A Single-Exponential Protocol for Envy-Free Chore Division

Preliminary version; latest version at https://cgi.cse.unsw.edu.au/~haziz/EFchores.pdf

HARIS AZIZ, UNSW Sydney, Australia

We give deterministic protocols for complete envy-free chore division among n agents with arbitrary nonnegative nonatomic costs, using single-exponentially many Robertson–Webb queries: $O(n^4\binom{2n}{n})$ for a self-contained protocol, and $O(n^3 2^n)$ by transferring an existing cake-cutting subroutine. The key step reduces the problem to polynomially many calls to *certified trimming*, plus polynomially many queries. A certified allocation gives agents envy-free suffixes of distinct pieces, with each trimmed suffix linked to a whole piece by a chain of indifferences. For positive costs, each such suffix is the longest attainable on its piece in any envy-free suffix allocation, which yields a dynamic program over subsets of agents and pieces. A buffered repair step then allocates almost all of the residue without disturbing existing bundles, at a cost logarithmic in the inverse tolerance. We also use an inverse-function construction to show that certified cake and chore trimming are equivalent on positive inputs. This gives the faster bound, since the suffix allocations of SubCore [Ye and Bai, 2026] convert into certified cake trims.

1 Introduction

Suppose agents must divide a heterogeneous, divisible chore, such as a cleaning roster or a stretch of road to maintain. Agents may judge the cost of the same piece of work differently, and each should find her own share no more costly than anyone else's. This requirement is *envy-freeness*. How many questions about their costs suffice to divide all the work in this way?

A protocol learns about costs only by asking agents questions, or *queries*, in the standard model of Robertson and Webb [1998]. An evaluation query asks an agent for her cost of an interval. A cut query asks her to mark where, starting from a given point, the work reaches a given cost to her. For two agents a few queries suffice: one agent cuts the chore into two pieces she finds equally costly, and the other takes the piece she finds cheaper. For any number of agents, Aziz and Mackenzie [2016] gave the first envy-free cake-cutting protocol whose number of queries depends only on n, with a bound that is a tower of six n's, and Dehghani et al. [2018] adapted it to show that such a bound exists for chores as well. For cake, Ye and Bai [2026] give a deterministic $n^{O(1)}2^n$ query bound.

Our main result gives a single-exponential bound for chores, with arbitrary finite nonnegative nonatomic costs. Pieces may be disconnected, and an agent may assign zero cost to entire intervals.

Theorem 1. *For $n \geq 2$ agents with finite nonnegative nonatomic additive costs, a complete envy-free chore allocation can be computed deterministically in $O\left(n^4\binom{2n}{n}\right) = O(n^{7/2}4^n)$ Robertson–Webb queries and $2^{O(n)}$ additional exact arithmetic operations. The guarantee holds for every choice of valid cut answers, including on intervals of zero cost.*

Our protocol reduces complete envy-free chore division to polynomially many calls to certified trimming, with polynomially many additional queries. For positive costs, every certified allocation retains on each piece the longest suffix compatible with envy-freeness. This characterization gives the self-contained subset algorithm used in Theorem 1. The corresponding property for cake lets us obtain certified cake trims from the envy-free suffix allocations returned by the SubCore procedure of Ye and Bai [2026]. Our reduction from certified chore trimming to certified cake trimming then gives a deterministic $O(n^3 2^n)$-query protocol for complete envy-free chore division (Corollary 1).

Envy-freeness is easy to maintain for partial allocations. Giving everyone nothing is envy-free, and adding to each agent's bundle her part of an envy-free division of some of the remaining work keeps the allocation envy-free, because costs are additive. The difficulty is the work not yet allocated, the *residue*. Careful division steps tend to leave some of it, and the last of it must also be

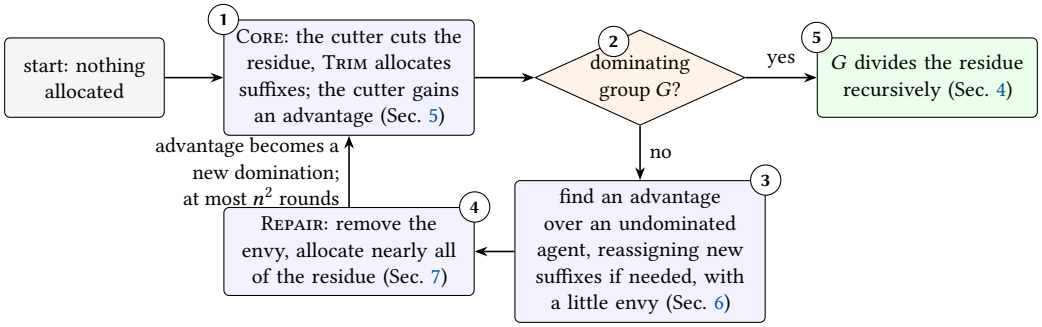

Fig. 1. The main loop of Divide (Algorithm 1), with the step numbers used throughout. The allocation stays envy-free after every step except the reassignment (3), and Repair (4) restores it. Table 2 states what each step guarantees.

allocated. The classical way out is an irrevocable advantage [Aziz and Mackenzie, 2016, Brams and Taylor, 1995, Peterson and Su, 2009]. Write $c_a(B)$ for agent a's cost of a piece B. Agent a's *advantage* over agent b is $c_a(X_b) - c_a(X_a)$, the amount by which she finds b's bundle X_b costlier than her own bundle X_a; envy-freeness says exactly that every advantage is nonnegative. We say that a *dominates* b when this advantage is at least her cost of the whole residue. Then a could take all of the residue herself and still not envy b, however the rest is divided. Once a nonempty proper group of agents dominates everyone outside it, the residue can be divided among the group's members alone, recursively. They have enough advantage to absorb the extra work, and the outsiders cannot object to their receiving more chores. A protocol therefore aims to create dominations, one at a time, while keeping its allocation envy-free.

The protocol in brief. Our protocol keeps one envy-free partial allocation and repeats a round of four steps (Figure 1) until a dominating group appears.

(1) In Core, a fixed agent, the cutter, divides the residue into n pieces she finds equally costly, and a trimming routine gives each agent the end part of a different piece without envy. One piece stays whole, so the cutter gains an advantage over its holder.

(2) If some group now dominates everyone else, its members divide the residue recursively (5).

(3) Otherwise the cutter may already dominate the holder of the whole piece, and her advantage is then wasted. We look for an advantage of some agent over a bundle she does not yet dominate. If necessary, we pass the suffixes that Core has just allocated around a cycle of agents: every agent keeps her earlier bundle and receives one of the new suffixes on top of it. This may create a little envy.

(4) Repair cuts the residue into a large share and a few small *buffers* for each bundle, all costing every agent nearly the same, and places the buffers in front of the bundle and the share behind it. It then calls Trim again, which gives each agent a suffix of a different one of these pieces without envy. The parts cut off lie within the buffers and return to the residue: since each buffer costs more than the envy, the cuts never reach a bundle (Section 7). The advantage found in step (3), of an agent over a bundle she did not yet dominate, survives with only a small loss, while the residue shrinks below it. So she now dominates whoever holds that bundle.

Dominations persist even when bundles change holders, so every round adds at least one dominated pair. There are at most n^2 such pairs, so a dominating group appears within n^2 rounds.

Two ingredients. The first, the trimming routine Trim, is used in steps ① and ④ and is the only source of the exponential factor. Given n pieces, it cuts some of them from the front and gives each agent the remaining *suffix* of a different piece, without envy. To avoid trimming more than necessary, every trim is justified by an indifference: the holder of a trimmed suffix finds it exactly as costly as another agent's suffix, in a chain ending at a whole piece (Figure 2). The certified-order routine of Psomas [2026] enumerates orders of agents and pieces. We develop a longest-proposal recurrence that avoids enumerating these orders. With positive costs, the suffixes of such an allocation are unique even though their holders are not: on every piece, they are the longest suffixes that any envy-free allocation of suffixes of the same pieces can give. We therefore compute them by a dynamic program with one state per pair of equal-size sets of agents and pieces, $\binom{2n}{n} < 4^n$ states in all. In each state, every agent proposes on each piece the longest suffix she could take without envy, the longest proposal is kept, and a final matching decides who holds what. The same maximality property holds for cake; Section 8 makes the connection precise through two-way reductions between the certified trimming problems.

The second, Repair, gives each bundle a large share of the residue that costs every agent nearly the same, with small *buffers* placed in front. A second call to Trim can then remove a small envy by consuming buffers while leaving every bundle intact. Only buffers and a small rounding error remain unallocated. To build these nearly common shares, we adapt the consensus-splitting algorithm of Alon and Graur [2021]. It starts from an exact fractional split and repeatedly halves the few cells that are still shared fractionally. We use it with unequal shares and Robertson–Webb queries. Its query cost is logarithmic in the inverse tolerance, which matters because the tolerance can be as small as $2^{-\Theta(n \log n)}$. With m active agents, the protocol makes $O(m^2)$ calls to Trim and uses polynomially many queries for all other steps.

Connections beyond chore division. For positive costs and values, we give polynomial oracle reductions in both directions between certified chore and cake trimming, so a polynomial-query algorithm for either would give one for the other and, through our protocol, for complete envy-free chore division. The same inverse-function construction turns certified trimming into a problem about assignment markets (Section 8.3): pieces become unit-demand buyers, agents become indivisible goods, and utilities are continuous and decreasing in price but not necessarily quasilinear, as in Alaei et al. [2016]. Competitive equilibria correspond exactly to envy-free suffix allocations, and certified suffixes to componentwise minimum prices. Hence a polynomial-query algorithm for minimum-price equilibria in these markets would remove the exponential factor from our bounds.

Organization. Section 2 compares our ingredients with earlier work. Section 3 sets up the model and domination. Section 4 presents the whole protocol and proves Theorem 1, taking as given one lemma for each of the three subroutines. A reader who wants the overall argument can stop there. Sections 5–7 prove these lemmas in the order in which the steps run: trimming and the cutter, then reassignment, then repair. One three-agent running example is followed throughout. Section 8 connects certified trimming for cake and chores to market equilibrium. Appendix **??** collects the notation. The remaining appendices establish existence, handle zero costs, illustrate a rotation, count the cost-record queries, and give self-contained proofs of the reductions (Appendix E).

