On the Complexity of the Circuit Width Problem

Zhengfeng Ji
Department of Computer Science and Technology, Tsinghua University, Beijing, China

Yinchen Liu
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Zhe'ou Zhou
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Abstract

Montanaro associated to every quantum circuit over the gate set $\{H, Z, \mathrm{CZ}, \mathrm{CCZ}\}$ a degree-three polynomial over $\mathbb{F}_2$, and showed that the strong simulation cost of the circuit is governed by the minimum number of qubit wires needed to realize that polynomial. We study this minimum-width problem. We prove that deciding whether a degree-three polynomial has width at most k is NP-complete. The reduction is parsimonious enough to imply that, for every $\alpha < 49/48$, the corresponding gap version is NP-hard. We also prove the same inapproximability threshold for degree-two polynomials by a twin-copy replacement of the cubic constraints. On the positive side, we give a nondeterministic search algorithm using only $O(k \log(en/k))$ nondeterministic bits, and a deterministic fixed-parameter algorithm running in $k^{6k+o(k)}n + O(m)$ time on an input with n symbols and m monomials.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory; Theory of computation → Problems, reductions and completeness; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases IQP circuits, circuit width, NP-completeness, approximation hardness, parameterized algorithms

Digital Object Identifier 10.4230/LIPIcs.TQC.2026.7

Related Version *Full Version*: `https://arxiv.org/abs/2606.18201` [18]

Funding *Zhengfeng Ji*: National Natural Science Foundation of China (Grant No. 12347104), National Key Research and Development Program of China (Grant No. 2023YFA1009403), Beijing Science and Technology Planning Project (Grant No. Z25110100810000), and Beijing Natural Science Foundation (Grant No. Z220002).
Yinchen Liu: Beijing Natural Science Foundation (Grant No. QY26408).

Acknowledgements We thank the anonymous reviewers for their helpful comments and suggestions, especially on the importance of better characterizing the degree-2 instances used in our hardness reduction and of clarifying the role of potentially tractable special cases.

1 Introduction

Quantum computation is widely believed to outperform classical computation on tasks such as simulating quantum systems [11] and factoring large integers [25]. Since Yao's equivalence theorem [27], the quantum circuit model has been the standard formalism for studying this power. A central theme is to relate circuit amplitudes to classical counting problems: quantum computation can be simulated using counting oracles, giving $\mathsf{BQP} \subseteq \mathsf{PP} \subseteq \mathsf{P}^{\#\mathsf{P}}$, and is closely connected to counting-based classes such as AWPP [3, 12]. Similar counting viewpoints appear in linear optics, Jones-polynomial approximation, Ising partition functions, IQP circuits, and boson sampling [1, 21, 13, 5, 2, 8].

For circuit amplitudes, this connection can be made explicit through an algebraic path-integral representation. Dawson et al. [9] showed that amplitudes of circuits built from Toffoli and Hadamard gates can be expressed as normalized counts of solutions to polynomial equations over $\mathbb{F}_2$. Montanaro [23] refined this framework for the gate set $\mathcal{Z} = \{H, Z, \mathrm{CZ}, \mathrm{CCZ}\}$. Given a circuit $\mathcal{C}$ on w qubits over $\mathcal{Z}$, one obtains a degree-3 polynomial $f_{\mathcal{C}}$ over $\mathbb{F}_2$, with no constant term, such that the amplitude $\langle 0^w|\mathcal{C}|0^w\rangle$ is proportional to $\mathrm{gap}(f_{\mathcal{C}})$, the difference between the number of zeros and ones of $f_{\mathcal{C}}$. Thus a BQP-complete quantity is encoded by a classical gap-counting problem whose cancellations reflect quantum interference.

This representation naturally leads to a width-minimization problem. Montanaro defined the width $w(f)$ of a polynomial f to be the minimum number of qubits in any circuit $\mathcal{C}$ over $\mathcal{Z}$ such that $f_{\mathcal{C}} = f$. If f has n variables and width $w(f)$, then a quantum computer can approximate $\mathrm{gap}(f)$ to absolute error $2^{(n+w(f))/2}/3$. Hence smaller width gives better approximate counting precision. Efficiently computing, or even approximating, $w(f)$ would therefore provide a clean combinatorial route toward approximate counting for quantum amplitudes, and was posed by Montanaro as a natural open problem [23].

In this paper, we undertake a systematic study of the quantum circuit width problem, formally denoted as CircuitWidth(f, k), which asks whether the width $w(f) \leq k$. We show that CircuitWidth(f, k) is NP-complete, and that even approximating the circuit width within a small constant factor remains NP-hard. Furthermore, we extend our hardness results to degree-2 polynomials, demonstrating that both the circuit width problem and its approximate version remain NP-hard. This extension has broad implications because many quantum gate sets yield degree-2 polynomials: for instance, circuits using $\{H, \mathrm{CS}\}$ or Clifford+T gates produce polynomials with at most quadratic terms modulo 4 or 8 depending on the gate set [26]. Thus, our approximation hardness result establishes that the same computational intractability is universal across all these diverse gate sets. Moreover, since degree-2 polynomials without linear terms correspond to graphs, circuit width in this setting can be interpreted as a graph parameter related to treewidth or pathwidth, but carrying a quantum information-theoretic interpretation concerning the arrangement of Hadamard gates and diagonal two-qubit gates. Consequently, our results rule out efficient width optimization across all these practically relevant quantum computing scenarios on classical or quantum computers unless unlikely complexity-theoretic collapses occur. We further complement these results with algorithmic upper bounds, including a nondeterministic search algorithm with succinct witnesses and an efficient fixed-parameter tractable algorithm parameterized by the circuit width.

Our results have several consequences. First, they answer Montanaro's open question by showing that circuit-width minimization is a genuine computational bottleneck in the approximate-counting approach to quantum amplitudes. Second, circuit width provides a natural framework for minimizing qubit usage in quantum computation, with potential relevance to practical implementations and compilation of quantum algorithms. Our hardness results indicate that, in the worst case, even quantum computers cannot solve this problem efficiently or even approximately. Any further progress in this direction must therefore rely on heuristic methods and exploit the structure of circuits for specific algorithms. Finally, our hardness and fixed-parameter results suggest that circuit width behaves more like a structural parameter, analogous to treewidth for tensor-network simulation [22] or linear rank-width for decision-diagram simulation [6], than a direct measure of intrinsic quantum computational power.

1.1 Our results

We summarize our main results on the circuit width problem. The results are twofold: hardness results and algorithmic results.

For the hardness of circuit width, we rule out the possibility of efficiently computing the width of a degree-3 polynomial f with no constant term by proving that the problem of deciding $w(f) \leq k$ (denoted by CircuitWidth(f, k)) is NP-complete.

▶ **Theorem 1.1** (See also Corollary 4.10)**.** CircuitWidth *is* NP*-complete.*

In fact, we are able to prove an inapproximability result: CircuitWidth is even NP-hard to approximate within a multiplicative factor of about 1.02 (even restricted to instances with $k = \Theta(n)$), and hence it is impossible to approximate the width efficiently (in BQP power) within additive $O(\log n)$ error unless NP $\subseteq$ BQP.

▶ **Theorem 1.2** (Informal, see also Theorem 5.3)**.** *For any constant $0 < \epsilon < 1/48$, it is* NP*-hard to distinguish degree-3 polynomials f with no constant term and $w(f) \leq k$ from those with $w(f) \geq (\frac{49}{48} - \epsilon)k$.*

The previous hardness results crucially rely on cubic terms (CCZ gates) to enforce complex geometric constraints. However, many quantum computational models naturally correspond to degree-2 polynomials. A natural question thus arises: does the approximation hardness persist when we restrict to *only* degree-2 polynomials? We answer this affirmatively by showing that approximation hardness persists with the same factor:

▶ **Theorem 1.3** (Informal, see Theorem 5.5)**.** *The approximation hardness of* CircuitWidth *holds for degree-2 polynomials: for any constant $0 < \epsilon < 1/48$, it is* NP*-hard to distinguish quadratic f with $w(f) \leq k$ from those with $w(f) \geq (\frac{49}{48} - \epsilon)k$.*

Finally, from the viewpoint of conditional lower bounds, our reduction from 3-SAT also yields an exponential lower bound under the standard Exponential Time Hypothesis (ETH) [16]:

▶ **Theorem 1.4** (See also Theorem 4.11)**.** *Assuming the Exponential Time Hypothesis* (ETH)*, no algorithm can solve* CircuitWidth(f, k) *in time $2^{o(n)}$.*

On the positive side, we provide a nondeterministic polynomial-time algorithm for the search version of CircuitWidth(f, k) with improved witness complexity:

▶ **Theorem 1.5** (Informal, see Theorem 6.3)**.** *There exists a nondeterministic polynomial-time algorithm that, given a witness of size $2\log_2 \binom{n}{k} = O(k \log(en/k))$ bits, either constructs a quantum circuit on at most k qubits with associated polynomial f, or correctly concludes that no such circuit exists. Enumerating all endpoint witnesses gives a deterministic $\binom{n}{k}^2 \cdot \mathrm{poly}(n)$-time algorithm. When $k = \Theta(n)$, this is $2^{O(n)}$, matching the ETH lower bound in Theorem 1.4 up to constant factors in the exponent.*

Taking k as a constant, this yields an XP algorithm with runtime $n^{2k+O(1)}$. Moreover, from the perspective of parameterized complexity [10, 7], the nondeterministic algorithm can be easily transformed into a placement of CircuitWidth in the parameterized complexity class W[1]. A natural question then arises: is the problem W[1]-complete, or is it fixed-parameter tractable (FPT) with respect to parameter k? We answer the latter affirmatively:

▶ **Theorem 1.6** (Informal, see also Theorem 6.5)**.** CircuitWidth(f, k) *is fixed-parameter tractable with respect to parameter k. There exists an algorithm with runtime $k^{O(k)} \cdot n$ that solves the search version of* CircuitWidth(f, k)*.*

The concrete runtime is $k^{6k+o(k)} \cdot n$, which is faster than the XP algorithm when $k \leq n^{1/4-o(1)}$, making it more efficient and practical for polynomially small k.

