hexagon:2611.00004v1

A SUBLINEAR TREEWIDTH EXPONENT FOR BINARY CONSTRAINT SATISFACTION

TROY NGUYEN

Abstract. We give a deterministic algorithm for binary constraint satisfaction whose exponent in the input size is sublinear in treewidth on a class of graphs with unbounded treewidth. There are a class $\mathcal{G}$ of graphs of bounded maximum degree, a constant c, and a function f such that one algorithm correctly decides every binary CSP instance and, on instances I with primal graph $G \in \mathcal{G}$, runs in time

$$f(G)\|I\|^{c\,\mathrm{tw}(G)/\sqrt{\log \mathrm{tw}(G)}}.$$

This gives a counterexample to Conjecture 1.4 of Marx (2010) as stated. The algorithm follows from a domain-uniform counting bound: for every fixed maximum degree Δ, complete-table instances with k variables and domain size $D \geq 2$ can be counted in time $L^{O_\Delta(1)} D^{O_\Delta(k/\sqrt{\log k})}$ for sufficiently large k. The constants are independent of D and of the constraint relations. We obtain this bound from the tensor-network recursion of Claude (Anthropic), and use attributed graph foundations from OpenAI to construct the required expander family. Support pruning connects the counting routine to sparse tuple-list inputs without expanding the original domain. The algorithm, its native finite-alphabet machine implementation, and the conjecture counterexample are formalized in Lean 4.

Contents

1. Introduction 1
2. Instances, encodings, and machine model 4
3. The imported counting routine 5
4. A counting bound uniform in the domain 6
5. Sparse inputs and the original domain 10
6. Expanders and a sublinear treewidth exponent 12
7. Formal verification and provenance 14
Appendix A. Dependency revisions and reproduction 15
Appendix B. Map from the proof to the formal development 15
References 16

1. Introduction

The Exponential Time Hypothesis (ETH) asserts that deciding 3-SAT on n variables requires time $2^{\Omega(n)}$ [3]. It has served as a common source of running-time lower bounds for parameterized

Date: October 31, 2026.

2020 *Mathematics Subject Classification.* 68Q25, 68Q17, 68R10, 03B35.

Key words and phrases. constraint satisfaction, treewidth, subexponential algorithms, exact counting, exponential time hypothesis, formal verification.

Research provenance. Troy Nguyen of Bothell, Washington initiated and directed this AI-assisted investigation and selected its research objectives. GPT 6.1 Sol developed the downstream mathematical argument, Lean formalization, and manuscript in response to that direction. The imported algorithmic and graph-theoretic foundations are credited in Sections 1.3 and 7.2.

and exact algorithms. The refutations of ETH by OpenAI and by Claude (Anthropic), accompanied by Lean developments, change that setting [6, 7, 1, 8]. A substantial part of exact and parameterized complexity must consequently be reassessed. A conditional lower bound remains a valid implication, but its role as an explanation of algorithmic optimality changes when its premise fails. Questions previously approached through reductions from satisfiability must now be examined through the algorithms themselves: which running-time bounds can be attained, and which structural parameters continue to govern them?

Constraint satisfaction is a natural place to begin. It encompasses graph coloring and the evaluation of conjunctive database queries [4]. A binary CSP asks for an assignment of values to variables subject to arbitrary relations on pairs of variables. The primal graph records which variables occur together in a constraint. If this graph has treewidth w, dynamic programming gives a running time with a domain exponent of order w [2]. Whether that exponent can be substantially reduced has been a central question in the structural complexity of CSP.

We show that a sublinear exponent is attainable on a class of graphs with unbounded treewidth. The domain is unrestricted: its size may grow with the number of variables, and the constraint relations may vary arbitrarily from instance to instance. The underlying improvement is an upper bound for exact counting that is uniform in the domain size. Combining it with a family of bounded-degree expanders gives a counterexample to a conjectured treewidth lower bound.

Throughout, log denotes the natural logarithm. We use k for the number of CSP variables, D for the domain cardinality, and w for treewidth. The degree bound Δ is fixed before the input is given. Running times count steps of a deterministic finite-alphabet machine. The distinction between the complete-table length L and the sparse instance size $\|I\|$ is specified in Section 2.

1.1. **Main results.**

Theorem 1.1. *There exist an isomorphism-closed class $\mathcal{G}$ of finite simple graphs, a constant $c \geq 0$, a function $f : \mathcal{G} \to \mathbb{R}_{\geq 0}$, and one deterministic algorithm $\mathcal{A}$ with the following properties.*

1. *The class $\mathcal{G}$ has bounded maximum degree and unbounded treewidth; every graph in $\mathcal{G}$ has treewidth at least two.*
2. *The algorithm $\mathcal{A}$ correctly decides every valid binary CSP instance.*
3. *For every instance I whose primal graph G belongs to $\mathcal{G}$, the running time is at most*

$$f(G)\|I\|^{cw/\sqrt{\log w}}, \qquad w = \mathrm{tw}(G). \tag{1.1}$$

The graph-dependent factor is independent of the domain and the constraint relations.

The quantitative input to this theorem is the following counting result. A complete-table instance specifies an edge mask and the binary relation table for each ordered pair of distinct variables; its exact format is given below.

Theorem 1.2. *For every integer $\Delta \geq 1$, there exist a single deterministic counting algorithm, constants $A_\Delta, C_\Delta \geq 0$, and nonnegative integers p_Δ, k_Δ such that the algorithm correctly counts the satisfying assignments of every complete-table binary CSP instance. If the instance has $k \geq k_\Delta$ variables, domain cardinality $D \geq 2$, maximum primal degree at most Δ, and canonical input length L, its running time is at most*

$$A_\Delta L^{p_\Delta} D^{C_\Delta k/\sqrt{\log k}}. \tag{1.2}$$

All constants and the algorithm are chosen before k, D, and the instance.

For fixed D, the nonpolynomial factor in (1.2) is $2^{o(k)}$. For growing D, the statement is $D^{o(k)}$ with a uniform exponent: the saving is measured against the D^k assignment space. This uniformity is essential to the treewidth consequence.

In the context of strengthening the ETH-conditional lower bound of Theorem 1.3, Marx proposed Conjecture 1.4 [5, p. 88]. Its displayed statement contains no ETH hypothesis and asserts that

no globally correct binary-CSP algorithm has running time $f(G)\|I\|^{o(\mathrm{tw}(G))}$ on a graph class of unbounded treewidth, for any graph-dependent factor f.