2 Related Work

Brams and Taylor [1995] gave an unbounded envy-free cake protocol for any number of agents, and Peterson and Su [2009] developed the corresponding idea of irrevocable advantage for chores. Aziz and Mackenzie [2016] gave the first bounded cake protocol for arbitrary n, and Dehghani et al. [2018] adapted it to chores, giving the first bounded chore protocol. Most of our ingredients refine theirs. The framework is that of Aziz and Mackenzie [2016]: a cutter divides the residue

Table 1. How the protocol builds on earlier ingredients.

Ingredient	Earlier work	This paper
Core, domination	cutter, trimming, domination as progress [Aziz and Mackenzie, 2016]; chore version with suffixes [Dehghani et al., 2018]	used as is
Trimming	recursive SubCore [Aziz and Mackenzie, 2016, Dehghani et al., 2018]; $(n!)^2$ certified orders [Psomas, 2026]	unique certified suffixes; $\binom{2n}{n}$ subset states (Sec. 5)
Repair	buffered; cost linear in inverse tolerance [Psomas, 2026]	nearly common shares by rounding [Alon and Graur, 2021]; cost logarithmic (Sec. 7)
Reassignment	exchanges create an advantage [Aziz and Mackenzie, 2016, Dehghani et al., 2018]; cake lemma [Psomas, 2026, Lemma 5]	chore version (Lemma 2)

into equal pieces, trimming allocates part of it and leaves some piece whole, domination measures progress, and, in the chore adaptation, a dominating group takes the residue recursively; their routine SubCore constructs a partial allocation of the cutter's pieces. In the intuitive account of Aziz and Mackenzie [2020], Core is their simplified Core Protocol, our reassignment plays the role of GoLeft, and Repair replaces repeated Core calls; its common shares make their Discrepancy Protocol unnecessary. Dehghani et al. [2018] transposed this framework to chores, using a trimming routine that gives each agent a suffix of a different piece. Ours fixes neither which pieces stay whole nor who holds them: it computes the unique certified suffixes from smaller instances in $\binom{2n}{n}$ states. Its longest-proposal formula is the capped-suffix counterpart of the maximum-offer characterization of highest equilibrium prices in Alaei et al. [2016, Theorem 2], with suffix length as price. For positive costs, inverting suffix functions and interchanging agents and labels connects certified chore trimming to cake trimming and minimum-price equilibrium (Section 8). Our proof uses a counting argument on the labels whose suffixes grow. Unlike the GoLeft Protocol, which reattaches extracted trimmings to particular allocated pieces, Repair blends the trimmings of the current Core call into shares and buffers that cost every agent nearly the same, so a single call to Trim suffices.

For cake, Ye and Bai [2026] obtain a deterministic $n^{O(1)}2^n$ query bound, Psomas [2026] gives an elementary $2^{O(n \log n)}$ protocol, and Mackenzie and Suzuki [2026] give a randomized protocol with polynomial expected query complexity. Ye and Bai reduce the number of calls to the recursive SubCore of Aziz and Mackenzie [2016]. Our self-contained protocol computes the trims by a subset dynamic program. We follow the organization of Psomas [2026] into trimming, repair, and domination. We adapt the certificate, buffered Repair and reassignment lemma of Psomas [2026], but trim over $\binom{2n}{n}$ subset states instead of $(n!)^2$ orders, and repair at a cost logarithmic rather than linear in the inverse tolerance. Table 1 summarizes which ingredients come from earlier work.

For chores the indifferences run from a trimmed piece's holder towards an earlier holder, and the dominating group receives the residue. To construct the shares used in repair, we adapt the fractional rounding argument of Alon and Graur [2021] to prescribed shares and ordinary cut queries. Our existence proof for trimming uses envy-cycle exchanges [Lipton et al., 2004].

In concurrent work, Farhadi and Hajiaghayi [2026] pool compensation across partial allocations to obtain a deterministic $2^{3n \log_2 n + O(n)}$ query bound for chores. Each Core call uses $2^{O(n)}$ queries, but their compensation protocol budgets $2^{\Theta(n \log n)}$ calls because each of $O(n)$ filtering or merger steps can shrink the pool of partial allocations by a factor of up to $n+1$. They also give polynomial expected query bounds for cake and chores. Both papers use subset dynamic programming to compute coordinatewise greatest trims under positive costs. They build anchored prefixes by testing

possible last agent–piece pairs, while our recurrence takes the longest proposed suffix on each piece. Buffered repair lets us complete chore division with polynomially many certified trims, giving a self-contained $O(n^4\binom{2n}{n})$ bound. Our reduction to certified cake trimming transfers the SubCore bound of Ye and Bai [2026], yielding $O(n^3 2^n)$ deterministic queries (Corollary 1).

3 Preliminaries

An instance is a tuple $(N, C, (c_i)_{i\in N})$, where $N = [n]$ is a finite set of agents, $C = [0, 1]$ is the chore, and each c_i is a finite nonnegative nonatomic additive cost measure. A nonnegative integrable cost density is a special case. A **piece** is a finite union of intervals. Costs need not be normalized. Multiplying an agent's costs by a positive constant preserves all her comparisons. Within a recursive call, n denotes the number of participating agents, relabeled as $[n]$. Its input chore may be disconnected.

An **allocation** $X = (X_i)_{i\in N}$ consists of pairwise disjoint pieces, one per agent. Its **residue** is $R = C \setminus \bigcup_i X_i$. The allocation is **complete** if the residue is empty. The empty piece is permitted in a partial allocation. We consistently assign endpoints using half-open intervals, except for the right endpoint of the chore. Nonatomicity makes these endpoint choices irrelevant to costs. An allocation X is **envy-free** if $c_i(X_i) \le c_i(X_j)$ for all agents $i, j \in N$. Thus each agent receives a least-cost bundle from her own perspective. Envy-freeness compares allocated bundles and places no restriction on how agents value the residue. For partial allocations X and Y on disjoint parts of the chore, write $X \oplus Y$ for the allocation giving $X_i \cup Y_i$ to each agent. Additivity implies that the union of two envy-free allocations for the same agents is envy-free.

Domination. We measure progress by the chore version [Dehghani et al., 2018] of domination [Aziz and Mackenzie, 2016, 2020, Sec. 3.2], a form of irrevocable advantage [Brams and Taylor, 1995, Peterson and Su, 2009]. In an envy-free partial allocation X with residue R, agent i **dominates** agent j if $c_i(X_j) - c_i(X_i) \ge c_i(R)$. A **dominating group** is a set $\emptyset \ne G \subsetneq N$ such that every member of G dominates every agent outside G.

If i dominates j, giving the whole residue to i still creates no envy towards j. Domination is also permanent under the operations we use. Adding an envy-free allocation Y of part of the residue preserves every existing domination: i's advantage over j changes by $c_i(Y_j) - c_i(Y_i) \ge 0$, while her cost of the residue can only fall. Section 4 shows that domination also survives when bundles change holders. Once a dominating group is found, the residue can be divided recursively among its members.

Queries and ordered pieces. An evaluation query returns $c_i([x, y])$. A cut query specifies an agent i, a starting point x, and a feasible cost v, and returns any $y \ge x$ with $c_i([x, y]) = v$ [Robertson and Webb, 1998]. All computation uses exact arithmetic on query answers. Ties in choices made by the protocol follow fixed orders.

A piece is an ordered list of disjoint component intervals. A prefix follows this order and a suffix is what remains after removing a prefix. The order may differ from the physical order on $[0, 1]$. We write $P +\!\!+ Q$ for concatenation. This lets Repair place buffers before a bundle, wherever those pieces lie physically. To cut a prefix with specified cost, locate its component from the costs already recorded. If the target is at an endpoint, we may select that endpoint. Otherwise one ordinary cut within the component suffices. A suffix mark can be found in the same way by subtracting the target from the total cost. The protocol keeps every active agent's cost of every component interval it uses; Appendix D counts the queries needed to keep these records.

4 The Protocol at a Glance

This section presents the whole protocol and proves Theorem 1. It treats the three subroutines Core, Reassign and Repair as black boxes. Each is described by a lemma stating exactly what the next step needs, and Sections 5–7 prove these lemmas. Table 2 summarizes one round, and Table **??** in Appendix **??** collects the notation.

4.1 The main loop

Divide (Algorithm 1) handles one or two agents directly. Otherwise it fixes a cutter r for the whole call and starts from the empty allocation, whose residue is the entire chore. Between rounds, its state is an envy-free partial allocation X with residue R. Each round runs the steps of Figure 1.

ALGORITHM 1: Divide(N, C)

Input: A nonempty set of agents N and an ordered chore C.
Output: A complete envy-free allocation.

if $|N| = 1$ **then return** C to its sole agent
if $|N| = 2$ **then**
  Let one agent cut C into two equal-cost pieces and the other choose a least-cost piece. **return** this allocation
Fix a cutter r, set $X_i \leftarrow \emptyset$ for all i, and set $R \leftarrow C$ ▷ *invariant: X envy-free*
while *true* **do**
  (1) $(Y, U) \leftarrow$ Core(r, N, R) ▷ *Lemma 1*
  (2) **if** $(X \oplus Y, U)$ *has a dominating group* G **then** ▷ *graph search, Section 4.2*
    (5) Divide U recursively among G and add the resulting shares to $X \oplus Y$
    return this allocation
  (3) $(a, b, \pi, \alpha) \leftarrow$ Reassign(r, X, Y, U), and set $D_i \leftarrow X_i \cup Y_{\pi(i)}$ for every i ▷ *Lemma 2*
  (4) $(W, \sigma, R') \leftarrow$ Repair(N, D, U, α), and set $(X, R) \leftarrow (W, R')$ ▷ *Lemmas 3 and 5*

Labels and holders. Bundles change hands during a round, which is the main source of bookkeeping. We therefore name each old bundle by a *label*, the agent who held it at the start of the round, and write $\sigma(j)$ for the agent who holds label j at the end:

$$X_j \xrightarrow{\text{(3) add } Y_{\pi(j)}} D_j = X_j \cup Y_{\pi(j)} \xrightarrow{\text{(4) add } Z_j} W_{\sigma(j)} = D_j \cup Z_j. \tag{1}$$

Here π reassigns the suffixes Y_i returned by Core, so that agent i receives $Y_{\pi(i)}$, and Repair adds Z_j to label j. Only new parts are added and nothing is removed, so the old bundle X_j stays whole on label j throughout. At the next round, labels are reset to the current holders.