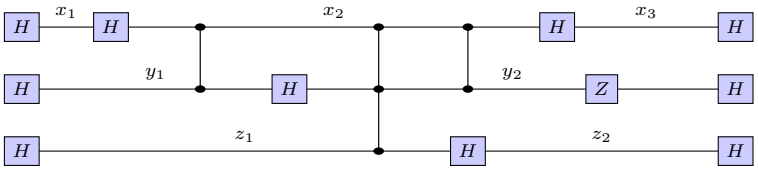

Figure 1 Circuit-to-polynomial map. The displayed circuit contributes $y_2 + x_1x_2 + x_2x_3 + y_1y_2 + z_1z_2 + x_2y_1 + x_2y_2 + x_2y_2z_1$.

1.2 Organization

Section 2 gives the main ideas of the reductions and algorithms. Section 3 records the circuit-polynomial correspondence, the interval model, and the complexity notation. Section 4 gives the 3-SAT reduction, including the main gadget definitions and proof sketches for completeness and soundness. Section 5 derives approximation hardness and the degree-two extension. Section 6 gives the structural characterization, the short nondeterministic witness, and the high-level fixed-parameter algorithm.

2 Technical overview

We first recall the circuit-polynomial correspondence. Normalize a circuit by placing an H gate at the beginning and end of every qubit wire. Each open wire segment between consecutive H gates receives a symbol. A Z gate on symbol u contributes u; a CZ gate between two simultaneously active symbols u, v contributes uv; a CCZ gate contributes uvw; and an internal H gate between adjacent symbols u, v on the same wire also contributes uv. Thus the polynomial has the form

$$f_{\mathcal{C}}(x) = \sum_{u \in \mathcal{S}} \alpha_u u + \sum_{\{u,v\} \subseteq \mathcal{S}} \beta_{uv} uv + \sum_{\{u,v,w\} \subseteq \mathcal{S}} \gamma_{uvw} uvw,$$

with coefficients in $\mathbb{F}_2$.

Hardness overview

For hardness, the important observation is the relation between the number of symbols, the number of internal H gates, and the number of wires. If a circuit has k wires and h internal Hadamard gates, then it has exactly $k + h$ symbols. A quadratic monomial can be supplied either by a CZ gate or by an internal H adjacency. In our reduction the number of symbols and the number of available quadratic monomials are arranged so that achieving the budget $k = 1 + 6m$ is possible only if all relevant quadratic monomials are used as Hadamard adjacencies. Once this happens, the polynomial no longer merely asks for certain overlaps: it forces a nearly rigid partition of the symbols into wires.

We distinguish variables from literals as follows. Symbols x, y, z name Boolean variables, while ℓ_x, ℓ_y, ℓ_z denote the corresponding literals, each either positive or negated. For an occurrence based on variable x, the local wire always uses $x_c, \widehat{x}_c, \overline{x}_c$; the literal ℓ_x is represented by x_c if it is positive and by $\overline{x}_c$ if it is negated. The same convention applies to y and z.

The reduction uses five kinds of objects. There is one base symbol b, whose support can be viewed as spanning the full time interval. For each clause c, a truth-marker wire contains three symbols $t_c, \widehat{t}_c, \widetilde{t}_c$, with t_c serving as the local truth reference. For each literal occurrence, a variable wire contains $x_c, \widehat{x}_c, \overline{x}_c$; the two possible orders of the end symbols

encode the truth value. Synchronization terms force all occurrences of the same Boolean variable to make the same choice. Finally, each clause is checked by two binary OR gadgets: p_c is the output of the first disjunction $\ell_x \vee \ell_y$, and o_c is the final output $p_c \vee \ell_z$ of the whole clause. The cubic term bt_co_c forces this final output to be true.

The proof of soundness has two layers. The first layer is a counting and isolation argument: in a width-$(1 + 6m)$ realization, the base symbol is isolated on one wire, and each clause contributes exactly six three-segment wires: one truth-marker wire, three variable-occurrence wires, and two OR wires. The second layer is geometric. Cubic monomials involving b force the other two symbols to overlap at a common time. Since b can be taken to be active for the whole interval, these terms become vertical overlap constraints. The OR wire can overlap the marker t_c exactly when at least one input literal does, and therefore every clause must be satisfied.

Approximation hardness is obtained by the same construction applied to Max-3-SAT. If a formula leaves Δ clauses unsatisfied, then each such clause must spend one additional wire. Hence the optimum width is $1 + 6m + \Delta$. Håstad's 7/8 inapproximability theorem [14] gives $\Delta \geq (1/8 - o(1))m$, which turns into the threshold $(1 + 6m + m/8)/(1 + 6m) \to 49/48$.

The degree-two result replaces each cubic constraint by many quadratic constraints between twin copies. The key combinatorial fact is a survival lemma: in any near-optimal realization, for each symbol family, many copies are not used as exceptional connectors. Among surviving copies, the complete pairwise quadratic constraints simulate the pairwise intersections that a cubic term would have enforced. Helly's theorem for intervals then recovers a common intersection, so the degree-two instance preserves the original soundness argument.

Algorithmic overview

The algorithms use a structural characterization. Guessing a circuit is the same as partitioning the symbols into k ordered lists, one per wire, and then asking whether all overlaps and Hadamard adjacencies can be scheduled in a common time order. This can be checked by building a directed constraint graph on interval endpoints and testing acyclicity. The nondeterministic algorithm therefore only guesses the sets of first and last symbols. Given the right endpoints, a greedy procedure repeatedly extends an earliest-ending current head; a geometric argument shows that such a head has at most one possible unused neighbor. For the deterministic FPT algorithm, a width-k realization gives a path decomposition of the connection graph of width at most k. We run dynamic programming over a tree decomposition, storing only boundary wire order, coarse information about forgotten subpaths, and a boundary topological order. The transition verification is handled by the boundary invariant described in Section 6.4; the important point for the overview is that the state count is $k^{6k+o(k)}$ per bag.

3 Preliminaries

3.1 Associated polynomials and width

We use the circuit-to-polynomial representation of Montanaro [23]. A circuit over $\{H, Z, \mathrm{CZ}, \mathrm{CCZ}\}$ is first put into a normalized form by adding an H gate at the beginning and end of every qubit wire. A *symbol* is a maximal open wire segment between two consecutive Hadamard gates. We write $\mathcal{S}$ for the set of symbols and $\mathrm{supp}(u) = (L_u, R_u)$ for the open time interval during which u is active.

The associated polynomial $f_{\mathcal{C}} \in \mathbb{F}_2[\mathcal{S}]$ is obtained by adding one monomial for each gate interaction. A Z gate on u contributes u. A CZ gate on overlapping symbols u, v contributes uv. A CCZ gate on symbols u, v, w with a common time intersection contributes uvw. Finally, an internal Hadamard gate between two consecutive symbols u, v on the same wire also contributes the quadratic term uv. All sums are over $\mathbb{F}_2$, so identical monomials cancel in pairs.

▶ **Definition 3.1** (Support and overlap). *The support of a symbol u in a realizing circuit is the interval* $\mathrm{supp}(u) = (L_u, R_u)$. *Two symbols u, v overlap if $L_u < R_v$ and $L_v < R_u$. A set of symbols has a common overlap if the intersection of their supports is nonempty.*

A cubic term requires a common overlap of three intervals. We repeatedly use the following special case of Helly's theorem.

▶ **Theorem 3.2** (Helly's theorem for intervals [15]). *For intervals on the line, if every two intervals in a finite family intersect, then all intervals in the family have a common intersection.*

In particular, for three symbols, pairwise overlap is equivalent to common overlap. This observation is what later allows us to replace cubic constraints by quadratic constraints between twin copies.

