being 0.415 bits, and the same search recovered the reading sets of the family itself.

Shallow circuits. From a basis seed the phase is global and the m − 1 exchanges form a path, so Theorem 17 applies. At eight cities the 21-wire circuit reaches 64 of the 5,040 orderings and equals six single-qubit rotations with classical decoding. The amplitudes matched the tour recurrence within 1.1×10^{-16} at every size. The measurement entropy is (m − 1) times the binary entropy of $\sin^2 \beta$, which is 4.6665 bits at eight cities, and it is the same on every map: from this start the output carries no information about the distances. The benchmark is informative only from the uniform start.

Concentration of the output. Table 6 gives the measurement entropy from the uniform start.

Cities	Capacity (bits)	Measurement entropy (bits)	Entropy deficit (bits)	Shortest tour, circuit	Shortest tour, guess
4	2.58	2.58	0.005	0.373	0.333
5	4.58	4.10	0.483	0.237	0.083
6	6.91	6.21	0.699	0.0521	0.0167
7	9.49	8.64	0.847	0.0265	0.0028
8	12.30	11.28	1.020	0.0036	0.0004

Table 6. Measurement entropy of the routing circuit, computed in the verifier. The deficit is capacity minus measurement entropy. The last two columns give the chance of measuring a shortest tour; each map has two, a tour and its reverse.

The eight-city angle scan found a largest one-layer deficit of 1.067 bits. Repeating the benchmark angles gave 1.435 bits at two layers and 1.330 at three. The scan maximises concentration, which is not the same as maximising the chance of a shortest tour.

Where the concentration goes. Let C be the integer length of the measured tour and S the indicator that C is minimal. For any statistic T of the tour, let P_T and U_T be its distributions under the circuit and under uniform guessing. Factor P(x)/U(x) into the ratio for the class of x and the conditional ratio within the class. Taking the P-weighted logarithm gives the chain rule [17]:

$$D(P\|U) = D(P_T\|U_T) + \Sigma_t\, P_T(t)\, D(P(\cdot|t)\|U(\cdot|t)). \tag{10}$$

Grouping first by length and then by success gives $D(P\|U) \geq D(P_C\|U_C) \geq D(P_S\|U_S)$. At eight cities the three values are 1.02044, 0.64341 and 0.00668 bits. Thus 0.37703 bits distinguish tours of the same length. The last comparison weighs success and failure together, and its small value is compatible with an 8.95-fold boost to a rare success. Relative entropy has no sign, so the success probability should be reported alongside it.

Information about a hidden optimum. A deficit says how concentrated the output is. It does not by itself say what the output is informative about. For that, fix the benchmark map and hide a uniformly random permutation G of the names of its m non-depot cities. The circuit is run on the relabelled cost table. Let X be one decoded output in the new names, and let Y be the shortest tour of the relabelled map with its two directions identified. The observer knows the base map and the sampling protocol and does not know G. Let r be the relabelling that sends each city of a shortest ordering v of the base map to the city in the reflected position, and put M(x) = [P(x) + P(rx)]/2.

Proposition 23 (relabelling experiment). Suppose the base map has exactly two shortest orderings, v and its reverse. Then:

(i) X is uniform on the m! orderings.

(ii) I(G;X) = D(P‖U), the entropy deficit.

(iii) $I(Y;X) = \log_2 m! - H(M)$.

(iv) $I(Y;X) \leq D(P\|U)$, with equality exactly when P(rx) = P(x) for every x.

(v) For k samples drawn with the same hidden G, $I(Y;X_1, \ldots, X_k) \leq k\, D(P\|U)$, so fixing Y with certainty needs at least H(Y)/D(P‖U) samples.

Every exact encoding with exact decoding preserves all of these quantities.

Proof. The preparation is uniform, the costs are relabelled with the map, and slot exchanges commute with a relabelling of cities. Conditional on G = g the output law is therefore $P(g^{-1}x)$. Averaging over G gives (i), and H(X|G) = H(P) gives (ii). Exactly two relabellings produce a given Y, namely g and g∘r, so conditional on Y the output law is a relabelling of M, which gives (iii) by the Shannon identity [17]. Since Y is a function of G, $I(Y;X) \leq I(G;X)$. The difference is H(M) − H(P), which by strict concavity of entropy vanishes exactly when the two halves of M agree. For (v), the samples are independent given G, so $I(G;X_1, \ldots, X_k) \leq k\, I(G;X)$, and Y is a function of G. An exact encoding followed by exact decoding leaves the joint law of G and the decoded samples unchanged. ■

The experiment concerns relabellings of one map at each size. It gives the deficit a meaning: it is the information one sample carries about how the cities were renamed, and the part of it that concerns the optimum is I(Y;X).

Cities	Tour deficit, I(G;X)	Length deficit	Success deficit	I(Y;X)	I(Y;X) per 1,000 CNOTs
4	0.00525	0.00510	0.00492	0.00506	0.04
5	0.48307	0.39993	0.15471	0.40739	1.70
6	0.69927	0.62723	0.03550	0.52437	0.87
7	0.84691	0.66571	0.05254	0.65142	0.74
8	1.02044	0.64341	0.00668	0.68918	0.58

Table 7. Information in one decoded sample, in bits, from the recurrence of this section. The first three columns compare the circuit with uniform guessing on the fixed map, and the first is also I(G;X) by Proposition 23. The fourth is the information about the hidden optimum. At four, five and six cities it was also computed directly from the joint law of Y and X, with the same result. The last column divides it by the whole-circuit selector count of Table 1.

At eight cities $H(Y) = \log_2(7!/2) = 11.2992$ bits and H(Y|X) = 10.6100 bits. A sample is a shortest tour with probability 0.00354966, about once in 282, against 2/5,040 for a uniform guess. By Proposition 23 at least 11.2992/1.02044 = 11.07 samples, and so at least twelve, are needed to fix the optimum with certainty.

Using the information. Table 8 tests whether those bits can be used. A likelihood decoder scores every relabelling g by the sum of $\log P(g^{-1}x)$ over the samples and names the shortest tour of the best one. The obvious alternative keeps the best tour seen, which succeeds only when a shortest tour is itself sampled.

Samples	Likelihood decoder	Best tour seen
4	0.060	0.014
8	0.239	0.028
12	0.498	0.042
16	0.696	0.055
24	0.902	0.082
32	0.957	0.108
48	0.996	0.157
64	0.998	0.204

Table 8. Probability of naming the shortest tour at eight cities. The decoder column is the fraction of 2,000 simulated trials for each sample count, with a standard error of at most 0.011. The last column is exact: one minus the chance that no sample is a shortest tour.

The decoder reaches even odds at twelve samples, the smallest number that Proposition 23 allows for certainty, and 0.90 at twenty-four. Keeping the best tour seen reaches even odds at about 195 samples. Most of the information in a run is therefore carried by tours that are not shortest. The decoder needs the output law P of the base map, which the observer of this experiment has by assumption. It shows that the information is real and usable, and it is no routing method. That samples which miss the optimum still carry information about it is the basis of recursive methods [19, 20]; the decoder measures how much there is in the ideal case. These are ideal probabilities; site geometry, hardware noise and repetition costs are outside this calculation.

Decoding from the samples alone. The decoder of Table 8 uses the output law of the base map. A second decoder receives only a batch of measured tours with a common depot. It treats tours as undirected cycles, includes the two depot edges, and counts the occurrences N(e) of each undirected edge e in the batch. It returns a cycle t that maximises the sum of N(e) over the edges of t, breaking ties uniformly. This is the cycle that disagrees least with the batch: two cycles s and t on n cities differ in 2n − 2|s ∩ t| edges, so maximising the summed overlap minimises the summed difference. Edge frequency in populations of tours is an established idea [21]. No distance, tour length, output law or training map enters the decoder.

Evaluation. The scoring rule was fixed before the runs. The circuit, angles and map generator are those of this section. The original map was tested with 8,192 batches at each size, and 32 further maps, generated with seeds 41000 to 41031, with 1,024 batches each. Only the verifier used the cost table: to draw samples from the exact output law and to check optimality. A tour and its reverse count as one route, and each of the 32 maps has a single shortest route, so a uniform guess succeeds once in 2,520. Keeping the best tour seen needs a cost evaluator, which anyone who built the circuit has. Its success probability is exact: the chance that the batch contains an optimum. The last rule adds the consensus tour to the sampled ones and keeps the shortest.

Samples	Consensus alone	Optimum named though never sampled	Best tour seen	Best seen or consensus	Consensus alone, original map
12	2.68	2.19	4.03	**6.22**	1.71
24	6.23	4.91	7.89	**12.80**	3.36
64	17.69	12.09	19.55	**31.64**	7.95
256	42.62	15.58	56.94	**72.52**	12.73
1,024	61.63	2.29	94.93	**97.22**	10.13
4,096	73.46	0.00	99.98	**99.98**	2.70

Table 9. Chance, in percent, of naming the shortest route at eight cities from the samples. Columns two to five are means over the 32 further maps. The best-seen column is exact. The others are fractions of simulated batches, and the standard error across maps is 4.6 points at 1,024 samples. The fifth column is the fourth plus the third.

What the table shows. Alone, the consensus decoder is the weaker rule on average at every batch size, and it is not consistent. In the limit of many samples its tour is the shortest on 27 of the 32 further maps, and a longer tour on the other five and on the original map, where the consensus has length 288 against 283. On that map its success rises to about 13% at 256 samples and then falls to 2.7% at 4,096, as the wrong answer is estimated more precisely. Its value is as a candidate. In 12.1% of all batches of 64 samples it names the optimum although the batch contains no optimal tour, so adding it to the sampled tours raises the success rate from 19.6% to 31.6%. At 256 samples the gain is from 56.9% to 72.5%. Part of the information that the likelihood decoder extracts with full knowledge of the circuit is therefore available with none.

Identifiability and scope. Two cost tables can have different optima and induce the same law of sampled tours, for example under sampling that ignores the costs. Every sample-only rule then behaves identically on both, so success requires some relation between the sampling bias and the target. Edge consensus assumes that edges which are collectively frequent assemble into a short tour. The experiment scores all 2,520 cycles by enumeration and makes no claim about larger maps, where finding the consensus cycle is itself a routing problem.

Redundancy and yield at eight cities. Table 10 applies Propositions 21 and 22 to the four registers, with the evolution counts of Tables 2 and 4 and $F = (1 - \varepsilon)^G$. The label register has the largest predicted yield at every rate. Its 8.7 bits of redundancy exceed the useful amount R_u until F falls below 0.0024, which at 630 CNOTs means a fault rate of about 1% for each gate. The 28 further bits of the one-hot register are never used: in the formula a 49-wire circuit matches the label register only if it has at most 632 to 635 CNOTs, against the label register's 630. The conditional register, with 1.7 bits and more than twenty times the gates, is last.

Register	Wires	R, bits	Random word valid	Flips always detected	Evolution CNOTs	Yield, ε = 10^{-4}	Yield, ε = 10^{-3}	Yield, ε = 3 × 10^{-3}
One-hot	49	36.70	9.0×10^{-12}	3	2,088	0.828	0.126	0.002
Label	21	8.70	2.4×10^{-3}	1	630	**0.958**	**0.542**	**0.151**
Cross-row	18	5.70	1.9×10^{-2}	0	658	0.954	0.517	0.124
Conditional	14	1.70	0.31	0	above 13,632	below 0.144	0.000	0.000

Table 10. Redundancy R and predicted yield at eight cities, in bits for each shot, by the closed form of Proposition 22 with the ideal law of this section, whose deficit is 1.0204 bits. Wires are data wires. The one-hot count is the best kit result of Table 2, and the conditional count is its cost formula alone, so its yields are upper bounds.

Faults at the level of gates. The model treats a faulty run as a random word. To test it, the evolution was written in CNOTs and one-wire gates for all four registers at five cities and for the label and cross-row registers at six: the lowering of Section 8 for the label register, with each controlled exchange made of Toffoli gates of six CNOTs; the blocks of Section 7 for the cross-row register; the better of the two Qiskit circuits for the one-hot register; and, for the conditional register, Qiskit's generic synthesis of its 32-dimensional unitary. Each circuit without faults reproduced the tour recurrence within 1.5×10^{-15}. After every CNOT, with probability ε, a uniformly random non-identity Pauli operator was applied to its two wires. Preparation and measurement were ideal. A run was kept when its data word was valid and, for the two registers with an auxiliary wire, that wire read zero; the auxiliary wire is counted in R. The kept law was computed exactly without faults, exactly or from 1,500 sampled patterns for one fault, and from 300 to 1,500 sampled patterns for each number of faults from two to six. Larger numbers, which hold less than 4×10^{-4} of the probability at the rates shown, were given the law for six.

Cities	Register	Wires in circuit	CNOTs	R, bits	ε = 10^{-4}	ε = 3 × 10^{-4}	ε = 10^{-3}	ε = 3 × 10^{-3}
5	Label	9	134	4.42	**0.4757** (0.4762)	**0.4612** (0.4628)	**0.4140** (0.4185)	**0.3041** (0.3129)
5	Cross-row	7	128	2.42	0.4749 (0.4753)	0.4590 (0.4601)	0.4077 (0.4105)	0.2908 (0.2948)
5	Conditional	5	423	0.42	0.4480 (0.4434)	0.3874 (0.3757)	0.2398 (0.2151)	0.0672 (0.0444)
5	One-hot	16	306	11.42	0.4680 (0.4685)	0.4394 (0.4407)	0.3522 (0.3556)	0.1878 (0.1925)
6	Label	16	422	9.09	**0.6686** (0.6703)	**0.6114** (0.6159)	**0.4478** (0.4578)	**0.1849** (0.1955)
6	Cross-row	13	426	6.09	0.6662 (0.6695)	0.6050 (0.6136)	0.4331 (0.4517)	0.1669 (0.1850)

Table 11. Yield in bits for each shot under gate faults at rate ε, from the simulation, with the closed form of Proposition 22 in brackets. The ideal deficits are 0.4831 bits at five cities and 0.6993 at six. The largest simulated yield in each column and size is in bold.

What the simulation shows. The closed form is close for the three registers with at least 2.4 bits of redundancy: it exceeds the simulated yield by at most 4.3% up to $\varepsilon = 10^{-3}$ and by at most 10.8% at 3×10^{-3}. It is too high because single faults are not scrambles. A fault that changes only phases leaves the word valid, so a run with one fault is kept with probability 0.22 on the label register at five cities and 0.15 on the one-hot register, where scrambling would give 0.047 and 0.0004; with four faults the kept fractions fall to 0.061 and 0.003. For the conditional register, with 0.42 bits, the formula is too low, by up to a third of the simulated value, because its kept faulty runs still carry information.