4.2 What each step guarantees

Step (1): the cutter. Core is the chore form [Dehghani et al., 2018] of the Core protocol [Aziz and Mackenzie, 2016, 2020, Sec. 3.1]. In Core(r, N, C), agent r cuts C into n pieces of equal cost to her, and Trim gives each agent a suffix of a different piece without envy (Section 5). Trim makes $O(n^2\binom{2n}{n})$ queries (Lemma 7) and is the only part of the protocol whose cost is exponential. Some piece stays whole, and this gives the cutter an advantage over its holder whenever she assigns positive cost to what was trimmed off.

Lemma 1 (Cutter's advantage; proved in Section 5.4). *For every agent r and piece C, Core(r, N, C) returns an envy-free allocation Y of parts of C, with residue $U \subseteq C$, using one call to Trim*

Table 2. One round of DIVIDE. The column "afterwards" lists everything that the next step relies on.

Step	What it does	What holds afterwards	By
start	—	X envy-free, with residue R	—
① CORE	CORE(r, N, R): the cutter r cuts R into n pieces of equal cost to her; TRIM gives each agent a suffix Y_i of a different piece	$X \oplus Y$ envy-free, with residue $U \subseteq R$; some $h \neq r$ has $c_r(Y_h) - c_r(Y_r) \geq \frac{c_r(U)}{n-1}$	L1
② test	build the non-domination graph Γ of $(X \oplus Y, U)$	a dominating group G exists (go to ⑤), *or* Γ is strongly connected and every $c_i(U) > 0$	§4.2
③ REASSIGN	pick an arc $a \to b$ of Γ; if needed, rotate only the *new* suffixes: $D_i = X_i \cup Y_{\pi(i)}$	a little envy: $c_i(D_i) \leq c_i(D_j) + \alpha\, c_i(U)$; an advantage: $c_a(D_b) - c_a(D_a) \geq (K-1)\alpha\, c_a(U)$	L2
④ REPAIR	REPAIR(N, D, U, α): place $n-1$ buffers before each D_j and a large share T_j after it; TRIM again	W envy-free; each D_j intact, now held by $\sigma(j)$; additions differ by $\leq 2n\alpha\, c_i(U)$; residue $c_i(R') \leq 2n^2\alpha\, c_i(U)$	L3
end	$(X, R) \leftarrow (W, R')$	a dominates $\sigma(b)$, though she did not dominate b; old dominations follow their labels; so the number of dominated pairs rises	L4, L5
⑤ recurse	the members of G divide U recursively	complete and envy-free	Thm 1

and $O(n^2)$ other queries, so $O(n^2\binom{2n}{n})$ queries in all. If $c_r(U) > 0$, then $c_r(Y_h) - c_r(Y_r) \geq c_r(U)/(n-1)$ for some agent $h \neq r$.

Step ②: testing for a dominating group. The **non-domination graph** Γ of an envy-free allocation has an arc $i \to j$ whenever $i \neq j$ and i does not dominate j (it is the complement of the domination graph of Aziz and Mackenzie, 2020, Fig. 4). A proper sink strongly connected component of Γ is a dominating group, because no arc leaves it. Conversely, a dominating group has no outgoing arc, so Γ cannot be strongly connected. We can therefore find such a group, or establish that none exists, using a standard graph search. If none exists, every agent assigns positive cost to the residue, since an agent with zero cost for it dominates everyone and forms a dominating group on her own.

Step ③: a new advantage. The cutter's advantage may be useless for progress, because she may already dominate the holder h of the whole piece. Step ③ finds an advantage of some agent a over the bundle on a label b that she does not yet dominate. It may pass the new suffixes around a cycle to do so, at the price of a little envy, measured by a tolerance α. With $K = 4n^2$, the advantage is at least $K-1$ times this envy, which is what step ④ needs.

LEMMA 2 (A USEFUL REASSIGNMENT; PROVED IN SECTION 6). *Let (X, R) be envy-free and let $(Y, U) = $ CORE(r, N, R). If $(X \oplus Y, U)$ has no dominating group, then $c_i(U) > 0$ for every i, and REASSIGN, using only recorded costs, finds agents a, b with a not dominating b in (X, R), a permutation π, and $K^{-(n+1)} \leq \alpha \leq K^{-2}$ such that the bundles $D_i = X_i \cup Y_{\pi(i)}$ satisfy*

$$\begin{aligned} c_i(D_i) &\leq c_i(D_j) + \alpha c_i(U) \quad (i, j \in N), \\ c_a(D_b) - c_a(D_a) &\geq (K-1)\alpha c_a(U). \end{aligned} \tag{2}$$

Step ④: repair. REPAIR receives the slightly envious bundles D_j and the residue U. It removes the envy without cutting into any D_j, and allocates nearly all of U, adding nearly the same cost to every bundle in every agent's eyes. Its output may move a bundle to a new holder.

Lemma 3 (Repair; proved in Section 7). *Let D be a partial allocation with residue U, and let $0 < \alpha \le 1/(8n^2)$. Suppose $c_i(U) > 0$ and $c_i(D_i) \le c_i(D_j) + \alpha c_i(U)$ for all i, j. Then Repair(N, D, U, α) returns an envy-free allocation W, a bijection σ from labels to final holders, and disjoint additions $Z_j \subseteq U$ such that $W_{\sigma(j)} = D_j \cup Z_j$. Its residue R' satisfies, for every agent i and labels j, l,*

$$c_i(R') \le 2n^2\alpha\, c_i(U), \qquad \left|c_i(Z_j) - c_i(Z_l)\right| \le 2n\alpha\, c_i(U).$$

The procedure uses one call to Trim and $O(n^5 \log(n/\alpha))$ other queries.

4.3 Domination follows its label

The same agent need not keep a bundle throughout the protocol. We therefore follow the *label* of each old bundle when counting progress. The next lemma shows that a domination of a bundle passes to whoever holds it next, as long as the bundle only grows by work taken from the residue.

Lemma 4 (Domination follows its label). *Let (X, R) be envy-free, and let W be an envy-free allocation with residue $R' \subseteq R$. Suppose that, for a bijection σ, each bundle $W_{\sigma(j)}$ consists of X_j and pieces of $R \setminus R'$. If agent i dominates j in (X, R), then i dominates $\sigma(j)$ in (W, R').*

Proof. Agent i does not envy $W_{\sigma(i)}$, and $W_{\sigma(i)}$ consists of X_i and pieces of $R \setminus R'$. Hence

$$c_i(W_{\sigma(j)}) - c_i(W_i) \ge c_i(X_j) - c_i(W_{\sigma(i)}) \ge c_i(X_j) - c_i(X_i) - c_i(R \setminus R').$$

If i dominates j in (X, R), the right-hand side is at least $c_i(R) - c_i(R \setminus R') = c_i(R')$. □

Distinct labels have distinct holders, so no agent's number of dominated agents decreases when bundles change hands.

4.4 Progress, and the proof of Theorem 1

Each round turns the advantage from step ③ into a domination. The choice $K = 4n^2$ leaves enough of the advantage to cover the final residue. From agent a's perspective, in units of $\alpha c_a(U)$, the bounds of Lemmas 2 and 3 are

Quantity	Bound
Advantage before repair	at least $K - 1$
Possible loss from unequal additions	at most $2n$
Remaining residue	at most $2n^2$

Since $K - 1 - 2n \ge 2n^2$ for $n \ge 2$, the advantage becomes a domination. The next lemma also shows that every earlier domination survives, so each round that does not return makes progress.

Lemma 5 (Progress in one round). *Every nonreturning iteration of Divide preserves envy-freeness and strictly increases*

$$\Phi(X, R) = \left|\{(i, j) \in N^2 : c_i(X_j) - c_i(X_i) \ge c_i(R)\}\right|.$$

Consequently, before recursing, Divide makes $O(n^2)$ calls each to Core and Repair.

Proof. *Repair applies and restores envy-freeness.* Lemma 2 gives exactly the input envy bound of Lemma 3, with the same α. It also gives $c_i(U) > 0$ and $\alpha \le K^{-2} \le 1/(8n^2)$. Thus repair returns an envy-free allocation W with residue R' and keeps each old bundle on its label, as in (1).

The surviving advantage creates a new domination. Let $\sigma(j)$ be the final holder of label j. Agent a may change labels, but envy-freeness gives $c_a(W_a) \le c_a(W_{\sigma(a)})$. Hence Lemmas 2 and 3 imply

$$\begin{aligned} c_a(W_{\sigma(b)}) - c_a(W_a) &\ge c_a(D_b) - c_a(D_a) + c_a(Z_b) - c_a(Z_a) \\ &\ge (K - 1 - 2n)\alpha c_a(U) \\ &\ge 2n^2\alpha c_a(U) \ge c_a(R'). \end{aligned}$$

	P_1	P_2	P_3
agent 1	1	1	1
agent 2	1	2	4
agent 3	1	2	5

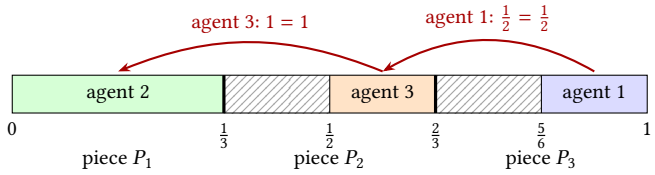

Fig. 2. The running example. Left: each agent's cost of each third, spread evenly within it. Right: the allocation Y computed by Core in the first round, with agent 1 as the cutter. Agent 2 holds P_1 whole, and agents 3 and 1 hold the last halves of P_2 and P_3. Each arrow is the tie that justifies a trim (Section 5.1), pointing from a holder to an earlier holder. The hatched halves form the residue U.

Agent a therefore dominates the final holder of label b, although she did not dominate b in (X, R).