Corollary 1.3. *Theorem 1.1 gives a counterexample to Conjecture 1.4 of Marx* [5] *as stated.*

Proof. As $w \to \infty$,

$$\frac{cw/\sqrt{\log w}}{w} = \frac{c}{\sqrt{\log w}} \longrightarrow 0.$$

Apply Theorem 1.1. □

The ETH-conditional Theorem 1.3 of [5] excludes the smaller exponent $o(w/\log w)$ under ETH. Theorem 1.1 concerns the separate conjecture with exponent $o(w)$; the conditional theorem remains valid.

1.2. **Why the domain parameter matters.** For a fixed domain and fixed arity, subexponential CSP algorithms are closely tied to the failure of ETH [4]. The growing-domain problem requires a stronger uniformity statement. Representing a domain value by $\lceil \log_2 D \rceil$ bits uses only $O(k \log D)$ assignment bits, but an arbitrary binary relation need not have a short 3-CNF description on those bits.

For example, a direct encoding forbids each disallowed pair by a clause of length $O(\log D)$. Converting all these clauses to width three can introduce $O(D^2 \log D)$ auxiliary variables per relation. A subexponential 3-SAT algorithm applied after this conversion does not by itself establish a bound of the form (1.2) uniformly in D. A reduction from 3-SAT to CSP establishes hardness in the opposite direction and supplies no such algorithm.

Our argument works with the tensor-network counting routine directly. The data of the relations are accessed through polynomial-time procedures, while the network capacities depend on $\log D$. This separates the cost of reading and evaluating the input from the exponential cost of searching its assignments.

1.3. **Prior work and source dependencies.** The algorithmic starting point is the exact signed tensor-network recursion of Claude (Anthropic) [1, 8]. Its sums of products are already presented as weighted binary counting CSPs. We use its quantitative clique routine and grouping bounds; the general identification of pairwise counting with tensor-network evaluation is part of that prior work.

OpenAI independently gives subexponential exact counting algorithms for fixed-width satisfiability [6]. We also reuse graph-theoretic proofs from the OpenAI mathematics repository [9]: the spectral expander family and weighted-tree separator foundations. The graph sources are imported by source port, with their original attribution retained. These graph dependencies are distinct from the imported Anthropic counting routine.

The results proved here connect that routine to arbitrary-domain bounded-degree binary CSPs, retain uniform constants through native simulation, and transfer the bound to sparse inputs and the precise treewidth conjecture. The downstream development contains the explicit encoders, concrete relation procedures, cost estimates, global fallback, and finite graph and treewidth transfers used in the proof. Section 7 identifies the checked endpoints and the exact source revisions.

More precisely, the counting engine and its grouping theorem are imported results. The application here supplies a factorized network for an arbitrary binary relation, bounds its setting lengths and row capacities in terms of $\Delta \operatorname{bits}(D)$, and implements its queries without enumerating the grouped state spaces. Choosing the imported clique level as a function of k then yields constants uniform in D. The sparse representation and graph transfers are further steps needed to obtain the statement of Conjecture 1.4. These are consequences and adaptations of the imported machinery, rather than an independent subexponential counting engine. Formal verification establishes the stated dependency chain; it does not establish priority over other possible applications of that machinery.

2. Instances, encodings, and machine model

2.1. **Binary CSPs.** A sparse instance consists of $k \in \mathbb{N}$ declared variables, a domain $[D] = \{0, \dots, D-1\}$, and a list of constraints

$$c = (a_c, b_c, R_c), \qquad a_c, b_c \in [k], \qquad R_c \subseteq [D]^2.$$

Each relation is represented by an explicit list of allowed pairs. Repeated constraints and repeated tuple entries are permitted. A constraint with $a_c = b_c$ requires $(z(a_c), z(a_c)) \in R_c$ and hence acts as a unary constraint. A valid instance has all scope indices below k. Values listed outside $[D]$ are immaterial to satisfaction and are discarded when the support is computed.

An assignment $z : [k] \to [D]$ satisfies the instance if

$$(z(a_c), z(b_c)) \in R_c \quad \text{for every listed constraint } c.$$

The primal graph is the simple graph on $[k]$ with an edge between two distinct variables precisely when some constraint has that scope, in either order. Its degree counts distinct neighbors, independently of the number of repeated constraints.

For the counting theorem, a complete-table instance is a simple graph on $[k]$ together with a relation $R_{ab} \subseteq [D]^2$ for each edge $\{a, b\}$, oriented by $a < b$. Its solution count is

$$Z(I) = \sum_{z:[k]\to[D]} \prod_{\substack{a<b \\ \{a,b\}\in E(G)}} \mathbf{1}_{\{(z(a),z(b))\in R_{ab}\}}. \tag{2.1}$$

The empty-domain count is one when $k = 0$ and zero otherwise. Self-constraints in the sparse input are handled by the normalization of Section 5; no diagonal edge test is silently added to (2.1).

2.2. **Input size.** Let m be the number of constraints and t the total number of listed tuples. Form a list F of natural-number fields consisting of the domain cardinality, the three scope fields $(a_c, b_c, |R_c|)$ for each constraint, and the two coordinates of each listed tuple. Thus

$$n_F := |F| = 1 + 3m + 2t.$$

Let $\mathrm{bits}(x)$ be the binary length of x, with $\mathrm{bits}(0) = 0$. We use the explicit field-size measure

$$N := \|I\| = k + m + t + n_F + \sum_{x\in F} \mathrm{bits}(x) + 1. \tag{2.2}$$

It charges the declared variables, list structure, and binary data. The use of zero bits for the natural number zero affects this size by at most the already charged number of fields.

The native sparse serialization writes the fields at the common width

$$B_F = 1 + \sum_{x\in F} \mathrm{bits}(x).$$

Five terminated unary headers specify k, B_F, m, t, n_F, followed by the n_F binary fields. Its physical bit length is

$$\ell = k + B_F + m + t + n_F + 5 + B_F n_F. \tag{2.3}$$

This encoding has a total decoder and an exact round trip on every canonical instance. Both encoding and decoding are polynomial-time procedures.

Lemma 2.1. *For every sparse instance,*

$$\ell \le (N+5)^2 \le 36N^2. \tag{2.4}$$

Consequently, if $r \ge 0$ and $2r \le h$, then for every $A \ge 0$,

$$A\ell^r \le A36^r N^h. \tag{2.5}$$

Proof. Equation (2.2) gives $N = k + m + t + n_F + B_F$. In particular, $B_F, n_F \le N$ and $N \ge 1$. Equation (2.3) therefore implies $\ell \le N + 5 + N^2 \le (N+5)^2$. Since $N + 5 \le 6N$, the second inequality follows. Raising to r and using $N \ge 1$ proves (2.5). □

The complete-table encoding has unary headers for k, D and three reserved parameter fields, followed by k^2 edge bits and k^2D^2 relation bits. The reserved fields are zero on the original input; the algorithm computes its grouping and clique parameters internally. Its canonical length is

$$L = k + D + 5 + k^2 + k^2D^2. \tag{2.6}$$

If the internally prepared fields are g, b, K, the corresponding length is

$$E = k + D + g + b + K + 5 + k^2 + k^2D^2. \tag{2.7}$$

2.3. **Native computation.** The formal implementation uses Mathlib's finite-program deterministic stack-machine model, with a fixed finite number of stacks and finite alphabet on every stack. For a fixed machine, each program block performs a bounded number of stack operations. Integers and lists are encoded explicitly; arithmetic on an unbounded integer is not a unit-cost primitive.

