Iterated-Logarithmic Additive Hardness for Bin Packing

Zhao Song*

October 13, 2026

Abstract

The question of obtaining a constant-additive approximation for bin packing was highlighted in the third of ten open problems in Williamson and Shmoys' *The Design of Approximation Algorithms* [WS11, Chapter 17, Problem 3]. We prove that there is an absolute constant $c > 0$ such that, unless P = NP, no deterministic polynomial-time algorithm for one-dimensional bin packing always returns a feasible packing using at most $\mathrm{OPT} + c \log^* \mathrm{OPT}$ bins, where OPT is the minimum number of bins. In particular, no such algorithm has additive error $o(\log^* \mathrm{OPT})$.

The best known upper bound, due to Hoberg and Rothvoss [HR17], is $\mathrm{OPT} + O(\log \mathrm{OPT})$, achieved by a randomized algorithm with expected polynomial running time. OpenAI [Ope26] previously showed that, unless P = NP, no deterministic polynomial-time algorithm guarantees a packing using at most $\mathrm{OPT} + O(1)$ bins.

We conjecture that the optimal additive approximation error for bin packing is $\Theta(\log \mathrm{OPT})$.

*magic.linuxkde@gmail.com. The author would like to thank Victor Reis and Lichen Zhang for very helpful discussions.

1 Introduction

In offline one-dimensional bin packing, the input consists of N items of sizes in $(0, 1]$, and the objective is to pack them into as few unit-capacity bins as possible. Let OPT denote the minimum number of bins and OPT_f the optimum of the Gilmore–Gomory configuration LP [GG61]. The question of obtaining a constant-additive approximation was highlighted in the third of ten open problems in Williamson and Shmoys' textbook *The Design of Approximation Algorithms* [WS11, Chapter 17, Problem 3].

Fernandez de la Vega and Lueker [FL81] gave, for every fixed $\epsilon > 0$, a linear-time algorithm using at most $(1+\epsilon)\mathrm{OPT} + O(\epsilon^{-2})$ bins. Karmarkar and Karp [KK82] obtained an additive $O(\log^2 \mathrm{OPT})$ guarantee. Rothvoss [Rot13] improved this guarantee to $O(\log \mathrm{OPT} \log\log \mathrm{OPT})$. Hoberg and Rothvoss [HR17] subsequently obtained the bound $\mathrm{OPT} + O(\log \mathrm{OPT})$ by a randomized algorithm with expected polynomial running time.

OpenAI [Ope26] proved that, unless P = NP, no deterministic polynomial-time algorithm guarantees a packing using at most $\mathrm{OPT} + O(1)$ bins. They also proved that the additive configuration-LP gap $\mathrm{OPT} - \mathrm{OPT}_f$ is unbounded.

1.1 Our Result

We state our result as follows.

Theorem 1.1 (Informal version of Theorem 4.1)**.** *There is an absolute constant $c > 0$ such that, unless* P = NP*, no deterministic polynomial-time algorithm for one-dimensional bin packing always returns a feasible packing using at most* $\mathrm{OPT} + c \log^* \mathrm{OPT}$ *bins. In particular, no such algorithm has additive error* $o(\log^* \mathrm{OPT})$.

We conjecture that the optimal additive approximation error for bin packing is $\Theta(\log \mathrm{OPT})$.

Year	Authors	Reference	Upper Bound/Lower Bound
1981	Fernandez de la Vega–Lueker	[FL81]	$(1+\epsilon)\mathrm{OPT} + O(\epsilon^{-2})$
1982	Karmarkar–Karp	[KK82]	$\mathrm{OPT} + O(\log^2 \mathrm{OPT})$
2013	Rothvoss	[Rot13]	$\mathrm{OPT} + O(\log \mathrm{OPT} \log\log \mathrm{OPT})$
2017	Hoberg–Rothvoss	[HR17]	$\mathrm{OPT} + O(\log \mathrm{OPT})$
2026	OpenAI	[Ope26]	No $\mathrm{OPT} + O(1)$ guarantee
2026	This work	Theorem 1.1	No $\mathrm{OPT} + o(\log^* \mathrm{OPT})$ guarantee

Table 1: Upper and lower bounds for offline one-dimensional bin packing.

Organization of the paper. Section 2 reviews the OpenAI proof technique and explains our extension from constant to iterated-logarithmic additive hardness. Section 3 introduces our notation and definitions and states Håstad's satisfiability-gap theorem. Section 4 proves our main result by presenting the construction and its local properties, establishing completeness and soundness, and verifying a uniform polynomial-time implementation as the additive allowance grows.

2 Technique Overview

2.1 OpenAI's Technique

We organize the hardness proof of OpenAI [Ope26] into four steps. Throughout Section 2.1, the additive allowance $c \geq 0$ is a fixed integer, and $\rho \in (0, 1/48)$ is a fixed constant.

Step 1: Start with a vertex-cover gap. Apply Håstad's satisfiability gap with $\epsilon := 1/16$. Given a formula with M clauses, form its literal-slot graph and define $n := 3M$ and $k := 2M$. Each clause contributes a triangle, and complementary literal slots are adjacent. The identity $\tau(G) = 3M - \mathrm{MAXSAT}(\Phi)$ gives

$$\text{YES: } \tau(G) = k, \qquad \text{NO: } \tau(G) \geq k + \frac{n}{48}. \tag{1}$$

Step 2: Encode two competing choices at each vertex. A $2d-1$-phase auxiliary construction gives two trees. Each has a marking respecting the depth quotas and hitting every path at least d times; when both trees share those quotas, some path remains unhit. Items in five roles encode this competition: rows, anchors, global auxiliaries, local auxiliaries, and flags. Their sizes are perturbations of $1/5$: successive scales encode subclass patterns, vertex-label order, and geometric feasibility. Nested intervals represent tree nodes, and shifted deadlines force coverage of test intervals. Local items implement the shared quotas, so global items must fund one side. Repeated edge positions and a limited permit supply couple the choices at adjacent vertices. The inventory consists of $5B$ items, each of size greater than $1/6$.

Step 3: Recover a cover from an arbitrary near-optimal packing. Write $K_{\mathrm{src}} := 25(c+1)(1+5\cdot 100^{11})$ for the source's error budget, and choose $d \geq \max\{72K_{\mathrm{src}}, 120K_{\mathrm{src}}/\rho\}$. A packing into $B+c$ bins has at most $5c$ unused item slots. Score conservation bounds the number of items outside the prescribed patterns; label-prefix conservation bounds the number of defective anchors at each vertex by $3K_{\mathrm{src}}$, independently of the gadget size. The tree obstruction forces a large global allocation on one side. A permit-counting argument shows that the vertices receiving more than $d/8$ plus-side units form a vertex cover, giving

$$\mathrm{OPT} \leq B + c \quad \Longrightarrow \quad \tau(G) \leq k + \frac{60K_{\mathrm{src}}n}{d}. \tag{2}$$

The chosen d makes this error at most ρn, below the gap in Eq. (1).

Step 4: Obtain hardness for a fixed allowance. A cover of size k determines the globally funded side at each vertex; separate tree markings and edge permits complete a packing into B bins. Step 3 rules out $B+c$ bins in the NO case. Hence

$$\text{YES: } \mathrm{OPT} = B, \qquad \text{NO: } \mathrm{OPT} > B + c \qquad (c \text{ fixed}). \tag{3}$$

For fixed c and ρ, the trees and depth quotas are constants, and the explicit rational instance is constructible in polynomial time. This proves hardness for each fixed additive allowance. Allowing c to grow requires bounding the dependence of the tree sizes and depth quotas on c.

2.2 From Constant to Iterated-Logarithmic Additive Hardness

Our four steps correspond to those in Section 2.1. We keep the combinatorial mechanisms and make the size and bit-cost bounds explicit as c grows.

Step 1: Retain the starting gap. We keep Step 1 of Section 2.1 and Eq. (1) unchanged. In particular, $k = 2n/3$, and the NO case has a cover gap of $n/48$. Our improvement comes from increasing the packing allowance while preserving this fixed source gap.

Step 2: Bound the tree growth explicitly. We make the tree-size analysis in Step 2 of Section 2.1 explicit by tracking the number of leaves after each phase. The original spine and continuation construction gives

$$N_0 := 1, \qquad N_{a+1} := \frac{2(4^{N_a} - 1)}{3}. \tag{4}$$

Let $E_0 := 1$ and $E_{j+1} := 2^{E_j}$. Eq. (4) implies $N_a \le E_{2a-1}$ for $a \ge 1$, so the final leaf count after $2d - 1$ phases is at most E_{4d-3}. Since soundness permits d to be proportional to $c + 1$, the tower height grows linearly with $c + 1$. Inverting this growth yields the iterated logarithm.

Step 3: Refine the cover bound and choose a growing allowance. We refine the encoding in Step 2 and the cover extraction in Step 3 of Section 2.1. Radix six for role balances and binary weights for subclass balances give $P_0 := 2^5 \cdot 6^4$ and $K := 20(c + 1)(1 + 5P_0)$. For $d \ge 1000K$, a sharper allocation count using $k = 2n/3$ and the threshold $2d/5$ yields the following refinement of Eq. (2):

$$\mathrm{OPT} \le B + c \quad \Longrightarrow \quad \tau(G) \le k + \frac{20Kn}{d}. \tag{5}$$

Taking $d := 1000K$ makes this error at most $n/50$, below $n/48$ (Lemma 4.7). Set $D := 20000(1 + 5P_0)$ and $h := \log^* n$. We now replace the fixed allowance in Eq. (3) by

$$c := \lfloor h/(8D) \rfloor - 1, \qquad d := D(c + 1) = 1000K. \tag{6}$$

For $h \ge 32D$, this gives $d \le h/8$ and $c \ge h/(16D)$. Thus $4d - 3 < h - 1$, so the tree bound in Step 2 and $E_{h-1} < n$ give fewer than n leaves. The recurrence also bounds depth and total quota by the leaf count, yielding fewer than n^2 vertices per tree (Lemma 4.8).