▶ **Definition 3.3** (Width). *For a degree-three polynomial $f \in \mathbb{F}_2[\mathcal{S}]$ with no constant term, the* width $w(f)$ *is the minimum number of qubit wires in a normalized circuit $\mathcal{C}$ over $\{H, Z, \mathrm{CZ}, \mathrm{CCZ}\}$ such that $f_{\mathcal{C}} = f$.*

▶ **Definition 3.4** (Decision and search versions). *The decision problem* CircuitWidth(f, k) *asks whether $w(f) \le k$. The search version asks either to output a width-k circuit realizing f, or to certify that no such circuit exists.*

The input polynomial is specified by its nonzero monomials. We write $n = |\mathcal{S}|$ for the number of symbols and m for the number of monomials. In the search version, an output circuit may be represented combinatorially by ordered symbol lists, one list per qubit wire, together with the choice of which quadratic monomials are realized by internal Hadamard gates rather than CZ gates.

3.2 Complexity and algorithmic preliminaries

We use 3-SAT in the standard form where each clause contains three literals. For a formula Φ, let m be the number of clauses and let $\mathrm{OPT}(\Phi)$ be the maximum number of clauses simultaneously satisfiable. The problem Max-3-SAT asks for this optimum.

Håstad's theorem states that, for every $\varepsilon > 0$, it is NP-hard to distinguish satisfiable 3-CNF formulas from formulas in which at most a $(7/8+\varepsilon)$-fraction of clauses are satisfiable [14]. We also use the Exponential Time Hypothesis and the sparsification lemma in the standard consequence that linear-size 3-SAT cannot be solved in $2^{o(m)}$ time unless ETH fails [16, 17].

A parameterized algorithm is fixed-parameter tractable with respect to k if its running time is $g(k)n^{O(1)}$ for some computable function g [10, 7]. We use standard tree-decomposition terminology [24, 4, 7]. A tree decomposition of a graph G is a tree whose nodes carry bags of vertices so that every vertex appears in a connected set of bags and every edge is contained in some bag. The width is one less than the maximum bag size. A path decomposition is the special case in which the decomposition tree is a path. The treewidth $\mathrm{tw}(G)$ is the minimum width of a tree decomposition of G, and the pathwidth $\mathrm{pw}(G)$ is the minimum width of a path decomposition of G.

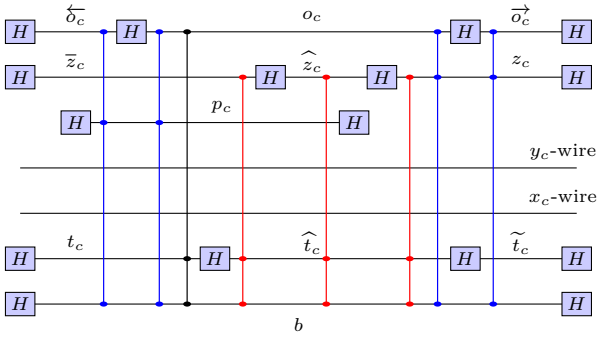

Figure 2 Clause gadget for $c = (\ell_x \vee \ell_y \vee z)$. The figure shows the base and marker wires, the compressed first OR output p_c, the z wire, and the final OR wire. The columns through p_c and o_c are OR-gadget CCZ constraints; the columns through the z wire are marker–literal constraints; and the remaining column through t_c and o_c is the pinning term bt_co_c.

4 The SAT reduction

Let Φ be a 3-CNF formula with clause set C, $|C| = m$. We construct a polynomial f_Φ and set the target width to $k = 1 + 6m$. The target should be read as one global wire plus six local wires per clause. The reduction is designed so that this budget is tight: any attempt to realize a quadratic term by a CZ gate instead of by an internal Hadamard adjacency would consume too much room and force an extra wire. Thus the width bound turns algebraic monomials into a rigid interval geometry.

We use the variable/literal convention from the overview: x, y, z are Boolean variables, while ℓ_x, ℓ_y, ℓ_z are literals. If $\ell_x = x$, its input symbol is x_c; if $\ell_x = \bar{x}$, its input symbol is $\overline{x}_c$, and similarly for y, z.

There is one global base symbol b. Geometrically, b is the anchor that allows a cubic monomial buv to behave like a vertical overlap constraint between u and v: since the base wire may span the entire construction, requiring b, u, v to meet at a common time is equivalent to requiring the two local intervals u and v to meet. This is the mechanism that lets the reduction implement Boolean tests using interval geometry.

For each clause $c = (\ell_x \vee \ell_y \vee \ell_z)$, introduce the eighteen symbols

$$t_c, \widehat{t}_c, \widetilde{t}_c, \qquad x_c, \widehat{x}_c, \overline{x}_c, y_c, \widehat{y}_c, \overline{y}_c, z_c, \widehat{z}_c, \overline{z}_c, \qquad \overleftarrow{p_c}, p_c, \overrightarrow{p_c}, \qquad \overleftarrow{o_c}, o_c, \overrightarrow{o_c}.$$

Let $\mathcal{S}_c$ denote this set of eighteen symbols. The intended six wires for c are the truth-marker wire, the three variable-occurrence wires, and two OR wires. Each of these wires contains three symbols. The middle symbol fixes the local role of the wire, while the two end symbols are the parts whose position relative to the truth marker encodes a Boolean value or the input-output behavior of an OR gate.

4.1 The gadgets

The truth-marker chain is

$$G_{\mathrm{TM}_c} = \widehat{t}_c t_c + \widehat{t}_c \widetilde{t}_c.$$

Global truth-marker synchronization is enforced by

$$G_{\mathrm{TM}} = b \sum_{\{c,d\} \in \binom{C}{2}} (t_c t_d + \widehat{t}_c \widehat{t}_d + \widetilde{t}_c \widetilde{t}_d).$$

For every literal occurrence of a Boolean variable x in clause c, the local variable gadget is

$$G_{x_c} = b\widehat{t}_c(x_c + \widehat{x}_c + \overline{x}_c) + \widehat{x}_c(x_c + \overline{x}_c).$$

For each Boolean variable x, all occurrences are synchronized by

$$G_x = b \sum_{\{(\ell_1,c_1),(\ell_2,c_2)\}\in\binom{\mathrm{Occ}(x)}{2}} (x_{c_1}x_{c_2} + \widehat{x}_{c_1}\widehat{x}_{c_2} + \overline{x}_{c_1}\overline{x}_{c_2}),$$

where $\mathrm{Occ}(x)$ is the multiset of literal occurrences whose underlying Boolean variable is x, and the sum ranges over unordered pairs of distinct occurrences.

The binary OR gadget has input symbols u, v and output symbol q, with two endpoint decorations $\overleftarrow{q}, \overrightarrow{q}$. It is

$$G_{q=u\vee v} = b\overleftarrow{q}\,u + bqu + bqv + b\,\overrightarrow{q}\,v + \overleftarrow{q}\,q + \overrightarrow{q}\,q.$$

For a clause $c = (\ell_x \vee \ell_y \vee \ell_z)$, we set

$$G_c = G_{p_c=\ell_x\vee\ell_y} + G_{o_c=p_c\vee\ell_z} + bt_co_c,$$

where p_c is the intermediate output for $\ell_x \vee \ell_y$, and o_c is the output Boolean value of the whole clause. A positive literal x is represented by x_c, and a negative literal $\bar{x}$ by $\overline{x}_c$. The full polynomial is

$$f_\Phi = G_{\mathrm{TM}} + \sum_x G_x + \sum_{c\in C}(G_{\mathrm{TM}_c} + G_{x_c} + G_{y_c} + G_{z_c} + G_c).$$

The construction uses $1 + 18m$ symbols and has size polynomial in $|\Phi|$. If a clause contains repeated literals, distinct literal occurrences should be read as distinct algebraic symbols; we suppress the corresponding occurrence indices to keep the notation light. Only the local clause gadgets are needed to understand completeness. The global terms G_{TM} and G_x are used in soundness: they prevent a realization from choosing different coordinate systems for different clauses, and they force all appearances of a Boolean variable to agree on which endpoint represents the true literal.

4.2 Gadget behavior

▶ **Lemma 4.1** (Base and clause isolation). *In every realization of f_Φ, the base symbol b lies on a wire with no internal Hadamard gate. Moreover, no wire contains symbols from two distinct sets $\mathcal{S}_c$ and $\mathcal{S}_d$, $c \neq d$. Thus any realization on at most $1 + 6m$ wires must use one isolated base wire and exactly six local wires for each clause.*

▶ **Lemma 4.2** (Wire optimality). *If f_Φ has a realization on at most $k = 1 + 6m$ wires, then the realization uses exactly k wires, has exactly $12m$ internal Hadamard gates, and every quadratic monomial of f_Φ is realized by a Hadamard adjacency rather than by a* CZ *gate.*

▶ **Lemma 4.3** (Good clause structure). *In any realization of f_Φ on at most $1 + 6m$ wires, for every clause c, the eighteen symbols $\mathcal{S}_c$ occupy exactly six wires. Up to reversal of individual three-symbol chains, these wires are precisely*

1. *$t_c, \widehat{t}_c, \widetilde{t}_c$ with $\widehat{t}_c$ in the middle;*
2. *$x_c, \widehat{x}_c, \overline{x}_c$, with $\widehat{x}_c$ between x_c and $\overline{x}_c$, and similarly for y, z;*
3. *each OR triple $\overleftarrow{q}, q, \overrightarrow{q}$ lies on one wire, with $q \in \{p_c, o_c\}$ in the middle.*

The global truth-marker terms force all truth-marker wires to have the same orientation, up to reversing the entire time axis.