The ordering is the same in the simulation and in the formula. The label register has the largest yield at every rate and both sizes. At five cities the cross-row register has fewer wires and fewer CNOTs, 128 against 134, and still yields less, by 0.2% at $\varepsilon = 10^{-4}$ and 4.4% at 3×10^{-3}; what it gave up is redundancy. At six cities the gap at 3×10^{-3} is 9.7%. The one-hot register, with the most redundancy and more than twice the gates of the label register at five cities, is third. A choice by wires alone would take the conditional register and a choice by gates alone the cross-row register at five cities; the yield takes neither.

A second measure gives a different order. The chance that a shot is kept and is a shortest tour is slightly higher for the cross-row register than for the label register at every rate and both sizes: at $\varepsilon = 10^{-3}$, 0.2129 against 0.2099 at five cities and 0.0366 against 0.0360 at six. A kept faulty run still lands on a shortest tour at the chance rate, which is 8.3% and 1.7% at these sizes and 0.04% at eight cities. The yield counts what the kept runs say about the instance, and this measure counts hits, lucky ones included. Which of them matters depends on the decoder.

Larger maps. The recurrence of this section acts on orderings: the cost phase is diagonal and an exchange of positions permutes them. Run directly on the m! orderings, with no wires, it reaches twelve cities and 39.9 million orderings. Table 12 gives the quantities of Tables 6 and 7 at the same angles. The deficit grows by about 0.15 bits for each added city while the capacity grows by 2.3 to 3.5, so one sample resolves a falling share of the question: 10.5% of the capacity at five cities and 6.0% at twelve. The success probability rises from 2.8 to 42 times chance. The number of samples at which the best tour seen is as likely as not to be optimal rises from 3 to about 330,000, while the bound of Proposition 23(v) rises only from 7.4 to 16.1 samples. At eight cities the likelihood decoder reached roughly 50% success at 12 samples using the output law. The bound addresses certain recovery, a different success criterion. Whether a decoder without the output law can do so at larger sizes is the question these numbers pose; the consensus decoder of Table 9 does not settle it.

Cities	Capacity	Deficit	I(Y;X)	Success over chance	Samples for even odds, best seen	Sample bound	Mean deficit, further maps	Mean success over chance, further maps
5	4.58	0.4831	0.4074	2.84	3	7.42	0.584	3.13
6	6.91	0.6993	0.5244	3.13	13	8.45	0.694	4.53
7	9.49	0.8469	0.6514	9.55	26	10.03	0.827	6.58
8	12.30	1.0204	0.6892	8.95	195	11.07	0.947	9.31
9	15.30	1.0658	tie	8.91	784	tie	1.072	14.77
10	18.47	1.1976	0.8483	18.60	6,762	14.59	1.197	19.56
11	21.79	1.4024	1.0076	28.03	44,860	14.83	1.358	37.63
12	25.25	1.5083	1.0424	41.94	329,866	16.08	not run	not run

Table 12. The routing circuit at the benchmark angles, one layer, uniform start, computed on the orderings. Capacity, deficit and I(Y;X) are in bits. Columns two to seven are for the benchmark map of each size. The sample bound is H(Y)/D(P‖U). At nine cities that map has two different shortest routes, so the assumption of Proposition 23 fails and the two entries are omitted. The last two columns are means over 16 further maps for each size, and over 4 at eleven cities.

Angles tuned at each size for the smallest mean tour length gave nearly the same picture. The deficit changed by 0.26 bits at five cities and by at most 0.07 bits from six to eleven, and the tuned angles carried over to the further maps of the same size with mean deficits within 0.12 bits of those in the table.

A thermal control. Among all laws on the orderings with a given entropy, the Gibbs law $G(x) \propto \exp(-bL(x))$, with $b > 0$ chosen to give that entropy, has the smallest mean tour length: for any law Q, $D(Q\|G) \geq 0$ reads $b\,E_Q[L] \geq (\ln 2)\,H(Q) - \ln Z$, with H(Q) in bits, with equality for G. It is therefore a bound on what any sampler, quantum or classical, can do with a given deficit when the aim is a short tour on average. Single-layer circuits of this family are known to give outputs that resemble thermal laws [25], which makes the Gibbs law at the circuit's own deficit a natural control. Table 13 gives it, computed by enumeration.

Cities	Deficit	Circuit mean length	Gibbs mean length	Uniform mean length	Share of the Gibbs gain	Shortest tour, circuit	Shortest tour, Gibbs	I(Y;X), circuit	I(Y;X), Gibbs
5	0.4831	200.5	185.2	242.5	0.734	0.2366	0.2976	0.4074	0.3798
6	0.6993	378.9	373.8	420.0	0.888	0.0521	0.0838	0.5244	0.5367
7	0.8469	317.7	298.8	371.0	0.738	0.0265	0.0396	0.6514	0.7205
8	1.0204	421.5	402.9	480.9	0.761	3.55×10^{-3}	7.59×10^{-3}	0.6892	0.8053
9	1.0658	396.4	376.5	459.0	0.758	8.84×10^{-4}	1.98×10^{-3}	tie	tie
10	1.1976	321.0	293.5	388.2	0.710	1.03×10^{-4}	3.78×10^{-4}	0.8483	0.9224
11	1.4024	499.3	460.3	585.6	0.689	1.55×10^{-5}	7.57×10^{-5}	1.0076	1.1043
12	1.5083	476.1	430.9	569.8	0.675	2.10×10^{-6}	1.20×10^{-5}	1.0424	1.1629

Table 13. The circuit at the benchmark angles against the Gibbs law with the same deficit, in bits, on the benchmark map of each size. The share is the circuit's reduction in mean length below the uniform mean, divided by the Gibbs reduction. The shortest-tour columns are probabilities.

The circuit realises between 0.675 and 0.888 of the Gibbs reduction in mean length, and the share is lowest at the largest sizes. The Gibbs law puts 1.3 to 5.7 times as much probability on the shortest tour, and from six cities on it also carries more information about the hidden optimum. For each bit of deficit, then, the one-layer circuit is a less effective sampler than a thermal one at these sizes. With the tuned angles the share was 0.75 to 0.89 from five to eleven cities. The comparison does not include the cost of drawing thermal samples, which we did not measure, and it concerns one layer.

Changing the mixer angles at fixed measurement entropy

Keep the cost phase fixed and give the six adjacent exchange rotations independent angles β. Write P for the resulting tour distribution, N = 7!, and r for the city relabelling that reverses the optimal ordering. By Proposition 23, total relabelling information is $D = \log_2 N - H(P)$, and information about the unoriented optimum is $J = \log_2 N - H(M)$, with $M(x) = [P(x) + P(rx)]/2$. Varying β changes the physical evolution. This design experiment follows the exact compilation comparisons, in which the target evolution is fixed.

Proposition 24 (the budget of a sample). Under the assumption of Proposition 23, D = J + R, where $R = I(G;X|Y) = H(M) - H(P)$ is the Jensen–Shannon divergence between P and its reflection $x \mapsto P(rx)$. Given the optimum Y there are exactly two possible relabellings, which list the optimal route in opposite directions, so R is the information a sample carries about that direction. It lies between zero and one bit and is zero exactly when P equals its reflection. Along any change of the circuit that keeps the measurement entropy fixed, J rises by exactly the amount that R falls.

Proof. Y is a function of G, so the chain rule [17] gives $I(G;X) = I(Y;X) + I(G;X|Y)$, and the first two terms are D and J by Proposition 23. Their difference is $H(M) - H(P)$. The reflected law has the entropy of P, so this is the entropy of an even mixture minus the mean entropy of its two parts, which is the Jensen–Shannon divergence; it vanishes only for equal parts and is at most the entropy of the mixing weights, one bit. A relabelling g with g(v) equal to Y or its reverse is fixed by which of the two it is, since v lists every city. ■

At the benchmark angles, the 1.0204-bit deficit comprises 0.6892 bits about the shortest unoriented route and 0.3313 bits about orientation (values rounded separately).

Derivative. For any mixer angle, normalisation gives $\Sigma_x\, \partial P(x) = 0$. Differentiating the entropy therefore gives $\partial D = \Sigma_x\, \partial P(x) \log_2[NP(x)]$. The same calculation first gives ∂J in terms of ∂M. Since r is an involution and $M(rx) = M(x)$, changing variables in the reflected half of that sum gives $\partial J = \Sigma_x\, \partial P(x) \log_2[NM(x)]$. These formulas apply at the positive probabilities used here. Differentiating the amplitude recurrence in Section 9 gives $\partial P = 2\,\mathrm{Re}(\bar{A}\, \partial A)$ for each angle, where the bar denotes complex conjugation.

Direction and finite step. Use the Euclidean metric on the six angles in radians. Let $u = \nabla D/\|\nabla D\|$ and $v = \nabla J - u(u\cdot\nabla J)$. When both norms are nonzero, $d = v/\|v\|$ leaves D stationary to first order and gives the largest first-order increase in J among unit tangent directions. Indeed, any such direction w obeys $w\cdot\nabla J = w\cdot v \leq \|v\|$, with equality at d. Starting from $\beta_0 = (-0.5, \ldots, -0.5)$, apply $\beta_1 = \beta_0 + 0.05d + zu$, choosing the root z closest to zero in the numerical search that restores $D(\beta_1) = D(\beta_0)$. The implicit function theorem gives a local correction because the derivative along u is $\|\nabla D\|$.

Original map. The entropy deficit stayed at 1.020442 bits. Information about the hidden optimum rose from 0.689179 to 0.693786 bits per sample. The probability of drawing the shortest route itself fell from 0.35497% to 0.35285%. Thus two circuits with the same measurement entropy and gate pattern can allocate different amounts of information to the identity of the optimum.

Transfer. Freeze d from the original map. On each new map recompute u from its output distribution, project the frozen d perpendicular to this u, normalise, and apply the same finite step and correction. Selecting the update requires the simulated tour probabilities on the new map. Only the evaluator uses that map's optimal tour to score J and check eligibility. Of 32 new eight-city maps, three had additional optimal orderings and fell outside the two-ordering formula. All remaining 29 gained information, by a mean of 0.003238 bits per sample (standard error across maps 0.000287). The largest absolute change in D was below 6.5×10^{-15} bits. Direct success probability increased on 11 maps and fell on average by 0.00304 percentage points.

Control. Random unit directions inside the surface of fixed D, taken with the same step and correction, changed J by −0.0003 bits on average on the original map: of 200 directions 43% raised it, and the largest gain was 0.0040 against 0.0046 for d. On the 29 maps, with 60 directions each, the mean change was −0.0001 and 49% raised it. The frozen direction did better than 94% of the random ones and obtained 76% of the gain of each map's own direction. The transfer is therefore a property of the direction and not of the constraint. By Proposition 24 it lowered R on every one of the 29 maps: it makes the output law more nearly the same in the two directions of the optimal route. Estimation of the direction from finite measurement samples remains open.

What the reallocation is worth. Information about the optimum and hits on it are two uses of one deficit, and here they pull apart. Let s be the probability of a shortest tour. Inside the surface of fixed D, the direction of steepest increase of s has cosine −0.14 with d on the original map, and the corresponding cosine is negative on 18 of the 29 further maps, with mean −0.10. Following a direction with D restored at every step, until the quantity stops rising, gives the laws of Table 14: one for J, one for s, and one for a bound on the error of likelihood decoding, the sum over wrong relabellings g of the twelfth power of $\Sigma_x \sqrt{(P(x)P(g^{-1}x))}$. All four laws have the same deficit, and so the same bound of Proposition 23(v), 11.07 samples.

Law	J, bits	R, bits	Shortest tour	Mean length	Best seen, even odds at	8	12	16	24
Benchmark angles	0.6892	0.3313	0.3550%	421.5	195	0.252	0.479	0.694	0.909
Most information	0.7506	0.2698	0.0767%	468.2	903	0.194	0.414	0.601	0.846
Most hits	0.6813	0.3391	0.3880%	425.8	178	0.246	0.500	0.705	0.903
Smallest error bound	0.6966	0.3238	0.0977%	456.0	709	0.208	0.482	0.673	0.910

Table 14. Four laws of the eight-city circuit with one deficit, 1.020442 bits, on the original map. Shortest tour is the probability of drawing one; chance is 0.0397%. The last four columns are the success of the likelihood decoder of Table 8 from 8, 12, 16 and 24 samples, from 4,000 trials each, with standard error at most 0.008. Best seen is the number of samples at which the best tour seen is as likely as not to be optimal. Each climb stops at a local maximum.

Maximising J locally increased it from 0.6892 to 0.7506 bits per sample. Direct hits fell from 0.3550% to 0.0767%, and the twelve-sample decoder fell from 47.9% to 41.4%. Optimising direct hits instead reduced the number of samples needed for an even chance of seeing an optimum from 195 to 178. The law with the smallest error bound gave 72% fewer direct hits than the benchmark (0.0977% versus 0.3550%), yet twelve-sample decoding success was similar (48.2% versus 47.9%). The common 11.07-sample lower bound concerns certain recovery. It does not determine either the even-odds threshold or the decoder's success at a fixed sample count.

Transfer depended on the size and objective of the update. A single step along the original map's hit-maximising direction raised the direct-hit probability on 13 of 32 further maps. For the endpoint optimised for J, the original deficit could be restored on 28 of 29 eligible maps. J rose on 17 of those 28, while direct-hit probability fell on all 28. On the first twelve evaluated, mean twelve-sample likelihood-decoding success fell from 0.45 to 0.39. The consistent gain across 29 maps applies to the small projected step.

Training on the deficit. For N valid orderings and output law P, the objective $D = \log_2 N - H(P)$ uses the circuit outputs and requires no optimal-tour labels. We trained seven parameters, the cost angle and the six mixer angles, by ascent on D from the benchmark angles, and for comparison by descent on the mean tour length. The likelihood decoder still uses a reference output law. The three settings give different results even on the same eight-city map (Table 15).

Angle selection	Deficit (bits)	Decoded optimum, 12 samples	Direct shortest-tour hit
Benchmark	1.02	49%	0.355%
Train for short tours	1.11	51%	0.332%
Train on deficit	1.51	73%	0.108%

Table 15. The original eight-city map under three settings of the seven parameters. The decoder combines twelve samples and knows the output law. The last column gives the probability that one sample is an optimal tour.

Decoding and transfer. On the original map training raised decoder success by 24 percentage points, while a shortest tour was drawn about 3.3 times less often. On twelve further maps, training the seven parameters separately for each map raised mean twelve-sample decoding success from 45% to 66%, with an improvement on all twelve. With the cost angle held fixed and only the six mixer angles trained, the deficit on the original map reached 1.49 bits and the twelve-map mean reached 64%. The trained parameters also transfer. Applied unchanged to the 29 further maps, those of the original map raised the deficit on all 29, from 0.96 to 1.33 bits on average, and on the twelve decoded maps they raised success from 45% to 67%, again on all twelve. Retraining for each map was not needed.