Step 4: Make the construction uniform and translate the gap. The fixed-parameter runtime argument in Step 4 of Section 2.1 is replaced by Lemma 4.9: $n \le B \le 1203n^5$, each item size has an encoding of $O(n^2 \log(n+2))$ bits, and one algorithm outputs all $5B$ items in polynomial bit time. The exponent is independent of c. Since $\log^* B \le h + 1$ for sufficiently large n, the choice in Eq. (6) converts the fixed-allowance gap in Eq. (3) into the growing gap

$$\text{YES: } \mathrm{OPT} = B, \qquad \text{NO: } \mathrm{OPT} > B + \epsilon_0 \log^* B, \qquad \epsilon_0 := \frac{1}{64D}. \tag{7}$$

Since the optimum in the YES case is B, an algorithm with an additive guarantee of $\epsilon_0 \log^* \mathrm{OPT}$ returns at most $B + \epsilon_0 \log^* B$ bins in that case. Every feasible packing uses more bins in the NO case. Comparing the output with this threshold distinguishes the cases. After renaming variables, exhaustive search over all assignments decides the promise problem for formulas with a bounded number of clauses. This proves Theorem 4.1, including the exclusion of an $o(\log^* \mathrm{OPT})$ guarantee.

3 Preliminaries

For a positive integer n, we define $[n] := \{1, 2, \cdots, n\}$. We also set $[0] := \varnothing$. For a condition E, its indicator $\mathbf{1}_E$ is one if E holds and zero otherwise. Cardinalities and sums over multisets count multiplicity.

Definition 3.1 (Iterated logarithm)**.** *All logarithms have base two. For $x > 0$, we define $\log^* x$ as the number of successive logarithms needed to obtain a value at most one; in particular, $\log^* x := 0$ for $0 < x \leq 1$.*

We consider offline bin packing with unit-capacity bins, so all item sizes are known in advance. The objective is to minimize the number of bins used.

Definition 3.2 (Bin packing)**.** *A bin-packing instance is a nonempty indexed list $I := (a_i)_{i \in [N]}$ of rational sizes $a_i \in (0, 1]$. A feasible packing is a partition $\mathcal{P}$ of $[N]$ such that $\sum_{i \in S} a_i \leq 1$ for every $S \in \mathcal{P}$. We define*

$$\mathrm{OPT}(I) := \min\{|\mathcal{P}| : \mathcal{P} \text{ is a feasible packing of } I\},$$

and abbreviate it as OPT *when the instance is clear. Items with equal sizes remain distinct.*

Every item is listed explicitly, with its size encoded by a binary integer numerator and a positive binary integer denominator. Our polynomial-time guarantees refer to the total binary input length.

Definition 3.3 (Additive approximation guarantee)**.** *For $g : \mathbb{Z}_{\geq 1} \to \mathbb{R}_{\geq 0}$, an algorithm has an additive guarantee g if, on every instance I, it returns a feasible packing using at most $\mathrm{OPT}(I) + g(\mathrm{OPT}(I))$ bins. No monotonicity is assumed for g.*

Our reduction uses vertex cover as an intermediate problem between satisfiability and bin packing. Independent sets provide the link between satisfying assignments and small vertex covers.

Definition 3.4 (Vertex cover and independent set)**.** *Let $G = (V, E)$ be a finite simple undirected graph. A vertex cover is a set $C \subseteq V$ meeting every edge, and we define $\tau(G) := \min\{|C| : C \text{ is a vertex cover of } G\}$. A set $S \subseteq V$ is independent if no edge has both endpoints in S; equivalently, $V \setminus S$ is a vertex cover.*

A literal is a Boolean variable x or its negation $\neg x$; these two literals are complementary. A CNF formula is a conjunction of clauses, each a disjunction of literals. A literal occurrence, or slot, is a position in a clause; distinct positions remain distinct even when they contain the same literal.

Definition 3.5 (MAXSAT)**.** *Let Φ be a CNF formula with variable set V and clauses $(C_j)_{j \in [M]}$. We define*

$$\mathrm{MAXSAT}(\Phi) := \max_{\sigma : V \to \{0,1\}} |\{j \in [M] : \sigma \text{ satisfies } C_j\}|.$$

Thus $\mathrm{MAXSAT}(\Phi)$ is the maximum number of clauses that can be satisfied simultaneously by one truth assignment. In particular, Φ is satisfiable if and only if $\mathrm{MAXSAT}(\Phi) = M$.

We use the following satisfiability gap with perfect completeness.

Theorem 3.6 (Håstad [Hås01, Theorem 6.5])**.** *Fix a constant $\epsilon \in (0, 1/8)$. For Boolean CNF formulas Φ with $M \geq 1$ clauses, each containing exactly three literal occurrences, the following promise problem is NP-hard:*

1. YES: *Some assignment satisfies all M clauses.*
2. NO: *Every assignment satisfies at most $(7/8 + \epsilon)M$ clauses.*

4 Iterated-logarithmic additive hardness

Section 4.1 states the main hardness theorem and proves it using the reduction lemmas below. Section 4.2 describes the packing construction and establishes its local properties. Section 4.3 proves completeness by constructing a packing in B bins for every YES instance of the satisfiability-gap promise, while Section 4.4 proves soundness by ruling out packings in at most $B + c$ bins for every NO instance. Section 4.5 bounds the size of the competing trees and constructs them in polynomial time. Finally, Section 4.6 bounds the number of items and their encoding lengths and proves that the entire reduction runs in polynomial bit time for our choice of growing additive allowance.

4.1 Main result

We state the main hardness result with an explicit absolute constant.

Theorem 4.1 (Iterated-logarithmic additive hardness, formal version of Theorem 1.1)**.** *Let* $D := 20000(1 + 160 \cdot 6^4)$ *and* $\epsilon_0 := 1/(64D) > 0$*. If a deterministic polynomial-time algorithm always returns a feasible packing using at most*

$$\mathrm{OPT} + \epsilon_0 \log^* \mathrm{OPT}$$

bins, then P = NP*. This holds even when the input is an explicit list of rational sizes in* $(1/6, 1)$*, each encoded in binary. In particular, assuming* P ≠ NP*, no deterministic polynomial-time algorithm has an additive guarantee* $o(\log^* \mathrm{OPT})$ *on all such instances.*

Proof. By Theorem 3.6, with $\epsilon := 1/16$, it is NP-hard to distinguish satisfiable formulas with exactly three literal slots per clause from formulas with at most 15/16 of their clauses satisfiable. Given such a formula with M clauses, define $n := 3M$ and $h := \log^* n$. First assume that $h \geq 32D$. Lemmas 4.6, 4.7, and 4.9 give a uniform polynomial-time reduction with

$$c := \lfloor h/(8D) \rfloor - 1, \qquad n \leq B \leq 1203n^5,$$

where, in the bounds on B, the first step follows from the lower inventory bound in Lemma 4.9-(a), and the second step follows from its upper inventory bound. For $h \geq 32D$,

$$c \geq \frac{h}{8D} - 2 \geq \frac{h}{16D}, \tag{8}$$

where the first step follows from the definition of c and $\lfloor x \rfloor \geq x - 1$, and the second step follows from $h \geq 32D$. In a YES instance the optimum is exactly B by Lemma 4.6. In a NO instance every feasible packing uses more than $B + c$ bins by Lemma 4.7.

For $n \geq 1203$, we have

$$n \leq B \leq 1203n^5 \leq n^6 \leq 2^n, \tag{9}$$

where the first step follows from the lower inventory bound in Lemma 4.9-(a), the second step follows from its upper inventory bound, the third step follows from $n \geq 1203$, and the fourth step follows from $n^6 \leq 2^n$, valid for integers $n \geq 64$. Consequently,

$$h \leq \log^* B \leq h + 1, \tag{10}$$

where the first step follows from $h = \log^* n$, Eq. (9), and monotonicity of $\log^*$, and the second step follows from Eq. (9) and $\log^*(2^n) = 1 + \log^* n$. Suppose the algorithm in the theorem exists. On a YES instance, its bin count is at most

$$B + \epsilon_0 \log^* B \leq B + \frac{h+1}{64D} \leq B + \frac{h}{32D} \leq B + c, \tag{11}$$

where the first step follows from $\epsilon_0 = 1/(64D)$ and Eq. (10), the second step follows from $h \geq 1$, and the third step follows from Eq. (8) and $h/(16D) \geq h/(32D)$. On a NO instance, feasibility forces its output to use more than $B + c$ bins, so its output bin count distinguishes the two cases.

To obtain one decision algorithm for every input size, let $E_0 := 1$ and $E_{j+1} := 2^{E_j}$, and fix

$$n_0 := \max\{1203, E_{32D-1} + 1\}.$$

For $n \geq n_0$, construct the instance, run the approximation algorithm, and answer YES exactly when its bin count is at most $B+c$. For $n < n_0$, parse the formula, rename the variables that occur in it, and test all assignments. At most $3M = n$ variables occur, so this branch uses at most 2^{n_0} assignment tests, each with polynomial input parsing and evaluation cost. Arbitrarily long original variable names therefore do not affect the polynomial running-time bound. Both branches have deterministic polynomial running time in the original encoding length, so the NP-hard promise problem would be decidable in polynomial time. This implies P = NP.