▶ **Definition 4.4** (Encoding). *For a clause $c = (\ell_x \vee \ell_y \vee \ell_z)$ satisfying Lemma 4.3, a symbol*

$$s \in \{x_c, \overline{x}_c, y_c, \overline{y}_c, z_c, \overline{z}_c, p_c, o_c\}$$

encodes TRUE *if s overlaps the truth marker t_c, and encodes* FALSE *otherwise. The literal ℓ_x is represented by x_c when $\ell_x = x$ and by $\overline{x}_c$ when $\ell_x = \bar{x}$, and similarly for y, z.*

▶ **Lemma 4.5** (Encoding well-definedness). *In every realization satisfying Lemma 4.3, for each variable occurrence in a clause c, exactly one of x_c and $\overline{x}_c$ overlaps t_c. The same statement holds for the y- and z-occurrence wires.*

▶ **Lemma 4.6** (Variable-value realizability). *For every Boolean variable x:*

(a) *In any realization on at most $1 + 6m$ wires, all occurrences whose underlying variable is x induce the same value: x_c overlaps t_c for every such occurrence c if and only if this common value is* TRUE.

(b) *Given any truth assignment, the truth-marker and variable-occurrence wires can be placed so that all marker–occurrence and synchronization cubic monomials are realized and each occurrence encodes the assigned value.*

▶ **Lemma 4.7** (OR correctness). *For a binary OR gadget $G_{q=u\vee v}$ in a good clause c:*

(a) *In any realization on at most $1 + 6m$ wires, if q overlaps t_c, then at least one of u, v overlaps t_c.*

(b) *Fix the input wires carrying u and v. There is a placement of the $\overleftarrow{q}, q, \overrightarrow{q}$ wire that realizes exactly the quadratic and cubic monomials in $G_{q=u\vee v}$; moreover, if at least one of u, v overlaps t_c, the placement can be chosen with q overlapping t_c.*

▶ **Lemma 4.8** (Clause correctness). *For any good clause $c = (\ell_x \vee \ell_y \vee \ell_z)$:*

(a) *If o_c overlaps t_c, then at least one of the three literal symbols representing ℓ_x, ℓ_y, ℓ_z overlaps t_c.*

(b) *Given fixed literal-occurrence wires, if at least one represented literal overlaps t_c, then the p_c- and o_c-wires can be placed so that all monomials in G_c are realized and o_c overlaps t_c.*

Proof sketches for Lemmas 4.1–4.3 and 4.5–4.8 are deferred to Appendix A.1.

▶ **Theorem 4.9** (Completeness and soundness). *For the polynomial f_Φ constructed above,*

$$\Phi \text{ is satisfiable} \iff w(f_\Phi) \le 1 + 6m.$$

Proof sketch. Completeness follows by placing the variable-occurrence wires according to a satisfying assignment and then applying Lemmas 4.6 and 4.8. For soundness, Lemmas 4.1–4.3 force the intended six-wire structure in every clause; Lemmas 4.5 and 4.6 decode a global assignment; and Lemma 4.8 plus the pinning term bt_co_c show that each clause is satisfied. ◀

▶ **Corollary 4.10.** CircuitWidth *is* NP*-complete.*

Proof sketch. Membership in NP follows because a certificate consists of a partition of symbols into at most k ordered wires and a specification of which quadratic monomials are Hadamard adjacencies; the interval constraints can be checked by a topological-sort test as in Lemma 6.1. Hardness is Theorem 4.9. ◀

▶ **Theorem 4.11.** *Unless* ETH *fails,* CircuitWidth *has no $2^{o(n)}$-time algorithm, where n is the number of symbols, even on instances with $k = \Theta(n)$.*

Proof sketch. By the sparsification lemma [17], it is enough to consider 3-SAT formulas with N variables and $m = O(N)$ clauses. The reduction has $1 + 18m$ symbols and $k = 1 + 6m = \Theta(n)$. A $2^{o(n)}$-time algorithm for CircuitWidth would therefore solve these 3-SAT instances in $2^{o(m)}$ time, contradicting ETH. ◀

5 Approximation hardness and degree-two inputs

5.1 Gap amplification from Max-3-SAT

▶ **Definition 5.1** (Approximation version). *For $\alpha > 1$,* CircuitWidth-α-Approx *asks us to distinguish instances with $w(f) \le k$ from instances with $w(f) \ge \alpha k$, promised that one case holds.*

▶ **Theorem 5.2** (Max-3-SAT reduction). *Let Φ be a 3-CNF formula with m clauses, and let $\mathrm{OPT}(\Phi)$ be the maximum number of satisfiable clauses. The construction above can be realized with $1 + 6m + (m - \mathrm{OPT}(\Phi))$ wires, and no realization can use fewer wires.*

Proof sketch. The extra-wire accounting is deferred to Appendix A.1. Each unsatisfied clause costs one additional wire, and each additional wire can repair at most one clause obstruction. ◀

▶ **Theorem 5.3.** *For every constant $\alpha < 49/48$,* CircuitWidth-α-Approx *is* NP*-hard.*

Proof sketch. By Håstad's theorem [14], it is NP-hard to distinguish satisfiable formulas from formulas for which at least $(1/8 - o(1))m$ clauses remain unsatisfied. The corresponding widths are at most $1 + 6m$ in the first case and at least $1 + 6m + (1/8 - o(1))m$ in the second. The ratio tends to $(6 + 1/8)/6 = 49/48$. ◀

5.2 Removing cubic terms

We now describe the degree-two transformation as an explicit recipe. The input is the cubic polynomial f_Φ from Section 4; every cubic monomial in that polynomial has the form buv, with the global base symbol b and two non-base symbols u, v. Fix a constant $N > 60$ and a sufficiently large polynomially bounded integer M. The degree-two polynomial f'_Φ is built as follows.

- **(i)** Replace b by a base family $B = \{b^1, \dots, b^{Nm}\}$, and replace every other symbol u by a twin family $U = \{u^1, \dots, u^M\}$.
- **(ii)** Add all quadratic terms inside each family. These clique constraints are not meant to simulate gates of the original circuit; they make it expensive for many copies of the same symbol to be used as exceptional connectors.
- **(iii)** For each original quadratic monomial uv, add only the matching terms $u^i v^i$, $1 \le i \le M$. Thus the i-th non-base copies reproduce the original Hadamard-adjacency structure independently for each index i.
- **(iv)** For each original cubic monomial buv, add the three complete bipartite graphs between the corresponding families:

$$\sum_{r,i} b^r u^i + \sum_{r,j} b^r v^j + \sum_{i,j} u^i v^j.$$

This is the only place where a cubic constraint is removed: the required common intersection is replaced by all three pairwise-overlap requirements among copies of b, u, v.

The yes-case realization uses M parallel non-base copies of the original six-wires-per-clause construction, together with the remaining base copies on separate base wires, for baseline width $(6M + N)m$.

The clique and bipartite terms make the replacement robust: near-optimal realizations have many surviving copies, and every surviving copy triple for a replaced term buv satisfies the original common-intersection condition by Helly's theorem. The accounting is deferred to Appendix A.2.

▶ **Lemma 5.4** (Twin survival). *In any near-optimal realization of $f^{(2)}$, all but a bounded number of copies in each family are* surviving*: they are not used as flexible connectors for symbols outside their intended family interactions. For surviving copies, the quadratic replacement of a cubic term buv enforces a common intersection in the same sense as the original cubic term.*

Proof sketch. See Appendix A.2. ◀

▶ **Theorem 5.5.** *For every constant $\alpha < 49/48$,* CircuitWidth-α-Approx *remains* NP*-hard when the input polynomial is restricted to degree at most two.*

Proof sketch. Apply the twin transformation to the gap instances of Theorem 5.2. The surviving copies simulate the original cubic constraints by Lemma 5.4, so a near-optimal degree-two realization induces a near-optimal realization of the original cubic instance, up to the predetermined lower-order slack. Taking M sufficiently large makes this slack negligible compared with the $m/8$ gap from Håstad's theorem [14], preserving every approximation factor below $49/48$. ◀

6 Algorithms

We now turn to the algorithmic side of the width problem. The guiding point is that, once the symbols have been assigned to ordered wire lists, the remaining question is purely one-dimensional: can the active intervals of the symbols be scheduled so that all monomials of the polynomial are realized by the permitted H, Z, CZ, and CCZ gates? This scheduling problem is exactly an acyclicity test for a directed constraint graph. Thus the algorithmic task splits into two parts. First, find or certify a plausible partition of the symbols into k ordered lists. Second, run a topological-sort check on the endpoint constraints induced by these lists.