We first construct a stepped program whose initialization, transition, and output procedures have polynomial-time implementation certificates. The imported native simulation supplies one machine and an integer q, fixed before the input, such that a run using T abstract steps and state size at most S gives a native run within

$$q(T+1)(\ell + S + 1)^q \tag{2.8}$$

steps on a bit input of length ℓ. The state-size term is retained throughout the argument.

3. The imported counting routine

We record the interface used from the Anthropic formalization [8]. This isolates the algorithmic dependency from the CSP construction and estimates proved below.

3.1. **Network data and admissibility.** A network on g nodes has, for each node a, a setting length ℓ_a, a set $S_a \subseteq \{0,1\}^{\ell_a}$, and an integer weight $W_a : S_a \to \mathbb{Z}$. Each pair of distinct nodes has an integer interaction $\Psi_{ab} : S_a \times S_b \to \mathbb{Z}$, a nonnegative integer capacity c_{ab}, and integer factors $f_{ab}(z, \theta)$, where $z \in S_a$ and $\theta \in \{0,1\}^{c_{ab}}$. The capacities are symmetric. With positive integer parameters P, B, the required conditions are

$$\ell_a \le P, \qquad \sum_{b \ne a} c_{ab} \le P, \tag{3.1}$$

$$\Psi_{ba}(z', z) = \Psi_{ab}(z, z'), \qquad \Psi_{ab}(z, z') = \sum_{\theta \in \{0,1\}^{c_{ab}}} f_{ab}(z, \theta) f_{ba}(z', \theta), \tag{3.2}$$

$$|W_a(z)| < 2^B, \qquad |\Psi_{ab}(z, z')| < 2^B, \qquad |f_{ab}(z, \theta)| < 2^B. \tag{3.3}$$

The conditions involving two nodes are imposed for $a \ne b$, and all conditions on settings are imposed on the indicated legal settings. Diagonal capacities play no role. The network value and total capacity are

$$Z(\mathcal{N}) = \sum_{z \in \prod_a S_a} \prod_a W_a(z_a) \prod_{a<b} \Psi_{ab}(z_a, z_b), \qquad U = \sum_a \sum_{b \ne a} c_{ab}. \tag{3.4}$$

In particular, U counts ordered pairs, so each shared capacity is counted twice.

The algorithm is given the setting lengths and capacities, together with four procedures: membership in S_a, evaluation of W_a, evaluation of Ψ_{ab}, and evaluation of f_{ab}. Their correctness is required for all queries in the domains above, with membership tested on strings of length ℓ_a. Their execution time and state size must be bounded. In the stepped-program interface used here each query is one abstract transition whose implementation is separately certified polynomial-time;

the native simulation charges that implementation through (2.8). Setting lengths and capacities have certified unary implementations. Thus admissibility includes both the mathematical identities and concrete access procedures, not only bounds on the number of settings. These requirements are respectively `Net.Is`, `Procs.Give`, and `Procs.Keeps` in the imported development.

3.2. **Quantitative recursion and grouping.**

Proposition 3.1 (Imported recursion interface)**.** *There are integers $C_0, C_1 \ge 0$, a constant $A_0 \ge 0$, a strictly increasing sequence of integers $K_j \ge 2$, and rates $e_j \ge 0$, with*

$$K_{j+1} \le K_j^{10}, \qquad e_j \le \frac{A_0}{\sqrt{\log_2 K_j}}. \tag{3.5}$$

The levels admit a uniform polynomial-time bounded computation. At level j, the clique routine evaluates a network with K_j nodes, settings consisting of $K_j - 1$ blocks of λ bits at each node, and interactions requiring agreement of the two copies of each shared block. Its integer local tables have magnitude below $2^{B'}$. Its abstract operation and state-size budget is

$$\mathcal{T} = \left\lceil 2^{e_j K_j^2 \lambda} \big(K_j(\lambda + B')\big)^{C_0 K_j^2} \right\rceil. \tag{3.6}$$

For a network satisfying Section 3.1 on $g = 3K_j^8$ nodes with parameters P, B and total capacity U, grouping reduces evaluation to that clique routine using parameters

$$\lambda = \left\lceil \frac{U + gP/K_j}{K_j^2} \right\rceil, \qquad B' = \left\lceil \frac{2gP}{K_j} \right\rceil + g^2 B, \tag{3.7}$$

and the resulting abstract operation bound is

$$T \le \left(K_j^g + \mathcal{T}\, 2^{2gP/K_j}\right) \big(g(P+B)\big)^{C_1}. \tag{3.8}$$

The same program is used at every level.

The interfaces above are supplied by the imported recursion and grouping theorems. In the downstream program, the grouping size is converted to its level by a certified adapter, and every table query is replaced by a concrete polynomial-time procedure on the input. The replacement gives a run in at most $3T$ abstract steps. Its state bound has the form

$$S \le F_\Delta E^{a_\Delta} + \mathcal{T}, \tag{3.9}$$

where F_Δ, a_Δ are fixed independently of k and D; we establish the required input-size estimate in the next section. Thus (3.8) is applied to an implemented relation provider, with the cost of its finite procedures included in native simulation.

4. A counting bound uniform in the domain

4.1. **Balanced groups and concrete tables.**

Let I be a complete-table instance with maximum degree at most Δ. Put $b = \mathrm{bits}(D)$; for $D \ge 2$, we have $b \ge 1$ and $D \le 2^b$. A value in $[D]$ is represented by a b-bit word, with words representing values outside the domain rejected.