Finally, suppose an algorithm instead has a nonnegative additive bound $g(t) = o(\log^* t)$. There is an integer $T_g \geq 2$ such that $g(t) \leq \epsilon_0 \log^* t$ for every integer $t \geq T_g$. Replace the threshold by

$$n_0 := \max\{T_g, 1203, E_{32D-1} + 1\}.$$

Because $B \geq n \geq T_g$, we have $g(B) \leq \epsilon_0 \log^* B$, so the comparison in Eq. (11) holds on every YES instance above this threshold, while feasibility alone handles NO instances. No bound on g at the optimum of a NO instance and no monotonicity assumption on g are needed. The threshold may depend on the hypothesized algorithm and its promised function g, but it is one fixed integer for the decision algorithm. Its existence does not require a procedure for recovering T_g from a description of g: one admissible threshold is fixed in the algorithm's finite description, rather than supplied as advice that varies with input length. The exhaustive-search branch handles all remaining inputs in polynomial time, giving the same contradiction. □

4.2 Construction and local properties

We first specify the common construction and its notation. The four lemmas below isolate the tree, geometric, encoding, and defect properties used by the reduction proofs.

Input graph and parameters. Let Φ be a formula with $M \geq 1$ clauses, each containing exactly three literal slots, and define $n := 3M$ and $k := 2M$. Form a simple graph G by making the three slots of each clause a triangle and joining slots carrying complementary literals, removing duplicate edges; write $m := |E(G)|$. Fix a nonnegative integer c and an integer d, with

$$P_0 := 2^5 \cdot 6^4, \qquad K := 20(c+1)(1+5P_0), \qquad d \geq 1000K. \tag{12}$$

Competing trees. Start an auxiliary rooted tree with one vertex and perform $2d-1$ phases. If the current leaves are $(u_i)_{i\in[N]}$ at depth s, the next phase adds N depths and sets $p_{s+i} := 2^{i-1}$ for $i \in [N]$. At u_i, attach 2^i paths of i edges that are disjoint except at u_i, and color half of their endpoints red and half blue. For $i < N$, attach to each endpoint $1+2^N-2^i$ paths of $N-i$ edges that are disjoint except at that endpoint. All added vertices are new; only spine endpoints are colored. When $i = N$, no continuation is needed. Thus the new leaves all have depth $s+N$. After the last phase, let L be the common leaf depth and define $P := \sum_{\ell\in[L]} p_\ell$. In two separate copies retain all complete paths containing at least d red endpoints and at least d blue endpoints, respectively. These unions are T^+ and T^-; retain every vertex on each chosen path without contracting edges. A marking is a set of vertices at positive depths; it hits a path if it contains a vertex of that path. Write $V_\ell(T)$ for the vertices of T at depth ℓ.

Packing parameters. With these trees and quotas, define

$$R := 100(P + nK + 1), \qquad t := d(|V(T^+)| + |V(T^-)|) + P, \qquad B := nt + 4dmR.$$

Coordinates. Index the vertices of G by $[n]$ and its edges by $[m]$. A position is a pair $r = (e, j) \in [m] \times [R]$, ordered lexicographically. Set

$$g_{\text{geom}} := \frac{1}{16(R+1)^m}, \qquad \theta_e := (R+1)^{e-1} - 1,$$
$$H_{e,j} := j(R+1)^{e-1} - 1, \quad b_{(e,j)} := \frac{1}{2} + g_{\text{geom}} H_{e,j}, \qquad \beta_e := g_{\text{geom}} \theta_e.$$

Write $g := g_{\text{geom}}$ and $H_r := H_{e,j}$ for $r = (e, j)$. For each vertex v, define $\mathcal{R}_v := \{(e, j) : v \in e,\ j \in [R]\}$ and $J_v := d|\mathcal{R}_v| = dR \deg(v)$. Let D_* be the maximum of one and the child counts in the retained trees, and set

$$A := 100(D_* + 1), \qquad \lambda_\ell := g_{\text{geom}} A^{-\ell}, \qquad \Delta := \lambda_L / 10.$$

Give each vertex label v its own copy of the following cells. The roots of T^+ and T^- have cells $[0, 1]$ and $[4, 5]$. For a parent cell with left endpoint a, its jth child at depth ℓ has cell $[a + 4j\lambda_\ell, a + (4j + 1)\lambda_\ell]$. Add the job cell $[b_r - \Delta, b_r]$ for each $r \in \mathcal{R}_v$. The test intervals are all leaf cells and job cells with their right endpoints removed; write $\mathcal{I}_v$ for their union.

Item inventory. An item has a subclass, a coordinate $w(i)$, and, for rows, anchors, and local auxiliaries, a vertex label. The five roles and their subclasses are

role	X	A	U	D	F
subclasses	$\mathrm{T}^+, \mathrm{T}^-, \mathrm{S}$	Z, Y	$\mathrm{U}^+, \mathrm{U}^-, \mathrm{W}$	$\mathrm{D}_M, \mathrm{D}_G$	$\mathrm{F}_T, \mathrm{F}_M, \mathrm{F}_G$

The complete inventory is as follows; each stated multiplicity specifies the number of distinct physical copies.

1. At each label v, every node of T^+ or T^-, including its root, contributes d rows of subclass T^+ or T^-. Their baseline b and coordinate w are the cell's right endpoint. Add P rows of subclass T^+ with $b := 1$ and $w := 1$. Define $t_+ := d|V(T^+)| + P$ and $t_- := d|V(T^-)|$, so $t = t_+ + t_-$. For each $r \in \mathcal{R}_v$, add d short and d long rows of subclass S, all with baseline b_r; their coordinates are $w := b_r - \Delta$ and $w := b_r$, respectively.

2. Form a multiset $\mathcal{B}_v$ from all t tree-row baselines, including padding, and d copies of b_r for each $r \in \mathcal{R}_v$. For a leaf test interval, use the d rows of that leaf; for a job test interval, use its d added baseline copies. Move these designated right-endpoint copies to the corresponding left endpoints. The resulting $t + J_v$ numbers are deadlines z; create one Z anchor with $w := -z$ for each. For each $r = (e, j) \in \mathcal{R}_v$, also create d key anchors of subclass Y with $w := -b_r - \beta_e$. All these anchors have label v.

3. Create $nt_+ + \sum_{v \in [n]} J_v$ global auxiliaries of subclass U^+, with length one for dk copies and zero for the rest. Create nt_- auxiliaries of subclass U^-, with length one for $d(n - k)$ copies and zero for the rest. For both subclasses define $w := -\ell$, where ℓ is the item's length. For each edge e, create dR permits and dR nonpermits of subclass W, with coordinates $w := \beta_e$ and $w := \beta_e + \Delta$.

4. At label v, create $t + J_v$ local auxiliaries of subclass D_M: for every $\ell \in [L]$, exactly p_ℓ copies have length λ_ℓ; the remaining $t + J_v - P$ have length zero. Their coordinate is the negative of their length. Also create J_v copies of subclass D_G with $w := 0$ and label v.

5. Create nt flags of subclass F_T, and $\sum_{v\in[n]} J_v$ flags of each subclass F_M and F_G, all with $w := 0$.

For real a, write $B_v(a) := \#\{b \in \mathcal{B}_v : b \le a\}$, counting multiplicity in the original baseline multiset.

Encoding the sizes. For any collection of items, let N_s and N_r count its subclass-s and role-r items. Use the following nine homogeneous equations: first $N_r - N_{\mathrm{F}} = 0$ for $r = \mathrm{X}, \mathrm{A}, \mathrm{U}, \mathrm{D}$ in that order, then

$$\begin{aligned} N_{\mathrm{T}^+} + N_{\mathrm{T}^-} - N_{\mathrm{F}_T} &= 0, \\ N_{\mathrm{Z}} - N_{\mathrm{F}_T} - N_{\mathrm{F}_M} &= 0, \\ N_{\mathrm{U}^+} + N_{\mathrm{U}^-} - N_{\mathrm{F}_T} - N_{\mathrm{F}_M} &= 0, \\ N_{\mathrm{D}_M} - N_{\mathrm{F}_T} - N_{\mathrm{F}_M} &= 0, \\ N_{\mathrm{T}^-} - N_{\mathrm{U}^-} &= 0. \end{aligned}$$

Let $\delta_j(i)$ be item i's coefficient in the jth left side, indexed from 0 to 8, and define

$$p(i) := \sum_{j=0}^{3} 6^j \delta_j(i) + 6^4 \sum_{j=0}^{4} 2^j \delta_{j+4}(i).$$

Set $q(i) := 3^v$ on roles X, D with label v, $q(i) := -2 \cdot 3^v$ on role A with label v, and $q(i) := 0$ on the other roles. Output the rational size

$$a_i := \frac{1}{5} + \frac{p(i)}{1000 P_0} + \frac{q(i)}{10^6 P_0 \cdot 3^n} + \frac{w(i)}{10^9 P_0 \cdot 3^n} \tag{13}$$

for each physical copy.

A table tuple consists of five distinct physical item copies whose subclasses follow one of the following patterns, in role order $(\mathrm{X}, \mathrm{A}, \mathrm{U}, \mathrm{D}, \mathrm{F})$:

M	T^+	Z	U^+	D_M	F_T
M	T^-	Z	U^-	D_M	F_T
M	S	Z	U^+	D_M	F_M
G	S	Y	W	D_G	F_G

The first three patterns have kind M, and the last has kind G. A table tuple is good when its $\mathrm{X}, \mathrm{A}, \mathrm{D}$ labels agree; feasibility is a separate condition. For a group of a row, two auxiliaries, and a flag compatible with an M pattern, write b for its row baseline and define its completion by

$$h := b - \Delta \mathbf{1}_{\{\text{row is short}\}} - \text{global length} - \text{local length}. \tag{14}$$

Associate with this group the half-open interval $[h, b)$ before choosing a deadline anchor. Each retained tree has a marking that hits every complete path at least d times. Under the combined depth quotas, no pair of markings hits every complete path in both trees.