Let G_f denote the *connection graph* of f. Its vertex set is the symbol set $\mathcal{S}$, and two symbols are adjacent whenever they occur together in a quadratic monomial or as part of a cubic monomial. We use the following harmless cleanup throughout this section. If a quadratic term uv appears while a cubic term uvw also appears for some w, then temporarily delete the quadratic copy. Any realization of uvw already places u, v, w on three distinct wires with a common overlap, and hence the deleted term can be restored at the end by placing a CZ gate inside that overlap. After this cleanup, a quadratic term that is used as a direct adjacency on a wire is never simultaneously forced as a subterm of a CCZ constraint.

6.1 Checking fixed ordered lists

Suppose first that we are given ordered lists $L_1, \ldots, L_k$, one for each wire. The lists include only the real symbols; conceptually, each list is also bracketed by a local source and a local sink, which are identified with global endpoint vertices S and E. Consecutive real symbols on a list are the only pairs that may be separated by an internal H gate. Therefore every

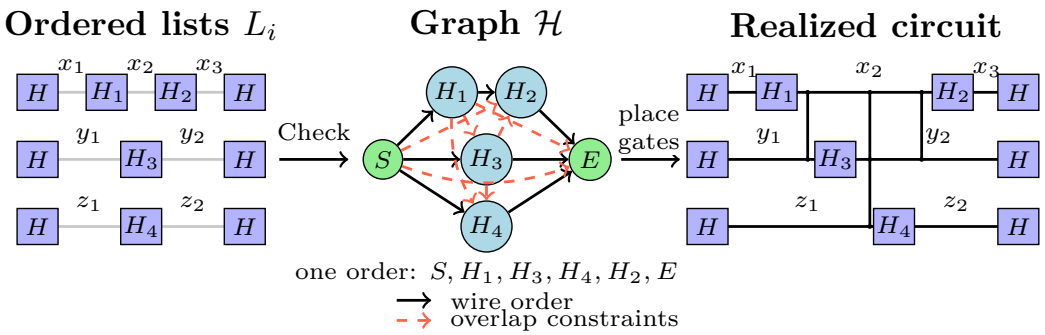

Figure 3 The CHECK routine for fixed ordered lists. Left: the proposed wire lists. Middle: the endpoint digraph $\mathcal{H}$, where black edges encode wire order and dashed red edges encode overlap constraints from monomials. Right: any topological order of $\mathcal{H}$ gives gate positions and hence a realizing circuit with H, CZ, and CCZ gates placed consistently.

consecutive real pair (u, v) must correspond to a quadratic monomial uv of f. Quadratic monomials not used in this direct way are interpreted as CZ-overlap constraints, and cubic monomials are interpreted as CCZ-overlap constraints.

For fixed lists we build a directed graph $\mathcal{H}$. Its vertices are endpoint or link vertices, representing the positions of H gates. For a symbol u, write $L(u)$ and $R(u)$ for the link immediately before and immediately after u on its list. If u and v are consecutive on a wire, then $R(u) = L(v)$. The wire order gives the directed path from S through the links on each list to E. A remaining quadratic monomial uv adds the two precedence constraints

$$L(u) \to R(v), \qquad L(v) \to R(u),$$

which are exactly the inequalities saying that the intervals of u and v overlap. A cubic monomial uvw adds the same two constraints for each of the three pairs. The construction is illustrated in Figure 3.

▶ **Lemma 6.1** (Structural characterization). *A polynomial f has a width-k realization with prescribed ordered lists $L_1, \ldots, L_k$ if and only if every adjacent real-symbol pair in the lists is a quadratic monomial of f, and the directed graph $\mathcal{H}$ constructed above is acyclic. Given the lists, the check and the corresponding circuit construction take $O(m + n)$ time.*

Proof sketch. A realizing circuit gives real times to all H gates, so every wire-order edge and every monomial-induced precedence edge in $\mathcal{H}$ points forward in time. Thus $\mathcal{H}$ is acyclic. Conversely, suppose $\mathcal{H}$ is acyclic and choose a topological order. This order assigns times to all endpoints, so each symbol u occupies the interval $(L(u), R(u))$. The two directed inequalities for a quadratic monomial uv are equivalent to the intersection of these two intervals, allowing a CZ gate unless uv was already used as a direct H-adjacency. For a cubic monomial uvw, the six directed inequalities force the three intervals to be pairwise intersecting; by Helly's theorem for intervals (Theorem 3.2), they have a common intersection, where we place a CCZ gate. Linear terms are realized by Z gates inside their symbol intervals. The graph has $O(n)$ vertices and $O(m + n)$ edges, so Kahn's topological-sort algorithm [19] gives the stated running time and also constructs the schedule. ◀

We denote this linear-time subroutine by $\textsc{Check}(f, L_1, \ldots, L_k)$. The rest of the section explains how to find suitable ordered lists. The nondeterministic algorithm guesses only the endpoints of the lists and reconstructs the interior greedily; the deterministic algorithm stores the same information locally in a tree-decomposition dynamic program.

6.2 Endpoint witnesses and greedy reconstruction

The witness only names the two endpoints of every wire. Namely, it gives a set S_{head} of the k leftmost symbols and a set S_{tail} of the k rightmost symbols, using $2\log_2\binom{n}{k} = O(k\log(en/k))$ nondeterministic bits. The rest of the wire order is then forced greedily: start one partial list at each head, keep CurrentHeads as the current right endpoints of unfinished lists, and let U be the symbols not yet placed.

In each round, first close any list whose current endpoint already lies in S_{tail}. Then look for an open endpoint v that has exactly one neighbor u in U in the connection graph G_f. Such a neighbor is the only possible next symbol on that wire, so append u, replace v by u in CurrentHeads, and remove u from U. If unplaced symbols remain but no endpoint has a unique unplaced neighbor, the guessed endpoints are rejected. After all symbols are placed, CHECK verifies the resulting ordered lists. Figure 4 illustrates this greedy reconstruction on the example from Figure 3.

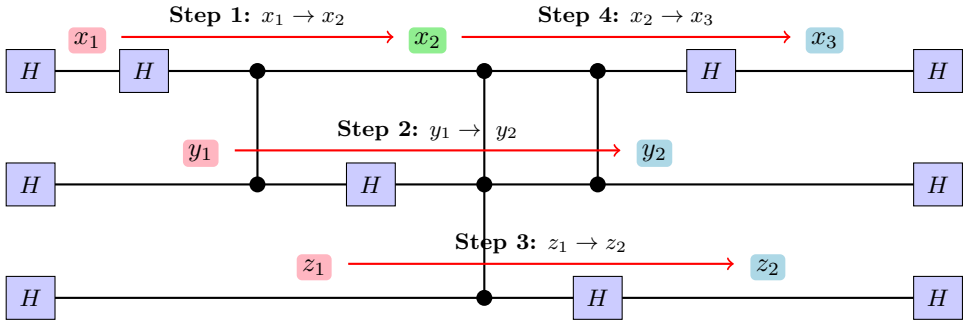

Figure 4 Greedy reconstruction from endpoint sets. For $f_0 = x_1x_2 + x_2x_3 + y_1y_2 + z_1z_2 + x_2y_1 + x_2y_2 + x_2y_2z_1$, the witness says that the three wires start at x_1, y_1, z_1 and end at x_3, y_2, z_2. The algorithm repeatedly extends a current endpoint by its unique unplaced neighbor in G_f, producing $L_1 = (x_1, x_2, x_3)$, $L_2 = (y_1, y_2)$, and $L_3 = (z_1, z_2)$, and then calls CHECK to verify the lists.

The unique-neighbor property is the reason the endpoint witness is enough.

▶ **Lemma 6.2** (Unique neighbor and greedy extension). *If S_{head} and S_{tail} are exactly the symbols at the start and end of each wire in some valid width-k circuit realizing f, then after the procedure closes the current heads in S_{tail}, at every nonterminal step there exists an open current head $v \in$ CurrentHeads with exactly one neighbor among the untouched symbols in G_f. This neighbor is the true next symbol on the same wire. Consequently, the greedy procedure with the correct witness reconstructs ordered lists compatible with a valid realization.*

Proof sketch. Maintain the invariant that each partial list built by the algorithm is a prefix of a true wire list. The closing step is valid because a true tail has no successor on its own wire. After the closing step, every current head is an open head. Choose one whose active interval ends earliest in the realizing circuit. Its true successor is an untouched neighbor because an internal H gate between consecutive symbols contributes the corresponding quadratic monomial. No untouched symbol on another wire can also be adjacent to the earliest-ending head: such an adjacency would require overlap, but every untouched symbol starts to the right of the current frontier on its own wire, whose endpoint is no later than the start of that symbol. Thus the only untouched neighbor is the true successor. Induction proves the lemma. ◀

▶ **Theorem 6.3** (Endpoint witness). *For the search version of* CircuitWidth(f, k)*, there is a nondeterministic polynomial-time algorithm using* $2\log_2 \binom{n}{k} = O(k \log(en/k))$ *nondeterministic bits. Enumerating all endpoint witnesses gives a deterministic* $\binom{n}{k}^2 \mathrm{poly}(n)$*-time algorithm.*

Proof sketch. The witness is the pair $(S_{\mathrm{head}}, S_{\mathrm{tail}})$. With the correct pair, Lemma 6.2 reconstructs compatible lists, and Lemma 6.1 then constructs a realizing circuit. With an incorrect pair, the algorithm accepts only if Check verifies an actual width-k realization, so every accepting branch is sound. Deterministically trying all two endpoint sets gives the claimed running time. ◀

6.3 From greedy reconstruction to bounded pathwidth

The same greedy view also gives the structural observation used by the fixed-parameter algorithm: if the circuit exists, then G_f has a path decomposition with bags of size at most $k+1$. Intuitively, the moving set of current heads is a separator between the symbols already reconstructed and the symbols not yet touched.