Estimating the gradient from samples. For one occurrence of $\exp(-i\theta S)$ with $S^2 = I$, shifting θ by $\pm\pi/4$ gives laws P_+ and P_- with $\partial\theta P = P_+ - P_-$. At positive probabilities the entropy derivative is the difference between the two shifted expectations of $-\log_2 P(X)$, evaluated using the unshifted law. Negating it gives the deficit derivative. On the benchmark map the six exact shifted derivatives agreed with independent amplitude differentiation within 5.0×10^{-16} bits per radian in maximum absolute error. A fitted model based on neighbouring cities supplies an approximate surprise score when probabilities are unknown. The reported directional agreement was 0.93 from 24,000 simulated shots. The fitted score introduces approximation error beyond sampling noise.

Training from sampled outputs. When updates to the six mixer angles used simulated samples, with the cost angle fixed, the final deficit was 1.48 bits, against 1.49 for the exact-gradient comparator. On twelve further maps mean decoding success rose from 45% to 62% and improved on every map. Training consumed about 1.4 million simulated shots. Reusing trained settings could distribute that cost over multiple decoding tasks. The experiments do not establish a break-even reuse count or a hardware advantage.

Nine and ten cities. The same training was run on the orderings at nine and ten cities, on maps with one shortest route, from the benchmark angles (Table 16). The mean deficit rose by 0.48 bits at nine cities and by 0.44 at ten. Twelve-sample decoding rose from 36% to 61% at nine cities and from 29% to 49% at ten, and every one of the six maps improved. Direct hits fell by a factor of about three at nine cities and about two at ten. The decoder scores all m! relabellings for every sample, which is what limits the test to ten cities.

Cities	Maps	Parameters	Deficit, bits	I(Y;X), bits	Shortest tour	8	12	16	24
9	4	Benchmark	1.099	0.799	0.0563%	0.138	0.355	0.580	0.869
9	4	Trained on deficit	1.581	1.196	0.0187%	0.295	0.605	0.808	0.956
10	2	Benchmark	1.255	0.868	0.0094%	0.104	0.292	0.560	0.832
10	2	Trained on deficit	1.692	1.239	0.0053%	0.206	0.494	0.764	0.966

Table 16. Deficit training at nine and ten cities, means over the maps. Each map was trained by itself on its eight or nine parameters. The last four columns give the success of the likelihood decoder from that many samples, from 600 trials for each map at nine cities and 250 at ten.

Uniform noise. Mix the output with the uniform law at weight $1 - \lambda$, as in the kept law of Proposition 22. The deficit falls for both settings and their order does not change. At $\lambda = 0.6$ the trained law keeps 0.57 bits against 0.38 and decodes from 24 samples in 49% of trials against 31%, when the decoder knows the noisy law. When it knows only the noiseless law the figures are 41% and 22%. The gradient

of the noisy deficit points along the noiseless one, with cosine 1.0000 at the start for λ from 0.8 to 0.2, and parameters trained on the noisy deficit at λ = 0.2 reach a noiseless deficit of 1.503 bits against 1.514. In this fault model noise costs samples and does not move the target of training. Gate-level faults were not simulated for the trained circuits.

What the decoder must know. The likelihood decoder holds the output law of the map, which here comes from exact simulation. Table 17 replaces it by decoders that know less, on 29 further maps, with parameters trained on the original map and carried over unchanged. The thermal-model decoder knows the distance table in reference names and no output law. It names the relabelling under which the sampled tours are shortest in total, which is the likelihood rule for a Gibbs law at any temperature. The canonical decoder knows neither the table nor the law of the map at hand. It scores a candidate route by a law learned on 40 other maps whose optima are known, each written in coordinates in which its optimum is the first ordering.

Parameters	Mean deficit, bits	Shortest tour	Best tour seen	Edge consensus	Canonical law	Thermal model	Output law
Benchmark	0.964	0.355%	0.083 / 0.160	0.067 / 0.148	0.106 / 0.212	0.357 / 0.725	0.879 / 0.989
Trained on length	1.075	0.368%	0.088 / 0.170	0.077 / 0.164	0.125 / 0.226	**0.410 / 0.791**	0.895 / 0.989
Trained on deficit	1.334	0.211%	0.049 / 0.094	0.017 / 0.036	**0.157 / 0.266**	0.062 / 0.177	**0.960 / 0.997**

Table 17. Chance of naming the shortest route of an eight-city map from 24 and from 48 samples, for five decoders that know progressively more, averaged over 29 further maps with 400 trials each. Parameters were trained on the original map and carried over. Bold marks the best setting for the three decoders that hold a law or a model.

The answer depends on the pairing of objective and decoder. Deficit training helps the decoders that hold a law: the output law itself, 0.88 to 0.96 at 24 samples, and the canonical law, 0.11 to 0.16. It harms the others. Best seen falls from 0.083 to 0.049, edge consensus from 0.067 to 0.017, and the thermal model from 0.36 to 0.06, because the added concentration no longer follows tour length. Training on length does the opposite. It leaves the deficit near its starting value and makes the output more thermal, and the thermal-model decoder then names the hidden optimum from 48 samples in 79% of trials, where the best tour seen does so in 17% and the output law in 99%. On the deficit-trained parameters the canonical decoder names an optimum that was never sampled in 15% of trials at 24 samples.

Two conclusions follow. The reference law cannot be dropped from deficit training: without a law the deficit-trained circuit is the worst of the three settings. And a decoder that needs no simulation does exist, but it pairs with the usual length objective. The rule is to train the quantity that the decoder models. The thermal-model decoder is defined inside the relabelling experiment; with the names known, the distance table settles an eight-city map by enumeration. Its use is to show how much of the decoding power of Table 8 survives without the output law: on the length-trained parameters its success is 46% of that of the output law at 24 samples and 80% at 48.

Pruning at equal gate cost. At a budget of 176 CNOTs for the cost phase, the total variation distance from the original output was 0.277 for coefficient-size ranking, 0.252 for Fisher sensitivity ranking and 0.215 for greedy pruning that accounted for joint effects. On twelve further maps the mean distances

were 0.276 for coefficient size and 0.277 for sensitivity ranking, so the advantage of sensitivity ranking did not transfer. Coefficient size can be a good proxy for sensitivity in this circuit, although the finite deletion effect also depends on the state and interference. The 176-CNOT ceiling is common to all three methods. Size ranking used 176 CNOTs and Fisher and greedy pruning each used 174 on the original map.

Exact completion of unused codes. We apply the diagonal-completion principle of [40] to the routing cost table. Seven labels occupy a three-bit register, leaving an eighth code free in each row. No valid ordering uses that code, and none puts the same label in two neighbouring slots. The diagonal phase may therefore assign arbitrary values to every table entry in which either slot holds the unused code or both slots hold the same label: 22 of the 64 entries of a table on two slots, and one of the eight entries of each end table. If two cost functions agree on every valid ordering, their diagonal phase operators agree on every basis vector in the valid subspace and hence on every superposition in it. The mixer preserves that subspace, so induction over layers preserves the ideal output law. Changing the free entries changes the Walsh coefficients and can make some vanish. On the eight-city instance, a completion chosen to cancel Walsh coefficients removed 20 of them from each of the six pair tables and one from each end table. It reduced the cost phase from 343 rotations to 199 (42.0%) and from 342 CNOTs to 258 (24.6%), and the completed formula agreed with the tour length on every valid ordering to 5×10^{-13}. These are phase-block counts. Exactness on valid inputs leaves open how faults that reach unused codes behave.

Choosing circuit depth under noise. For each candidate depth let G count the CNOTs, P be the ideal tour law, N the number of valid tours, and W the number of possible measured words including the flag. With independent per-CNOT fault probability ε, the scrambling model of Proposition 22 assigns accepted mass $\mu_i = FP_i + (1 - F)/W$ to tour i, where $F = (1 - \varepsilon)^G$. Write $A = \Sigma_i \mu_i$. Choose the depth maximising $Y = \Sigma_i \mu_i \log_2(N\mu_i/A)$, in bits per attempted shot. Computing this prediction requires the ideal law at each candidate depth. Earlier work also treats noise-aware depth selection for QAOA [31]. Here the objective includes the probability of discarding a run.

When another layer pays. For the large-redundancy approximation $Y \approx FD$, $D = \log_2 N - H(P)$, the next layer improves the score precisely when $\ln(D_{next}/D_{now}) > -\Delta G \ln(1 - \varepsilon)$, provided both deficits are positive. Evaluate the score directly when either is zero. A later layer may recover from a local decline. A stronger stopping test uses Proposition 22 at the next gate count with P concentrated on one tour. This upper-bounds all future yields in the scrambling model if gate counts never fall. To see this, at fixed F the accepted fraction is constant and the divergence is convex in P. Its maximum is attained at a point mass, with the same value for every tour by symmetry. Reducing F mixes the accepted mass with uniform mass of zero yield. Convexity of the mass-weighted divergence implies that the maximum cannot increase. Stop when the best yield already seen equals or exceeds this upper bound.

Four-city depth test. We used 33 symmetric maps, three CNOT error rates (0.01%, 0.1%, 0.3%), and one to eight layers at fixed angles. Six data wires encode six valid orderings. A seventh wire must read zero for a shot to count. After each CNOT, exact density-matrix evolution averages a uniformly chosen nonidentity two-wire Pauli fault at the specified rate. Preparation and measurement are ideal. We scored the accepted tour distribution per attempted shot. Each of the 99 comparisons uses the best of eight depths under this gate-level channel as its reference. The gate decomposition used here costs 96 CNOTs per layer on all 33 maps.

Depth choices. The full yield formula selected the reference depth in 95/99 comparisons. Its mean percentage loss against the reference yield was 0.033%, and its greatest loss was 1.79%. Scanning the simpler FD score gave 93/99 matches and 0.119% mean loss. Stopping at its first decline gave 87/99

and 1.208%. Choosing the largest noiseless deficit gave 63/99 and 13.995%. At 0.3% error the point-mass bound certified a stopping point by layer eight for 31/33 maps under the scrambling model. At the two lower rates it certified none within eight layers. These results concern one fixed-angle circuit family and an information objective. The test did not measure shortest-tour recovery or device performance.

Best depths and baseline. At 0.01% error the numbers of maps whose best depth was one through eight were respectively 12, 12, 1, 1, 1, 5, 1 and 0. At 0.1% error, one layer won on 21 maps and two layers on 12. At 0.3%, the corresponding counts were 25 and 8. Always choosing one layer therefore matched the best depth in 58/99 cases, with 11.90% mean relative yield loss. Of the 99 comparisons, 90 favoured one or two layers. The full formula selected the correct depth in all nine deeper cases, all at the lowest error rate.

Role of redundancy. Only six of 128 measured words pass the validity check. The full formula and FD selected the same depth in 97/99 comparisons. Their two disagreements both occurred at 0.3% error, where the full formula correctly selected one layer. The incremental benefit observed here is two decisions. Its size in registers with less redundancy, or in families where deeper circuits routinely win, remains open.

10 Transfer to an independently generated fermion model

We froze the selector and used OpenFermion 1.8.1 to generate a spinful Fermi–Hubbard Hamiltonian on open rectangular lattices [7]. With hopping amplitude one and on-site interaction four, our convention is

$$H = -\Sigma_{\langle i,j\rangle,\sigma} \left(a^{\dagger}_{i\sigma} a_{j\sigma} + a^{\dagger}_{j\sigma} a_{i\sigma} \right) + 4 \Sigma_i n_{i\uparrow} n_{i\downarrow}.$$

Apply the Jordan–Wigner transform. Drop the identity term as a common phase. Supply one singleton Pauli generator for each remaining term, preserving the library's order, with step size 0.23. At site (i, j) initialise one occupied mode with bit index $2(jn_x + i) + ((i + j) \bmod 2)$, where the parity is taken on i + j. This is one specified first-order product step. Equivalence here concerns that ordered circuit.

Why two qubits disappear. Each hopping Pauli string flips two mode bits of the same spin. Occupation terms have no flip mask. In a connected graph with k vertices, the binary span of edge incidence vectors is exactly the even-weight subspace: every edge has even weight, and paths from a root to each other vertex give k − 1 independent vectors with ones at the two endpoints. The spin-up and spin-down graphs are disjoint connected copies, so their combined rank is 2(k − 1) = 2k − 2. The two conserved parity bits are exactly the deficit from the 2k physical wires. This proves the compressed widths of six and ten. The proof applies even though individual Pauli rotations need only preserve parity; conservation of exact particle number by the complete Hamiltonian is a stronger property.

Lattice	Data qubits	Selector alone	TKET direct	TKET after selector	Qiskit direct	Qiskit after selector
2 × 2	8 → 6	136	45	**37**	95	100
3 × 2	12 → 10	308	**86**	87	**205**	228

Table 18. CNOT counts for the same product steps. TKET's winning pass was GreedyPauliSimp both before and after reduction. Each Qiskit entry is its best of the two tested passes. On the smaller lattice the combination saves 8/45 = 17.8% of CNOTs as well as two data qubits. On the larger lattice the same combination costs one extra gate. Basis-state preparation needs only single-wire flips.

Capacity of the step. At the end of the specified step the state lies in the sector with fixed numbers of up and down particles: 36 words on the smaller lattice, a capacity of 5.17 bits, and 400 on the larger, 8.64 bits, of which 361 are occupied after one step. Six wires therefore meet the floor of Proposition 1 on the smaller lattice, and ten wires are one above the floor of nine on the larger. The missing bit is knowledge shared by the modes of one spin, whose count is fixed. Parity is the only linear part of that knowledge; by criterion (2) the rest needs a code whose minimum distance exceeds twice the particle number. During the step the individual rotations leave the sector, with up to 48 and 494 occupied words and a norm of up to 0.04 outside it, so a register at the floor would also have to treat each hopping pair as one generator. The flat pass of Section 6 does that regrouping from the list of single rotations: 28 rotations become 11 terms on the smaller lattice and 46 become 19 on the larger, 8 and 14 of them hops. The conserved counts it finds give capacities of 5.17 and 8.64 bits, and at the boundaries of the regrouped terms the state stays inside the sector, with at most 36 and 361 occupied words.

Numerical check. For each input term (P, c), explicitly update a full vector by $a' = \cos(0.23c)\, a - i \sin(0.23c)\, Pa$, evaluating Pa from (1). Independently simulate the reduced gate list, insert its amplitudes at s + By, and fill other entries with zero. Maximum absolute errors were 3.43×10^{-15} and

5.74×10^{-15} or smaller. We also simulated the compiled greedy TKET gates on each indicated initial state, resolving implicit wire permutations, before and after compression. The SDK circuits passed an amplitude comparison up to global phase at tolerance 10^{-10}.

Dependence on the basis. Theorem 7 leaves the basis B free, and the selector fixes it by the scan order of Section 4. Downstream gate counts depend on that choice. In an independent re-run with pytket 2.18.5 the first-come basis reproduced 45 and 37 on the smaller lattice, and 40 other valid bases gave between 37 and 46. On the larger lattice, generated with $n_x = 3$ and $n_y = 2$, the same re-run gave 89 and 85 against the 86 and 87 of Table 18, and the other bases gave between 94 and 125. The one-gate difference on the larger lattice is therefore inside the variation caused by term order and basis choice. The first-come basis was the best of those tried on both lattices.