Lemma 4.2 (Competing-tree properties)**.** *The competing trees have the following properties.*

(a) *The trees T^+ and T^- are nonempty and have all leaves at depth L.*

(b) *Each T^σ, for $\sigma \in \{+,-\}$, has a marking with at most p_ℓ vertices at each depth ℓ and at least d vertices on every root-to-leaf path.*

(c) *If markings S^+ and S^- satisfy*

$$|S^+ \cap V_\ell(T^+)| + |S^- \cap V_\ell(T^-)| \le p_\ell \qquad (\ell \in [L]),$$

then at least one retained tree has a root-to-leaf path avoiding its marking.

Proof. **Proof of Part (a)**. A path choosing the same color in every phase shows that each retained tree is nonempty. Retaining complete paths without contracting edges preserves depths and leaves no new leaf before depth L.

Proof of Part (b). At depth $s+i$ there are p_{s+i} endpoints of each color. Marking the retained red endpoints in T^+, or the retained blue endpoints in T^-, therefore respects the depth quotas and hits every path at least d times.

Proof of Part (c). A marking with at most p_ℓ vertices at depth ℓ cannot hit every auxiliary path: at u_i, the first i new depths allow only 2^i-1 marks, leaving a spine unhit. If $i < N$, the later depths allow $2^N - 2^i$ marks, leaving one continuation unhit; if $i = N$, the spine already reaches the phase boundary. Repeating this choice through the phases gives an unhit complete path. Each complete auxiliary path has $2d-1$ colored endpoints and therefore belongs to at least one retained tree. Projecting markings from both retained trees to the auxiliary tree preserves depths and can only merge marks. Since every auxiliary path belongs to at least one retained tree, markings hitting all paths in both trees would project to a quota-respecting marking hitting every auxiliary path. This contradicts the unhit-path argument. □

The coordinate scales separate tree cells and job positions while preserving their order. The same geometry bounds every item coordinate and gives an exact formula for the deadline prefixes.

Lemma 4.3 (Geometry and deadline prefixes)**.** *The construction has the following properties.*

(a) *For every $\ell \in [L]$,*

$$\Delta \le \lambda_\ell/10, \qquad \lambda_1 + 2\Delta \le \frac{6}{5}\lambda_1 \le \frac{3g}{500} < g.$$

At each vertex label, child cells lie inside their parent cells, and nonroot cells lie in the first tenth of their root cell. Distinct depth-ℓ cells across the two trees have gaps at least $3\lambda_\ell$. Consequently an interval of length at most $\lambda_\ell + \Delta$ meets at most one such closed cell; if it contains a leaf point, that cell is the leaf's depth-ℓ ancestor.

(b) *The position coordinates satisfy $0 \le H_{e,j} \le R(R+1)^{m-1} - 1 < (R+1)^m$ and $b_r \in [1/2, 9/16)$. Distinct closed job cells have gaps at least $g - \Delta$. An interval of length at most $\lambda_1 + \Delta$ meets at most one job cell. At each vertex label, all test intervals are pairwise disjoint.*

(c) *The position and edge orders satisfy*

$$\begin{aligned} H_{r'} - H_r &\ge \theta_e + 1 \qquad \text{if } r = (e,j) \text{ and } r' > r, \\ \theta_f - \theta_e &\ge H_{e,R} + 1 \quad \text{if } f > e. \end{aligned}$$

(d) *Every item coordinate satisfies $|w(i)| \le 6$. For every vertex label v and every real a,*

$$\#\{\mathrm{Z} \text{ anchors of label } v \text{ with } z \le a\} = B_v(a) + d\mathbf{1}_{\{a \in \mathcal{I}_v\}}.$$

Proof. **Proof of Part (a)**. The coordinate definitions give $\lambda_\ell = gA^{-\ell}$, $A = 100(D_*+1) \geq 200$, and $\Delta = \lambda_L/10$. The first bound in Part (a) follows from $\lambda_L \leq \lambda_\ell$. For the second chain of inequalities in Part (a), the first step follows from $\Delta \leq \lambda_1/10$, the second step follows from $\lambda_1 = g/A$ and $A \geq 200$, and the third step follows from $g > 0$. The right endpoint of the last child is at most $(4D_* + 1)\lambda_\ell$ beyond its parent's left endpoint. For $\ell > 1$ this is less than $\lambda_{\ell-1}$; for $\ell = 1$ it is less than $g/25 < 1/10$. Thus cells are nested, and nonroot cells lie in the first tenth of their root cell. Sibling depth-ℓ cells have gaps $3\lambda_\ell$; cells in different parents inherit larger gaps, and the two root cells have gap three. Induction therefore gives gaps at least $3\lambda_\ell$ between any two distinct depth-ℓ cells across both trees. An interval of length at most $\lambda_\ell + \Delta < 3\lambda_\ell$ can meet at most one such closed cell. If it contains a leaf point, that cell is the leaf's ancestor at depth ℓ.

Proof of Part (b). Since $e \in [m]$ and $j \in [R]$, the formula $H_{e,j} = j(R+1)^{e-1} - 1$ gives $0 \leq H_{e,j} \leq R(R+1)^{m-1} - 1 < (R+1)^m$. Consequently $b_r = 1/2 + gH_r \in [1/2, 9/16)$. Consecutive job baselines differ by at least g, so distinct closed job cells have gaps at least $g - \Delta$. Since $\lambda_1 + \Delta < g - \Delta$ by Part (a), an interval of length at most $\lambda_1 + \Delta$ meets at most one job cell. Since $\Delta < g \leq 1/16$, job cells lie strictly to the right of $1/10$ and to the left of one. They are therefore disjoint from all leaf cells. Together with the cell gaps, this proves that all test intervals at each vertex label are pairwise disjoint.

Proof of Part (c). For Part (c), the next position after $r = (e, j)$, when it exists, increases H by $(R+1)^{e-1} = \theta_e + 1$, including a transition to the next edge. Also $\theta_{e+1} - \theta_e = R(R+1)^{e-1} = H_{e,R} + 1$. Later positions or edges only increase these differences.

Proof of Part (d). All tree-cell endpoints and all deadlines lie in $[0, 5]$, and every auxiliary length is at most one. The coordinate formulas also give $0 \leq \beta_e < 1/16$ and $0 < \Delta < 1/16$. The inventory assigns coordinates of the forms b, $b_r - \Delta$, $-z$, $-b_r - \beta_e$, the negative of an auxiliary length, β_e, $\beta_e + \Delta$, and zero. Each coordinate therefore has absolute value at most six.

Moving d designated baseline copies from a right endpoint r_0 to a left endpoint $\ell_0 < r_0$ changes the prefix count at a by

$$d(\mathbf{1}_{\{\ell_0 \leq a\}} - \mathbf{1}_{\{r_0 \leq a\}}) = d\mathbf{1}_{\{\ell_0 \leq a < r_0\}}.$$

The test intervals are pairwise disjoint and use distinct designated copies. Summing these changes over all test intervals at label v gives the identity in Part (d), including their left and right endpoints. □

The inventory and scalar encoding realize the prescribed patterns through the bin-capacity constraint. We collect the resulting size, conservation, and feasibility properties for use in both directions of the reduction.

Lemma 4.4 (Inventory, encoding, and tuple feasibility)**.** *The construction has the following properties.*

(a) All multiplicities are nonnegative. Each role has exactly B items, and each of the roles $\mathrm{X}, \mathrm{A}, \mathrm{D}$ *has $t + 2J_v$ items of label v. Every item satisfies $|p(i)| < P_0$ and $a_i \in (1/6, 7/30)$, and the complete inventory satisfies $\sum_i p(i) = \sum_i q(i) = 0$.*

(b) A five-item tuple has zero primary total if and only if it is a table tuple. Every feasible five-item tuple has nonpositive primary total; if that total is zero, its secondary total is nonpositive as well. In a feasible table tuple, write v_X, v_A, v_D for the labels of its $\mathrm{X}, \mathrm{A}, \mathrm{D}$ *items. Then $v_X, v_D \leq v_A$, and its secondary total is zero if and only if $v_X = v_A = v_D$.*

(c) A good tuple is feasible if and only if its coordinate sum is nonpositive. In particular, a good M *tuple with completion h, baseline b, and deadline z is feasible if and only if $h \leq z$; always $h \leq b$.*

(d) A good G *tuple whose key has position* $r = (e, j)$*, row has position* r'*, and* W *item comes from edge* f *is feasible if and only if*

$$b_{r'} - b_r + \beta_f - \beta_e + \Delta(\mathbf{1}_{\{nonpermit\}} - \mathbf{1}_{\{row\ is\ short\}}) \le 0.$$

Every feasible such tuple satisfies $r' \le r$ *and* $f \le e$*. If such a feasible tuple has a long row and* $r' = r$*, its* W *item is either a permit of edge* e *or an item from an earlier edge.*

Proof. **Proof of Part (a).** The roots imply $t_+, t_- \ge d$, and the plus padding gives $t \ge P + 2d$. Thus the global length-one stocks fit their respective inventories because $0 \le k \le n$, and the P positive local items fit the $t + J_v$ local slots. Every other multiplicity is nonnegative by its definition. Since $\sum_{v\in[n]} J_v = 2dmR$, each of the roles X, A, D has $t+2J_v$ items at label v, and hence $nt+2\sum_{v\in[n]} J_v = B$ items in total. The global auxiliary count is $nt_+ + \sum_{v\in[n]} J_v + nt_- + 2dmR = B$, and the flag count is $nt + 2\sum_{v\in[n]} J_v = B$. Therefore there are exactly $5B$ items.