▶ **Lemma 6.4** (Path decomposition from a realization). *If* $w(f) \le k$*, then* $\mathrm{pw}(G_f) \le k$*, and hence* $\mathrm{tw}(G_f) \le k$*.*

Proof sketch. Run the greedy procedure using the true endpoint sets. Keep an auxiliary set S_t of size k, retaining the tail symbol of a wire even after that wire is closed. Let S_0 be the head set. Whenever the greedy process replaces a head v by its successor u, set $S_t = (S_{t-1} \setminus \{v\}) \cup \{u\}$. Use bags $B_0 = S_0$ and $B_t = S_{t-1} \cup S_t$. Each bag has size at most $k+1$, so the width is at most k. Each symbol appears in a contiguous subsequence of bags. A direct H-adjacency is covered by the transition bag where the successor is introduced. An overlap edge (coming from CZ or CCZ gates) cannot jump over the frontier; otherwise an earliest-ending current head would have an untouched neighbor other than its true successor, contradicting Lemma 6.2. Therefore every edge of G_f is covered, and the bags form a path decomposition. ◀

Given the polynomial, we first build G_f in $O(m)$ time. We then run, for example, Korhonen's single-exponential 2-approximation algorithm for treewidth [20]. If it certifies that $\mathrm{tw}(G_f) > k$, then we reject by Lemma 6.4. Otherwise we obtain a tree decomposition of width $O(k)$ and convert it into a standard nice tree decomposition with leaf, introduce, forget, and join nodes, as recalled in the full version [18].

6.4 Dynamic programming on a tree decomposition

We now describe the explicit dynamic program. Let x be a node of the nice tree decomposition, let B_x be its tree bag, and let V_x be the set of symbols appearing in the subtree rooted at x. The table $\mathcal{D}_x$ stores *realizable states*. Each state encodes a partial wire assignment and a boundary topological order for the symbols in B_x. A state is realizable if it can be extended to a valid circuit configuration using all symbols in V_x.

Intuitively, a valid circuit configuration for V_x consists of:

- **wire assignments:** a partition of symbols in V_x into k ordered lists $L_1, \ldots, L_k$;
- **edge types:** an assignment ρ describing, for each pair of consecutive symbols on the same list, whether they are adjacent by a direct internal H gate or are not connected inside the current partial solution;

such that the induced partial endpoint graph $\mathcal{H}[V_x(\rho)]$ is acyclic. This captures exactly the structure needed to realize the symbols in V_x: the lists determine the wire assignment, the edge types encode local connectivity, and acyclicity of $\mathcal{H}[V_x(\rho)]$ ensures that the relevant H gates can be consistently ordered in time. At the root, where $V_x = \mathcal{S}$, such a configuration is precisely a complete circuit layout.

A state in $\mathcal{D}_x$ has two components $(\mathcal{W}_x, \mathcal{O}_x)$:

1. $\mathcal{W}_x = (W_1, \ldots, W_k, \tau)$. This is a partition of the boundary symbols in B_x into k ordered lists W_i, each with endpoint sentinels s, e, together with an edge type τ on consecutive entries of each W_i:
   - $\tau(u, v) = 1$: real symbols u, v are adjacent on the wire, so they are connected by a direct internal H gate;
   - $\tau(u, v) = 2$: the two visible entries are connected through forgotten real symbols in $V_x \setminus B_x$, and each consecutive real-symbol pair on that hidden path is connected by a direct internal H gate;
   - $\tau(u, v) = 0$: the two visible entries are not connected inside the current subtree.
2. $\mathcal{O}_x$. This is a topological ordering of the boundary endpoint vertices

$$\{S, E\} \cup \{L(u), R(u) : u \in B_x\},$$

after the identifications induced by type-1 pairs, with S first and E last.

The formal realizability condition is the following. A state $(\mathcal{W}_x, \mathcal{O}_x)$ is stored in $\mathcal{D}_x$ if and only if there exist ordered lists $L_1, \ldots, L_k$ partitioning V_x and edge types $\rho \in \{0, 1\}$ on consecutive entries in each L_i such that:

- **(i)** each boundary list W_i is a subsequence of L_i with order preserved;
- **(ii)** τ extends to ρ, meaning that type-0 and type-1 boundary claims remain unchanged, while a type-2 boundary relation is realized by a nonempty path of type-1 direct H-adjacencies through forgotten real symbols;
- **(iii)** the partial constraint graph $\mathcal{H}[V_x(\rho)]$ is acyclic, and $\mathcal{O}_x$ is the restriction of one of its topological orderings to the boundary vertices $\{S, E\} \cup \{L(u), R(u) : u \in B_x\}$.

Following the standard bottom-up dynamic-programming paradigm on nice tree decompositions [7], the four nice-node transitions are local. At a leaf node, the unique state has empty boundary lists and the order $S < E$. At an introduce node, a new symbol is inserted into one visible type-0 gap of one boundary wire; the transition chooses the two new boundary types, inserts $L(u), R(u)$ into the boundary order, and checks all monomial constraints that have become visible. At a forget node, the symbol is removed from the boundary list after all constraints that would become internal have been checked; the two exposed boundary gaps are merged, recording type 2 if a hidden direct-H path has been completed. At a join node, two child states are compatible only if they agree on the visible wire order and on the boundary topological order; the type information is then merged. The common boundary order certifies acyclicity of the union, because any directed cycle would either already lie in one child or would force two boundary vertices to appear in opposite orders. For details, see the full version [18].

At the root r, the bag is empty. The algorithm accepts exactly when $\mathcal{D}_r$ is nonempty. In that case, backpointers recover the ordered lists and the direct-H adjacencies, and a final call to Check from Lemma 6.1 constructs the actual circuit. If $\mathcal{D}_r$ is empty, then no valid width-k circuit exists.

▶ **Theorem 6.5** (Fixed-parameter algorithm). *Given a degree-three polynomial f on n symbols with m monomials and an integer k, the search version of* CircuitWidth(f, k) *can be solved in $k^{6k+o(k)}n + O(m)$ time.*

Proof sketch. By Lemma 6.4, the treewidth preprocessing is sound: if the connection graph has treewidth greater than k, no width-k realization exists; otherwise we obtain a nice decomposition of width $O(k)$. For each bag, the number of possible ordered boundary wire structures, three-valued type assignments, and boundary endpoint orders is $k^{6k+o(k)}$. The introduce, forget, and join transitions are bounded by the same factor, and the number of nice bags is linear in n. The DP invariant above proves by induction that a state is stored exactly when it has a compatible extension over the processed subtree. Hence a nonempty root table yields a circuit, recovered by backpointers and Lemma 6.1, while an empty root table certifies that none exists. Building G_f and performing the cleanup take $O(m)$ time, so the total running time is $k^{6k+o(k)}n + O(m)$. ◀

7 Conclusion

We proved that the width of the polynomial representation of IQP-type circuits is computationally hard to determine and hard to approximate below 49/48, with the same approximation hardness persisting for degree-two inputs. At the same time, the problem has a strong structural side: width-k realizations force low pathwidth in the connection graph, which leads to compact nondeterministic witnesses and to a deterministic FPT algorithm. The main open directions are to close the approximability gap, improve the dependence on k in the dynamic program, and identify important circuit families for which width can be optimized efficiently.

References

1 Scott Aaronson. A linear-optical proof that the permanent is #p-hard. *arXiv preprint*, 2011. arXiv:1109.1674.

2 Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 333–342, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1993636.1993682.

3 Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997. doi:10.1137/S0097539795293639.

4 Hans L Bodlaender. A tourist guide through treewidth. *Acta cybernetica*, 11(1-2):1–21, 1993. URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3417.

5 Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proc. R. Soc. A*, 467(2126):459–472, 2011.