11 Circuit controls and tests beyond routing

Fundamental exchange chain. For an open chain of N sites with N = 4 or 8 local colours, consider the exchange Hamiltonian proportional to $\Sigma_i S_{i,i+1}$ [8]. Initialise site i with colour i. Apply one even-bond sweep followed by one odd-bond sweep, with rotation angle 0.23 on each bond. For the input fixture, construct the local matrix that sends |a,b⟩ to |b,a⟩ and use its Pauli traces to obtain the coefficients. This produces the input independently of the selector's expansion routine. The reference simulator explicitly exchanges the site labels. Both chain outputs agreed within 5.6×10^{-16} after phase alignment.

Chain	Selector	Qiskit best	TKET best	Explicit exchange circuit
Four sites and four colours	96	72	54	**48**
Eight sites and eight colours	336	609	476	**168**

Table 19. Logical CNOTs per complete step. Initial basis labels require no CNOTs. At eight sites, automatic recognition reduces the count relative to both SDK configurations, but the explicitly structured circuit halves the selector's cost.

Derivation of the control. A two-qubit swap has eigenvalue −1 only on the singlet Bell state; the three triplets have eigenvalue +1. The inverse Bell preparation (CNOT then Hadamard) encodes the singlet as 11. Thus the sign of an entire register swap is the parity of its w singlet indicators. XOR these indicators into a clean auxiliary using relative-phase Toffolis, phase the auxiliary by 2t, then apply the exact inverse of the compute network. The compute phases cancel because the intervening gate is diagonal. Undo the Bell basis changes. The Bell transforms cost 2w CNOTs round trip; the relative-phase Toffolis cost 6w round trip. This gives 8w per bond, against 16w in the frozen selector. We checked each bond as a unitary on all data inputs with a clean auxiliary for w = 2, 3.

Capacity of the chain control. The chain starts from a basis state, has no diagonal generator and uses each bond once, and the bonds form a path. Theorem 17 applies. The eight-site step reaches 128 of the 40,320 orderings, each by a single path, and equals seven single-qubit rotations with classical decoding, on 7 wires instead of 24. The four-site step uses 3 wires instead of 8. Both agreed with direct simulation to machine precision. Table 19 prices the exchange block as a unitary on arbitrary inputs, which is what matters when the block sits inside a deeper circuit. As a one-step experiment from a basis state, the control needs no two-qubit gate.

Transverse-field Ising control. Use an open eight-site chain, bond angle 0.23 followed by field angle 0.17, with a computational-zero input [9]. In the ZZ + X convention the single-site fields span all eight flip directions. The selector retains eight qubits and uses 14 CNOTs, tying the best SDK counts. In the rotated XX + Z convention the seven independent adjacent-pair flips have even parity. The selector uses seven qubits and 12 CNOTs; the best SDK counts are 14. A global basis change also rotates the physical initial state. These two rows describe separate basis-state experiments. Comparisons between compilers use the same state within each row.

Independent circuit files. As a separate coverage check, we applied the Pauli-rotation reduction of Theorem 7 with terminal computational-basis readout to all 43 original circuit files in the small tier of PNNL's QASMBench at commit 357b942396d5c2b7cbc1c229c585a6ef5ccaebac. The adapter assumes a known zero input and allows classical decoding of the final measurements. It reduced width on 10/43 files (23.3%). Six of the 43 (14.0%) also used fewer total gates and fewer CNOTs. Of the ten reductions, six produced effectively deterministic outputs (one outcome above probability 10^{-10}). The four reductions with multiple occupied outputs (4/43, 9.3%) were cat_state_n4, qpe_n9, simon_n6 and

variational_n4. Of these, cat_state_n4 and qpe_n9 also used fewer CNOTs. The decoded distributions agreed with direct simulation to within 8×10^{-16} in total variation. Of the remaining files, 24 had full rank, five used dynamic operations and four failed to parse. These percentages describe files in this fixed corpus under the stated readout contract. The other compiler passes were not evaluated in this audit.

We compiled both the original QASMBench circuit and its reduced implementation with Qiskit optimisation level 1, the U3/CX basis, seed 7 and unrestricted connectivity. Appendix A gives the commands for reproducing the per-file comparisons.

Independent capacity audit. We repeated the coverage check by a different method and extended it to all three tiers of the collection, 132 original files. The audit follows two things gate by gate from the zero input. The first is the list of parities of wires that are constant on every word that can carry amplitude. It uses only which entries of each gate matrix are non-zero, so it is sound for any gate and exact for reversible classical gates, and it runs at any wire count. Subtracting the number of independent tracked parities from n gives the tracked dimension d, the dimension of the affine space certified by these constraints. This bounds the true occupied affine-hull dimension from above. A linear sketch can therefore remove at least n − d wires for the final words, under the stated readout contract. The second is the set of occupied words itself: on all amplitudes up to 20 wires, and word by word at any width while fewer than 2^{16} words are occupied. Its logarithm is the capacity of Section 2, taken at the end of the file and at its peak. Both analyses are known in compilers as basis-state and constant propagation [38, 39]. Their use here is to measure capacity.

Tier	Files	Analysed	Tracked dimension below wire count	Wires a linear sketch removes	Simulated	Peak capacity below tracked dimension	Tracker sound
Small	43	34	10	27 of 154	34	6	34 of 34
Medium	25	19	9	121 of 352	16	5	16 of 16
Large	64	50	20	2,499 of 6,906	20	4	20 of 20
All	132	103	39	2,647 of 7,412	70	15	70 of 70

Table 20. Capacity audit of QASMBench at the commit above. Files with measurement before the end, reset or classical control (18), files the parser rejected (8) and files above 30 MB (3) were left out. A file counts as simulated when its occupied words could be followed to the end. The last column counts simulated files whose final words all satisfy the tracked parities.

On the small tier the audit finds the tracked dimension below the wire count on 10 of 43 files, the number found by the rotation method above. The four files with several occupied outputs are the same four, and the other six end in a single word, as they do there. The two methods need not agree in general: the rotation method absorbs Clifford gates into a frame, and the tracker follows classical gates exactly and so sees through arithmetic on a fixed input. Neither stock pipeline removes these wires. On the ten files Qiskit at optimisation level 3 retained all active wires, and TKET's contextual simplification, which knows the zero input and the final measurement, cleared one file completely and left the other nine at full width.

The larger tiers show what the redundancy consists of. All 20 large files with a tracked dimension below the wire count belong to four classes: adders and multipliers on a fixed input, where the tracked

dimension is zero and the file computes one classical word; cat and GHZ states, dimension one; and W states, dimension n − 1. These files are easy by any method. A circuit whose peak capacity is small can be simulated by following its words, so on such files the audit measures simplicity and not a saving of quantum resources. The routing registers of Section 9 differ in kind: their capacity $\log_2 m!$ grows with the problem, and the redundancy removed there sits beside a state that still has to be prepared on a quantum device.

On 15 of the 70 simulated files, the peak capacity is below the final tracked dimension. The tracker can miss linear constraints, so this comparison alone does not establish a limit on every linear sketch. The clearest case is the W state, which is the one-hot row of Section 6 in public form: the 380-wire file has tracked dimension 379, a final capacity of 8.57 bits and a peak of 9.57. Others are oracle circuits whose work wires hold fixed functions of a few inputs: the 11-wire satisfiability file never occupies more than 32 words. Recognising such structure at a scale where it matters is the task of Sections 5 to 7, and the collection contains no instance of it beyond these. Under this audit the public suite supports the affine reduction as a general pass and does not test the re-encoding passes.

11.1 Information and decoding beyond routing

Experiment. We compared a five-facility quadratic assignment on equally spaced sites and a six-vertex weighted Max-Cut with one bit per vertex. The first uses adjacent whole-register exchanges. The second uses single-bit X rotations and has no one-hot rows. In each task G is a uniform permutation of the names. A decoder sees twelve independent output words from the same hidden G and must identify the relabelled optimum Y, ignoring reversal or the naming of the two sides. The reference instance and angle settings are known. A calibrated reader scores samples using the ideal reference law. A cost-only reader uses the Gibbs law of the standardised reference cost at unit inverse temperature. Each sums likelihood over the permutations giving the same candidate answer.

Information identity. Let P be the reference output law on N words, X the observed word, Q the average of P over all name permutations, and U uniform on the words. Fix one oriented representative of the optimum on the reference instance, and let Z be its relabelled image. Then Y is a function of Z and Z a function of G. With all entropies in bits,

$\log_2 N - H(P) = D(Q \,\|\, U) + I(Y; X) + I(Z; X \mid Y) + I(G; X \mid Z)$.

Proof. Every conditional law given G is a permutation of P. Therefore $I(G; X) = H(Q) - H(P)$. Subtract $H(Q)$ from $\log_2 N$ and apply the mutual-information chain rule to Y, Z and G. In assignment, the action on orderings is transitive, so Q is uniform. The full oriented optimal ordering specifies G, making the last term zero. Reflection leaves exactly two orientations for each eligible optimum. In Max-Cut, vertex permutations preserve Hamming weight. Q is uniform within each weight, and its divergence from U is exactly the divergence between the output weight histogram and the fair-binomial histogram. Z identifies the image of one fixed optimal side. Permutations within each side can still affect the output law. Complement symmetry of cut values alone does not eliminate either additional term.

Methods. Both tasks start in the uniform superposition and use one diagonal cost phase followed by mixers with a common angle. For assignment, the cost of ordering π is the sum over $i < j$ of $w(\pi_i,\pi_j)(j - i)$. For Max-Cut, the cost is minus the total weight crossing the cut. Independent integer pair weights range from 1 to 100. Both costs are centred and scaled to unit standard deviation over all words. We chose angles separately per instance from the same 21 × 21 grid, minimising mean cost or maximising the deficit. Optima were used only for eligibility and evaluation. One of sixteen assignments had an additional tied optimum and was excluded. All sixteen initial graphs qualified. Cases receive equal weight. Decoder percentages average 512 simulated twelve-sample trials per instance, paired across objectives. A trial uses a fresh hidden permutation and counts recovery of the exact unoriented optimum.

Task	Goal	Deficit	Across orbits	Optimum	Orientation	Within sides
Assignment	Cost	0.952	0	0.724	0.229	0
Assignment	Deficit	1.053	0	0.759	0.294	0
Max-Cut	Cost	0.658	0.392	0.051	0.027	0.189
Max-Cut	Deficit	1.633	1.255	0.064	0.028	0.287

Table 21. Mean bits per sample on the initial instances. The last four columns sum to the deficit up to rounding.

Task	Instances	Training goal	Calibrated	Cost only
Assignment	15	Cost	64.3%	43.6%
Assignment	15	Deficit	73.8%	26.8%
Max-Cut	16	Cost	33.3%	32.3%
Max-Cut	16	Deficit	30.2%	1.3%
Fresh Max-Cut	16	Cost	33.1%	31.1%
Fresh Max-Cut	16	Deficit	30.7%	1.3%
Fresh Max-Cut	16	Relabelling information	49.9%	18.5%

Table 22. Twelve-sample recovery, averaged within each cohort. In this table, only the follow-up uses the relabelling-information training score.

What transferred. In assignment, deficit training improved calibrated recovery on all fifteen instances and lowered thermal-model recovery on fourteen. In the initial Max-Cut cohort, most of the increased deficit came from the weight histogram. Mean twelve-sample calibrated recovery changed by −3.1 percentage points in Table 22. An independent sixteen-graph cohort reported a change of +0.2 points, with improvements on nine graphs. The direction varied across cohorts, giving no consistent recovery gain from deficit training.

Exploratory follow-up. After observing the Max-Cut result, we switched the objective to $I(G; X) = H(Q) - H(P)$, which removes the concentration shared by every relabelling and can be evaluated without the optimum. On sixteen fresh graphs, this objective raised calibrated recovery by 16.8 percentage points over cost training and beat cost training on fifteen graphs. Cost training still gave the higher average for the thermal reader. These are small, dense, positively weighted instances. Fifteen of the sixteen fresh graphs and all initial graphs had balanced optimal cuts. A calibrated reader here has access to a reference law calculated by exact simulation. Neither search over a common grid nor the twelve-sample comparison measures an end-to-end speed advantage.

Fixed angles on held-out graphs. We fitted one angle pair per objective by averaging the objective over the original sixteen Max-Cut graphs and searching the same 21 × 21 grid. All four pairs stayed fixed during testing. Of 32 fresh graphs, 31 had a unique optimal partition up to exchange of sides. For each graph the decoder knew the exact reference law and the reference optimum. It inferred the partition after an unknown uniform permutation of vertex names. This tests transfer of circuit settings under graph-specific calibration. The selected relabelling-information angle $\gamma = \pi$ is the upper edge of our search. Larger values may improve the training score. Table 23 reports the settings selected within this grid.

We compared expected cut weight, entropy deficit, information about the hidden permutation $I(G; X)$, and information about the optimal partition $I(Y; X)$. The last objective uses the reference optimum during training. To evaluate it, average the output law over all vertex permutations to obtain Q, then average over permutations preserving the reference partition as an unordered pair to obtain R. Each conditional law given Y is a relabelling of R, so $I(Y; X) = H(Q) - H(R)$. Goal-oriented experimental design already uses expected information gain about a chosen quantity [41]. Here we compare the resulting single-sample training criteria against exact partition recovery at four sample budgets.

Training goal	1 sample	3 samples	6 samples	12 samples
Expected cut weight	15.5%	19.1%	23.4%	30.4%
Entropy deficit	10.9%	13.5%	19.3%	31.1%

Training goal	1 sample	3 samples	6 samples	12 samples
Permutation information	12.9%	17.9%	26.3%	43.5%
Partition information	11.0%	14.7%	21.5%	33.7%

Table 23. Mean probability of recovering the optimal partition on 31 held-out six-vertex graphs. Each column uses the same calibrated decoder with the indicated number of samples.

The one-sample column is the exact success probability of the maximum-posterior decoder. The other columns average 512 paired trials per graph. At twelve samples, training for permutation information gained 13.1 percentage points over expected cut weight and beat training for partition information on 24 of 31 graphs. Each trial holds the hidden permutation fixed across samples. Several observations could therefore help resolve the labelling jointly, although this comparison does not establish the mechanism.

11.2 Independent circuit inputs and finite reference samples

An externally authored circuit. We instantiated the excitation_preserving ansatz from Qiskit 2.2.3 [42] with linear entanglement, two repetitions and angle (1 + k mod 11)/32 radians for parameter k, starting at zero. We supplied the known basis seed with one occupied wire and called the existing hopping recogniser. In the compiled path, transpilation at level zero (seed 17) produced only U gates and CNOTs. The existing lifting and frame-closure routines received this gate list. The hopping recogniser received the resulting flat rotations and seed, without any row labels, exchange boundaries or ansatz name.