The domination count increases. Each final bundle $W_{\sigma(j)} = X_j \cup Y_{\pi(j)} \cup Z_j$ consists of X_j and pieces of $R \setminus R'$. By Lemma 4, every old domination of label j passes to its new holder $\sigma(j)$. For each agent these holders are distinct, so no old counted pair is lost. The new domination of label b by a is an additional pair. Reindexing bundles by their final holders therefore gives $\Phi(W, R') > \Phi(X, R)$. The definition includes self-pairs and always gives $\Phi \le n^2$. Each nonreturning iteration calls each subroutine once, and the returning iteration makes one final Core call. □

Proof of Theorem 1. Induct on the number of agents. The cases of one or two agents are immediate. At the recursion point, write $V = X \oplus Y$, with residue U, and let F be the recursive allocation of U among G. Comparisons within G remain envy-free by additivity. Because G is dominating, for $i \in G$, $j \notin G$, and for $i \notin G$, $j \in G$, respectively,

$$c_i(V_i \cup F_i) \le c_i(V_i) + c_i(U) \le c_i(V_j), \qquad c_i(V_i) \le c_i(V_j) \le c_i(V_j \cup F_j).$$

Comparisons within the complement of G hold in V, and the recursive call allocates all of U. For m active agents, $\log(1/\alpha) = O(m \log m)$. Lemmas 1 and 3, with the bound of Lemma 7 for the call to Trim in each, together with the upkeep of component records (Appendix D), bound the cost of each cutter or repair call by $O(m^2\binom{2m}{m} + m^6 \log m)$. Multiplying by the $O(m^2)$ calls in Lemma 5 gives $O(m^4\binom{2m}{m})$ queries before recursing, since the exponential factor absorbs the additive polynomial overhead. The active population strictly decreases and $\binom{2m}{m}/\binom{2m-2}{m-1} = 4 - 2/m \ge 3$ for $m \ge 2$. Summing over the population sizes visited proves the claimed bound. The number of records and subset states is $2^{O(n)}$. Each proposal uses polynomial work in the records, as do linear elimination and matching. This proves the exact-arithmetic bound. □

4.5 The running example, one round

Three agents divide $C = [0, 1]$, with costs spread evenly within each third and totals as in Figure 2. We follow the first round, step by step.

1. Agent 1 is the cutter, and her equal-cost pieces are the thirds P_1, P_2, P_3. Agents 2 and 3 both find P_1 cheapest, so no allocation of whole thirds is envy-free. Core keeps P_1 whole and the last halves of P_2 and P_3 (Figure 2; Section 5 shows how Trim finds this). The residue U consists of the first halves of P_2 and P_3, and costs the three agents 1, 3 and 7/2. Lemma 1 is tight: agent 1 finds agent 2's whole piece costlier than her own by $1/2 = c_1(U)/2$.
2. Nobody dominates anybody. For instance, agent 3's largest advantage, 3/2 over agent 1, is below $c_3(U) = 7/2$. So Γ has all six arcs.
3. Here $K = 36$. The arcs $1 \to 2$, $2 \to 1$ and $3 \to 1$ all carry large advantages, so Reassign moves nothing (case 1 in Section 6). Taking the first arc in a fixed order, it selects $a = 1$, $b = 2$, the identity permutation and $\alpha = K^{-2} = 1/1296$; the other two arcs would give the

same permutation and tolerance. Thus $D = Y$: label 1 carries agent 1's half of P_3, label 2 carries P_1, and label 3 carries agent 3's half of P_2.

④ Repair adds to each label a large share and two buffers, costing every agent nearly a third of her cost of U (Figure 5). By Lemma 3, the additions differ by at most a $2n\alpha = 1/216$ fraction, and the new residue costs at most a $2n^2\alpha = 1/72$ fraction, of each agent's cost of U. Since $c_1(U) = 1$, agent 1's advantage of $1/2$ over label 2 becomes a domination:

$$c_1(W_{\sigma(2)}) - c_1(W_1) \;\geq\; \tfrac{1}{2} - \tfrac{1}{216} \;>\; \tfrac{1}{72} \;\geq\; c_1(R').$$

Agents 2 and 3 find the old bundle on label 1 costlier than those on labels 2 and 3 by at least a third of their cost of U, far more than these bounds. So after repair they hold labels 2 and 3, and both dominate agent 1, who holds label 1.

⑤ Hence $\{2, 3\}$ is a dominating group. Adding the envy-free allocation of the next Core call preserves these dominations, so the test in the second round finds a dominating group, and Divide recurses after that call.

5 Trimming and the Cutter

This section proves Lemma 1 (step ①). Its main part is the trimming routine Trim, which Core uses in step ① and Repair uses again in step ④. Trimming makes a chore cheaper for its holder, but also more tempting to everyone else. We therefore seek an envy-free allocation in which each trim is justified by an indifference: its holder finds her suffix exactly as costly as an earlier holder's suffix.

5.1 Certified suffix allocations

A *suffix allocation* gives each agent a suffix of a different labeled piece. A *label* names the input piece, and its *holder* is the agent who receives the suffix. We index suffixes by label and bundles by agent: S_j is the suffix on label j, and $\sigma(j)$ is its holder, whose bundle is S_j. A suffix is *trimmed* if some of its piece was cut off, and *whole* otherwise. The allocation is *certified* if it is envy-free and the holders can be ordered so that each receives a whole piece or finds her own suffix as costly as one assigned earlier. Following these indifferences leads to a whole piece.

Running example. Inside Core the labels are the cutter's pieces, so in Figure 2 label j is the third P_j. (Inside Repair, and in Section 4.5, labels are old bundles instead.) The certified allocation keeps P_1 whole and the last halves of P_2 and P_3. Agent 2 holds label 1 whole. Agent 3 holds label 2 and finds its half exactly as costly as P_1. Agent 1 holds label 3 and finds its half exactly as costly as the half on label 2. So the order 2, 3, 1 certifies the allocation. Exchanging the holders of labels 1 and 2 gives another certified allocation with the same suffixes.

A counting test. Whether an allocation is certified can be read off from the suffixes alone. For an envy-free suffix allocation S, let $\mathcal{M}_i(S)$ be the set of labels j minimizing $c_i(S_j)$.

Lemma 6 (Certification test). *An envy-free suffix allocation S is certified if and only if every nonempty set H of trimmed labels satisfies*

$$\left|\{i : \mathcal{M}_i(S) \subseteq H\}\right| < |H|. \tag{3}$$

In words, no set of trimmed labels contains all the cheapest labels of as many agents as it has labels.

Proof. Suppose S is certified, and fix H. By envy-freeness, each agent counted on the left of (3) holds a label in H. In a certifying order, the earliest holder of an H-label holds a trimmed suffix, so she ties a label assigned earlier, which lies outside H. She holds an H-label but is not counted,

giving the strict inequality. Conversely, draw an arc from each holder to those she ties. If some holders cannot reach a whole piece, their labels form a nonempty set H of trimmed labels. All their cheapest labels lie in H, contrary to (3). Ordering holders by distance to a whole piece now gives a certificate. □

Taking H to be the set of all labels shows that a certified allocation keeps some piece whole. Condition (3) depends only on the suffixes. In the running example, $\mathcal{M}_1 = \{2, 3\}$ and $\mathcal{M}_2 = \mathcal{M}_3 = \{1, 2\}$. Of the sets of trimmed labels, only $H = \{2, 3\}$ contains all the cheapest labels of some agent, and it does so for agent 1 alone.

Lemma 7 (Certified trimming). *Given each of n agents' component costs on n disjoint ordered pieces, Trim computes a certified suffix allocation using $O\left(n^2\binom{2n}{n}\right)$ Robertson–Webb queries, including when costs vanish or pieces are empty.*

First assume positive costs on every interval of positive length and nonempty input pieces; Appendix B removes this assumption. Under this assumption a certified suffix allocation always exists. Appendix A proves this by a limit argument used only to establish existence. We next characterize the suffixes (Section 5.2) and compute them by a finite dynamic program (Section 5.3).

5.2 Uniqueness of certified suffixes

Certified allocations need not be unique: in the running example, two of them differ in their holders. The structural fact underlying the recurrence is that their suffixes cannot differ. The following proposition concerns allocations alone, independently of any algorithm.

Proposition 1 (Greatest suffixes). *Assume positive costs, and let S be a certified allocation. On every label, S has the longest suffix of any envy-free suffix allocation of the same pieces to the same agents, whoever holds it. Consequently, all certified allocations have the same suffixes, though possibly different holders.*

The proposition follows from a slightly stronger statement, which tolerates envy towards one label. The dynamic program of Section 5.3 uses this stronger form to show that no proposal is too long.

Lemma 8 (Envy towards one label). *Assume positive costs, and let S be a certified allocation. If S' is a suffix allocation of the same pieces to the same agents in which agents may envy only the holder of label j, then $S'_j \subseteq S_j$.*

Proof. Suppose the labels lengthened in S' form a set H containing j. Let agent i hold label $k \in H$ in S', and let $l \notin H$. Since any envy in S' points to $j \in H$, and costs strictly increase with suffix length, $c_i(S_k) < c_i(S'_k) \le c_i(S'_l) \le c_i(S_l)$. Thus $\mathcal{M}_i(S) \subseteq H$ for every one of the $|H|$ agents holding H in S'. Every label in H is trimmed in S, since it is lengthened in S'. This contradicts Lemma 6. □

Proof of Proposition 1. Apply Lemma 8 to an envy-free S', for every label j. Two certified allocations are both envy-free, so each has every suffix of the other, and their suffixes coincide. □

Two agents. Restrict the running example to agents 2 and 3 and pieces P_1, P_3. Both prefer P_1, so one takes it whole and the other takes a suffix of P_3 that she finds exactly as costly: the last quarter for agent 2, or the last fifth for agent 3. Only the quarter works: if agent 3 held the fifth, agent 2 would envy it, since it costs her $4/5 < 1$. The quarter is also the longer suffix, as Proposition 1 predicts. On P_1, P_2 instead, both candidates are the last half, so either agent can hold it, as in the exchange of holders noted above.