Partition the k variables into g groups V_G by their residues modulo g. Every group has size at most

$$s = \lfloor k/g \rfloor + 1.$$

A setting of group G is a concatenation, in increasing variable order, of one b-bit block per variable in V_G. The set S_G consists exactly of words whose blocks decode to values in $[D]$; hence $\ell_G = b|V_G|$. Write x_a for the value of variable a in a setting. For an edge with $a < v$, put

$$r_{av}(x, y) = \mathbf{1}_{\{(x,y) \in R_{av}\}}.$$

The local weight and the interaction between distinct groups are

$$W_G(x) = \prod_{\substack{a<v,\ \{a,v\}\in E(I)\\ a,v\in V_G}} r_{av}(x_a, x_v), \tag{4.1}$$

$$\Psi_{GH}(x,y) = \prod_{\substack{a<v,\ \{a,v\}\in E(I)\\ \text{one endpoint in } V_G,\ \text{one in } V_H}} r_{av}(z_a, z_v), \tag{4.2}$$

where z agrees with x on V_G and with y on V_H. Empty products are one. This definition makes $\Psi_{GH}(x,y) = \Psi_{HG}(y,x)$.

For each unordered pair of groups, list its crossing edges once, ordering endpoints by group and then ordering the resulting pairs lexicographically. Use this same list in both orientations. If it has m_{GH} edges, set $c_{GH} = bm_{GH}$. A label θ is a word of c_{GH} bits with one block t_e per crossing edge e. It is legal when every block represents a value of $[D]$. The interpretation of t_e is always the value of the smaller *original variable*, irrespective of group order. For $e = \{a, v\}$ with $a < v$, its contribution to the factor at either endpoint is

$$\phi_{a,e}(x_a, t_e) = \mathbf{1}_{\{x_a = t_e\}}, \qquad \phi_{v,e}(x_v, t_e) = r_{av}(t_e, x_v). \tag{4.3}$$

For the endpoint u of e in V_G, define

$$f_{GH}(x,\theta) = \mathbf{1}_{\{\theta \text{ is legal}\}} \prod_{e \text{ crossing } G,H} \phi_{u,e}(x_u, t_e). \tag{4.4}$$

When there are no crossing edges, the label is the empty word and the factor is one. Nonedges carry no domain labels and introduce no multiplicity.

Lemma 4.1 (Factorization and value)**.** *The factors in* (4.4) *satisfy* (3.2)*, and the network value in* (3.4) *equals* $Z(I)$*.*

Proof. Fix legal settings x, y of two distinct groups. Labels with an illegal block contribute zero. Legal labels are in bijection with independent choices $t_e \in [D]$ on the crossing edges. Distributing the finite sum over these choices gives

$$\sum_{\theta\in\{0,1\}^{c_{GH}}} f_{GH}(x,\theta) f_{HG}(y,\theta) = \prod_{\substack{e=\{a,v\}\ \text{crossing } G,H\\ a<v}} \left(\sum_{t\in[D]} \mathbf{1}_{\{z_a = t\}}\, r_{av}(t, z_v) \right) = \Psi_{GH}(x,y).$$

The last equality follows because exactly one term survives in each inner sum. This is a factorization through b bits per crossing edge and does not require a special constraint language or a low-rank relation table.

A collection of legal group settings is in bijection with an assignment $[k] \to [D]$. Every original edge occurs exactly once, either in a local weight or in an interaction between two groups. The summand in (3.4) is consequently the indicator that the assignment satisfies all constraints. Summing proves the value identity. The corresponding formal results are `BitNetwork.factor_lt` and `BitNetwork.value_eq_count`; the semantic merge is `Instance.coarsened_count`. □

For example, in a two-variable instance with one edge $a < v$ and relation $R_{av} \subseteq [D]^2$, place the variables in separate groups. The factors are $\mathbf{1}_{\{x=t\}}$ and $r_{av}(t, y)$, so their sum over legal b-bit labels is $r_{av}(x, y)$, for an arbitrary table R_{av}. Invalid labels contribute zero. The resulting network value is $|R_{av}|$.

The row-capacity and entry parameters used by the concrete provider are

$$P = \Delta bs, \qquad B = s + s^2 + 1. \tag{4.5}$$

The entry parameter is retained as stated, including its dependence on s. All required table values are evaluated by scans, products, and indexed accesses to the original finite tables. The procedures

manipulate the labels of the current query; they do not materialize the entire table of grouped assignments.

Lemma 4.2. *If $1 \le g \le k$, the grouped network satisfies the imported network contract with parameters* (4.5)*, and its total capacity U obeys*

$$gP \le 2\Delta bk, \qquad U \le 2\Delta bk. \tag{4.6}$$

Its value is $Z(I)$. The concrete query procedures are polynomial-time in the encoded input and current query.

Proof. A group setting has length $b|V_G| \le bs \le P$, since $\Delta \ge 1$. Each crossing edge needs b label bits and each variable has at most Δ neighbors, so $\sum_{H \ne G} c_{GH} \le \Delta b|V_G| \le P$. The capacities are symmetric. All local weights, interactions, and factors in (4.1)–(4.4) are zero or one, so their magnitudes are strictly below 2^B. Both P and B are positive. Lemma 4.1 supplies the remaining factorization and value requirements. These are the conditions checked by `BitNetwork.is_valid`. Summing the row bounds gives $U \le gP$. Finally,

$$g(\lfloor k/g \rfloor + 1) \le k + g \le 2k.$$

Membership checks inspect the setting length and the decoded blocks. The other procedures evaluate the displayed products by scans and indexed accesses to the input tables, rejecting illegal labels before evaluation. They enumerate neither $[D]$ nor all group settings. For $g \le k$, the setting and label lengths and the numbers of factors are polynomial in k and b; table lookup is polynomial in the complete-table input length. The formal provider satisfies the mathematical and execution contracts by `ProviderGive.isNetwork`, with polynomial implementation certificates in `ProviderMetadata.procs_keeps`. The abstract query budget is one step, with its actual native implementation charged through (2.8). □

4.2. **Choosing the clique level.** For $k \ge K_0^{32}$, select the largest j with $K_j^{32} \le k$, and put

$$K = K_j, \qquad g = 3K^8. \tag{4.7}$$

Strict increase of the levels guarantees that this selection is finite. The selected level satisfies

$$K^{32} \le k < K^{320}. \tag{4.8}$$

Indeed, maximality gives $k < K_{j+1}^{32}$, and (3.5) gives $K_{j+1}^{32} \le K^{320}$. In particular, $K \le k$, $K^2 \le k$, and $g \le k$.

The selection is implemented using the bounded level computer, with a unary cutoff determined by k. This avoids evaluating an unbounded next level as an intermediate value. Once K is obtained, the group count and domain width are computed by fixed polynomial-time procedures.