6 Bin Cheng, Ziyuan Wang, Ruixuan Deng, Jianxin Chen, and Zhengfeng Ji. Breaking the treewidth barrier in quantum circuit simulation with decision diagrams. arXiv:2510.06775, 2025.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

8 Alexander M. Dalzell, Aram W. Harrow, Dax Enshan Koh, and Rolando L. La Placa. How many qubits are needed for quantum computational supremacy? *Quantum*, 4:264, May 2020. doi:10.22331/q-2020-05-11-264.

9 Christopher M. Dawson, Andrew P. Hines, Duncan Mortimer, Henry L. Haselgrove, Michael A. Nielsen, and Tobias J. Osborne. Quantum computing and polynomial equations over the finite field z2. *Quantum Info. Comput.*, 5(2):102–112, March 2005. doi:10.26421/QIC5.2-2.

10 Rodney G Downey and Michael R Fellows. *Fundamentals of Parameterized Complexity*, volume 4. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

11 R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6–7):467–488, 1982. doi:10.1007/BF02650179.

12 Lance Fortnow and John Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59(2):240–252, 1999. doi:10.1006/jcss.1999.1651.

13 Keisuke Fujii and Tomoyuki Morimae. Commuting quantum circuits and complexity of ising partition functions. *New Journal of Physics*, 19(3):033003, March 2017. doi:10.1088/1367-2630/aa5fdb.

14 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001. doi:10.1145/502090.502098.

15 Ed. Helly. Über mengen konvexer körper mit gemeinschaftlichen punkte. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1923. URL: http://eudml.org/doc/145659.

16 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

17 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

18 Zhengfeng Ji, Yinchen Liu, and Zhe'ou Zhou. On the complexity of the circuit width problem, 2026. arXiv:2606.18201.

19 A. B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962. doi:10.1145/368996.369025.

20 Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–192. IEEE, 2021. doi:10.1109/FOCS52979.2021.00026.

21 Greg Kuperberg. How hard is it to approximate the jones polynomial? *Theory of Computing*, 11(6):183–219, 2015. doi:10.4086/toc.2015.v011a006.

22 Igor L. Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008. doi:10.1137/050644756.

23 Ashley Montanaro. Quantum circuits and low-degree polynomials over F2. *Journal of Physics A: Mathematical and Theoretical*, 50(8):084002, January 2017. doi:10.1088/1751-8121/aa565f.

24 Neil Robertson and Paul D Seymour. Graph minors. i. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983. doi:10.1016/0095-8956(83)90079-5.

25 Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. doi:10.1137/S0097539795293172.

26 Ziyuan Wang, Bin Cheng, Longxiang Yuan, and Zhengfeng Ji. FeynmanDD: Quantum Circuit Analysis with Classical Decision Diagrams. In *Proceedings of the 37th International Conference on Computer Aided Verification*, Zagreb, Croatia, July 2025.

27 Andrew Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361, 1993. doi:10.1109/SFCS.1993.366852.

A Technical details deferred from the main text

A.1 Deferred SAT-gadget checks

Base isolation

The symbol b appears only in cubic monomials. If an internal Hadamard gate were adjacent to b, it would contribute a quadratic monomial bs. Since the two adjacent supports are disjoint, the same monomial cannot also be produced by a CZ gate and canceled over $\mathbb{F}_2$. This contradicts the definition of f_Φ. The same argument applies to a Hadamard transition between symbols from two different clauses, because no cross-clause quadratic monomial is present in f_Φ. Hence clause symbol sets cannot share wires, which proves Lemma 4.1.

Wire optimality and good clauses

The construction has $1 + 18m$ symbols. A circuit with w wires and h internal Hadamard gates has $w + h$ symbols, so $w \le 1 + 6m$ implies $h \ge 12m$. On the other hand, f_Φ has exactly $12m$ quadratic monomials: two for each of the six local chains in every clause. Each internal Hadamard gate contributes one quadratic monomial, and an adjacent pair of symbols has disjoint supports, so this monomial cannot be canceled by a CZ gate. Hence $h \le 12m$, and equality proves Lemma 4.2.

Together with Lemma 4.1, this forces the two quadratic monomials in each intended chain to be Hadamard adjacencies. A wire containing four local symbols would create a length-three Hadamard chain and hence a quadratic monomial not present in f_Φ. Thus each clause needs at least six local wires, and the global budget leaves exactly six. The listed quadratic pairs then uniquely force the six three-symbol chains. The terms in G_{TM} align the roles $t, \widehat{t}, \widetilde{t}$ across clauses, up to one global time reversal, proving Lemma 4.3.

Encoding and synchronization

The marker–occurrence cubic monomials

$$b\widehat{t}_c x_c, \qquad b\widehat{t}_c \widehat{x}_c, \qquad b\widehat{t}_c \overline{x}_c$$

force $\widehat{t}_c$ to overlap all three symbols $x_c, \widehat{x}_c, \overline{x}_c$. In the good-clause structure, x_c and $\overline{x}_c$ lie on opposite sides of $\widehat{x}_c$, while $\widehat{t}_c$ lies between t_c and $\widetilde{t}_c$. Hence exactly one of $x_c, \overline{x}_c$ overlaps t_c, proving Lemma 4.5.

The truth-marker synchronization terms force every pair of truth-marker wires to have the same role order, up to the same global reversal. For two occurrences of x, the terms

$$b x_{c_1} x_{c_2}, \qquad b\widehat{x}_{c_1} \widehat{x}_{c_2}, \qquad b\overline{x}_{c_1} \overline{x}_{c_2}$$

force the two occurrence wires to have the same orientation relative to their truth-marker wires. Thus the overlap pattern of x_c and $\overline{x}_c$ with t_c is independent of the clause. Conversely, given an assignment, choose the common truth-marker orientation and orient each occurrence wire according to the assigned value. The same overlaps realize the required marker–occurrence and synchronization terms, proving Lemma 4.6.

OR and clauses

In $G_{q=u\vee v}$, the quadratic terms $\overleftarrow{q}\, q$ and $\overrightarrow{q}\, q$ place q between its two decorated endpoints. The cubic terms involving u force the junction between $\overleftarrow{q}$ and q to lie in $\mathrm{supp}(u)$, and the terms involving v force the junction between q and $\overrightarrow{q}$ to lie in $\mathrm{supp}(v)$. Thus $\mathrm{supp}(q)$ is

contained in the interval spanned by $\mathrm{supp}(u) \cup \mathrm{supp}(v)$. If neither input overlaps t_c, this interval cannot cross t_c, so q cannot overlap t_c. Conversely, if one input meets t_c, put the corresponding junction point in that intersection and the other junction point in the other input support. This realizes the four cubic terms and makes q overlap t_c, proving Lemma 4.7.

The clause gadget is the cascade $p_c = \ell_x \vee \ell_y$ and $o_c = p_c \vee \ell_z$. Applying Lemma 4.7 to the two OR gadgets gives both directions of Lemma 4.8; the final monomial bt_co_c is realizable exactly when o_c overlaps t_c, since b is active for the whole interval.

Gap accounting

Let $\Delta = m - \mathrm{OPT}(\Phi)$. An assignment satisfying $m - \Delta$ clauses is realized by the intended six wires on every satisfied clause. For each unsatisfied clause, one extra wire realizes the final output constraint locally without making o_c overlap the truth marker, giving $1+6m+\Delta$ wires. Conversely, a realization using $1 + 6m + r$ wires has the width-optimal six-wire form on all but at most r clauses. On those good clauses, the OR constraints and bt_co_c imply satisfaction under the decoded assignment. Hence each extra wire can repair at most one clause obstruction, proving Theorem 5.2.

A.2 Degree-two twin replacement

The key point is that a clique of quadratic constraints leaves almost all copies geometrically usable. If three clique copies lie on one wire in order $a - b - c$, then ac cannot be realized; if two wires each contain two clique copies, the earlier internal H-adjacency makes a crossed pair disjoint. Thus at most two copies in such a clique can share a wire with another clique copy. For a replaced cubic term buv, the family cliques and the three complete bipartite graphs form a clique on the copies of b, u, v. Hence at least $Nm - 2$ base copies and $M - 2$ copies of each non-base symbol survive this term. Union-bounding over all cubic-term families gives at least $M - 20$ globally surviving copies of every non-base symbol and at least $Nm - 60m$ globally surviving copies of b: the worst non-base losses are 20 for truth markers, 8 for variable-occurrence symbols, and 10 for OR symbols, while the base losses are $6 + 18m + 6n + 16m + 2m \le 60m$, using $n \le 3m$. Fix a clause c. The sharper local count loses at most $26 + 48 + 22 = 96$ indices among its 18 local symbols, so at least $M - 96$ indices j make all copies s^j, $s \in \mathcal{S}_c$, globally surviving. Since $N > 60$, choose one globally surviving base copy b^r. For every original cubic term buv, the quadratic replacement forces b^r, u^j, v^j to pairwise intersect; by Theorem 3.2, they have a common intersection, exactly as the original CCZ constraint required. Finally, distinct surviving base copies and distinct surviving clause copies cannot share wires: a consecutive shared-wire pair would either have no matching quadratic term or contradict survival for a tripartite replacement. Thus each surviving clause copy uses six wires if it is good and at least seven otherwise. If g is the number of good surviving clause copies, then

$$w(f'_\Phi) \ge Nm - 60m + 7\sum_c |J_c| - g.$$

A circuit with at most $(7M + N)m - tM - 750m$ wires would force $g > tM$, so more than t clauses have a good copy. The decoding lemmas from the cubic construction then give an assignment satisfying more than t clauses. This proves Lemma 5.4 and the degree-two gap.