Equal role counts make the four role equations vanish on the full inventory. The combined $\mathrm{T}^+, \mathrm{T}^-$ count is nt, which equals the F_T count. The Z count, the combined $\mathrm{U}^+, \mathrm{U}^-$ count, and the D_M count all equal $nt + \sum_{v\in[n]} J_v$, which is the combined $\mathrm{F}_T, \mathrm{F}_M$ count. Finally, the T^- and U^- counts both equal nt_-. Thus all nine equation totals vanish, giving $\sum_i p(i) = 0$. At label v, the secondary total is $(1 - 2 + 1)3^v(t + 2J_v) = 0$, so $\sum_i q(i) = 0$ as well.

Each coefficient $\delta_j(i)$ belongs to $\{-1, 0, 1\}$, so

$$|p(i)| \le \frac{6^4 - 1}{5} + 6^4(2^5 - 1) < 2^5 \cdot 6^4 = P_0, \tag{15}$$

where the first step follows from the triangle inequality and the two geometric sums, the second step follows from $(6^4 - 1)/5 < 6^4$, and the third step follows from the definition of P_0. Also $|q(i)| \le 2 \cdot 3^n$ by definition, and $|w(i)| \le 6$ by Lemma 4.3-(d). Hence Eq. (13) gives

$$|a_i - 1/5| \le \frac{1}{1000} + \frac{2}{10^6 P_0} + \frac{6}{10^9 P_0 \cdot 3^n} < \frac{1}{30},$$

where the first step follows from the triangle inequality, Eq. (15), the bound $|q(i)| \le 2 \cdot 3^n$, and Lemma 4.3-(d), and the second step follows from $P_0 \ge 1$ and $n \ge 1$. This proves $a_i \in (1/6, 7/30)$.

Proof of Part (b). For a collection of five items, each role-equation total is an integer of magnitude at most 5. If the primary total is zero, four successive reductions modulo 6, each followed by division by 6, force the four role equations. All five role counts are equal and sum to five, so each equals one. Each remaining equation is now a difference of two indicators, so its total belongs to $\{-1, 0, 1\}$. After the four divisions, successive reduction modulo 2 and division by 2 force all five remaining totals to vanish. With flag F_T, these equations require a tree row, Z, D_M, and U^+ or U^- of the same sign as the row. With flag F_M, they require $\mathrm{S}, \mathrm{Z}, \mathrm{U}^+, \mathrm{D}_M$. With flag F_G, they require $\mathrm{S}, \mathrm{Y}, \mathrm{W}, \mathrm{D}_G$. These are precisely the four table patterns. Conversely, every table pattern satisfies all nine equations and therefore has zero primary total.

Define

$$\gamma := \frac{1}{1000P_0}, \qquad \mu := \frac{\gamma}{1000 \cdot 3^n}, \qquad \nu := \frac{\mu}{1000}.$$

For five items, the total contribution of the two scales below the primary scale has absolute value at most $10 \cdot 3^n\mu + 30\nu$, and

$$\frac{10 \cdot 3^n\mu + 30\nu}{\gamma} = \frac{1}{100} + \frac{3}{100000 \cdot 3^n} < 1, \qquad \frac{30\nu}{\mu} = \frac{3}{100} < 1, \tag{16}$$

where, in the first chain, the first step follows from the definitions of μ and ν, and the second step follows from $n \geq 1$; in the second chain, the first step follows from $\nu = \mu/1000$, and the second step follows from $3 < 100$. By the first inequality in Eq. (16), a positive integer primary total would make the sum of sizes exceed one. If the primary total is zero, the second inequality in Eq. (16) shows that a positive integer secondary total would also make the sum exceed one. This proves the asserted nonpositivity statements. In a feasible table tuple, the secondary total is $3^{v_X} + 3^{v_D} - 2 \cdot 3^{v_A} \leq 0$. If $v_X > v_A$, then $3^{v_X} \geq 3 \cdot 3^{v_A}$, contradicting this inequality; the same argument applies to v_D. Thus $v_X, v_D \leq v_A$. If either inequality is strict, then $3^{v_X} + 3^{v_D} < 2 \cdot 3^{v_A}$; if both are equalities, the secondary total is zero. This proves the label assertions.

Proof of Part (c). A good tuple has zero primary and secondary totals, so its size sum is $1 + \nu \sum_i w(i)$. Since $\nu > 0$, it is feasible exactly when its coordinate sum is nonpositive. For a good M tuple the coordinate sum is $h - z$, by the definition of the completion. Thus feasibility is equivalent to $h \leq z$, and nonnegative lengths imply $h \leq b$.

Proof of Part (d). For a good G tuple, adding the row, key, auxiliary, local, and flag coordinates gives exactly the left side of the inequality in Part (d). By Part (c), the tuple is feasible if and only if this sum is nonpositive.

If $r' > r$, Lemma 4.3-(c) and $\theta_f \geq 0$ make that left side at least $g(\theta_e + 1) - g\theta_e - \Delta = g - \Delta > 0$, a contradiction. Independently, if $f > e$, the second inequality in that part and $H_{r'} \geq 0$, $H_r \leq H_{e,R}$ make it at least $-gH_{e,R} + g(H_{e,R} + 1) - \Delta = g - \Delta > 0$. Here $g > \Delta$ follows from Lemma 4.3-(a). Therefore every feasible good G tuple satisfies $r' \leq r$ and $f \leq e$. If the row is long and $r' = r$, a same-edge nonpermit would leave the positive coordinate sum Δ. Thus a same-edge item must be a permit, and the only other possible edge indices satisfy $f < e$. □

A packing with few extra bins leaves few items outside the prescribed tuple patterns. Conservation of the labeled inventories then bounds the number of anchors whose tuples have mismatched labels.

Lemma 4.5 (Items and anchors outside good tuples)**.** *For every feasible packing into at most $B + c$ bins, the following hold.*

(a) Fewer than K items lie outside table tuples.

(b) For each label $v \in [n]$, at most $3K$ anchors of label v are outside good tuples, counting the Z *and* Y *anchors together. The same bound applies to every subset of the anchors of label v.*

Proof. Let $N_{\text{bin}} \leq B + c$ be the number of bins in the packing. All tuples counted below are bins of this fixed packing.

Proof of Part (a). By Lemma 4.4, every item has size greater than $1/6$, so every bin contains at most five items and $N_{\text{bin}} \geq B$. Call a bin full if it contains five items. The number of unused slots relative to five per bin is $5N_{\text{bin}} - 5B \leq 5c$. Each nonfull bin contributes at least one unused slot, so there are at most $5c$ nonfull bins and at most $20c$ items in those bins.

Lemma 4.4 also gives $|p(i)| \leq P_0$, zero total primary score over the inventory, and a nonpositive integer primary total in every full bin. Consequently, the sum of the magnitudes of the negative full-bin primary totals equals the primary total of the nonfull-bin items and is at most $20cP_0$. Every negative full-bin total is at most -1, so at most $20cP_0$ full bins have negative primary total. The full bins with zero primary total are precisely the table tuples, by Lemma 4.4. It follows that

$$\#\{\text{items outside table tuples}\} \leq 20c + 5(20cP_0) = 20c(1 + 5P_0) < K, \tag{17}$$

where the first step follows from the bounds of $20c$ items in nonfull bins and $20cP_0$ full bins with negative primary total, each containing five items, the second step follows from factoring out $20c$, and the third step follows from $K = 20(c+1)(1+5P_0)$ and $c < c+1$.

Proof of Part (b). For a table tuple, write v_X, v_A, v_D for the labels of its X, A, D items. Lemma 4.4 gives $v_X, v_D \le v_A$. For $a \in \{0\} \cup [n]$, define $S := [a]$ and fix $r \in \{\mathrm{X}, \mathrm{D}\}$. Let $O_r(S)$ and $O_A(S)$ count, respectively, the role-r items and anchors with labels in S that lie outside table tuples. Let $E_r(S)$ count the table tuples whose role-r label belongs to S and whose anchor label does not belong to S. The original inventories of both roles with labels in S have the same cardinality, namely $\sum_{v \in S}(t + 2J_v)$. Every table tuple whose anchor label belongs to S consumes a role-r item with label in S, because $v_r \le v_A$. Thus the number of role-r items with labels in S in table tuples equals the number of anchors with labels in S in table tuples plus $E_r(S)$. Subtracting these counts from the equal original inventories gives

$$E_r(S) = O_A(S) - O_r(S) \le O_A(S) < K, \tag{18}$$

where the first step follows from the equal original role inventories and the fact that the role-r tuple count exceeds the anchor tuple count by $E_r(S)$, the second step follows from $O_r(S) \ge 0$, and the third step follows from Eq. (17), since $O_A(S)$ counts only anchors outside table tuples.

Fix an anchor label v. If an anchor of label v lies in a table tuple with a mismatched X label, then $v_X < v$ and this tuple is counted by $E_X([v-1])$. The analogous assertion holds for a mismatched D label and $E_D([v-1])$. Each table tuple has one anchor, so Eq. (18) shows that each of these two sets contains fewer than K anchors. Eq. (17) shows that fewer than K anchors in total lie outside table tuples. A table tuple is good exactly when its X and D labels both match its anchor label, so the union of these three sets contains every anchor of label v outside good tuples. Their union has at most $3K$ anchors, with the Z and Y anchors counted together. Every subset of the anchors of label v inherits this bound. □

4.3 Completeness of the reduction

A vertex cover determines which side receives the length-one global items at each label. The construction properties then allow every item to be assigned to a feasible bin.

Lemma 4.6 (Completeness of the packing reduction)**.** *For the instance constructed in Section 4.2 with any admissible parameter choice, if Φ is satisfiable, then* $\mathrm{OPT} = B$.

Proof. The literal-slot graph has $n = 3M$ vertices, $1 \le m \le n^2$ edges, and cover target $k = 2M = 2n/3$. An independent set selects consistent literals from distinct clauses, which an assignment can satisfy simultaneously. Conversely, selecting one true slot from every satisfied clause gives an independent set. Hence

$$\tau(G) = 3M - \mathrm{MAXSAT}(\Phi).$$

If Φ is satisfiable, then $\tau(G) = k$. This graph can be constructed in time polynomial in the length of the explicit formula encoding, including its variable names.