Lemma 4.3. *For the parameters in* (4.5)–(4.8)*, the actual clique arguments in* (3.7) *satisfy*

$$K^2\lambda \le 5\Delta bk, \tag{4.9}$$

$$\lambda \le 5\Delta bk, \qquad B' \le 12\Delta bk^4, \tag{4.10}$$

$$K(\lambda + B') \le 17\Delta bk^5. \tag{4.11}$$

Proof. Using the ceiling bound and (4.6),

$$K^2\lambda \le U + gP/K + K^2 \le 2\Delta bk + 2\Delta bk + \Delta bk.$$

Here $K^2 \le k$ and $\Delta b \ge 1$. This proves (4.9) and the first part of (4.10). Since $s \le k + 1$, we have $B \le 7k^2$. Thus

$$B' \le 2gP/K + 1 + g^2B \le 4\Delta bk + 1 + 7k^4 \le 12\Delta bk^4.$$

Multiplying the sum of the two bounds by $K \le k$ proves (4.11). □

4.3. **Uniform absorption of the overheads.** Write

$$R = \frac{bk}{\sqrt{\log k}}. \tag{4.12}$$

The next elementary estimates hold for every $b \geq 1$. This is where uniformity in the domain parameter enters the cost analysis.

Lemma 4.4. *Suppose $K \geq 2$, $b \geq 1$, and $K^{32} \leq k \leq K^{320}$. Then*

$$\sqrt{\log k} \leq 18K, \tag{4.13}$$

$$3K^8 \log K \leq 54R, \tag{4.14}$$

$$\frac{1}{K} \leq \frac{18}{\sqrt{\log k}}, \tag{4.15}$$

$$K^2(\log k + \log b + 1) \leq 5796R. \tag{4.16}$$

Moreover $R \geq 1$.

Proof. The upper bound on k gives $\log k \leq 320 \log K \leq 320K$, which implies (4.13). Multiplying its left side by $3K^8 \log K$, using $\log K \leq K$ and $K^{10} \leq k$, gives (4.14). Equation (4.15) is a rearrangement of (4.13).

For (4.16), use $\log k \leq 320K$ and $\log b \leq b$ to obtain

$$\log k + \log b + 1 \leq 322bK.$$

After multiplying by $K^2\sqrt{\log k}$, the resulting expression is at most

$$K^2(322bK)(18K) = 5796bK^4 \leq 5796bk.$$

Finally $\log k \leq k - 1 \leq k^2$ for $k \geq 2$, so $\sqrt{\log k} \leq k$ and $R \geq 1$. □

Lemma 4.5. *There is a constant c_0, independent of k and b, such that the integer clique budget of* (3.6) *satisfies*

$$\mathcal{T} \leq \exp(c_0 R). \tag{4.17}$$

The constant may depend on Δ and the fixed imported routine.

Proof. Let X be the expression inside the ceiling in (3.6). The level window implies

$$\sqrt{\log k} \leq 18\sqrt{\log_2 K}, \qquad e_j \leq \frac{18A_0}{\sqrt{\log k}}.$$

Consequently,

$$e_j K^2 \lambda \log 2 \leq 90A_0\Delta(\log 2)R.$$

By (4.11),

$$\log\bigl(K(\lambda + B')\bigr) \leq c_\Delta(\log k + \log b + 1), \qquad c_\Delta = \log 17 + \log \Delta + 6.$$

Equation (4.16) now gives

$$\log X \leq \bigl(90A_0\Delta \log 2 + 5796C_0c_\Delta\bigr)R.$$

All factors defining X are at least one. Hence $\lceil X \rceil \leq 2X$, and the additive $\log 2$ in its logarithm is absorbed using $R \geq 1$. □

The remaining exponential factor in (3.8) is bounded by

$$\frac{2gP}{K} \leq \frac{4\Delta bk}{K} \leq 72\Delta R. \tag{4.18}$$

Also $\log(K^g) = 3K^8 \log K \leq 54R$. Combining these bounds with Lemma 4.5, the sum in (3.8) is at most $\exp(c_1 R)$ for a fixed c_1. Since $g, b, K, k \leq E$,

$$g(P + B) \leq (2\Delta + 7)E^3.$$

The actual counter therefore has bounds

$$T \le F_1 E^{3C_1} \exp(c_1 R), \tag{4.19}$$

$$S \le F_2 E^{3C_1+2} + \exp(c_0 R), \tag{4.20}$$

with fixed constants F_1, F_2. The second estimate includes the encoded context, query words, and grouping state in (3.9). In particular, the state bound is used as an input to native simulation rather than omitted from the time analysis.

4.4. **Native time and global correctness.** The program first checks the size and degree conditions of the fast branch. If they fail, it uses exact enumeration. The enumeration ranges over fixed-width words, rejects illegal domain values, and sums the indicators in (2.1). For $D \le 1$ it uses width zero, with an explicit empty-domain case. Thus the chosen counter is correct on every canonical complete-table input, including inputs outside the fast branch.

On the fast branch, substitute (4.19) and (4.20) into (2.8), with the constant-factor step replacement and guard state included. For fixed q, the result is at most

$$AE^p \exp(cR) \tag{4.21}$$

for constants A, p, c fixed before k, D, I. To see the absorption explicitly, write $a = 3C_1 + 2$. Since $E \ge 1$ and $c_0 \ge 0$,

$$E + S + O(1) \le O(F_2 + 1)E^a \exp(c_0 R).$$

Raising to q changes the exponential coefficient by qc_0 and the polynomial degree by aq. All such changes are independent of D.

The prepared length satisfies $E \le 4L$: the original length is (2.6), and the additional fields satisfy $g, K \le k$ and $b \le D$. Finally, for $D \ge 2$,

$$b = \lfloor \log_2 D \rfloor + 1 \le 2 \log_2 D,$$

so

$$\exp\left(\frac{cbk}{\sqrt{\log k}}\right) \le D^{(2c/\log 2)k/\sqrt{\log k}}.$$

This proves Theorem 1.2. The rate is sublinear because $(\sqrt{\log k})^{-1} \to 0$.

5. Sparse inputs and the original domain

The original domain cardinality is a binary field of the sparse instance. Constructing all D^2 relation entries before inspecting the tuple lists would not be polynomial-time in that input. We first restrict to the values supported by the listed tuples.

5.1. **Support pruning.** Let $S_I \subseteq [D]$ be the set of in-domain values that appear as a coordinate of some listed tuple, and put $M = |S_I|$. Since there are t tuples,

$$M \le 2t. \tag{5.1}$$

Compute the deduplicated list of these values and use its positions as the new domain $[M]$. Denote the decoding injection by $d : [M] \to [D]$.

Lemma 5.1. *If the constraint list is nonempty, the original sparse instance is satisfiable if and only if it has a satisfying assignment taking all its values in S_I.*

Proof. Only the forward implication requires proof. In a satisfying assignment, the value of every variable occurring in a constraint appears in a satisfying tuple and hence belongs to S_I. A satisfied constraint supplies at least one value in S_I. Assign that value to every variable that occurs in no constraint. Satisfaction is unchanged. □

This argument applies even when some variables are isolated in the primal graph. Such a variable may still carry self-constraints, which must be retained when the sparse relations are normalized.