At 4, 6, 8, 10, 32 and 64 source wires, the inferred widths were 2, 3, 3, 4, 5 and 6. At 64 wires, the 1,956 input gates yielded 444 lifted rotations. The largest absolute difference over the 64 × 64 valid-sector matrix entries was 1.24×10^{-13}. Through ten wires we checked every valid input against Qiskit's full statevector evolution. At 32 and 64 wires we instead propagated every column of the one-excitation matrix through the original Qiskit gates. The local-word certificate applies to the recognised groups. The amplitude comparison also checks the numerical gate lifting at the chosen parameters. The 64-dimensional sector is easy to simulate classically. This experiment tests recovery from an external source, specifically for the hopping branch.

Fixed circuit files. Two fixed QASMBench source files were also tried at the Section 11 commit. We supplied any leading X preparation as the seed and stripped terminal readout. The frame-closure check declined iswap_n2 and wstate_n3. These were targeted probes for exchange or single-excitation structure. Section 11.3 tests a different external source for the rectangle branch.

Learning the decoder from examples. Each query gives twelve words from a single hidden permutation of a known reference instance. The decoder has the reference optimum. We drew a separate calibration set from the same circuit on the unpermuted instance for each graph and angle setting, then reused it across queries. For cuts, a word and its bitwise complement have identical ideal probability. Count the occurrences of each unordered pair and add one pseudocount. A candidate permutation receives the product of these smoothed counts for the inverse-relabelled query words. The normalisation cancels. Summing over permutations giving the same partition and taking the largest sum gives the decision. The same reference sample prefixes give the different budgets. Finite calibration is simulated from the ideal circuit independently of query samples.

A dictionary stores at most as many observed keys as calibration shots. An unseen key uses the pseudocount. This removes the need to store or compute the exact law. Trying all n! permutations still takes factorial time. Calibration also costs fresh shots when the reference instance or circuit setting changes. The 2,048-shot column represents reusable calibration across many queries, not the total budget for solving one query.

Vertices / graphs	Setting	Chance	Cost reader	256 refs	2,048 refs	Exact law
6 / 31	cost	9.7%	28.2%	27.8%	29.8%	30.4%
6 / 31	label	9.7%	15.4%	38.8%	43.0%	43.5%
8 / 12	cost	2.8%	13.1%	11.4%	12.5%	13.2%
8 / 12	label	2.8%	6.0%	14.0%	16.5%	18.4%

Table 24. Optimal-partition recovery under all vertex permutations with twelve query samples. Columns headed "refs" use the complement-pooled frequency reader. Each instance has equal weight.

We reused the six-vertex settings $(\gamma, \beta)/\pi = (0.25, -0.10)$ for cost and $(1, -0.25)$ for relabelling. The second remains a grid-edge choice. For six vertices we reused the 31 held-out graphs and 512 trials per graph of Table 23. The eight-vertex test used 256 trials per eligible graph, all 40,320 permutations and the same unique-optimum criterion. With 256 reference samples, the relabelling setting won on 28 of 31 six-vertex graphs, with an 11.1-point mean advantage over cost training (paired graph bootstrap 95% interval 7.3 to 15.0 points). At eight vertices the mean gain narrowed to 2.6 points, with nine wins, one tie and two losses (interval −1.1 to 6.4). The latter cohort leaves the sign of the population gain unresolved. Intervals resample graphs, using 20,000 replicates and seed 33, conditional on their realised calibration samples.

Larger state spaces. Larger tests used all basis states for the simulation but restricted the hidden name change to a uniform cyclic shift. The reference optimum and its images determine the candidates. This makes likelihood evaluation tractable and the identification problem easier. A pair reader fits smoothed Bernoulli frequencies for vertices on opposite sides, or for facilities adjacent in an ordering. For a feature vector f, its word score is the sum of $f_i \log(p_i/(1-p_i))$; omitted constants cancel. Here $p_i = (\text{count}_i + ½)/(\text{reference shots} + 1)$. The reader needs only $n(n-1)/2$ frequencies. Feature dependence makes this a composite score. The code also reports histogram, four-bit and cost-bin controls where applicable.

Problem	Setting	Chance	Cost reader	Pair 256	Pair 2,048	Exact law
12-vertex cut	cost	8.3%	65.3%	47.0%	59.8%	68.6%
12-vertex cut	label	8.3%	11.4%	43.0%	55.6%	95.7%
16-vertex cut	cost	6.2%	65.7%	39.7%	59.7%	67.5%
16-vertex cut	label	6.2%	10.6%	38.0%	54.6%	95.1%
8-facility assignment	cost	13.8%	34.5%	22.9%	31.5%	40.7%
8-facility assignment	label	13.8%	81.0%	59.6%	69.1%	99.8%

Table 25. Recovery under cyclic relabelling from twelve query samples. "Label" and "cost" denote the two fixed six-vertex settings for all tasks, including assignment. Pair budgets count calibration samples per instance and setting.

For sixteen vertices we also calculated single-sample information under the full vertex permutation group by averaging within Hamming weights, without enumerating the group. The label setting yielded 0.731 bits about the hidden permutation, against 0.302 for the cost setting. These are full-group information values. The decoding percentages in Table 25 use the cyclic subgroup. Twelve 12-vertex and twelve 16-vertex graphs qualified (4,096 and 65,536 basis words). Ten of twelve eight-facility assignments qualified (40,320 orderings). For each instance we ran 512 paired queries. The label setting at sixteen vertices reached 95.1% with the exact law, 54.6% with 2,048 reference samples and pair features, and 10.6% for the cost reader. At the cost setting the cost reader alone reached 65.7%. For assignment the cost reader at the label setting also exceeded the fitted pair reader. Low-dimensional calibration therefore supplies usable evidence in these tests, while a simpler cost reader can still be the better practical choice. No runtime comparison against a classical optimiser is implied.

11.3 External routing rectangles and a reader without a reference

Independent routing source. The published decomposition study [43] supplies a Qiskit implementation of a routing phase separator and a colour-parity mixer. We invoked its unchanged

build_phase_separator and build_mixer methods at both supported sizes, three and four variable positions, with one layer and the supplied diagonal one-hot seed. The authors use a four-wire controlled-rotation circuit, including phases on the exchanged words. Flattening to single-qubit U gates and CNOTs yielded 393 and 1,052 gates. The original recogniser found rectangle structure but declined both lists at its complete-family check. Regrouped diagonal terms and ordered partial exchanges need not admit the family schedule assumed by that lowering.

Ordered lowering. Retain the lifted and regrouped order. Recover the two grid directions as in Section 6 and require exactly one set bit on every line of the supplied seed. Encode the occupied column of each row by a binary label. For each moving group, inspect its local action exactly in rational Pauli arithmetic. Require two opposite active corners on four wires, equal real matrix elements a, and no other active words. If its two rows are s,t and its two columns i,j, emit a two-level rotation on those label registers coupling (i,j) to (j,i), with angle a. Keep every diagonal group in place, replacing the occupied bit at (s,i) by the equality predicate "label s equals i". This last step retains the extra phases in the upstream gate.

Valid-subspace certificate. Let E map a valid compact ordering to its physical one-hot word. A diagonal term has the same eigenvalue on both words by the equality predicate. A moving generator annihilates both words unless the two named labels occur on the specified rows in either order. In the active case each generator swaps those labels with coefficient a. Hence $H_k E = E\hat{H}_k$ group by group, and exponentiating and composing in order gives the corresponding circuit identity. This proof applies to the accepted lifted coefficients. The numerical lift from floating-point elementary gates is checked separately. Each emitted two-level rotation is an operation on the compact register. Section 11.4 gives its elementary-gate implementation and measured CNOT cost.

Checks and scope. The three-position input yielded 151 lifted rotations, nine recognised rectangles and a six-wire program from nine physical wires. The four-position input yielded 402 rotations, 24 rectangles and an eight-wire program from sixteen. We compared the entire compact valid-sector matrix with the original Qiskit gate matrices for all six and 24 basis inputs. Maximum entry errors after one common global phase were 3.30×10^{-15} and 7.45×10^{-15}. Compact leakage was zero to numerical precision. Both invalid zero-seed controls were declined. The new rule was developed after examining this source. The lift counts nontrivial Pauli rotations before regrouping. Each source layer includes two explicit RZ(0) gates. Including those identity placeholders gives 153 and 404 entries. Our final angle filter removed no further entries on either run. This was the development source, and the recognised hopping template in Section 11.2 remains a classically easy 64-state example.

Frozen-rule regression. We loaded the byte-identical ordered lowering function from the preceding release before testing a second source. On the eight-city source (49 wires, 1,309 rotations), the ordered rule returned 21 logical wires and 127 operations. The same result held for ten lists shuffled by commuting adjacent rotations, five random orders that preserve dependencies and one within-slot rectangle shuffle. We regenerated the four compiled inputs with Qiskit 2.6.2 (default and Rustiq) and TKET 2.18.5 (GreedyPauliSimp and PauliSimp). All four were accepted after lifting and frame closure. For a fixed normalised random superposition, we compared all 5,040 output amplitudes with the independent tour recurrence after removal of one common phase. The maximum discrepancy was below 5.4×10^{-13}. This tests logical equivalence on this fixture. Gate costs of the two lowering strategies have yet to be compared. An additional Qiskit 2.2.3 default-synthesis run was declined on an unsupported moving term.

Held-out external source. Radzihovsky, Murphy and Swofford published a separate PyQuil routing implementation in 2019 [44]. We executed its original cost, mixer and basis-seed generator functions and flattened their returned Pauli terms without block labels. For three and four positions the frozen rule accepted 114 and 336 nonidentity rotations, respectively, yielding six and eight logical wires. We used integer edge weights from 1 to m^2 in row order, mixer angle 3/128 and cost angle 1/32. We compared all entries on the six- and 24-order sectors against independent matrix exponentials of the original generator blocks. The largest entry discrepancy was below 4.1×10^{-16} up to global phase.

Held-out gate inputs. Using PyQuil 4.23.0 to synthesise those same commuting blocks produced 1,518 and 4,512 elementary gates, excluding the supplied seed preparation. At the initial angles the three-position circuit passed. The four-position circuit was declined before ordered lowering because the lift could not close its trailing Clifford frame. In a follow-up with cost angle 1/128, both complete gate lists passed, with valid-sector entry errors below 4.0×10^{-16}. The older optimisation driver and quantum virtual machine were not run. The generator source and lowering rule were fixed throughout. This is a held-out implementation test. It does not estimate coverage on arbitrary routing circuits.

Reference-free control. Given twelve cut words, return the observed word with smallest total Hamming distance to the others, measuring distance to a cut as the smaller distance to either orientation. Break ties uniformly among minimising sample indices. This sample medoid needs neither a reference graph, its optimum, calibration samples, nor a search over permutations. It uses $O(nk^2)$ bit comparisons for n vertices and k samples. Vertex relabelling preserves every distance and carries the output cut along with the inputs, so scoring before relabelling gives the same recovery rate.

On fresh weighted complete graphs (seeds 91100–91131 per size), 31, 32, 30 and 31 instances had a unique optimal partition at 6, 8, 12 and 16 vertices. We simulated 512 independent twelve-word queries per graph at each fixed angle pair from Table 23. At the cost setting, returning the first sample recovered the optimum in 11.35%, 3.87%, 0.39% and 0.050% of trials. The medoid recovered it in 20.71%, 6.57%, 0.79% and 0.095%. At the label setting its rates were lower still: 7.35%, 1.05%, 0.19% and 0.013%. At sixteen vertices the optimum appeared anywhere among the twelve cost-setting samples in only 0.53% of trials, an empirical ceiling for every rule that returns one of those samples. The earlier calibrated decoder can propose an unseen answer using the known reference optimum. The present control gives up that information and has low absolute recovery at scale.

11.4 Gate cost and angle coverage on external routing layers

Paired compilation. We compiled source and recovered layers with Qiskit 2.2.3, optimisation level 3, seed 17, unrestricted connectivity and the same U/CX basis. Both sides exclude preparation, readout and classical decoding. The counts include all diagonal phases and all mixer operations. No auxiliary wires were introduced. Depth is the transpiler's elementary-gate depth. All measurements describe one noiseless logical layer. Tübingen uses the Section 11.3 parameters. Stanford uses $\beta = 3/128$, $\gamma = 1/128$ for three and four positions and $\gamma = 1/256$ for five and six. The larger runs extend the authors' generator and use the same row-ordered integer weights as the smaller Stanford tests.

Compact gate synthesis. For a diagonal group, expand each equality predicate on its touched label bits in the Walsh basis, add equal masks and emit one parity computation, RZ and uncomputation per nonzero mask. For each two-word rotation choose a differing pivot bit. CNOTs from the pivot to every other differing bit map the two words onto a pair differing on the pivot alone. Hadamards on the pivot convert the required RX to a uniformly controlled RZ; its angle table has 2a at the required control pattern and zero elsewhere. Undo the CNOTs. Within each uninterrupted run on the same two slots, different unordered label pairs have disjoint support, so their rotations commute and angles for repeated pairs add. We apply this exact simplification before elementary-gate synthesis.

Source	Positions / orders	Wires	CNOTs	All gates	Depth
Tübingen	3 / 6	9 → 6	198 → 261	367 → 456	303 → 399
Tübingen	4 / 24	16 → 8	528 → 296	978 → 559	212 → 269
Stanford	3 / 6	9 → 6	444 → 128	981 → 231	693 → 209
Stanford	4 / 24	16 → 8	1,320 → 233	2,914 → 445	1,773 → 377
Stanford	5 / 120	25 → 15	2,920 → 2,334	6,438 → 4,234	3,754 → 3,786
Stanford	6 / 720	36 → 18	5,460 → 3,631	12,026 → 6,778	6,693 → 6,076

Table 26. Elementary gates and depth of the original published layer and the recovered layer, in that order. Repeated exchanges are combined on the compact side. The next control applies the same opportunity to the source.

Control for duplicate exchanges. Stanford's layer visits each pair of distinct cities in both orders. We verified equality of each duplicate generator. Within a time-step, distinct unordered pairs act on disjoint valid label pairs; equal-city terms are diagonal and invariant under exchanging the two slots. We kept these diagonal terms and applied each distinct exchange once with double angle before compiling the one-hot source. At three through six positions this control used 240, 708, 1,560 and 2,910 CNOTs, with depths 353, 748, 1,255 and 1,874. The compact circuit still won on CNOT count at three and four positions. At five and six the one-hot control won on CNOT count and depth. At six positions the compact circuit cut the original CNOT count by 33.5% but used 24.8% more than the improved one-hot control. The unmerged compact CNOT counts were 188, 413, 3,592 and 6,090. The supplement also records the generic controlled-RX decomposition.

Verification. We compared the compiled compact circuit with independent source-sector evolution up to one global phase. At three and four positions we checked every valid input column and the amplitudes outside the sector. At five and six the unchanged recogniser processed gate streams of 9,980 and 18,660 instructions. We checked all compact amplitudes for one normalised random superposition spanning each valid sector, including leakage, against direct action of the original generator blocks. At six positions the maximum valid-amplitude difference was below 1.7×10^{-15} and the leakage norm below 8.6×10^{-15}. These larger numerical checks sample an input state; the local equivalence argument is in Section 11.3.