Packing a satisfiable instance. Fix a vertex cover C of size k. We use the markings from Lemma 4.2-(b) and the tuple criteria from Lemma 4.4. At each label v, use all tree rows in type M tuples. At each position use its d long rows in M if $v \in C$, and its d short rows otherwise. Give the d plus-root rows length-one U^+ items when $v \in C$; otherwise give the d minus-root rows length-one U^- items. Give every other row assigned to M a length-zero global item of the subclass prescribed by its pattern, and reserve zero local items for these funded roots. On the unfunded tree, use its

separate marking from Lemma 4.2-(b). At each marked depth-ℓ node, assign a length-λ_ℓ local item to one of that node's rows. The depth quotas supply these items. If $s_v \le P$ rows receive them, then $t + J_v - s_v - d \ge P - s_v$ remaining slots suffice for every unused positive local item; fill the other slots with zeros. Attach F_T flags to tree rows and F_M flags to job rows.

For each resulting group, let h be its completion from Eq. (14) and let b be its baseline. The funded roots cover every test interval on their side d times through intervals $[h, b)$. The separate marking and the nesting property in Lemma 4.3 give at least d coverings of every leaf interval on the other side. When the plus side is unfunded, its d short rows cover each job interval. Consequently every test interval has coverage at least d. The baseline multiset of these groups is exactly $\mathcal{B}_v$. For each real a, the number of completions at most a is therefore at least the number of baselines at most a, plus $d\mathbf{1}_{\{a \in \mathcal{I}_v\}}$. This is the deadline count by Lemma 4.3-(d). Sorting the completions and deadlines and pairing them in order gives $h \le z$ for every pair. Attach the corresponding Z anchors; each resulting good M tuple is feasible by $h \le z$ and Lemma 4.4.

At each position, pair its remaining d rows with the d keys and D_G items of the same label, and attach F_G flags. These rows are short at selected vertices and long otherwise. Using a W item from that exact edge, the coordinate sum is

$$\Delta(\mathbf{1}_{\{\text{nonpermit}\}} - \mathbf{1}_{\{\text{row is short}\}}).$$

Thus long rows require permits and short rows accept either kind. Since C covers every edge, at most one endpoint supplies long rows; the dR permits suffice for them, and all remaining slots are short rows that consume the remaining permits and nonpermits. Lemma 4.4 therefore makes all these G tuples feasible. This exhausts every inventory and yields $t + J_v$ bins of type M and J_v bins of type G at label v, totaling B feasible bins. By Lemma 4.4, every item has size greater than $1/6$ and there are $5B$ items. No bin can contain six items, so at least B bins are required, proving $\mathrm{OPT} = B$. □

4.4 Soundness of the reduction

A packing with at most $B + c$ bins forces a small vertex cover in the literal-slot graph. The gap between $k + n/50$ and $k + n/48$ then excludes such a packing in the NO case of the fixed gap problem.

Lemma 4.7 (Soundness of the packing reduction)**.** *For every formula Φ and every admissible parameter choice in Section 4.2, the corresponding packing instance satisfies*

$$\mathrm{MAXSAT}(\Phi) \le \frac{15M}{16} \quad \Longrightarrow \quad \mathrm{OPT} > B + c.$$

Proof. Suppose that at most $15M/16$ clauses of Φ can be satisfied and that a feasible packing uses $N_{\mathrm{bin}} \le B + c$ bins. All tuples counted below are bins of this fixed packing. We can show

$$\tau(G) \ge 3M - \frac{15M}{16} = k + \frac{n}{48}, \tag{19}$$

where the first step follows because every independent set selects consistent literals from distinct clauses and therefore has size at most $15M/16$, while the complement of a maximum independent set is a minimum vertex cover; the second step follows from $k = 2M$ and $n = 3M$. By Lemma 4.5, at most $3K$ anchors of each label lie outside good tuples, jointly for both anchor subclasses. We use the geometry and tuple criteria from Lemmas 4.3 and 4.4 to extract a small vertex cover in five steps.

Step 1: Force coverage at every test point. Let $\mathcal{M}_v$ denote the set of good M tuples with label v, using the completions from Eq. (14). Let x_v^+ and x_v^- count those using a length-one U^+ and U^- item, respectively. Distinct tuples use distinct physical copies, so

$$\sum_{v\in[n]} x_v^+ \le dk, \qquad \sum_{v\in[n]} x_v^- \le d(n-k), \qquad \sum_{v\in[n]} (x_v^+ + x_v^-) \le dn, \tag{20}$$

where the first step follows from the inventory of dk length-one U^+ items, the second step follows from the inventory of $d(n-k)$ length-one U^- items, and the third step follows from adding the first two bounds. For real a, define the three counts

$$\begin{aligned} A_v(a) &:= \#\{T \in \mathcal{M}_v : b(T) \le a\}, \\ C_v(a) &:= \#\{T \in \mathcal{M}_v : h(T) \le a\}, \\ I_v(a) &:= \#\{T \in \mathcal{M}_v : h(T) \le a < b(T)\}. \end{aligned}$$

Since $h(T) \le b(T)$, the condition $h(T) \le a$ holds exactly when one of the mutually exclusive conditions $b(T) \le a$ and $h(T) \le a < b(T)$ holds, including when endpoints coincide. Consequently

$$C_v(a) = A_v(a) + I_v(a). \tag{21}$$

By Lemma 4.3-(d), the number of deadlines at most a is $B_v(a) + d\mathbf{1}_{\{a\in\mathcal{I}_v\}}$. For $a \in \mathcal{I}_v$, Lemma 4.5 leaves at least $B_v(a) + d - 3K$ good anchors among these deadlines. Their matched completions satisfy $h \le z \le a$, hence

$$C_v(a) \ge B_v(a) + d - 3K. \tag{22}$$

Let N' count positions in $\mathcal{R}_v$ with $b_r \le a$. These positions have $2dN'$ actual job rows and dN' keys. At least $dN' - 3K$ keys lie in good G tuples, and Lemma 4.4 places their distinct rows in the same baseline prefix. Thus at most $2dN' - (dN' - 3K) = dN' + 3K$ job rows from that prefix can belong to $\mathcal{M}_v$. This upper bound remains valid if $dN' - 3K < 0$. The tree rows contribute at most their original prefix count, and $\mathcal{B}_v$ includes exactly dN' job baselines in this prefix. It follows that $A_v(a) \le B_v(a) + 3K$, and therefore

$$I_v(a) = C_v(a) - A_v(a) \ge d - 6K \qquad (a \in \mathcal{I}_v), \tag{23}$$

where the first step follows from Eq. (21), and the second step follows from Eq. (22) and $A_v(a) \le B_v(a) + 3K$.

Step 2: Force a large allocation on one side. A plus-row interval has right endpoint at most one. A minus row is never short and has $h \ge 4 - 1 - \lambda_1 > 1$. Thus an interval can cover test points only on its own side. Choose an interior point of every leaf cell in both trees. An M tuple with both auxiliary lengths zero has an empty interval, except for a short job row whose interval is its job test interval; neither case covers a chosen leaf point. A tuple with global length zero and local length λ_ℓ has interval length at most $\lambda_\ell + \Delta$. By Lemma 4.3, it intersects at most one depth-ℓ cell across the two trees. If it intersects one, mark that cell's vertex. If its interval covers a chosen leaf point, the marked vertex is the leaf's depth-ℓ ancestor. Apply this marking rule to every such tuple of label v, merging repeated marks. At depth ℓ, the number of marks across both trees is at most the number of used length-λ_ℓ local items, hence at most p_ℓ. All these marks have positive depth.

If both x_v^+ and x_v^- were less than $d - 6K$, then Eq. (23) would force every chosen leaf point to be covered by a tuple with global length zero and positive local length: the length-one global tuples on that side supply at most x_v^+ or x_v^- coverings, and zero-length pairs supply none. The resulting marks would hit every root-to-leaf path in both trees, contrary to Lemma 4.2-(c). We have proved

$$x_v^+ \ge d - 6K \quad \text{or} \quad x_v^- \ge d - 6K \qquad \text{for every } v. \tag{24}$$

Step 3: Count long matches at each nonexcluded position. Exclude $r \in \mathcal{R}_v$ if its closed job cell meets the interval of a good M tuple of label v with global length zero and positive local length. Such an interval has length at most $\lambda_1 + \Delta$ and meets at most one job cell by Lemma 4.3. Since label v has only P positive local items, at most P positions are excluded. Let $L_{v,r}$ count good G tuples of label v with key at r and a long row at that same position.

Fix a nonexcluded r and define $a := b_r - \Delta/2$, so $a \in \mathcal{I}_v$. Job baselines have gaps at least $g > \Delta$, so those at most a are exactly the positions before r. Let N' be their number in $\mathcal{R}_v$, and let ξ count good matches of keys at r to rows at earlier positions. Good keys at earlier positions consume at least $dN' - 3K$ distinct rows from the earlier prefix. The ξ current-key matches consume ξ additional rows from it. Of its $2dN'$ job rows, at most $dN' + 3K - \xi$ can therefore occur in $\mathcal{M}_v$. Adding the original tree-row prefix yields

$$A_v(a) \le B_v(a) + 3K - \xi. \tag{25}$$

Let u count short rows at r used in $\mathcal{M}_v$. Intervals with length-one global items contribute at most x_v^+ coverings of a. Intervals with global length zero and positive local length contribute none, since r is nonexcluded. With both lengths zero, only a short row at r can cover a. Consequently $I_v(a) \le x_v^+ + u$; any double counting only increases this upper bound. We can show

$$B_v(a) + d - 3K \le C_v(a) = A_v(a) + I_v(a) \le B_v(a) + 3K - \xi + x_v^+ + u, \tag{26}$$

where the first step follows from Eq. (22), the second step follows from Eq. (21), and the third step follows from Eq. (25) and $I_v(a) \le x_v^+ + u$. Rearranging Eq. (26) gives $u - \xi \ge d - x_v^+ - 6K$. At least $d - 3K$ keys at r have good matches. Exactly ξ use earlier rows; Lemma 4.4 places all other matched rows at r. At most $d - u$ short rows at r remain available for these matches. It follows that

$$L_{v,r} \ge d - 3K - \xi - (d - u) = u - \xi - 3K \ge d - x_v^+ - 9K, \tag{27}$$

where the first step follows from subtracting the ξ earlier matches and at most $d - u$ available short rows from at least $d - 3K$ good key matches, the second step follows from collecting terms, and the third step follows from rearranging Eq. (26).