ALGORITHM 2: TRIM by longest proposals

Input: Agents N and ordered pieces $(P_j)_{j\in J}$, with $|N| = |J| = n$.
Output: A certified suffix allocation. For zero costs, run within the simulation of Appendix B.

foreach $I \subseteq N, L \subseteq J$ *with* $|I| = |L| \geq 1$, *in increasing order of size* **do**
  foreach $j \in L$ **do**
    For each $i \in I$, compute θ_i from the stored entry $S^{I\setminus\{i\},L\setminus\{j\}}$ and mark the longest suffix of P_j costing i at most θ_i
    Store the longest mark as $S_j^{I,L}$ and record its cost to all agents in N
return a perfect matching joining each agent i to a label in $\mathcal{M}_i(S^{N,J})$

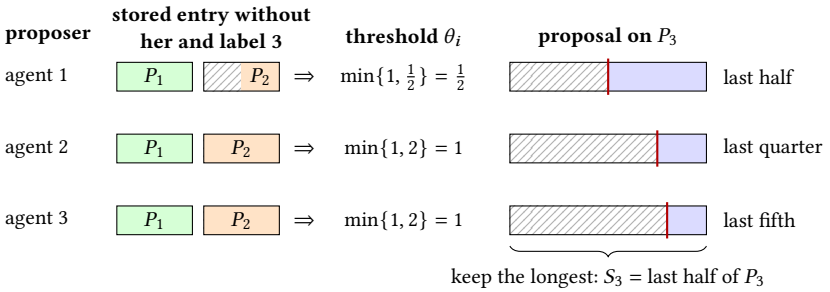

Fig. 3. Computing the suffix on label 3 in the full state of the running example. Each agent i looks up the stored entry for the other two agents and labels 1 and 2, which records suffixes only, not holders. Her threshold is her cheapest suffix there, and she proposes the longest suffix of P_3 within it. The table keeps the longest proposal. The final matching gives label 3 to agent 1, one of its longest proposers, so she receives exactly what she offered to take.

From uniqueness to an algorithm. The proposition also suggests how to compute certified suffixes: each agent proposes a suffix of each label, and we keep the longest proposal on that label. Lemma 8 will show that no proposal is too long. Smaller instances supply the limit on each agent's proposal. Suppose agent i holds label j in a certified allocation. Removing both leaves an envy-free allocation of the other labels to the other agents. The certified suffixes of that smaller instance are at least as long, so they cost i at least as much. Hence her own suffix costs her at most the cheapest suffix in the smaller certified allocation, whoever holds it. This bound is the threshold θ_i used next.

5.3 Computing certified suffixes

For agents $I \subseteq N$ and labels $L \subseteq J$ with $|I| = |L|$, write $S^{I,L}$ for the suffixes of a certified allocation of the pieces in L to the agents in I. Proposition 1 makes them unique. TRIM does not call itself. It fills a table with the entries $S^{I,L}$ in increasing order of $|I|$, computing each entry from entries one size smaller. On label $j \in L$, agent $i \in I$ proposes the longest suffix of P_j costing her at most

$$\theta_i = \min_{l\in L\setminus\{j\}} c_i\big(S_l^{I\setminus\{i\},L\setminus\{j\}}\big),$$

where $\theta_i = +\infty$ when $I = \{i\}$. Then $S_j^{I,L}$ is the longest of these proposals. Moreover, the holder of j in any certified allocation proposes exactly $S_j^{I,L}$: each label goes to one of its longest proposers, and she receives the suffix she offered to take. Figure 3 shows one entry being computed.

In the running example, Figure 3 computes label 3 in the full state. Without agent 1 and label 3, agents 2 and 3 keep P_1 whole and the last half of P_2, so agent 1's threshold is $1/2$ and she proposes

the last half of P_3. Without agent 2, or without agent 3, the remaining two agents keep P_1 and P_2 whole, so the thresholds of agents 2 and 3 are 1, and they propose the last quarter and the last fifth. The longest proposal, the last half, is S_3. On label 1, agents 2 and 3 both propose the whole piece, so P_1 stays whole although nobody assumed it would.

Correctness of the recurrence. Fix a state and let S be a certified allocation for it. Assume inductively that smaller states store their certified suffixes. We prove that no proposal exceeds S_j and that the holder's proposal reaches it.

For a proposal by i on j, take a certified assignment in the smaller state and add the proposed suffix for i. Agent i envies nobody by her threshold, and the other agents remain envy-free among themselves. Any resulting envy therefore points to j, so Lemma 8 bounds this proposal by S_j.

Conversely, let i hold j in S. Removing them leaves an envy-free allocation. The certified suffixes stored for the smaller state contain those of this restriction, so they cost i at least as much. Since i did not envy any suffix in the restriction, S_j meets her threshold. Her proposal therefore contains S_j, and by the first bound it equals S_j. So the holder of j is one of its longest proposers, which proves the recurrence. The identity of the holders in a smaller state is irrelevant to either argument, which is why the table stores only suffixes.

The final matching. Since $\mathcal{M}_i$ depends only on the suffixes, join each agent i to the labels in $\mathcal{M}_i(S^{N,J})$. Any certified allocation of the stored suffixes is a perfect matching in this bipartite graph. Every perfect matching is envy-free and, by Lemma 6, certified. By the converse argument above, each agent then holds a label on which her own proposal was a longest one, and she receives exactly that suffix. In the running example the graph joins agent 1 to labels 2 and 3 and agents 2 and 3 to labels 1 and 2. Its two perfect matchings are the two certified allocations described above.

Query count. There are $\sum_k \binom{n}{k}^2 = \binom{2n}{n}$ states. Each needs at most n^2 proposal cuts and n^2 evaluations of the retained suffixes. To cut or evaluate a trial suffix, use the recorded costs of the input components of P_j. Only the component containing its boundary requires an oracle query. Temporary proposal marks need no evaluation by other agents. The final matching uses only recorded comparisons. Appendix B proves the same bound for arbitrary nonnegative costs. This proves Lemma 7.

5.4 The cutter's advantage

Core uses Trim in place of SubCore, the trimming routine of Aziz and Mackenzie [2016]. In Core(r, N, C), agent r cuts C into n pieces of equal cost to her. We call Trim on those pieces and return its allocation Y, indexed by agent so that Y_i is agent i's suffix, together with its residue U.

Proof of Lemma 1. The allocation Y is envy-free by Lemma 7. The cutter makes $n-1$ cuts, and recording a cut for every agent takes at most $n+1$ queries (Appendix D), so Core uses $O(n^2)$ queries besides its call to Trim.

Write $\tau = c_r(C)/n$ and $x = c_r(Y_r)$. A whole piece, held by some h, costs τ. By envy-freeness every other allocated suffix costs at least x. Subtracting their costs from $n\tau$ gives $c_r(U) \le (n-1)(\tau - x)$, that is, $c_r(Y_h) - c_r(Y_r) = \tau - x \ge c_r(U)/(n-1)$. Positive residue cost implies $\tau > x$, and hence $h \ne r$. □

In the running example, $\tau = 1$ and $x = 1/2$, and the bound holds with equality (Section 4.5).

6 Finding a New Advantage

This section proves Lemma 2 (step ③). Throughout, (X, R) is envy-free, $(Y, U) =$ Core(r, N, R), and $(X \oplus Y, U)$ has no dominating group, so its non-domination graph Γ is strongly connected

ALGORITHM 3: REASSIGN(r, X, Y, U)

Input: The cutter r, an envy-free X with residue R, and $(Y, U) = \textsc{Core}(r, N, R)$, with no dominating group in $(X \oplus Y, U)$.

Output: An arc $a \to b$ of Γ, a permutation π of the new suffixes, and a tolerance α.

$K \leftarrow 4n^2$, and $g_{ij} \leftarrow \big(c_i(Y_j) - c_i(Y_i)\big)/c_i(U)$ for all i, j

$\Gamma \leftarrow$ the non-domination graph of $(X \oplus Y, U)$ ▷ *strongly connected*

if *some arc* $a \to b$ *of* Γ *has* $g_{ab} \geq K^{-n}$ **then** ▷ *case 1: a large gap; nothing moves*

  return $a, b, \pi =$ id, and $\alpha = \min\{K^{-2}, g_{ab}/K\}$

▷ *case 2: every arc of* Γ *has gap less than* K^{-n}

Pick h with $g_{rh} \geq 1/(n-1)$ ▷ *exists by Lemma 1*

Take a simple path $r = u_0, u_1, \ldots, u_\ell = h$ in Γ

$s \leftarrow$ the least index with $g_{u_s h} \leq K^{-(s+1)}$; set $a \leftarrow u_{s-1}$, $b \leftarrow u_s$, and $\alpha \leftarrow K^{-(s+1)}$

Take a simple path $h = v_0, v_1, \ldots, v_{\ell'} = b$ in Γ

Set $\pi(v_\nu) \leftarrow v_{\nu+1}$ for $\nu < \ell'$, $\pi(b) \leftarrow h$, and $\pi(i) \leftarrow i$ for every other agent ▷ *rotate*

return a, b, π, α

(Section 4.2). We adapt the reassignment lemma of Psomas [2026, Lemma 5] and the cycle exchanges of GoLeft [Aziz and Mackenzie, 2020, Sec. 3.6].

6.1 What is needed, and why

Progress requires a domination that did not exist before. CORE gives the cutter an advantage over the holder h of a whole piece (Lemma 1), but she may already dominate h, and then nothing is gained. The arcs of Γ show where dominations are missing. An arc $a \to b$ means that a does not dominate b in $(X \oplus Y, U)$, and hence not in (X, R) either, since adding the envy-free allocation Y preserves domination. We therefore want an arc $a \to b$ along which a has an advantage.

How large must this advantage be? It need not be large in absolute terms, only large compared with the envy we create to obtain it. Step ④ shrinks the residue to a small multiple of that envy, and changes a's advantage by at most another small multiple (Lemma 3). An advantage $K = 4n^2$ times the envy survives both (Section 4.4). This relative requirement is what makes the step possible: if no arc of Γ carries a large advantage, then moving bundles along arcs is cheap.