Finite angle sweep. For each of three and four positions, we regenerated the full Stanford gate circuit on the Cartesian grid $\gamma \in \{1,2,3,4,6,8,12,16\}/128$ and $\beta \in \{1,2,3,4,6,8\}/128$ radians. The 48 settings per size include both previously reported cost angles. With the same frozen lowering rule, lifting and recognition succeeded at 25/48 (52.1%) and 15/48 (31.3%), respectively. At three positions the accepted numerators for γ were 1,2,3,4,6; at four they were 1,2,3. For each of those cost angles the accepted β numerators were 1,2,3,4,6. Every decline occurred during closure of the trailing Clifford frame. Counts measure whether the pipeline returned a program. The grid was chosen after the original two angle results, and the separate amplitude checks cover the costed settings.

11.5 Spin-chain capacity test and group ordering

Workload and observable. We use an open chain of sixteen spin-3/2 sites. The Blume–Capel chain has been studied as a model for encoded quantum operations [45]. We designed a short echo to test compilation and readout. Initialise each site in level 1, numbering the four levels from 0 to 3. Apply $\exp(-0.75i\,A)$ to the outer pairs (0,1) and (2,3), followed by the same rotation on the middle pair (1,2). On each pair A exchanges its two levels and vanishes elsewhere. With m taking values −3/2, −1/2, 1/2 and 3/2, apply the diagonal phase generated by $0.25\,\Sigma_i\, m_i^2 - 0.8\,\Sigma_i\, m_i m_{i+1}$ and undo the preparation pulses. The control sets the interaction coefficient to zero. A full-register shot reads every site. We score the probability that site 8, counting from zero, returns to level 1.

Grouping rules. With interactions on, the count-ordered rule predicts a 67.15% central return on sixteen sites. The independent calculation gives 50.36%. The failure also occurs in the two-site echo with interactions off. When a later contraction introduces a predecessor, the count saved at group formation can place a readout pulse before a diagonal operation with which it does not commute. The topologically ordered rule uses the same groups. Map each source rotation to its final group. For every noncommuting pair in source order, add an edge from the earlier group to the later one if the groups differ. Emit each group only after all its predecessors, breaking ties by group content. Every noncommuting pair therefore retains its order. Proposition 13(ii) preserves the product, and Proposition 15 gives equivalence on the valid sector. Subsequent fusion moves rotations only across commuting neighbours.

Regression tests. We tested 543 random flat lists of hops and phases. Both rules accepted the same lists. The count-ordered rule gave 21 incorrect outputs and the topologically ordered rule gave zero. On routing fixtures from four to twelve cities, both recovered the same programs. The further-families suite gave the same widths and gate counts, and 20 circuits compiled with Qiskit or TKET gave the

same outputs. We observed no loss of coverage on these fixtures. A separate implementation of the topologically ordered rule reproduced the spin-echo results.
Independent checks. On two sites we evolved every one of the sixteen valid inputs through the one-hot source and compared every complex amplitude in the valid sector after aligning the global phase. The largest entrywise discrepancy was 0.314 for count-ordered recovery and below 6×10^{-16} for topologically ordered recovery, its fused form and the hand-written compact circuit. On sixteen sites we checked the central marginal separately. If p_a is the probability of level a after the preparation pulses, trace out the neighbours by weighting their two levels independently with $p_\beta p_c$. For each neighbour pair, the central return amplitude is the sum over a of p_a times its diagonal phase from the central anisotropy and the two adjacent bonds. Average the squared modulus over the neighbour pair. This gives 50.36% with the interaction and 93.91% without it. Exact propagation of the measured projector through the topologically ordered circuits agrees in both settings.

Table 27. Resource counts for the interacting 16-site echo. A weight-two Pauli rotation costs one tunable ZZ entangler plus local basis changes.

Implementation	Qubits	Two-qubit gates
One-hot source, including fusion control	64	432
Count-ordered recovery (incorrect echo)	32	204
Topologically ordered recovery	32	204
Topologically ordered recovery with fusion	32	140
Direct binary implementation	32	140

Processor and noise model. Quantinuum H2 specifies 56 qubits, all-to-all connectivity, four parallel two-qubit operations and a typical two-qubit average infidelity of 0.001 in the 17 September 2025 data sheet [46]. A greedy schedule for the 140 entanglers uses 42 slots, with at most four disjoint gates per slot and dependencies on each wire respected. This is a resource estimate. It excludes transport, single-qubit gate time and readout. We propagated the central return projector with an independent two-qubit depolarising channel after each entangler. At average infidelity 0.001, the fused circuit gives 93.82% return with interactions off and 50.07% with interactions on. The 43.75-point difference is close to the ideal 43.55-point difference. At 0.0015, a sensitivity run gives 93.77% with interactions off and 49.92% with them on, a difference of 43.85 percentage points. Only gates in the measured spin's causal past can affect its return probability. With interactions off, the fused circuit couples no neighbouring sites. Raising the error rate lowers both returns, but the interacting return falls further, so their difference grows slightly. Memory errors, state preparation, single-qubit errors and measurement errors were excluded. We did not run this circuit on H2.

Practical scope. The 50% width reduction crosses the H2 capacity limit for simultaneous full-chain encoding. Fusion also removes 67.6% of the source entanglers and matches the hand-written binary count. The local return probability has a cheap classical calculation, and this shallow full-chain circuit is also classically tractable. The benchmark therefore establishes a fit and signal test for a scientifically motivated model. Demand for this particular experiment, an end-to-end hardware runtime and an advantage over classical simulation remain open.

12 Interpretation and limits

What the proofs establish. For the affine branch, equation (4) holds on the whole encoded sector and for any rotation angles. For the exchange branch the coefficient certificate is an operator identity, valid on all data inputs. The slot and seed tests justify preparation on a specified orbit. For the one-hot front end the certificate is an identity on valid words only. It supports the two stated initial conventions and nothing else. Proposition 20 is a bound for disjoint reading sets under the uniform law on orderings; it says nothing about gates by itself. The history, cross-row and conditional registers are exact encodings, covered by Theorem 17 for the first and Theorem 6 for the other two. Regrouping and lifting are operator identities, and the hop pass, like the one-hot front end, is an identity on valid words. Proposition 22 is exact for its fault model and for no other. In every branch the correctness argument is independent of benchmark performance. A projector or angle choice can make a smaller subspace relevant for one particular question; this implementation does not attempt that further reduction.

What the entropy accounting shows. Capacity is a floor on wires and says nothing about gates; Table 4 shows gates rising steeply near the floor. Proposition 20 and Table 5 give a reason: positions that can be read separately cost wires, and on the family tested the gates fall only when every position can be. The history register is exact, and it is also a certificate that the circuit is classically easy: a product state followed by classical decoding. It identifies experiments that are smaller than they look, and it is no speed-up. The deficit and the mutual information of Table 7 belong to the circuit and the instance, and no exact compilation changes them. The mutual information is defined for relabellings of one map, and the decoder of Table 8 assumes that the output law of that map is known. The sample-only decoder of Table 9 assumes nothing, but alone it is weaker than keeping the best tour seen and settles on a wrong tour on six of the 33 maps; it is useful as an added candidate. Redundancy is the same surplus seen from the side of faults. The marginal yield of extra redundancy diminishes above the crossover, and in the fault simulations the register that kept the most information for each shot was neither the widest nor the narrowest nor the shortest. From five to twelve cities the deficit grows slowly, the best-seen rule needs rapidly more samples than the information bound, and a thermal law with the same deficit is the better sampler. As a control outside the exact contract, we dropped the 250 smallest of the 343 phase terms at eight cities. The phase cost fell from 342 to 176 CNOTs and the deficit from 1.020 to 0.997 bits, but the output distribution moved by 0.277 in total variation. That is a different calculation, and it is not used.

Cost of recognition. With T terms in a candidate exchange block, the single-one probes take roughly nT mask accumulations. A w-pair exchange has 4^w Pauli coefficients. Certifying it requires regenerating and comparing those entries. Slot alignment also tries up to w! common bit orders to find a labelling accepted by the current preparation routine. Consequently this approach can be expensive when a compactly specified exchange is wide. The one-hot front end avoids the 4^w comparison on its input side: its mixing generators have $4m(m - 1)$ strings and are probed on m^2 words. Neither route enumerates tours to select the routing branch. For fixed-point or symbolic implementations, use canonical sparse dictionaries with zero coefficients removed. The frozen code compares exchange coefficients exactly; it recognises the dyadic fixtures used here. A policy for noisy or approximately equal operators would require an error bound.

Scope of the numerical evidence. The selector and the grouped front end receive generator boundaries. Given single Pauli strings as separate generators, the front end declines, as Section 3 predicts, because the closed set is preserved by the grouped generators and not by their strings. The flat pass of Section 6 removes that requirement by rebuilding the boundaries, for one layer of the routing and assignment circuits and for rows of hops. It reads Pauli rotations, or gate lists of CNOTs and one-wire gates that lift

to them with a frame it can close, and it recognises grids of rectangle moves and rows of hops and no other structure. The Section 6 rectangle tests use our one-hot source. Section 11.3 adds one independently authored routing implementation at both sizes it supports, using a revised lowering rule. Section 11.2 evaluates the hopping recogniser on an external Qiskit template and reports two declined QASMBench files. Device routing and discrete fault-tolerant gate synthesis remain outside the tests. The fault simulations place faults after CNOTs only, with ideal preparation and measurement, at five and six cities; the eight-city entries of Table 10 are the closed form. The results up to twelve cities are exact enumerations of one layer and say nothing about hardware. The inputs encode an already-specified computation. The tests do not show automatic algorithm design from a natural-language objective, recognition from arbitrary gate sequences, or guaranteed compression. The front end ran as a separate module ahead of the selector, and its hand-off was verified at the level of generator dictionaries. The fixed-particle-number case of Section 3 is a statement about sketch width; no circuit was compiled for it. The cross-row lowering was verified with its new blocks at gate level and its unchanged exchange blocks at the level of their action on words. The conditional register was costed for its formula only. The repeated-label seed control is declined by the implemented permutation backend. A connected swap graph is required because the present preparation routine constructs all label permutations. A different routine could exploit orbits on disconnected components. The fixed-entropy study uses exact simulation of one eight-city layer. The four laws in Table 14 come from one map, and the endpoint searches reach local optima. The 29-map transfer result concerns the small projected step. Decoder success is estimated from finite trials. Gate-level acceptance depends on the angles and on the synthesis and frame-closure routines. Within the finite Stanford sweep of Section 11.4 the lift accepted 25 of 48 settings at three positions and 15 of 48 at four. These fractions describe the specified grid and implementation. They do not estimate coverage of arbitrary circuits or a continuous distribution of angles.

What "cheaper" means. We count all-to-all logical CNOTs under explicitly chosen decompositions. The routing and spin-chain selector uses one clean auxiliary during exchange evolution. The direct SDK implementations of those operators use data wires; shared routing preparation already supplies temporary workspace. Table 2 excludes preparation on both sides. Hubbard compression trades physical output wires for classical decoding. Reconstructing a coherent physical state, compiling on sparse connectivity, synthesising fault-tolerant rotations, or estimating wall-clock time gives a different comparison. The cost of one product step gives no bound on the number of steps needed for accurate Hamiltonian dynamics.

What the training experiments support. In routing and quadratic assignment, deficit training improved recovery for the decoder supplied with the reference output law. Cost training gave better results for the cost-only reader. On Max-Cut, the deficit can grow when the distribution of group sizes changes, even if that change reveals nothing about vertex names. The two cohorts comparing deficit with cost training showed no consistent calibrated gain. On held-out graphs, the preferred objective also depended on the sample budget (Table 23). The initial assignment simulation spans 5! = 120 orderings and the graph simulation spans $2^6 = 64$ words, compared with 7! = 5,040 orderings for the eight-city routing fixture. These smaller tests probe how the symmetry and decoder affect the result. Section 11.2 extends assignment to 40,320 orderings and graph partitioning to 65,536 words. It tests unrestricted hidden permutations on six and eight vertices and cyclic shifts on larger instances. The required calibration budget at hardware scale and sensitivity to gate-level noise remain open. Routing tests at nine and ten cities, the sample-trained reference model and the thermal reader are discussed in Tables 15–17. Prior work covers probability-based training objectives [33] and the use of unconstrained circuit behaviour in state preparation [34]. At twelve and sixteen vertices the cost-only

reader at its cost setting reached 65.3% and 65.7%. Across the two settings and both reference budgets, the best pair-reader results were 59.8% and 59.7%, respectively (Table 25). All of these large-graph comparisons restrict relabelling to cyclic shifts.

Conclusion. A compiler can recognise a one-hot program in supported flat rotations or gate sequences emitted by Qiskit and TKET, derive compact registers, and certify equivalent evolution on valid inputs when the recovered groups preserve the source dependencies. Under hidden relabelling, a second result separates a sample's information about the optimum from information about orientation. Those terms exhaust the deficit on the routing and assignment fixtures. A third result concerns how the samples will be used. Training for a distinctive output improved calibrated recovery in routing and assignment, while a cost-only decoder favoured cost training. On Max-Cut, the sample budget also changed which training goal performed best.

The eight-city fixture illustrates the compiler result with a reduction from 49 to 21 data wires. Its reference circuit gives 1.02 bits per sample about the hidden names, including 0.689 bits about the optimum. A sample-only decoder used alongside the best-sample rule raised recovery from 19.6% to 31.6% at 64 samples. On the 31 held-out Max-Cut graphs, cost training led at one and three samples. With twelve samples and graph-specific calibration, relabelling-information training raised recovery from 30.4% to 43.5% using settings fixed across graphs. The comparisons concern representation and decoding under the stated input and reference-law assumptions. General optimisation speedups remain open. With 256 reference samples, the corresponding six-vertex comparison was 27.8% against 38.8%. This reader uses one known reference instance and its optimum. Learning a law for an unrelated instance or finding its optimum from scratch is a different task. The eight-vertex unrestricted comparison remains inconclusive. On the first external source, ordered lowering cut logical width from sixteen to eight on its largest supported case, four positions with 24 valid orders. The second group's four-position generator list attained the same width with the rule frozen. At four positions the compact Tübingen layer used 44% fewer CNOTs but greater depth. At six positions the compact Stanford layer used 18 logical wires versus 36 and 3,631 CNOTs versus 5,460 in the original. Combining the duplicated exchanges in the one-hot source brought its count down to 2,910. The preferred encoding therefore depends on how the width and gate budgets are weighted.
The spin echo gives a concrete capacity threshold. Reducing 64 wires to 32 makes the simultaneous circuit fit H2, and the fused, topologically ordered implementation matches the direct binary gate count. The observable retains a large signal in the stated noise model. Topologically ordered grouping reproduces the independently calculated spin echo and the routing results on the regression fixtures.