Step 4: Extract a vertex cover. Define $C := \{v : x_v^+ > 2d/5\}$ and fix an edge with index e and endpoints v_1, v_2. The earlier edges provide exactly $2dR(e-1)$ items of subclass W and the same number of keys over all vertex labels. Lemma 4.5 leaves at least $2dR(e-1) - 3nK$ of those keys in good tuples. By the edge-order restriction in Lemma 4.4, each such tuple consumes a distinct W item from an earlier edge. At most $3nK$ earlier-edge items therefore remain available for keys of edge e. The bound $3nK$ applies to the earlier-edge inventory across all labels. Every tuple counted by $L_{v,r}$ with $r = (e, j)$ uses either an edge-e permit or an earlier-edge item, by Lemma 4.4. There are dR edge-e permits, so

$$\sum_{v \in \{v_1, v_2\}} \sum_{j \in [R]} L_{v,(e,j)} \le dR + 3nK. \tag{28}$$

If neither endpoint were in C, every nonexcluded position at either endpoint would satisfy

$$L_{v,(e,j)} \ge d - \frac{2d}{5} - 9K \ge \frac{11d}{20}, \tag{29}$$

where the first step follows from Eq. (27) and $x_v^+ \le 2d/5$, and the second step follows from $d \ge 1000K \ge 180K$. Each endpoint has at least $R - P$ nonexcluded positions. We can show

$$2(R-P)\frac{11d}{20} - (dR + 3nK) = \frac{d}{10}(R - 11P) - 3nK = \frac{89d}{10}P + (10d - 3)nK + 10d > 0, \tag{30}$$

where the first step follows from expanding and collecting terms, the second step follows from $R = 100(P + nK + 1)$, and the third step follows from $d \geq 1$, $P \geq 0$, and $nK > 0$. By Eq. (29), the total demand from the two endpoints is at least $2(R - P)11d/20$, which exceeds the upper bound in Eq. (28) by Eq. (30). This contradiction shows that every edge has an endpoint in C.

Step 5: Bound the size of the cover. Define $V_+ := \{v : x_v^+ \geq d - 6K\}$. Since $d \geq 1000K$, we have $d - 6K > 2d/5$, so $V_+ \subseteq C$. We can show

$$|V_+| \leq \frac{dk}{d - 6K}, \tag{31}$$

where the inequality follows from the plus budget in Eq. (20) and $x_v^+ \geq d - 6K > 0$ for every $v \in V_+$. At each vertex of V_+ charge its plus count, and at each other vertex charge its minus count. Eq. (24) implies that every charge is at least $d - 6K$. Exactly one side is charged at each vertex, so the total charges are at least $n(d - 6K)$. Every plus count outside V_+ is uncharged. Subtracting from the total budget dn gives

$$\sum_{v \notin V_+} x_v^+ \leq dn - n(d - 6K) = 6Kn, \tag{32}$$

where the first step follows from Eq. (20), the lower bound $n(d-6K)$ on the total charges, the fact that every plus count outside V_+ is uncharged, and nonnegativity of the other uncharged counts, and the second step follows from expanding and cancelling dn. We can show

$$|C \setminus V_+| \leq 15Kn/d, \tag{33}$$

where the inequality follows from Eq. (32) and $x_v^+ > 2d/5$ for every $v \in C \setminus V_+$; it also holds when $C \setminus V_+$ is empty. Since $k = 2n/3$ and $d \geq 1000K \geq 30K$,

$$\frac{dk}{d - 6K} - k = \frac{4Kn}{d - 6K} \leq \frac{5Kn}{d}, \tag{34}$$

where the first step follows from subtracting k over a common denominator and substituting $k = 2n/3$, and the second step follows from $d - 6K \geq 4d/5$. We can show

$$|C| \leq \frac{dk}{d - 6K} + \frac{15Kn}{d} \leq k + \frac{20Kn}{d} \leq k + \frac{n}{50},$$

where the first step follows from the disjoint decomposition $C = V_+ \cup (C \setminus V_+)$ and Eqs. (31) and (33), the second step follows from Eq. (34), and the third step follows from $d \geq 1000K$. Since C is a vertex cover and $n/50 < n/48$ for $n > 0$, this contradicts Eq. (19). Thus no packing uses at most $B + c$ bins, proving $\mathrm{OPT} > B + c$. □

4.5 Uniform construction of the competing trees

The auxiliary tree grows rapidly with the number of phases in the construction. Choosing the additive allowance proportional to $\log^* n$ keeps both the tree size and the time needed to generate its retained trees polynomial in n.

Lemma 4.8 (Uniform construction of the competing trees)**.** *Set $P_0 := 2^5 \cdot 6^4$ and $D := 20000(1 + 5P_0)$. For the literal-slot graph in Section 4.2, let $h := \log^* n$. If $h \geq 32D$, define*

$$c := \lfloor h/(8D) \rfloor - 1, \quad K := 20(c + 1)(1 + 5P_0), \quad d := D(c + 1).$$

Then the following properties hold.

(a) We have $c \geq 0$ and $d = 1000K \leq h/8$, so the parameter requirements in Eq. (12) hold.

(b) If N is the auxiliary tree's final leaf count, L is its depth, and P is the sum of its depth quotas, then

$$N < n, \qquad L \leq N - 1, \qquad P \leq N - 1.$$

The auxiliary tree and each retained tree have fewer than n^2 vertices. The maximum D_* of one and the child counts in the retained trees satisfies $1 \leq D_* \leq n^2$.

(c) A single deterministic algorithm computes these parameters and constructs the auxiliary tree and both retained trees T^+, T^-, with their separate markings, in polynomial bit time.

Proof. We bound the sizes of the trees and describe a deterministic construction.

Proof of Part (a). Write

$$D := 20000(1 + 5P_0), \qquad h := \log^* n, \qquad c := \lfloor h/(8D) \rfloor - 1, \qquad d := D(c + 1).$$

By assumption, $h \geq 32D$. Then $c \geq 0$, $d = 1000K$, and

$$d \leq h/8, \qquad c \geq \frac{h}{8D} - 2 \geq \frac{h}{16D},$$

where the first step follows from $d = D\lfloor h/(8D) \rfloor$ and $\lfloor h/(8D) \rfloor \leq h/(8D)$, the second step follows from $\lfloor h/(8D) \rfloor \geq h/(8D) - 1$, and the third step follows from $h \geq 32D$. Thus Eq. (12) holds, with absolute constants.

Proof of Part (b). We first derive the exact recurrence for the number of leaves in the auxiliary tree. If a phase starts with u leaves, leaf i receives 2^i disjoint spines of length i. Each endpoint has $1 + \sum_{j=i+1}^{u} 2^{j-1} = 1 + 2^u - 2^i$ continuation leaves. For $i = u$, the endpoint itself is the only continuation leaf. Consequently, define the leaf-count function by

$$f(u) := \sum_{i \in [u]} 2^i (1 + 2^u - 2^i).$$

Evaluating the geometric sums of 2^i and 4^i gives

$$f(u) = \frac{2(4^u - 1)}{3}. \tag{35}$$

Define $N_0 := 1$, $N_{a+1} := f(N_a)$, and $F := 2d - 1$; the final leaf count is $N := N_F$. Fix an integer $u \geq 1$. We can show

$$f(u) = \frac{2(4^u - 1)}{3} \geq 2u, \tag{36}$$

where the first step follows from Eq. (35), and the second step follows from $4^u = (1 + 3)^u \geq 1 + 3u$ by the binomial theorem, since all remaining terms are nonnegative.

We can show

$$f(u) - u \geq f(u) - (2^u - 1) = \frac{(2^u - 1)(2^{u+1} - 1)}{3} \geq 2^u - 1, \tag{37}$$

where the first step follows from $2^u = (1 + 1)^u \geq u + 1$ by the binomial theorem, the second step follows from substituting Eq. (35) and factoring, and the third step follows from $2^{u+1} - 1 \geq 3$ and $2^u - 1 \geq 0$.

We can show

$$f(u) = \frac{2(4^u - 1)}{3} < 2^{2u} \le 2^{2 \cdot 2^{u-1}} = 2^{2^u}, \tag{38}$$

where the first step follows from Eq. (35), the second step follows from $2(4^u - 1)/3 < 4^u = 2^{2u}$, the third step follows from $2^{u-1} = (1+1)^{u-1} \ge 1 + (u-1) = u$ by the binomial theorem and monotonicity of exponentiation, and the fourth step follows from $2 \cdot 2^{u-1} = 2^u$.