Measuring in units of the residue. If an agent assigned zero cost to U, she would dominate everyone and form a dominating group on her own. Thus $c_i(U) > 0$ for every i, and the *normalized gap*

$$g_{ij} = \frac{c_i(Y_j) - c_i(Y_i)}{c_i(U)}$$

is well defined, nonnegative by envy-freeness of Y, and zero when $i = j$. It is agent i's advantage over j's new suffix, measured in her own residues. Only the new suffixes Y move; each old bundle X_i stays on label i. We look for an arc $a \to b$, a permutation π, where agent i receives $Y_{\pi(i)}$, and a tolerance α such that

$$g_{i,\pi(i)} \leq \alpha \quad (i \in N), \qquad g_{a,\pi(b)} \geq K\alpha. \tag{4}$$

In words, every agent's new suffix costs her at most α residues more than her own, while a finds the new suffix on label b costlier than her original suffix Y_a by at least $K\alpha$ residues.

6.2 The algorithm

REASSIGN (Algorithm 3) distinguishes two cases.

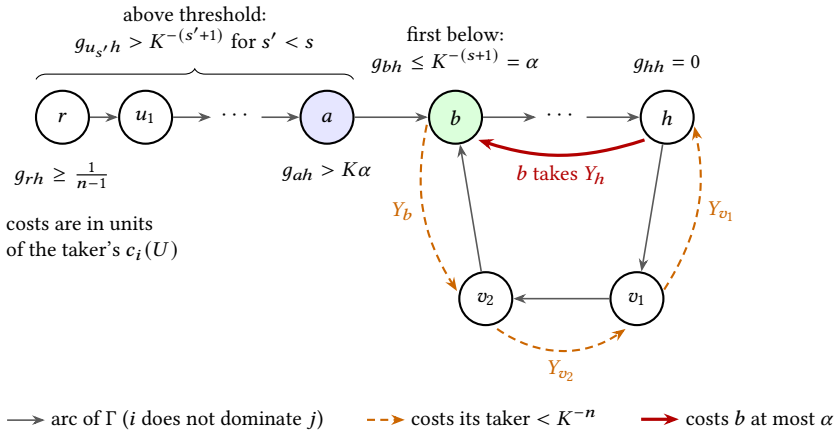

Fig. 4. Case 2 of Reassign. Walking from the cutter r to h, the gap $g_{u_s h}$ starts above its threshold $K^{-(s+1)}$ and ends at 0, so some arc $a \to b$ straddles the thresholds. Rotating the new suffixes around the cycle $h \to \cdots \to b \to h$ hands Y_h to b. Every agent's envy is at most $\alpha\, c_i(U)$, while a now finds b's bundle costlier than her own by at least $(K-1)\alpha\, c_a(U)$. Old bundles X_i never move.

Case 1: use an existing advantage. If some arc $a \to b$ already has a gap of at least K^{-n}, nothing needs to move. The tolerance α is set so that the gap is at least $K\alpha$, and capped at K^{-2} so that Repair applies.

Case 2: rotate along cheap moves. Otherwise every arc of Γ has a gap below K^{-n}. A small gap is a near-indifference, so passing a new suffix backwards along an arc costs its new holder very little. The cutter, however, has a large gap towards h. Walk along a path from r to h and compare each agent's gap towards h with a threshold that shrinks by a factor K per step (Figure 4). The gap starts above its threshold and ends at $g_{hh} = 0$, so some arc $a \to b$ straddles the thresholds: a finds Y_h much costlier than her own suffix, while b would accept Y_h at little extra cost. Now take a path from h back to b. Each agent on it takes her successor's suffix, and b takes Y_h. This closes a cycle. Every move costs its taker little, and a now finds the suffix on label b much costlier than her own.

Why the thresholds shrink geometrically. A fixed threshold would only give an arc from an agent above it to an agent below it, and the drop across that arc could be tiny. Thresholds that shrink by a factor K per step force a drop by a factor of more than K across a single arc: a's gap exceeds K times what b is allowed. The path has at most $n-1$ arcs, so the tolerance never falls below K^{-n} in case 2, nor below $K^{-(n+1)}$ overall.

The first round never rotates. In the first round, X is empty and case 1 always applies. Write $\tau = c_r(R)/n$ for the cost to the cutter of each of her pieces. Each suffix Y_j costs her at most τ, and U includes the part cut off the piece she holds, so $c_r(Y_j) - c_r(Y_r) \le \tau - c_r(Y_r) \le c_r(U)$. If r dominated some j, both inequalities would be equalities. Then every other trimmed part would cost her nothing, every other suffix would cost her τ, and r would dominate everyone. With no dominating group, r therefore dominates nobody. In particular $r \to h$ is an arc of Γ, and its gap is at least $1/(n-1) \ge K^{-n}$. Rotations appear only in later rounds, once X carries advantages of its own. Appendix C works through an example of a rotation.

6.3 Proof of Lemma 2

The non-domination graph Γ of $(X \oplus Y, U)$ is strongly connected. Its arcs were also undominated in (X, R), since adding an envy-free allocation preserves domination. We showed above that $c_i(U) > 0$ for every i, so the gaps g_{ij} are well defined. We first establish (4) in each case.

Case 1. If some arc $a \to b$ has $g_{ab} \geq K^{-n}$, take the identity permutation and $\alpha = \min\{K^{-2}, g_{ab}/K\}$. Then $g_{i,\pi(i)} = 0 \leq \alpha$ and $g_{a,\pi(b)} = g_{ab} \geq K\alpha$, so (4) holds, and $K^{-(n+1)} \leq \alpha \leq K^{-2}$.

Case 2. Every arc now has gap less than K^{-n}. By Lemma 1, some holder h satisfies $g_{rh} \geq 1/(n-1) > 1/K$. Follow a simple path $r = u_0, \ldots, u_\ell = h$ in Γ. Compare $g_{u_s h}$ with $K^{-(s+1)}$: the gap starts above its threshold and ends at zero. At the first index s with $g_{u_s h} \leq K^{-(s+1)}$, set

$$a = u_{s-1}, \qquad b = u_s, \qquad \beta = K^{-(s+1)}.$$

Then $a \to b$ is an arc, $g_{ah} > K\beta$, $g_{bh} \leq \beta$, and $K^{-n} \leq \beta \leq K^{-2}$, since $1 \leq s \leq \ell \leq n-1$.

Take a simple return path from h to b. Give each path agent before b her successor's bundle and give b the bundle Y_h, fixing everyone else. This is a cyclic permutation. Every move along the return path costs less than $K^{-n} \leq \beta$ in relative units, and the closing move costs b at most β. Thus (4) holds with $\alpha = \beta$. All choices use recorded costs.

Translate these bounds to the input of repair. Envy-freeness of X and Y gives $c_i(D_j) \geq c_i(X_i) + c_i(Y_i)$ for every j, whereas (4) gives $c_i(D_i) \leq c_i(X_i) + c_i(Y_i) + \alpha c_i(U)$. This proves the first inequality in (2). Also, since $c_a(X_b) \geq c_a(X_a)$ by envy-freeness of X,

$$c_a(D_b) - c_a(D_a) \geq \big(g_{a,\pi(b)} - g_{a,\pi(a)}\big) c_a(U) \geq (K-1)\alpha c_a(U).$$

The loss of one unit allows for a herself receiving a slightly costlier suffix during the rotation. □

How small the tolerance can be. The tolerance α may be as small as $K^{-(n+1)} = 2^{-\Theta(n \log n)}$. This is why Lemma 3 depends only logarithmically on $1/\alpha$: a repair whose cost grew linearly in $1/\alpha$ would by itself need $2^{\Theta(n \log n)}$ queries per call.

7 Repairing Envy and Shrinking the Residue

This section proves Lemma 3 (step ④). The input is an allocation D with residue U and a little envy: $c_i(D_i) \leq c_i(D_j) + \alpha c_i(U)$ for all i, j, where $0 < \alpha \leq 1/(8n^2)$. Labels are named after the agents who own the input bundles, so label j carries D_j and agent j is its *owner*; after repair its *holder* $\sigma(j)$ may be someone else.

7.1 What repair must achieve

Repair must do four things at once:

(i) remove the envy, so that the output is envy-free;
(ii) keep every input bundle D_j intact, so that old dominations survive (Lemma 4);
(iii) allocate almost all of U, so that the residue falls below agent a's advantage;
(iv) add nearly the same cost to every bundle, in every agent's eyes, so that the advantage itself survives.

Trimming removes envy, so (i) is easy on its own. But trimming the bundles D_j directly would cut into them and violate (ii). Adding an exactly equal-cost share of U to every bundle would serve (iii) and (iv), since it preserves every cost gap, but it would leave the envy in place. Repair does both. It attaches to every bundle a large share of U that costs every agent nearly the same, places $n-1$ small *buffers* in front, and calls Trim once. Trimming removes a prefix, so it must consume buffers before it can reach a bundle, and Section 7.4 shows that it never gets through. In the cake protocol of Aziz and Mackenzie [2016], further Core calls shrink the residue until an advantage becomes

domination [Aziz and Mackenzie, 2020, Sec. 3.2]; Repair achieves the required shrinkage in one call.

Example 1 (Why buffers are needed). Let $n = 3$ and $\alpha = 1/(8n^2) = 1/72$, the largest tolerance allowed. The input bundles are three blocks D_1, D_2, D_3, and the residue U is a block that costs every agent 72. Each agent's cost is spread evenly within each block:

	D_1	D_2	D_3	U
agent 1	**4**	8	5	72
agent 2	4	**4**	7	72
agent 3	7	4	**5**	72

So $\alpha c_i(U) = 1$. Agent 3 envies D_2 by exactly 1, and nobody else envies anybody. Suppose we give each bundle a third of U, placed behind it, and call Trim on the three resulting pieces. The equal thirds change no comparison, so agent 3 still envies label 2 by 1. Trim keeps labels 1 and 2 whole and trims the front of label 3 until agent 3 ties label 2. The front of label 3 is D_3, so a fifth of an input bundle returns to the residue. We return to this example with buffers in Section 7.3.

7.2 Nearly common shares

The construction needs large shares and small buffers whose costs are close to prescribed targets for every agent. What matters is the total error over all shares, rather than the error of each one.