A.3 Checking a fixed wire order

We also record the constructive content of Lemma 6.1. Suppose the symbols have already been partitioned into ordered lists $L_1, \ldots, L_k$. Consecutive symbols on a list are the only candidates for internal Hadamard adjacencies. Therefore the first check is purely algebraic: if two consecutive genuine symbols u, v are declared adjacent, then the quadratic monomial uv must appear in f. Otherwise the wire order would create a quadratic term that is not in the target polynomial.

After this check, all remaining monomials are interpreted as overlap requirements. For every symbol u introduce two endpoint vertices $L(u), R(u)$ and the edge $L(u) \to R(u)$. If u, v are consecutive on a wire and are realized by an internal Hadamard adjacency, identify $R(u) = L(v)$. A remaining quadratic monomial uv requires the two intervals to intersect, and this is equivalent to the two precedence constraints $L(u) \to R(v)$ and $L(v) \to R(u)$. A cubic monomial uvw requires a common intersection of three intervals; for intervals, by Helly's theorem, this is equivalent to the three pairwise intersections, so we add the same two edges for each of the three pairs.

If the resulting directed graph has a cycle, then no assignment of real endpoint times can satisfy all precedence constraints. Conversely, if it is acyclic, choose any topological order and place endpoints according to that order, with a small perturbation to separate equalities except those intentionally identified by Hadamard adjacencies. The added edges guarantee that every required pair of intervals overlaps. For every cubic monomial, the three pairwise overlaps imply a common overlap, so a CCZ gate can be placed at a common time. Quadratic monomials not used as Hadamard adjacencies are realized by CZ gates at their overlap times, and linear monomials are realized by Z gates. This produces a circuit realizing exactly f.

The construction is linear in the input size once the monomials are stored as incidence lists. The endpoint graph has $O(n)$ vertices and $O(m + n)$ edges, because each monomial contributes only constantly many precedence edges. A single topological sort either returns a schedule or certifies impossibility. This is the subroutine used by both the nondeterministic reconstruction and the final reconstruction step of the FPT algorithm.

A.4 Greedy reconstruction from endpoints

We expand the proof of Lemma 6.2. Assume that the guessed heads and tails are the true sets of first and last symbols of the wires in some valid realization. During the greedy procedure, maintain the invariant that every partial list already constructed is a prefix of the corresponding true wire. Equivalently, the current heads form a cut through the realizing circuit: all assigned symbols lie weakly to the left of this cut on their wires, and all unassigned symbols lie to the right.

At the beginning of each iteration, the algorithm closes every current head that belongs to S_{tail}. This is safe for the reconstruction step: such a symbol has no successor on its own wire, while any remaining overlap edges incident to it will be verified later by the independent check. If unassigned symbols remain, then some unclosed wire still has an unassigned successor.

Choose, among the remaining current heads, a head v whose right endpoint is earliest in the true schedule, and let u be its true successor. The pair v, u is an internal Hadamard adjacency, hence an edge of the connection graph, so u is an unassigned neighbor of v. We claim that u is the only such neighbor. If there were another unassigned neighbor z, then some monomial would force the interval of z to overlap the interval of v. But z lies to the

right of the current cut on its own wire. Its wire therefore has a current head whose right endpoint must occur no later than the left endpoint of z. Since z overlaps v, this would force that current head to end before or at the same time as v, contradicting the choice of v as the earliest-ending available head unless z is exactly the successor on the same wire.

Thus the selected open head has exactly one unassigned neighbor, and the greedy rule is forced to append its true successor. The prefix invariant is preserved, and the number of unassigned symbols decreases after each extension. Induction proves that with the correct endpoint witness the greedy procedure never gets stuck and reconstructs wire lists compatible with the original realization.

For soundness, no trust is placed in the witness. The greedy procedure may construct lists from an incorrect pair of endpoint sets, but the algorithm accepts only after the independent check of Lemma 6.1 succeeds. Thus any accepting branch outputs an actual width-k realization. The witness size is exactly the information needed to specify the two endpoint sets, namely $2\log_2 \binom{n}{k}$ bits.

A.5 Dynamic-programming states and transitions

We use the notation from Section 6.4. For a bag B_x, a state $(\mathcal{W}_x, \mathcal{O}_x)$ has the following formal meaning. The wire part $\mathcal{W}_x$ places the symbols of B_x into k ordered lists, each with sentinels s, e. For every consecutive visible pair (u, v) it stores a type $\tau(u, v) \in \{0, 1, 2\}$. Type 1 means that u, v are already a direct internal-Hadamard pair, and therefore uv must be a quadratic monomial of f. Type 2 means that u and v are connected by a nonempty path of forgotten direct-Hadamard pairs. Type 0 means that the processed part has not connected them by a Hadamard chain.

The order part $\mathcal{O}_x$ is a topological order of the visible endpoint vertices $S, E, L(u), R(u)$ for $u \in B_x$, after the identifications $R(u) = L(v)$ forced by type-1 pairs. It must respect the interval edges $L(u) \to R(u)$, the visible wire-order edges, and every currently visible overlap edge induced by a monomial whose relevant symbols are in the processed subtree. A state is stored exactly when it has an extension to all symbols in V_x whose full partial endpoint graph is acyclic and restricts to $\mathcal{O}_x$ on the boundary.

For a leaf bag, the unique state consists of k empty lists (s, e) and the order $S < E$. At an introduce node for a symbol u, we insert u into one visible type-0 gap of one wire. Inserting into a type-1 or type-2 gap is forbidden: type 1 is already a direct Hadamard adjacency, and type 2 hides a completed forgotten path. The two new neighboring pairs are marked type 0 or legal type 1, and the order inserts $L(u), R(u)$ in all possible positions consistent with the interval, wire, and newly visible overlap edges.

At a forget node for u, we remove u from the visible wire list. If the visible neighbors of u are a and b, the new type of (a, b) records whether the removed portion has become a hidden Hadamard path: two incident pieces that contain a type-1 or type-2 segment are merged into type 2, while two empty type-0 pieces merge into type 0. Before deleting $L(u), R(u)$, the transition checks all monomial constraints involving u whose other symbols have already appeared in the processed subtree; after the forget step, such constraints will no longer be visible from the boundary.

At a join node, the two child states are compatible only if they induce the same visible wire order and the same boundary topological order. Their type labels are then merged. A direct-adjacency claim cannot be made independently on both sides unless it is the same visible claim; hidden paths are merged as hidden information. This rule reflects the separator property of the tree decomposition: the two processed subinstances can interact only through the bag.

A.6 Correctness and size of the dynamic program

Correctness follows by induction over the nice tree decomposition. The leaf case is immediate. For an introduce node, any realizing extension of the parent state restricts to a realizing child state after deleting the newly introduced symbol. Conversely, if a child extension exists and the introduce checks pass, then the only new constraints involving u are boundary-visible; adding the checked endpoint vertices to the child's acyclic endpoint graph preserves acyclicity. The bag-separator property ensures that u has no unrecorded interaction with vertices outside the current subtree.

The forget step is the same invariant in reverse. A realizable child state can forget u precisely when all constraints that become internal after deleting u have been checked. The type of the merged boundary pair contains exactly the information an outside continuation may need: either the gap is still empty, or it already hides a completed Hadamard path and therefore cannot be split by a future insertion. No absolute timing information is needed because all timing constraints are represented by the acyclicity of the endpoint graph.

For a join node, let H_1 and H_2 be the two partial endpoint graphs. They intersect only on boundary endpoint vertices. If the two child states agree on the boundary topological order, any directed cycle in $H_1 \cup H_2$ either lies inside one H_i or crosses the boundary at least twice. In the latter case, the cycle would force two boundary vertices to appear in opposite orders in the common boundary order, a contradiction. Therefore the union is acyclic. The converse is obtained by restricting any parent realization to the two child subtrees.

Backpointers stored with every accepted transition reconstruct the ordered wire lists and the direct-Hadamard pairs. The final call to Check performs a topological sort of the full endpoint graph and places the remaining CZ and CCZ gates at the corresponding overlap times. Thus the DP finds a realizing width-k circuit exactly when one exists.

For a bag of size $O(k)$, assigning visible symbols to k ordered boundary lists with sentinels, choosing the three-valued pair types, and choosing a boundary endpoint order yields $k^{6k+o(k)}$ states after the standard Stirling-type simplification. The number of nice bags is linear in n, and transition enumeration is bounded by the same state factor. The treewidth preprocessing costs $2^{O(k)}n$, and building the connection graph costs $O(m)$, so the total running time is $k^{6k+o(k)}n + O(m)$.