5.2. **Normalization with isolated variables.** For each variable a, let $U_a \subseteq [M]$ be the set of values satisfying all self-constraints at a, after decoding by d. For each distinct pair a, b that occurs as a scope, form the relation on $[M]^2$ obtained by intersecting all constraints with that scope, with coordinates reversed where necessary, and requiring the two endpoint values to lie in U_a, U_b. Let J be the complete-table instance with these relations. Its primal graph is exactly that of the sparse instance: an edge is retained even when its normalized relation is empty or universal.

Define

$$Q(I): \quad U_a \neq \varnothing \quad \text{for every isolated vertex } a.$$

Lemma 5.2. *For every scope-valid sparse instance with a nonempty constraint list,*

$$\mathrm{SAT}(I) \quad \Longleftrightarrow \quad Q(I) \text{ and } Z(J) > 0. \tag{5.2}$$

All data defining J and the test $Q(I)$ are polynomial-time computable from the sparse input.

Proof. Use Lemma 5.1 to restrict to $[M]$. Every satisfying assignment yields a solution of J and a witness that each isolated U_a is nonempty. Conversely, take a solution of J. At an incident vertex, the edge relations already enforce its self-constraints. At an isolated vertex, choose any value in U_a using $Q(I)$. This change affects no binary edge. The resulting assignment satisfies every original constraint.

The support has size at most $2t$. All unary tests, relation intersections and coordinate reversals are finite scans of the tuple lists. Filling the k^2M^2 relation array after pruning is therefore polynomial-time. The implemented procedures compute these scans and their indices directly. □

For an empty constraint list, the answer is true precisely when $k = 0$ or $D > 0$. For a nonempty list, the decision program checks $Q(I)$ and, if it holds, runs the total dense counter on J and tests whether its count is positive. This defines one globally correct sparse decision algorithm. The test of $Q(I)$ is unnecessary on nonisolated graphs, where it is vacuously true, but is required for global correctness.

5.3. **Runtime on the sparse representation.**

Lemma 5.3. *Let ℓ be the native sparse bit length. After support pruning, the prepared complete-table length satisfies*

$$E \leq 17\ell^4 \tag{5.3}$$

whenever the selected fields satisfy $g, K \leq k$ and $b \leq M$.

Proof. The unary variable header gives $k \leq \ell$, and the tuple-count header together with (5.1) gives $M \leq 2\ell$. By (2.7),

$$E \leq 3k + 2M + 5 + k^2 + k^2M^2.$$

Using $\ell \geq 1$ bounds the right side by $17\ell^4$. □

For $M \geq 2$, apply the estimates of Section 4 to J. In the native simulation use $E + \ell$ as the common input-size bound; this also covers sparse inputs whose original domain field is much longer than the reduced tables. Lemma 5.3 gives $E + \ell \leq 18\ell^4$. Therefore there are fixed constants $F, a \geq 0$ and an integer $p \geq 0$, depending only on Δ, such that the runtime on sufficiently large bounded-degree nonisolated instances is at most

$$F\ell^p M^{\beta(k)}, \qquad \beta(k) = \frac{ak}{\sqrt{\log k}}. \tag{5.4}$$

The initialization and all sparse decoding and normalization procedures are included in this native bound.

Since $M \leq 2t \leq 2\ell$, equation (5.4) is at most

$$F2^{\beta(k)}\ell^{p+\beta(k)}. \tag{5.5}$$

For $M = 0$ or $M = 1$, the dense fallback enumerates zero-bit words in a constant number of abstract steps, with polynomial state size. Its native time is polynomial in ℓ. Enlarging F and p, but retaining their independence of the instance and the original domain, includes both cases in (5.5).

Proposition 5.4. *For every fixed $\Delta \geq 1$, one globally correct sparse decision algorithm has, on sufficiently large instances of maximum degree at most Δ and with no isolated vertices, running time at most*

$$F_\Delta(k)N^{2p+2\beta(k)}, \qquad F_\Delta(k) = F36^p72^{\beta(k)}, \qquad N = \|I\|. \tag{5.6}$$

The constants F, p, a are fixed before k, the original domain cardinality, and all relation data.

Proof. Apply Lemma 2.1 to (5.5) with exponent $p + \beta(k)$. Its multiplicative factor becomes $F2^{\beta(k)}36^{p+\beta(k)} = F36^p72^{\beta(k)}$. □

6. Expanders and a sublinear treewidth exponent

We use the attributed spectral family in the OpenAI graph development [9]. There is a fixed finite nonempty port set $\mathcal{D}$ and a family of finite regular port graphs with symmetric transition operator P, spectral parameter at most $1/2$, and cardinalities $N_n = r^{n+1}$ for a fixed integer $r > 1$. Let H_n be the underlying simple graph: loops are discarded and parallel ports give a single edge. Then

$$\max \deg(H_n) \leq |\mathcal{D}|. \tag{6.1}$$

The relevant separator estimate follows directly from the spectral certificate, so the passage to simple graphs does not require them to remain regular.

The imported construction chooses a finite base graph from a proved existence theorem and then iterates the graph operations. We use only the existence of this family with its stated certificates. No effective selection of the base graph, or algorithm for recognizing the resulting class, is claimed or needed in Theorem 1.1: the CSP algorithm is globally correct and its fast branch tests the size and degree conditions, not membership in the family.

6.1. **A linear lower bound on bag size.**

Lemma 6.1. *Let A, S be vertex sets in a member of the port family. Suppose every transition from a vertex of A to a vertex outside A ends in S. Put $\alpha = |A|/N$ and $\sigma = |S|/N$, where N is the number of vertices. Then*

$$\frac{1}{2}\alpha(1-\alpha) \leq \sigma. \tag{6.2}$$

In particular, if $1/4 \leq \alpha \leq 3/4$, then $|S| \geq N/16$.

Proof. Write $f = \mathbf{1}_{\{v \in A\}}$ and use the uniform mean and inner product. The spectral certificate gives

$$\langle f, Pf\rangle \leq \alpha^2 + \tfrac{1}{2}(\alpha - \alpha^2).$$

The escaping mass $\alpha - \langle f, Pf\rangle$ is bounded above by σ, since every escaping transition ends in S and the transition operator preserves the uniform mean. Rearrangement gives (6.2). For $\alpha \in [1/4, 3/4]$, its left side is at least $3/32$, which is greater than $1/16$. □

Lemma 6.2. *Every H_n has no isolated vertices.*

Proof. Since $N_n > 1$, the density of a singleton is strictly between zero and one. If its vertex were isolated in H_n, every port transition from that vertex would return to it. Apply (6.2) with that singleton as A and $S = \varnothing$, obtaining a contradiction. □

Lemma 6.3. *Every tree decomposition of H_n with maximum bag cardinality q satisfies*

$$N_n/16 \leq q. \tag{6.3}$$

Proof. Assign each graph vertex to one bag containing it and give each decomposition-tree node the number of vertices assigned to it as weight. A weighted centroid of the tree has every component of its deletion of weight at most $N_n/2$. Let S be its bag.