Let $E_0 := 1$ and $E_{j+1} := 2^{E_j}$. For $n > 1$, the definition of h is equivalent to $E_{h-1} < n \le E_h$. We prove by induction that

$$N_a \le E_{2a-1} \qquad (a \ge 1). \tag{39}$$

The base case is $N_1 = 2 = E_1$. For the induction step, we can show

$$N_{a+1} < 2^{2^{N_a}} \le 2^{2^{E_{2a-1}}} = E_{2a+1},$$

where the first step follows from $N_{a+1} = f(N_a)$ and Eq. (38), the second step follows from the inductive hypothesis $N_a \le E_{2a-1}$ and monotonicity of exponentiation, and the third step follows from applying the recurrence for E_j twice.

We can show

$$N \le E_{2F-1} = E_{4d-3} \le E_{4d} \le E_{\lfloor h/2 \rfloor} < n, \tag{40}$$

where the first step follows from $N = N_F$ and Eq. (39), the second step follows from $F = 2d-1$, the third step follows from monotonicity of E_j and $4d-3 \le 4d$, the fourth step follows from $4d \le \lfloor h/2 \rfloor$, and the fifth step follows from $\lfloor h/2 \rfloor \le h - 1$ and $E_{h-1} < n$. This bounds the final leaf count of the auxiliary tree specified in Section 4.2.

We can show

$$L = \sum_{a=0}^{F-1} N_a \le N - 1, \tag{41}$$

where the first step follows from the N_a depths added in phase a, and the second step follows from summing $N_a \le N_{a+1} - N_a$ (see Eq. (36)) and telescoping with $N_0 = 1$ and $N_F = N$.

We can show

$$P = \sum_{a=0}^{F-1} (2^{N_a} - 1) \le N - 1,$$

where the first step follows from summing the depth quotas $1, 2, 4, \ldots, 2^{N_a - 1}$ in each phase, and the second step follows from summing $2^{N_a} - 1 \le N_{a+1} - N_a$ (see Eq. (37)) and telescoping with $N_0 = 1$ and $N_F = N$.

Every vertex of the auxiliary tree has a descendant at its final depth L. At each depth the descendant leaf sets of different vertices are disjoint and nonempty. Thus there are at most N vertices at each positive depth, and at most $1 + LN$ vertices altogether. We can show

$$1 + LN \le 1 + N(N-1) \le N^2 < n^2,$$

where the first step follows from Eq. (41), the second step follows from $N \ge 1$, and the third step follows from Eq. (40).

Choose D_* to be the maximum of one and the actual child counts in the retained trees, so $1 \le D_* \le n^2$.

Proof of Part (c). Fix a deterministic order for the phase-start leaves, and generate the prescribed spines and continuation chains by loops, storing parent indices, depths, and endpoint colors. Every loop creates explicitly specified vertices or visits existing ones; its work is polynomial

in the resulting tree size and index lengths. To produce each retained tree, follow the path from each final leaf to the root, count its red and blue vertices, and retain that path on the appropriate side if its color count is at least d. This direct implementation requires $O(NL) = O(n^2)$ path visits per side by Eqs. (40) and (41). Keep original depths and mark the retained vertices of that side's color. This constructs both trees and their separate markings by one algorithm whose exponent is independent of d.

Also, h can be computed exactly without constructing large tower entries. Start with $z := 1$ and a counter initialized to zero. While $z < n$, increment the counter and replace z by $\min\{2^z, n\}$. Compute $\ell := \lceil \log_2 n \rceil$ from the binary representation of $n - 1$. If $z \geq \ell$, set the new value directly to n; otherwise form 2^z. Each stored integer has $O(\log(n + 2))$ bits, and the final counter equals $h \leq n$. This is a polynomial-time integer algorithm for the parameter choice.

□

4.6 Item encoding and polynomial-time implementation

The remaining construction expands the trees into a packing instance and assigns rational item sizes. The tree bounds yield polynomial bounds on both the number of items and the bit length of their coordinates.

Lemma 4.9 (Uniform explicit construction at growing allowance)**.** *Set* $P_0 := 2^5 \cdot 6^4$ *and* $D := 20000(1 + 5P_0)$*. For the literal-slot graph in Section 4.2, let* $h := \log^* n$*. If* $h \geq 32D$*, define*

$$c := \lfloor h/(8D) \rfloor - 1, \quad K := 20(c + 1)(1 + 5P_0), \quad d := D(c + 1).$$

Then the following properties hold.

(a) The corresponding packing instance consists of $5B$ *sizes in* $(1/6, 1)$*, with*

$$n \leq B \leq 1203n^5.$$

(b) Every size has a common-denominator representation of bit length $O(n^2 \log(n + 2))$*, and the full list has bit length* $O(n^7 \log(n + 2))$*, with constants independent of* n *and* c*.*

(c) A single deterministic algorithm constructs the corresponding packing instance in polynomial bit time, outputting all sizes explicitly.

Proof. Lemma 4.8-(c) constructs the trees, their markings, and the parameters in polynomial bit time. We use its notation N, L, P, D_* and the bounds in Lemma 4.8-(b) on the tree sizes and depth quotas.

Proof of Part (a). Lemma 4.4-(a) gives the item count and size range. Since $K \leq d \leq h/8 \leq n$ and $P < n$, we have

$$R = 100(P + nK + 1) \leq 300n^2, \qquad t = d(|V(T^+)| + |V(T^-)|) + P \leq 3n^3, \tag{42}$$

where the first step follows from the definition of R, the second step follows from $P < n$ (see Lemma 4.8-(b)), $K \leq n$, and $n \geq 1$, the third step follows from the definition of t, and the fourth step follows from $d \leq n$, $|V(T^+)|, |V(T^-)| < n^2$, $P < n$, and $n \geq 1$. The construction uses the same tree-row count t at every label. Its exact inventory identity is

$$B = nt + 4dmR, \qquad n \leq B \leq 3n^4 + 1200n^5 \leq 1203n^5, \tag{43}$$

where the first step follows from the definition of B, the second step follows from $t \geq 2d \geq 1$ and $4dmR \geq 0$, the third step follows from Eq. (42), $d \leq n$, and $m \leq n^2$, and the fourth step follows from $n \geq 1$.

Proof of Part (b). To bound the encoding length, we specify common denominators explicitly. Define

$$A := 100(D_* + 1), \qquad D_{\text{geom}} := 160(R+1)^m A^L, \qquad D_{\text{size}} := 10^9 P_0 \cdot 3^n D_{\text{geom}}.$$

The construction uses $g_{\text{geom}} = 1/[16(R+1)^m]$, $\lambda_\ell = g_{\text{geom}} A^{-\ell}$, and $\Delta = \lambda_L/10$. Nested cell endpoints are integer sums of these widths and root endpoints; job coordinates add integer multiples of g_{geom} to $1/2$. All their denominators therefore divide D_{geom}. The item-size encoding is

$$a_i = \frac{1}{5} + \frac{p(i)}{1000 P_0} + \frac{q(i)}{10^6 P_0 \cdot 3^n} + \frac{w(i)}{10^9 P_0 \cdot 3^n},$$

where the first step follows from Eq. (13). Here $p(i), q(i)$ are integers and $w(i)$ has denominator dividing D_{geom}. Hence every size has denominator dividing D_{size}, and

$$\log_2 D_{\text{size}} = O(1 + n + m \log(R+1) + L \log A) = O(n^2 \log(n+2)), \tag{44}$$

where the first step follows from the definitions of D_{size} and D_{geom} and the fixed value of P_0, and the second step follows from Eq. (42), $m \leq n^2$, and the bounds $A \leq 200n^2$ and $L < n$ supplied by the definition of A and Lemma 4.8-(b). The implicit constants may depend on the fixed value P_0, but not on c. Represent all sizes over D_{size}; since $0 < a_i < 1$, the bit lengths of their numerators satisfy the bound in Eq. (44). Reducing fractions is optional. By Eq. (43), there are $5B = O(n^5)$ items. Combining this count with Eq. (44) gives $O(n^7 \log(n+2))$ bits for the entire explicit rational list.

Proof of Part (c). All coordinates can be generated using the common denominator and integer arithmetic. Lemma 4.8-(c) bounds the cost of constructing the trees, while Eqs. (42) and (43) give polynomial bounds on the numbers of label and edge-position iterations, items, and sorting operations. The coordinate arithmetic uses integers of the bit length bounded in Eq. (44). Integer powers can be computed by repeated squaring. The powers 3^n, $(R+1)^m$, and A^L are each at most D_{size}, so their intermediate bit lengths also satisfy Eq. (44). Thus the construction runs in polynomial bit time, with an exponent independent of c. Its running time is also polynomial in the formula's encoding length, since this length is at least a constant multiple of the number of explicitly listed clauses.

□

References

[Che20] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Theory of Computing*, 16(4):1–50, 2020.

[FL81] W. Fernandez de la Vega and George S. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[GG61] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[HR17] Rebecca Hoberg and Thomas Rothvoss. A logarithmic additive integrality gap for bin packing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2616–2625, 2017.

[KK82] Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 312–320, 1982.

[Ope26] OpenAI. Additive hardness and unbounded configuration gaps in bin packing. OpenAI Math Release preprint, September 2026. September 24, 2026.

[Rot13] Thomas Rothvoss. Approximating bin packing within $O(\log \mathrm{OPT} \log \log \mathrm{OPT})$ bins. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 20–29, 2013.

[SZ26] Zhao Song and Xinzhi Zhang. An $O(1)$ bound for the kls constant. *arXiv preprint arXiv:2610.01447*, 2026.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

AI Disclosure and Acknowledgements

The AI tool used in this draft is Codex 6 Astra. The iterated-logarithm idea comes from Chen's paper on the Chinese remainder theorem [Che20] and from Song and Zhang's KLS paper [SZ26]. The author learnt about this problem from Pravesh Kothari while the author was a PhD student. The author was trying very hard to improve the $O(\log \mathrm{OPT})$ bound while the author was a Ph.D. student. However, a lot of attempts are failed. At the end of July 2026, Victor Reis told the author that this problem remained open. The author has been working on this problem again since then.