Appendix A Reproduction protocol

Use Python 3.12 and NumPy. The reference runs used Qiskit 2.6.2, pytket 2.18.5 and OpenFermion 1.8.1. We ran the rotation-method QASMBench check in Section 11 separately using Qiskit 2.2.3 for both the original and reduced circuits; the independent capacity audit used Qiskit 2.6.2. Each toolkit can be installed in a separate environment. Extract each archive into its own directory. The operator lists and programs supplied with the paper are listed below. The experiment can be repeated from the saved Hubbard input without installing OpenFermion; the generation command reproduces that input from the published model.

Routing. In the compiler comparison directory, run the following for eight cities. Repeat with 4, 5, 6 and 7 for Table 1. For additional maps use the flag `--seed` for the release driver and the environment variable `MAP_SEED` for the greedy TKET driver.

```
python test_common_input.py
python release_benchmark.py qiskit 8
python release_benchmark.py pytket 8
python greedy_tket.py 8
```

Use Qiskit optimisation level 3, basis U and CX, transpiler seed 11, and both default and Rustiq synthesis [10]. For TKET run PauliSimp, explicit commuting-set synthesis, and GreedyPauliSimp with seed zero, rebasing to TK1 and CX. The explicit commuting-set input and Qiskit receive the supplied commuting block boundaries. Default SDK settings otherwise follow the supplied drivers. Add the identical preparation count after compiling the evolution, with no optimisation across that boundary. The release drivers check full evolution unitaries for four and five cities. For the larger cases their results are gate counts. Our separate tour-amplitude checks extend to eight cities.

One-hot front end. In the one-hot directory, run:

```
python verify_onehot.py
python flat_frontend.py
python tket_onehot.py
python qiskit_onehot.py
```

The first script needs NumPy and SciPy only. It builds the one-hot and compact programs for the maps of Section 9, runs the front end, and performs the exact comparison, the exhaustive check on valid words, the full-register evolution at four and five cities, the five controls, and the sketch check for two set wires among fifteen. The check matrix in that last test is that of the double-error-correcting BCH code of length fifteen. The second script is the flat pass of Section 6 with its tests: the exact comparison at six sizes, the reordered lists and the random orders that respect every dependency (Python generator seed 11), the rotation-by-rotation run on the full register, the six controls, and the Hubbard step, which needs OpenFermion. The last two scripts produce the one-hot columns of Table 2 and the gate-list comparisons at four and five cities. Qiskit receives one evolution block for the cost generator and one for each partial swap, with the settings above. TKET receives the same strings in the same order and runs PauliSimp and GreedyPauliSimp with seed zero, rebasing to TK1 and CX. These runs used Python 3.13.18, NumPy 2.5.3 and SciPy 1.18.1 with the SDK versions above. The SHA-256 digest of `onehot_frontend.py` is:

```
8077e9bec0bfe90f1a4d478690a3a76c141fdf02d1b7dbc29280551b3b78d757
```

Lifting and further families. In the same directory, run:

```
python lift_circuits.py
python families.py
```

The first script lifts the Qiskit and TKET circuits of Table 2 by Proposition 16, compares each lifted list with its gate list on a random state at four cities (NumPy generator seed 5), runs the flat pass on the lifted lists at four to eight cities, and applies the five controls. Before the exact arithmetic of the pass, angles are reduced modulo π and written as fractions with denominators up to 10^6. The second script builds the assignment and colouring lists, shuffles them (Python generator seeds 3 and 5), runs the routing pass and the hop pass of Proposition 15, and checks the results on every assignment and every colouring. Its gate counts need pytket.

Entropy tests. In the same directory, run:

```
python entropy_tests.py
```

This script produces Tables 3, 4 and 6 to 8 and the capacity statements of Sections 9 to 11. It builds the history register and compares it with direct branching for the routing circuit from a basis seed and for both exchange chains. Its two controls, a second sweep of the same path and a ring of four bonds, are declined by the forest test, and in both some word is reached by more than one path. It runs the cross-row program at every size, checks the new exchange block as a unitary, lowers the cost formula on the conditional register, checks the exchange rule of Proposition 19 on every ordering, computes the measurement entropies and the angle scan, and follows the support of the Hubbard step, which needs OpenFermion.

Reconstructing Tables 6 to 8. Generate each cost table and amplitude vector by the recurrence of Section 9. Square the amplitude magnitudes to obtain P, and set U(x) = 1/m!. Sum each distribution in bins of identical integer tour length for the length column, and merge the bins into minimum and other for the success column. Compute each divergence as $\Sigma\, p \log_2(p/u)$. For I(Y;X), take either shortest ordering v, pair the labels v[j] and v[m − 1 − j] to build r, replace each probability by the mean of its value and that of its relabelled tour, and subtract the entropy of this vector from $\log_2$ m!. No angle fit is involved. For Table 8, build the table of log $P(g^{-1}x)$ for all orderings x and relabellings g, draw the samples from P with NumPy generator seed 2026, and run 2,000 trials for each sample count; by symmetry the hidden relabelling can be taken as the identity.

Sample-only decoder. Table 9 is produced by the following script, which needs NumPy.

```
python tour_consensus.py
```

The function consensus_decode takes a batch of sampled tours and a random generator for ties, and returns an undirected cycle. For each map the sampling generator is seeded with the map seed plus 100000, the batch sizes are run in increasing order, and batches are drawn in blocks of 256. The script also scores every cycle by its expected edge counts under the exact output law, which gives the limit of many samples; that diagnostic is computed after the sample-only decisions and does not feed them.

Reconstructing the fixed-entropy test and Table 14. In entropy_accounting_companion.zip, run the following command with NumPy and SciPy installed. The script produces both the small-step results and the four laws and decoder comparisons of Table 14.

OPENBLAS_NUM_THREADS=1 python entropy_budget.py

The training map uses seed 9026 and the evaluation maps use seeds 51000–51031. Seeds 51016, 51019 and 51031 have additional optima. The cost angle is 0.02. For the small-step correction, bracket roots on 21 equally spaced points from −0.20 to 0.20, refine sign-changing brackets with Brent's method and choose the root of smallest absolute value. For endpoint transfer the search interval is [−1, 1]. The random-direction controls use NumPy seed 1, with 200 directions on the training map and 60

per eligible evaluation map. The error-bound objective uses central differences with step 10^{-4}. Decoder trials use seeds 2027 and 2028.

Faults and yield. Tables 10 and 11 are produced by the following script, which needs Qiskit for the one-hot and conditional circuits.

```
python noise_yield.py
```

It enumerates the registers of Proposition 21 from four to eight cities, builds the gate lists, checks each against the tour recurrence without faults, and computes the kept law for zero to six faults with NumPy generator seeds 2031 and 2032 at five and six cities. Single faults are enumerated exactly except for the one-hot register at five cities and the label register at six, where 1,500 are sampled.

Larger maps. Tables 12 and 13 are produced by the following script, which needs NumPy and SciPy and about 3 GB of memory at twelve cities. The sizes can be given as arguments.

```
python scaling_information.py
```

Further maps use the seed 43000 + 100n + s for n cities, with s counted from zero. Tuned angles are found on the benchmark map by a grid of 15 by 13 points followed by a Nelder–Mead search, and the Gibbs law by bisection on b.

Readable positions. Table 5 and the checks of Proposition 20 are produced by the following script, which needs NumPy.

```
python trade_law.py
```

It builds the registers with a label positions for a from 0 to m, finds the largest set of positions with disjoint reading sets, expands the cost formula on each register and checks it on every ordering. The exhaustive search at m = 4 uses NumPy generator seed 12.

Training experiments. Tables 15 to 17 and the pruning, completion and sampled-gradient figures are produced by the following scripts, which need NumPy and SciPy.

```
python entropy_gradients.py
python training_gaps.py
python dontcare_completion.py
python deficit_method.py
```

The first script trains the seven parameters by normalised ascent with the cost angle scaled by 1/50, an initial step of 0.05, growth by 1.3 on success up to 0.1 and halving on failure down to 10^{-4}; it also runs the pruning rankings, the shift-rule check, the fitted neighbour model and the closed training loop from samples. The second trains the six mixer angles alone and decodes with carried-over parameters. The third completes the cost tables on unused codes by least squares on the Walsh coefficients. The fourth produces Tables 16 and 17 and the noise test. Its maps with one shortest route are the first found from seed 61000 at nine and ten cities, from 52000 for the 40 maps of the canonical law and from 51000 for the 29 test maps; the decoding generators are seeded 708 and 709 at nine and ten cities, 801 for the noise test and 900 for Table 17. It takes about twenty minutes, most of it at ten cities.

Capacity audit. Table 20 is produced by the following commands, which need Qiskit, and pytket for the second. The first argument is a clone of QASMBench at the commit given in Section 11.

```
python benchmark_audit.py QASMBench small medium large
python benchmark_baseline.py QASMBench adder_n10 adder_n4 basis_trotter_n4/basis_test_n4
cat_state_n4 fredkin_n3 pea_n5 qpe_n9 simon_n6 toffoli_n3 variational_n4
```

The audit expands user-defined gates to standard gates on at most three wires, drops final measurements, and lists files with earlier measurement, reset or classical control as dynamic.

Simulation on all amplitudes is limited to 20 wires and 3×10^{10} gate-amplitude products; the word-by-word simulator stops above 2^{16} occupied words or 4×10^{7} word visits. Where both simulators ran they agreed on the final and peak counts.

Basis dependence. The statement of Section 10 is produced by the following script, which needs pytket and OpenFermion.

```
python basis_sensitivity.py
```

Non-routing. In the non-routing directory, run:

```
python hubbard.py generate
python hubbard.py qiskit
python hubbard.py pytket
python benchmark.py qiskit
python benchmark.py pytket
```

The two Hubbard drivers consume the same serialised rows (x, z, c) and seed. The generator calls `fermi_hubbard(nx, ny, 1., 4., periodic=False, spinless=False)`, then `jordan_wigner`. The TKET driver performs the extra compiled-state checks described in Section 10. The same selector source is used in both packages. Its SHA-256 digest for `test_common_input.py` is:

```
e722063d1313b5187f43f79222798e7f351f2e0047224189ccba276469afd525
```

Companion materials. compiler_comparison.zip contains the routing compiler, SDK drivers and result logs. nonrouting_benchmark.zip contains the non-routing fixtures, generators, compiler and logs. entropy_accounting_companion.zip contains the one-hot front end, flat pass, lifting, further families and entropy experiments, including entropy_budget.py, the readable-positions script, the training scripts and the capacity audit, each with the log of its run.

QASMBench reproduction (Section 11). Extract the separate supplement qasmbench_coverage_audit.zip. With Qiskit 2.2.3 and NumPy installed, run python get.py in the extracted directory to fetch and verify the 43 original source files against the pinned Git manifest. Run python audit.py there to regenerate results.csv. It records inclusion status for every file and, for each reduction, both widths, both gate counts, occupied outputs and total variation distance.

Depth sweep (Section 9). Extract depth_selection_pilot.zip. Run python depth.py, python totals.py and python certify.py in sequence with NumPy installed. The map seeds are 9022 and 51000–51031. The cost angle is 0.02 and both mixer angles are −0.5. depth.csv records all depths, including noiseless checks against the ideal tour recurrence. decisions.csv reports relative losses against the gate-fault reference. certificates.csv records the earliest depth meeting the scrambling-model bound.

Depth test, independent check. The archive also contains depth_selection.py, a separate implementation of the four-city depth test on a label-register circuit with exact density matrices. Its circuit has 90 CNOTs for each layer and differs from that of the pilot archive, so its counts are close to those of Section 9 and not identical.

Cross-family replication (Tables 21 and 22). Extract cross_family_information.zip. Run OPENBLAS_NUM_THREADS=1 python test.py and then python verify.py with NumPy and SciPy. Seeds 73000–73015 generate the first two cohorts, with assignment seed 73000 excluded by the two-optimum rule. The follow-up graphs use 74000–74015. The grid covers γ from 0 to π and β from $-\pi/2$ to $\pi/2$ including endpoints, in row-major order with a 10^{-12} objective tie tolerance. results.csv and followup.csv contain every selected angle, information component, direct-hit probability and decoder score. The verification script checks evolution by dense matrix exponentiation and the information accounting by direct relative entropies.

Held-out Max-Cut transfer (Table 23). Extract maxcut_transfer_test.zip and run OPENBLAS_NUM_THREADS=1 python transfer.py with NumPy and SciPy. The archive includes test.py and the per-graph results in transfer.csv. Training uses seeds 73000–73015 and testing uses 75000–75031. In table row order, the selected (γ, β) pairs in units of π are (0.25, −0.10), (0.35, −0.35), (1.00, −0.25) and (0.35, −0.40). The decoder sums the joint sample likelihood over all permutations giving each candidate partition, then selects the highest posterior, breaking ties uniformly.
Independent-circuit and reader extension (Section 11.2 and Tables 24–25). Extract scale_calibration_recovery.zip. The README gives commands, version pins, the QASMBench source commit and paths. readers.py repeats the six-vertex queries, full8.py covers all eight-vertex permutations, and scale.py generates the cyclic cohorts. Larger cohorts draw seeds 86100–86111. All use the standardised cost and one-layer mixers of Section 11.1. external.py and wide.py check the unmodified hopping recogniser, and frozen_files.py records the two declined external inputs. Per-instance CSVs and circuit logs are included.

Section 11.3 reproduction. Extract external_routing_and_blind_reader.zip. Run rectangles.py with Qiskit 2.2.3, NumPy and SciPy, and with upstream and vendor on PYTHONPATH. The archive includes the upstream circuit module, its MIT licence, and the frozen routines it imports. Run blind.py with NumPy and SciPy to reproduce the new reader control. The archive includes per-instance results and both execution logs.

Frozen-rule audit. Extract frozen_routing_audit.zip and follow README.txt. The archive supplies the test scripts, logs, source commit and the hash of the published ordered-rule module. The eight-city check uses the original companion's source generators. The second-source test loads the upstream functions without the legacy optimiser and records both cost angles.

Section 11.4 reproduction. Extract external_gate_cost_and_coverage.zip. Its README gives the pinned source paths and execution commands for the gate counts, the source-side control, the membership of the angle grid and the larger compiled-state checks.

Section 11.5 reproduction. For the count-ordered counterexample, use spin_chain_capacity_study.zip. For the topologically ordered rule, extract topological_echo_check.zip, which contains flat_frontend.py and the independent echo driver recheck.py. Run python recheck.py with NumPy and SciPy and compare the output with rerun.json. The driver checks every valid two-site matrix entry, computes the sixteen-site central return, counts entanglers and scheduling slots, and evaluates both noise rates.

Independent checks. We used echo_order_check.py to compare the echo on its valid sector and order_fuzz.py to test random lists of hops and phases. We checked the cross-family and frequency-reader experiments with cross_family_check.py and freq_reader_check.py, and the two external routing sources with external_routing_check.py and second_source_check.py. On four to ten wires, external_hop_check.py compares the recovered program directly with Qiskit's operator. The operators agree to about 10^{-15}.