Lemma 9 (Approximate common shares). *Let $w_1, \dots, w_m \geq 0$ sum to one, let $0 < \delta < 1$, and normalize $c_i(U) = 1$ for all i. We can find disjoint pieces $A_q \subseteq U$ with $c_i(A_q) \leq w_q$ for every i, q, leaving residue U_0 with $c_i(U_0) \leq \delta$ for every i. Equivalently, the total deficit is*

$$\sum_{q=1}^{m} \big(w_q - c_i(A_q)\big) = c_i(U_0) \leq \delta.$$

The procedure uses $O(n^3 m \log(2nm/\delta))$ queries and polynomially many exact arithmetic operations in the size of its interval records, n, m, and $\log(1/\delta)$.

We prove this lemma in Section 7.5. A consequence used below is that, for any set F of slots,

$$\sum_{q \in F} w_q - \delta \leq c_i\left(\bigcup_{q \in F} A_q\right) \leq \sum_{q \in F} w_q. \tag{5}$$

Thus a union of many shares still incurs a total error of at most δ.

7.3 The construction

Repair (Algorithm 4) normalizes each agent's cost of U to 1 and cuts U into n^2 nearly common shares, with the targets in Table 3. On label j it places $n - 1$ buffers, then D_j, then a large share T_j. We call $D_j \cup T_j$ the *protected part* of label j (Figure 5). Since every buffer costs every agent nearly the same, it does not matter which buffers go to which label.

How the envy is removed. Repair does not remove the envy itself. It arranges the pieces so that one call to Trim, which always returns an envy-free allocation, removes it safely. Trimming removes envy in two ways. An envious agent's piece can be trimmed from the front until it ties another suffix, and because the buffers come first, this consumes buffer. Or the envious agent can take the bundle she envied while its holder takes another, which is why the holder of a label may change.

Table 3. The parameters of Repair(N, D, U, α), in units where every agent's cost of U is 1.

Symbol	Value	Role
B	$n(n-1)$	number of buffers, $n-1$ per label
p	2α	target cost of each buffer: more than the envy α plus the rounding loss δ
t	$(1-Bp)/n$	target cost of each large share; the shares carry almost all of U
δ	$\alpha/2$	total rounding loss allowed in Lemma 9

ALGORITHM 4: Repair(N, D, U, α)

Input: A partial allocation D with residue U and tolerance α satisfying Lemma 3.
Output: An envy-free allocation W retaining every D_j intact, the holder map σ, and the residue R'.

1 For this call, scale each cost measure so that $c_i(U) = 1$
2 Set $B \leftarrow n(n-1)$, $p \leftarrow 2\alpha$, $t \leftarrow (1-Bp)/n$, and $\delta \leftarrow \alpha/2$
3 Use Lemma 9 to make n large shares T_j of target t and B buffers $Q_{j,k}$ ($j \in N$, $1 \le k \le n-1$) of target p, with total deficit at most δ
4 For every label j, form $P'_j \leftarrow Q_{j,n-1} +\!\!+ \cdots +\!\!+ Q_{j,1} +\!\!+ (D_j +\!\!+ T_j)$
5 $W \leftarrow$ Trim$(N, (P'_j)_{j\in N})$, and let $\sigma(j)$ be the holder of label j
6 **return** W, σ, and $R' \leftarrow (U \cup \bigcup_j D_j) \setminus \bigcup_i W_i$

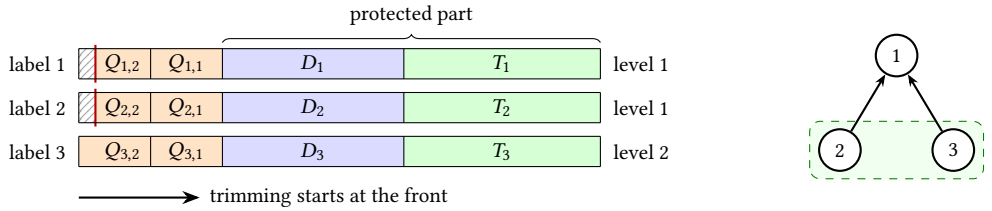

Fig. 5. Repair in the running example (not to scale), in the first round, where $D = Y$. Label 1 carries agent 1's suffix D_1 (the last half of P_3), label 2 carries agent 2's piece $D_2 = P_1$, and label 3 carries agent 3's suffix D_3 (the last half of P_2). Left: each label carries two buffers $Q_{j,2}, Q_{j,1}$, its input bundle D_j and a large share T_j. The cuts shown, in outer buffers, are one possible outcome, and the level counts the whole buffers kept. In every outcome each protected part $D_j \cup T_j$ survives. Right: in every outcome agents 2 and 3 both dominate agent 1 (Section 4.5). Here arrows show domination, unlike the arcs of the non-domination graph.

Example 1, continued. Here $p = 1/36$ and $t = 5/18$, so each buffer costs every agent 2 and each large share costs every agent 20. Every agent's cost is spread evenly on U, so blocks of the right lengths are exact common shares. Each label then carries 24 in addition to its input bundle, for every agent; write P'_j for the piece formed on label j:

	P'_1	P'_2	P'_3
agent 1	**28**	32	29
agent 2	28	**28**	31
agent 3	31	28	**29**

Agent 3 still envies label 2 by 1. Trim now removes this envy by trimming half of the outer buffer of P'_3, which costs 1 to everybody, and agent 3 ties label 2 at 28. Every input bundle is intact. Labels 1 and 2 have level 2 and label 3 has level 1, the additions cost 24, 24 and 23, and the residue is half a buffer.

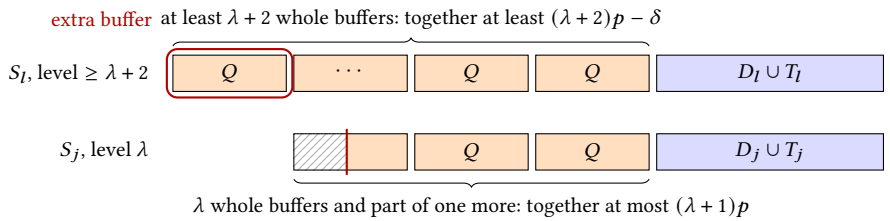

Fig. 6. Step 1 of the proof, through the eyes of the owner j of label j, with $c_j(U) = 1$. A suffix two or more levels higher carries at least one extra whole buffer (outlined), worth about $p = 2\alpha$ to her. This outweighs her input envy, $c_j(D_j) \leq c_j(D_l) + \alpha$, because by (5) all deficits together cost her at most $\delta = \alpha/2$.

7.4 Why every input bundle survives

We now prove Lemma 3. Use the normalized costs of Algorithm 4. Since $nt + Bp = 1$ and $Bp \leq 1/4$, the targets are valid. The output is envy-free by Lemma 7. It remains to show that no protected part is cut, and to bound the unequal additions and the residue.

Levels. Let S_j be the returned suffix on label j, whoever holds it. Its *level* $\ell(j)$ is -1 if it cuts into the protected part, and otherwise is the number of whole buffers it retains. A suffix of level $\lambda \geq 0$ contains its protected part, λ whole buffers, and at most part of one more buffer. A whole piece has level $n-1$. Repair succeeds exactly when no level is -1.

Step 1: two levels apart means strictly costlier. Let j and l be labels with $\ell(j) = \lambda$ and $\ell(l) \geq \lambda + 2$, and look through the eyes of the owner j of label j, who need not be its final holder. Label l carries at least one more whole buffer than label j could, even if S_j keeps part of its next buffer (Figure 6). By (5),

$$c_j(S_j) \leq c_j(D_j) + t + (\lambda+1)p, \qquad c_j(S_l) \geq c_j(D_l) + t + (\lambda+2)p - \delta.$$

The first bound holds also at $\lambda = -1$, since that suffix lies inside its protected part. The input envy bound $c_j(D_j) \leq c_j(D_l) + \alpha$ therefore gives

$$\ell(l) \geq \ell(j) + 2 \quad \Longrightarrow \quad c_j(S_l) - c_j(S_j) \geq p - \delta - \alpha = \tfrac{\alpha}{2} > 0. \tag{6}$$

In words: one extra buffer is worth $p = 2\alpha$, which outweighs both the envy α and the total rounding loss $\delta = \alpha/2$. Adjacent levels may differ by only part of a buffer, which is why two levels are needed.

Step 2: no protected part is cut. Suppose some protected part were cut, so that level -1 occurs. Level $n-1$ also occurs, since a certified allocation keeps some piece whole. With n labels and $n+1$ possible levels $-1, 0, \ldots, n-1$, some intermediate level μ is missing. Let $H = \{j : \ell(j) < \mu\}$, the labels below the gap; it is nonempty. Every label outside H is at least two levels above every label in H. By (6), the owner of each label in H finds her own label's suffix strictly cheaper than every suffix outside H, so all her cheapest labels lie in H. These owners are $|H|$ distinct agents, and every label in H is trimmed, since its level is below $\mu \leq n-2$. This contradicts the certification test, Lemma 6. Thus every protected part survives. The buffers are there exactly to make the owners below a gap strictly prefer their own side.

Step 3: only buffers and the rounding residue remain. Let $\sigma(j)$ be the final holder of label j. We now have $W_{\sigma(j)} = S_j = D_j \cup Z_j$, where Z_j consists of T_j and the retained buffers and buffer parts. For every i, j,

$$t - \delta \leq c_i(Z_j) \leq t + (n-1)p.$$

Hence $|c_i(Z_j) - c_i(Z_l)| \le (n-1)p + \delta \le 2n\alpha$. The residue contains only discarded buffer parts and U_0 from Lemma 9, so $c_i(R') \le Bp + \delta \le 2n^2\alpha$. Rescale to obtain the claims. There are $n + B = n^2$ slots, giving $O(n^5 \log(n/\alpha))$ queries beyond the single call to Trim. This proves Lemma 3. □

7.5 Computing the common shares

It remains to prove Lemma 9. We adapt the fractional rounding method of Alon and Graur [2021], who split a necklace into equal shares between thieves; here the shares are prescribed and unequal, and cells are cut with Robertson–Webb queries. The idea has three steps. First, meet every target exactly with a fractional assignment, in which every agent values every slot at exactly its target. Second, make all but a few cells integral without breaking any target, using the fact that the number of split cells is bounded by the number of equations. Third, repeatedly refine only the split cells. Their number stays at most nm while the cost of each one halves in every round, so after logarithmically many rounds they are negligible and can be discarded.