For a graph vertex outside S, all bags containing that vertex lie in a single component of the deleted tree, by the connectedness condition in a tree decomposition. Adjacent graph vertices outside S have a common bag, so they receive the same component label. Thus each component of $H_n - S$ has at most $N_n/2$ vertices.

If $|S| \geq N_n/16$, the conclusion follows. Otherwise the components outside S have total size greater than $15N_n/16$. Accumulate components until their union A first has size at least $N_n/4$. The last component added has size at most $N_n/2$, so $|A| \leq 3N_n/4$. Every edge from A to its complement ends in S, and the same holds for port transitions. Lemma 6.1 gives $|S| \geq N_n/16$, a contradiction. □

For a nonempty finite graph, the minimum possible maximum bag cardinality is $\mathrm{tw}(G) + 1$. Existence follows from the one-bag decomposition, and the minimum is attained over the natural numbers. Consequently,

$$N_n \leq 16\big(\mathrm{tw}(H_n) + 1\big), \qquad \mathrm{tw}(H_n) \leq N_n. \tag{6.4}$$

In particular, the family has unbounded treewidth, and every tail defined by a cardinality threshold still has unbounded treewidth. Treewidth, degree and nonisolation are preserved by graph isomorphisms.

6.2. **Proof of the main theorem.** Set $\Delta = |\mathcal{D}|$ and choose the globally correct sparse algorithm of Proposition 5.4. Let k_Δ be its fixed size threshold. Define $\mathcal{G}$ to consist of all graphs isomorphic to some H_n with

$$N_n \geq \max\{64, k_\Delta\}. \tag{6.5}$$

Equations (6.1) and (6.4) prove that this class has bounded maximum degree and unbounded treewidth. Lemma 6.2 supplies the hypothesis needed for the fast sparse bound.

Let I have primal graph $G \in \mathcal{G}$, and write $k = |V(G)|$, $w = \mathrm{tw}(G)$. From (6.4) and (6.5),

$$2 \leq w \leq k \leq 32w. \tag{6.6}$$

For $p, a \geq 0$, these inequalities imply

$$p + \frac{ak}{\sqrt{\log k}} \leq (p + 32a)\frac{w}{\sqrt{\log w}}. \tag{6.7}$$

Indeed, the denominator increases with k, the numerator satisfies $k \leq 32w$, and $1 \leq w/\sqrt{\log w}$ absorbs the constant p.

Choose

$$c = 2(p + 32a), \qquad f(G) = F36^p 72^{\beta(|V(G)|)},$$

with the constants of Proposition 5.4. Equation (6.7) bounds $2p + 2\beta(k)$ by $cw/\sqrt{\log w}$. Proposition 5.4 and $N \geq 1$ therefore give

$$\mathrm{time}_{\mathcal{A}}(I) \leq f(G)N^{cw/\sqrt{\log w}}.$$

The factor $f(G)$ depends only on the graph cardinality and the fixed algorithm constants. Global correctness was established before restricting the runtime analysis to $\mathcal{G}$. This proves Theorem 1.1.

The algorithm does not need a tree decomposition or a test for membership in $\mathcal{G}$. It reads the CSP, computes the support, checks the degree and size conditions for the fast routine, and uses the total fallback when necessary. The graph family and its treewidth enter the proof of the running-time bound.

7. Formal verification and provenance

The closed formal statement constructs a native machine, a graph class, a graph-dependent multiplicative factor, and an exponent function h satisfying

$$h = o(w), \qquad \text{time}(I) \leq f(G)\|I\|^{h(\text{tw}(G))}.$$

Correctness is quantified over every scope-valid sparse binary CSP instance. The runtime condition is quantified over all such instances whose primal graphs lie in the constructed class. The machine and constants are chosen before the input. No fast-solver hypothesis appears in the final theorem.

The development uses one version of Lean for both the counting algorithm and the graph argument. The required OpenAI graph sources were ported from source and checked under Lean 4.33.1, using the same pinned dependencies as the counting routine.

7.1. **Checked endpoints.** All downstream modules lie in the namespace `CSPConsequence`. The central endpoints are:

Endpoint	Mathematical content
`DenseCSPTheorem.exists_counter`	Globally correct native exact counting with the domain-uniform bound of Theorem 1.2.
`SparseNativeMachine.exists_native`	One finite-alphabet native decision machine correct on every valid sparse input.
`SparseUniformRuntime.fast_for_host`	The sparse runtime estimate for all supported domain sizes, including zero and one.
`GraphWidthNumber.tail_unbounded`	Unbounded numerical treewidth after any cardinality cutoff.
`MarxConjecture.counterexample`	The native sparse-input algorithm and graph-class witness of Theorem 1.1.
`MarxConjecture.not_statement`	Negation of the formalized conjecture statement.

The proposition `MarxConjecture.Counterexample` packages unbounded treewidth, a little-o exponent, finite machine alphabets, global correctness, and the runtime inequality. Bounded degree, isomorphism closure, the lower cutoff on treewidth, and the displayed formula for the exponent are properties of the witnesses in the proof of `counterexample`, obtained from the graph-family and exponent-transfer lemmas. The theorem `not_statement` is the immediate logical consequence of this construction; the comparison with the historical conjecture rests on the stated definitions and input representation.

The last two endpoints have kernel dependency reports containing only

`propext`, `Classical.choice`, `Quot.sound`.

The retained verification covers 82 downstream modules, with 269 printed standard-axiom reports, and the selected imported counting closure of 633 upstream modules. It records stable source hashes before and after each successful whole-file check, object hashes, compiler exit status and log hashes. Failed development runs are excluded from these receipts.

The Anthropic ETH formalization [8] separates its statements in `MainResults.lean` from their proofs in `Solution.lean`. In particular, `MainResults.ethIsFalse` proves the stated ETH refutation. For the present result we import the quantitative counting and grouping endpoints directly. Our local verification concerns their selected dependency closure and the new CSP theorem; it is not a replay of every module in the upstream ETH repository. The corresponding OpenAI native exact-counting and satisfiability endpoint is recorded in its separate Lean development [7].