Appendix B Six small reconstruction tests

Test B1. On two physical wires choose the sequence XX, ZI and YY with angles 0.37, 0.19 and −0.23. Here ZI denotes a Z on the higher-index wire. With seed 00 the only independent flip is 11. Choose that vector as the sole column of B. It maps logical zero to 00 and logical one to 11. Equation (5) gives the signed logical sequence X, Z, −X. In matrix order, acting on the initial logical zero, the result is

$$e^{-0.23iX}\, e^{-0.19iZ}\, e^{-0.37iX}\, (1, 0)^{T}.$$

Insert its two components into entries zero and three of a four-component vector and set entries one and two to zero. Compare that vector with the full two-qubit sequence applied to 00. As a second check, change the seed to 01. The YY sign becomes positive while the Z on the higher wire still acts as logical Z. This checks a nonzero affine offset without changing either physical generator or angle.

Test B2. Give three slots two bits each and seed them with labels 0, 1 and 2. Define two generators as Pauli dictionaries using (6): the first exchanges wire pairs (0,2) and (1,3), and the second exchanges (2,4) and (3,5). Each dictionary has sixteen distinct entries, all with coefficient 1/4. The incidence signatures in wire order are {1}, {1}, {1,2}, {1,2}, {2}, {2}. They recover the three slots without their labels being provided. Each supplied exchange has eigenvalues ±1. With a basis start and only the first bond executed for a test angle t, the two outputs should have amplitudes cos t and −i sin t, up to a common phase.

Now request the uniform orbit for the full two-edge program. The insertion procedure gives six states with equal positive amplitudes $1/\sqrt{6}$. The sum of these states is invariant under either swap, so a circuit containing only swap rotations changes it by a global phase. Insert a diagonal phase on a selected label bit before mixing to test nontrivial interference. Independently update the six amplitudes by the two-term recurrence in Section 9 and compare all six values, as well as the norm on all other outputs.

Counterexample for the certificate. Add $Q = (I - Z_0)(I - Z_1)/4$ to the first candidate generator. Every single-one probe has zero overlap with the computational-basis support of Q, so the proposed pair list is unchanged. But Q acts nontrivially whenever wires zero and one are both set. Equality with the reconstructed swap dictionary must fail, even if the probe stage succeeds. This example distinguishes the exact operator certificate from a heuristic based on probing alone.

Test B3. Place three cities on nine wires, with wire 3p + a set when city a is at position p, and seed city p at position p. Supply the two mixing generators (7) for positions (0,1) and (1,2). Each has three partial swaps and 24 Pauli strings. The supports give the signatures {1}, {1,2} and {2} on the three rows. Probing the first generator on the nine words with one set wire in each of rows 0 and 1 returns zero on the three words that repeat a city and the exchanged word on the other six. The pairing matches wire a of row 0 with wire a of row 1. The sketch sends each row to the two-bit label of its city, giving six label wires, and each generator becomes a sixteen-entry dictionary of Test B2. The label 3 never occurs.

For a diagonal check, take Z on wire 1, which is city 1 at position 0. On slot 0 it becomes the table with value −1 at label 1 and +1 at labels 0, 2 and 3. With Z_0 and Z_1 on the low and high label wires of that slot, the table is $(I + Z_0 - Z_1 + Z_0Z_1)/2$. Unlike the counterexample above, adding Q on wires 0 and 1 to the first mixing generator must leave the output of the front end unchanged: both wires lie in row 0, and no valid word sets two of them.

Test B4. Give three slots the labels 0, 1 and 2 and apply the exchange of slots 0 and 1 with angle a, then the exchange of slots 1 and 2 with angle b, from the basis seed. The two edges form a path. The history

bits 00, 10, 01 and 11 give the orderings 012, 102, 021 and 120, with amplitudes cos a cos b, −i sin a cos b, −i cos a sin b and −sin a sin b. The orderings 201 and 210 are never reached. Two wires, each rotated once, reproduce the distribution on a register of six.

Test B5. With the same three labels, c = 0 ⊕ 1 ⊕ 2 = 3. Store slots 0 and 1 on four wires. The word (0, 1) implies the label 3 ⊕ 0 ⊕ 1 = 2, and the word (2, 0) implies 1. By (9) the exchange of slot 1 with the implied slot adds $\tau = 3 \oplus l_0$ to slot 1. From (0, 1) this gives (0, 2), whose implied label is 3 ⊕ 0 ⊕ 2 = 1, as an exchange of the last two labels should.

Test B6. Write the circuit of Test B3 with a cost generator as single rotations: 21 cost strings, then eight strings for each of the six partial swaps, 69 rotations in all. Regrouping gives seven terms, one diagonal and six rectangle moves. The sides of the rectangles take two colours. One colour joins the wires 3p, 3p + 1 and 3p + 2 of each position p, and the other joins the wires a, 3 + a and 6 + a of each city a. Each of the position pairs (0,1) and (1,2) carries its three moves, so the positions are the slot lines, and the two blocks are the two mixing generators of Test B3. Removing any one of the 48 swap strings leaves a term that is not a rectangle move, and the pass declines.

References

[1] S. Bravyi, J. M. Gambetta, A. Mezzacapo and K. Temme. *Tapering off qubits to simulate fermionic Hamiltonians*. Symmetry-based qubit reduction. arXiv:1701.08213.

[2] S. Hadfield et al. *From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz*. Feasibility-preserving mixing operators. arXiv:1709.03489.

[3] Y. Lin, C. Zheng and C. Guo. *A Resource-Efficient Variational Quantum Framework for the Traveling Salesman Problem*. Compact binary registers and permutation-preserving evolution, preprint, 2026. arXiv:2605.00739.

[4] B. Bakó, A. Glos, Ö. Salehi and Z. Zimborás. *Prog-QAOA: Framework for resource-efficient quantum optimization through classical programs*. Quantum 9, 1663 (2025). doi:10.22331/q-2025-03-20-1663.

[5] M. Amy, P. Azimzadeh and M. Mosca. *On the CNOT-complexity of CNOT-PHASE circuits*. Parity-network synthesis. arXiv:1712.01859.

[6] A. Cowtan, W. Simmons and R. Duncan. *A Generic Compilation Strategy for the Unitary Coupled Cluster Ansatz*. Compilation of commuting Pauli terms. arXiv:2007.10515.

[7] J. R. McClean et al. *OpenFermion: The Electronic Structure Package for Quantum Computers*. arXiv:1710.07629. Operator generation is documented in the OpenFermion tutorial.

[8] T. Hakobyan. *The ordering of energy levels for su(n) symmetric antiferromagnetic chains*. Nuclear Physics B (2004). Publisher record. The permutation form of the local exchange is also described in P. Corboz, SU(N) Heisenberg models.

[9] IBM Quantum Learning. *Simulating nature*. Transverse-field Ising Hamiltonian and simulation setting. Course notes.

[10] IBM Quantum. Rustiq Pauli evolution synthesis. Quantinuum. Pauli synthesis and compiler transforms. Exact versions and invocations for the reported measurements appear in Appendix A.

[11] M. Steudtner and S. Wehner. *Fermion-to-qubit mappings with varying resource requirements for quantum simulation*. New Journal of Physics 20, 063010 (2018). Classical codes for fixed particle number. doi:10.1088/1367-2630/aac54f.

[12] A. Bärtschi and S. Eidenbenz. *Grover Mixers for QAOA: Shifting Complexity from Mixer Design to State Preparation*. Preparation of a uniform superposition of one-hot permutation states. arXiv:2006.00354.

[13] L. Binkowski and M. Schwiering. *Quantum Fisher-Yates shuffle: Unifying methods for generating uniform superpositions of permutations*. arXiv:2504.17965.

[14] N. P. D. Sawaya, A. T. Schmitz and S. Hadfield. *Encoding trade-offs and design toolkits in quantum algorithms for discrete optimization: coloring, routing, scheduling, and other problems*. Quantum 7, 1111 (2023). arXiv:2203.14432.

[15] A. Glos, A. Krawiec and Z. Zimborás. *Space-efficient binary optimization for variational quantum computing*. npj Quantum Information 8, 39 (2022). arXiv:2009.07309.

[16] C. Rotello, E. B. Jones, P. Graf and E. Kapit. *Automated detection of symmetry-protected subspaces in quantum simulations*. Physical Review Research 5, 033082 (2023). arXiv:2302.08586.

[17] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Chapter 2, Entropy, Relative Entropy, and Mutual Information. Wiley. doi:10.1002/0471200611.ch2.

[18] H. Wakaura. *Solving the traveling salesman problem on ⌈log_{2}(N − 1)!⌉ qubits with a variational quantum Kolmogorov–Arnold network*. Lehmer-code register at the floor. arXiv:2509.22752.

[19] S. Bravyi, A. Kliesch, R. Koenig and E. Tang. *Obstacles to Variational Quantum Optimization from Symmetry Protection*. Physical Review Letters 125, 260505 (2020). Recursive variable elimination from measured correlations. arXiv:1910.08980.

[20] J. R. Finžgar, A. Kerschbaumer, M. J. A. Schuetz, C. B. Mendl and H. G. Katzgraber. *Quantum-Informed Recursive Optimization Algorithms*. arXiv:2308.13607.

[21] W. Zhang and M. Looks. *A Novel Local Search Algorithm for the Traveling Salesman Problem that Exploits Backbones*. Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005), pages 343–350.

[22] P. Ma, Z. Xue, Z. Li, Q. Zhang, Y. Du, Z. Wang, Y. Zhou, S. Wang and X. Zhang. *Decompiling Quantum Assembly into Structured Programs*. Lifting of compiled circuits to Pauli rotations. arXiv:2610.05012.

[23] J. Yang, Y. Li and X.-H. Deng. *Representation-Dependent Recoverability in Quantum Compilation*. arXiv:2609.29210.

[24] R. Shaydulin, S. Hadfield, T. Hogg and I. Safro. *Classical symmetries and the Quantum Approximate Optimization Algorithm*. Output probabilities on symmetry orbits. arXiv:2012.04713.

[25] P. Díez-Valle, D. Porras and J. J. García-Ripoll. *Quantum Approximate Optimization Algorithm Pseudo-Boltzmann States*. Physical Review Letters 130, 050601 (2023). arXiv:2201.03358.

[26] S. Hu, V. Lopata, S. Sinno, S. Thuravakkath and P. Zuliani. *A Hybrid Classical-Quantum Annealing Algorithm for the TSP*. Edge frequencies in a pool of tours. arXiv:2605.09616.

[27] M. Streif, M. Leib, F. Wudarski, E. Rieffel and Z. Wang. *Quantum algorithms with local particle number conservation: noise effects and error correction*. arXiv:2011.06873.

[28] L. Botelho et al. *Error mitigation for variational quantum algorithms through mid-circuit measurements*. Post-selection for one-hot and related encodings. arXiv:2108.10927.

[29] R. C. Singleton. *Maximum distance q-nary codes*. IEEE Transactions on Information Theory 10, 116–118 (1964).

[30] P. Cartier and D. Foata. *Problèmes combinatoires de commutation et réarrangements*. Lecture Notes in Mathematics 85, Springer (1969). Words up to exchanges of commuting letters.

[31] Y. Pan, Y. Tong, S. Xue and G. Zhang. Efficient depth selection for the implementation of noisy quantum approximate optimization algorithm. Journal of the Franklin Institute 359, 11273–11287 (2022). doi:10.1016/j.jfranklin.2022.10.027.

[32] M. DeCross, E. Chertkov, M. Kohagen and M. Foss-Feig. Qubit-Reuse Compilation with Mid-Circuit Measurement and Reset. Physical Review X 13, 041057 (2023). doi:10.1103/PhysRevX.13.041057.

[33] J. Yao. Variational quantum state diagonalization with computational-basis probabilities. arXiv:2412.18257 (2024).

[34] H. Wang, D. B. Tan and J. Cong. *Quantum State Preparation Circuit Optimization Exploiting Don't Cares*. International Conference on Computer-Aided Design (2024). arXiv:2409.01418.

[35] J. I. Munro, R. Raman, V. Raman and S. S. Rao. *Succinct Representations of Permutations and Functions*. Theoretical Computer Science (2012). arXiv:1108.1983.

[36] A. Golynski. *Cell probe lower bounds for succinct data structures*. Proceedings of the Twentieth ACM-SIAM Symposium on Discrete Algorithms (SODA 2009), pages 625–634.

[37] Y. Alekseev, M. Göös, K. Myasnikov and D. Sokolov. *Sampling Permutations with Cell Probes is Hard*. Lower bounds for storing a permutation and answering queries for its entries. arXiv:2512.02724.

[38] J. Liu, L. Bello and H. Zhou. *Relaxed Peephole Optimization: A Novel Compiler Optimization for Quantum Circuits*. International Symposium on Code Generation and Optimization (CGO 2021), pages 301–314. arXiv:2012.07711.

[39] Y. Chen and Y. Stade. *Quantum Constant Propagation*. Static Analysis Symposium (SAS 2023), pages 164–189. Restricted simulation with a bounded number of occupied words.

[40] O. Friedewald, A. Shiri Sichani and C.-R. Shyu. Exact Diagonal Completion on Reachable Subspaces: Application to QAOA Placement. arXiv:2609.38734 (2026).
[41] S. Zhong, W. Shen, T. Catanach and X. Huan. Goal-Oriented Bayesian Optimal Experimental Design for Nonlinear Models Using Markov Chain Monte Carlo. DOI 10.1137/24M1649344.
[42] IBM Quantum. Qiskit circuit library, excitation_preserving. Tested with Qiskit 2.2.3. https://quantum.cloud.ibm.com/docs/en/api/qiskit/qiskit.circuit.library.excitation_preserving
[43] Using an A*-based framework for decomposing combinatorial optimization problems to employ NISQ computers. Quantum Information Processing (2023). DOI 10.1007/s11128-023-04115-w. Source: github.com/ekut-es/decomposition-for-quantum, mhp_qaoa.py, commit 89be2126764ff193300eae4b35e283e6eb6357eb.
[44] M. Radzihovsky, J. Murphy and M. Swofford. Stanford CS 269Q final project, travelling salesman problem (2019). Project report and executable source: github.com/murphyjm/cs269q_radzihovsky_murphy_swofford, commit 5b5b9c5d7122b4699e13d6df0af852b704981c74.
[45] S. Hoffman, Y. Chen, H.-P. Cheng and X.-G. Zhang. Universal quantum operation of spin-3/2 Blume-Capel chains. Physical Review A 105, 032604 (2022). https://doi.org/10.1103/PhysRevA.105.032604
[46] Quantinuum. H2 Product Data Sheet, System Fundamentals and Errors, version 4.00, 17 September 2025, Table 1, p. 3. https://docs.quantinuum.com/systems/_static/assets/data_sheets/Quantinuum%20H2%20Product%20Data%20Sheet.pdf