7.2. **Contributions and imported proofs.** Troy Nguyen, of Bothell, Washington, initiated and directed an AI-assisted investigation of consequences of the recent ETH refutations, with a focus on binary CSP and formal verification. Nguyen selected the research objectives and directed the development. GPT 6.1 Sol developed the bounded-degree, domain-uniform CSP argument, the

sparse-input and native-runtime transfers, the graph and treewidth connection, and the downstream Lean development and exposition.

The quantitative counting recursion, clique-level bounds, grouping routine, and native simulation infrastructure are reused from Claude (Anthropic) [1, 8]. The spectral expander construction and weighted-tree foundations are reused from the OpenAI mathematics repository [9]. The original source attributions are retained above the imported graph blocks, and a manifest records their copied ranges and compatibility changes. These imports are algorithmic and proof dependencies of the result, rather than claims of independent rediscovery.

7.3. **Scope of the result.** The degree bound is fixed. Theorem 1.1 exhibits a particular class of unbounded treewidth; it does not supply the same exponent on every graph class. The domain and all relation data remain unrestricted within the stated input representation. The native runtime includes parsing and normalization. The state-size estimates are used to prove the time bound, and no polynomial-space bound is asserted.

The mathematical statements have complete Lean proofs under the recorded imports. The source-level cross-references in Appendix B allow the conjecture counterexample to be compared directly with its machine model, CSP semantics, input-size measure, and graph class.

Appendix A. Dependency revisions and reproduction

The algorithmic dependency is pinned to the Anthropic revision

`16a5ce41304fcdaea4e0be7c132e1c361dc26da5.`

The graph source port is pinned to the OpenAI mathematics revision

`7ef5c0cc5ca4b0e072b7a050fca09ab24bc6fa3a.`

The development uses official Lean 4.33.1, revision

`819816b2e0a3bf405af45ae5c7af2491d8f5bee6,`

and mathlib revision

`0df444a360eaa60ab8c11dca51a86af692955474.`

The selected algorithmic endpoints are

```
CountedInstance.theRecursionMeetsTheHypothesis,
ExplicitCost.recursionSolves,
ExplicitCost.groupingExplicit,
ExplicitCost.groupingPoly,
GeneralPieces.PartA.K_lt_le.
```

Their theorem statements supply the effective level family and actual counting routine. The adapter and provider modules replace table queries with certified finite procedures before native compilation.

The companion source overlay contains the 82 checked downstream Lean modules, the source manifest, successful verification receipts, the graph port manifest and adaptation diff, and a sequential replay script. The script checks source hashes, orders downstream imports, recompiles each supplied module, and rejects failed compilations, holes and unexpected axiom reports. The supplied dependency lockfile is retained during reproduction. Complete instructions are in the artifact's `README.md`.

The selected 633-module upstream closure is sufficient for the imported counting endpoints. No independent second-kernel replay or full replay of all 1050 upstream modules is included in the verification claim.

Appendix B. Map from the proof to the formal development

The following groups identify where the mathematical constructions are verified. Source and object hashes, rather than filenames alone, identify the retained checked versions.

CSP semantics and the concrete network. `CSPNetwork`, `BalancedCSP`, `DomainCodec`, `FiniteEncoding`, `EncodedSettings`, `EncodedLabels`, `BitNetwork` and `BitNetworkValidity` establish the assignment codecs, balanced grouping, network contracts and exact value. The concrete query and factor refinements are assembled in `ProviderGive`; `OracleElimination` replaces the query interface by implemented procedures.

The uniform counting bound. `LevelSelection` and `BoundedSelection` implement the choice of level; `UniformClique` and `CounterExistence` connect it to the imported routine. `SelectedBounds`, `NumericOverheads`, `LogCost`, `CliqueBudget`, `RuntimeMetadata`, `OperationBound` and `CoreRuntime` establish the parameter, time and state estimates. `TotalCounter`, `NativeFastRuntime`, `SublinearRate` and `DenseCSPTheorem` give global correctness, the native bound and its domain-power form.

The original sparse input. `SparseSupport`, `SparseSupportDomain`, `SparseRelabelPreparation` and `SparseIsolates` prove the support and normalization results. `SparseFields`, `SparseFieldBits` and `SparseRawBits` implement the codec and its round trip. `SparseSizeBounds` and `SparseOrdinarySize` establish the size conversion. `SparseDecision`, `SparseDecisionRuns`, `SparseNativeMachine`, `SparseFastRuntime`, `SparseSmallRuntime`, `SparsePowerAbsorption` and `SparseUniformRuntime` assemble the globally correct native decision algorithm and the estimate covering every reduced domain size.

The graph class and conjecture. `GraphPinnedExpanderFamily` contains the attributed source port. `GraphFiniteMassSelection`, `GraphTreewidth` and `GraphFamilyFacts` establish the separator, degree and nonisolation facts. `GraphWidthNumber`, `GraphWidthTransport` and `SparseGraphTransfer` connect numerical treewidth and primal graphs to the actual family. `MarxConjecture` constructs all witnesses of the final theorem and proves the negation of the formalized conjecture.

References

[1] Claude (Anthropic), *A $2^{o(n)}$ time algorithm for 3-SAT*, preprint, October 2026, version 1, October 27, 2026. `https://hexagonmath.org/2610.00577v1`.

[2] Eugene C. Freuder, *Complexity of k-tree structured constraint satisfaction problems*, Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI), 1990, pp. 4–9. Proceedings paper.

[3] Russell Impagliazzo and Ramamohan Paturi, *On the complexity of k-SAT*, Journal of Computer and System Sciences **62** (2001), no. 2, 367–375. doi:10.1006/jcss.2000.1727.

[4] Iyad Kanj and Stefan Szeider, *On the subexponential time complexity of CSP*, Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013, pp. 459–465. Proceedings paper.

[5] Dániel Marx, *Can you beat treewidth?*, Theory of Computing **6** (2010), 85–112. doi:10.4086/toc.2010.v006a005.

[6] OpenAI, *The Exponential Time Hypothesis is false*, preprint and accompanying Lean development, October 25, 2026. OpenAI Math Release preprint.

[7] OpenAI, *Native exact-counting and satisfiability verification*, Lean 4 formalization accompanying the ETH refutation, revision `7ef5c0c`, 2026. `SqrtLogSpace/Final.lean`.

[8] Anthropic, *Formalization accompanying "A $2^{o(n)}$ time algorithm for 3-SAT"*, Lean 4 source repository, revision `16a5ce4`, 2026. Pinned source repository.

[9] OpenAI, *Spectral expander and weighted-tree proof foundations*, OpenAI mathematics repository, revision `7ef5c0c`, 2026. Pinned source repository. The companion `GraphPortManifest.json` records the exact source blocks and their adaptations.

Bothell, Washington, USA