Proof of Lemma 9. Fractions are bookkeeping variables, not physical allocations. Think of each slot as a recipient with target w_q. Initially U is one cell, and its fraction in slot q is w_q.

Keep few split cells. For a cell E, denote its current fraction in slot q by x_{Eq}. Preserve the equations $\sum_q x_{Eq} = 1$ for every cell E and $\sum_E c_i(E)x_{Eq} = w_q$ for every i, q. With κ cells there are at most $\kappa + nm$ constraints. If more than $\kappa + nm$ fractions are positive, the corresponding columns are linearly dependent. Move the fractions along a dependence until one becomes zero, keeping all fractions nonnegative. The dependence preserves each cell sum, so it has both signs. Only positive fractions are changed, and this ends with at most $\kappa + nm$ positive fractions. Cells assigned to one slot never change: their cell equation fixes their only positive fraction at one. As each cell needs at least one positive fraction, at most nm cells remain split between slots.

Refine the split cells. In each round refine only those split cells at every agent's cost midpoint, give each child its parent's fractions, and repeat the elimination. Every child costs each agent at most half its parent's cost. After $d = \lceil \log_2(nm/\delta) \rceil$ rounds the remaining split cells therefore cost each agent at most $nm2^{-d} \le \delta$ in total. Leave these cells in the residue and give every other cell to its slot. Their fractions account for all deficits. Each round uses at most n^2m prefix cuts, with at most n evaluations per cut. Gaussian elimination finds each dependence, and each move removes one positive fraction. Fixed pivot and tie orders make the procedure deterministic. □

8 Certified Trimming: Cake, Chores, and Equilibrium

The trimming problem has the same oracle complexity, up to polynomial overhead, for cake and chores. Its relation to minimum-price equilibrium uses the same construction. For the reductions, assume that pieces are nonempty and every value or cost is positive on each interval of positive length, so suffix functions are continuous and strictly increasing. All reductions use exact arithmetic and the component records of Appendix D; no bit-complexity claim for arbitrary real functions is intended.

8.1 Two-way reductions for certified trimming

For cake, a suffix allocation is certified if it is envy-free and its holders can be ordered so that each proper suffix is tied with an earlier holder's own suffix *by that earlier holder*. This reverses who witnesses a trim compared with chores. For both signs, the certified suffix vector is the coordinatewise greatest envy-free vector, and every envy-free matching of that vector is certified (Proposition 1 and Lemma 10).

Proposition 2 (Polynomial oracle equivalence). *Certified chore trimming on n agents and pieces reduces to certified cake trimming on n agents and pieces. The reverse reduction uses at most*

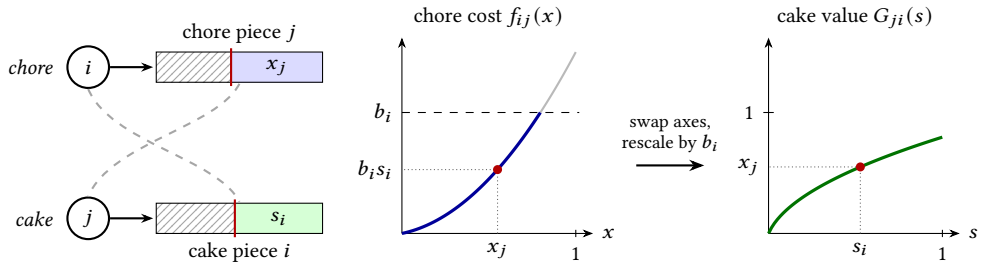

Fig. 7. The chore-to-cake construction (7) for one agent and one piece (not to scale). Left: agents and pieces exchange roles (dashed), and holding is reversed. If chore agent i holds the suffix of length x_j on chore piece j, then cake agent j holds the suffix of length s_i on cake piece i; trimmed fronts are hatched. Middle: only the part of f_{ij} below $b_i = \min_k f_{ik}(1)$ is used (thick). Right: swapping the axes and dividing costs by b_i turns it into the cake value $G_{ji}(s) = f_{ij}^{-1}(b_i s)$. The marked point is the same in both plots: the chore suffix costs agent i exactly $b_i s_i$, and the cake suffix is worth exactly x_j to agent j, as in (9).

$n+1$ agents and pieces. Each direction uses one target-solver call, $O(n^2)$ preprocessing and recovery queries, $O(1)$ original queries per simulated query, and polynomial additional work in the supplied records.

The construction inverts suffix functions and exchanges agents with pieces. If $f_{ij}(x)$ is chore agent i's cost of a suffix of relative length x on piece j, set $b_i = \min_j f_{ij}(1) > 0$ and define cake values

$$G_{ji}(s) = f_{ij}^{-1}(b_i s), \qquad 0 \le s \le 1. \tag{7}$$

Cake agent j represents chore piece j, and cake piece i represents chore agent i (Figure 7). Reversing the matching converts envy-free allocations in both directions. The induced maps preserve coordinatewise order, hence the greatest vectors and certification. The reverse reduction first adds a dummy agent and piece to normalize whole-piece values. Appendix E.2 proves both directions.

8.2 A faster query bound via cake trimming

We show how to turn any routine that assigns cake suffixes without envy and gives one agent a whole piece into a certified cake-trimming algorithm, enabling our reduction in Proposition 2 to transfer the SubCore bound of Ye and Bai [2026] to certified chore trimming.

Corollary 1. *Given the component costs, certified chore trimming for n agents can be computed deterministically in $O(n2^n)$ Robertson–Webb queries. Under the assumptions of Theorem 1, this gives a deterministic protocol for complete envy-free chore division using $O(n^3 2^n)$ queries.*

Proof. We first compute certified cake suffixes on positive inputs. With one agent, give her the whole piece. With $m \ge 2$ agents, run the modified SubCore of Ye and Bai [2026, Propositions 4.4–4.5] on the m pieces and one extra empty piece, with all benchmarks zero. In $O(m2^m)$ queries it assigns suffixes of distinct nonempty pieces without envy. Some agent h receives a whole piece P, since the last step assigns a whole uncontested piece. Give P to h. On each other piece keep the longest suffix that h values at most as much as P. This contains the suffix assigned by SubCore, by envy-freeness. Recurse on these pieces and the remaining agents.

The previous suffixes of the remaining agents still form an envy-free allocation on the shortened pieces. By induction the recursive allocation is certified, so Lemma 10 implies that each recursive suffix contains the previous suffix on its label. Each remaining agent receives her favorite recursive suffix, so she obtains at least her previous value and weakly prefers her share to P. Agent h weakly prefers P by construction. Prepend h to the recursive certifying order. Each suffix received whole

in the recursive call is either a whole input piece or a suffix that h trimmed to the value of P. In the latter case h witnesses the trim. Every other trim keeps its recursive witness.

At size m, trimming and recording the shortened pieces adds $O(m^2)$ queries. Summing $O(m2^m + m^2)$ over $m \le n$ gives $O(n2^n)$. Proposition 2 transfers this bound to positive chore trimming. For zero costs or empty pieces, Appendix B simulates the positive routine with constant cost per query and $O(n^2)$ further queries, preserving whole pieces and all certificate equalities. Finally, the proof of Theorem 1 counts $O(m^2)$ trimming calls at each active population size m, with polynomially many other queries. Summing $O(m^3 2^m)$ over these sizes gives $O(n^3 2^n)$. □

8.3 Connection to minimum-price equilibrium

Create one buyer per chore label j and one good per chore agent i, with outside-option utility zero and utility $u_{ji}(q_i) = f_{ij}^{-1}(b_i - q_i)$ for good i at price q_i. Extend f_{ij} linearly with slope one outside $[0, 1]$. These utilities are continuous, strictly decreasing in price, and potentially nonquasilinear, as in Alaei et al. [2016].

Proposition 3 (Polynomial oracle reduction). *Certified chore trimming reduces to componentwise minimum-price competitive equilibrium in this market. Each utility evaluation or inverse query uses at most one Robertson–Webb query; setup and recovery use $O(n^2)$ additional queries and polynomial computation in the supplied records.*

The connection is explicit: on the equilibrium price range $0 \le q_i \le b_i$,

$$u_{ji}(q_i) = G_{ji}(1 - q_i/b_i).$$

Thus prices encode the fractions removed from the transformed cake pieces. Greatest retained suffixes correspond to minimum prices. Appendix E.3 gives the proof and the analogous cake-market correspondence. Both certified trimming problems therefore reduce to minimum-price equilibrium; these reductions identify a common algorithmic target without supplying a polynomial-query solver.

9 Conclusions

Our longest-proposal recurrence over subsets gives a self-contained deterministic protocol for complete envy-free chore division using $O(n^4\binom{2n}{n})$ Robertson–Webb queries. Corollary 1 sharpens the bound to $O(n^3 2^n)$ by reducing certified chore trimming to cake trimming. Both implementations use buffered repair to preserve old bundles while allocating almost all of the residue. Together with reassignment, repair turns useful advantages into dominations within polynomially many rounds.

The exponential factor comes entirely from certified trimming. For positive costs and values, the two-way reductions show that finding a polynomial-query algorithm for certified chore trimming is equivalent to doing so for certified cake trimming. The minimum-price equilibrium connection provides another formulation of this remaining problem.

AI Acknowledgements

We used LLMs, in particular ChatGPT-6.0 Astra and Claude Opus-5.5, to adapt ingredients from prior work, to help with the exposition and to develop and simplify arguments. The interactions led to some insights and connections which could be beneficial for further work.

References

Saeed Alaei, Kamal Jain, and Azarakhsh Malekian. 2016. Competitive Equilibria in Two-Sided Matching Markets with General Utility Functions. *Operations Research* 64, 3 (2016), 638–645. https://doi.org/10.1287/opre.2016.1509

Noga Alon and Andrei Graur. 2021. Efficient Splitting of Necklaces. In *48th International Colloquium on Automata, Languages, and Programming (Leibniz International Proceedings in Informatics, Vol. 198)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 14:1–14:17. https://doi.org/10.4230/LIPIcs.ICALP.2021.14

Haris Aziz and Simon Mackenzie. 2016. A Discrete and Bounded Envy-Free Cake Cutting Protocol for Any Number of Agents. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 416–427. https://doi.org/10.1109/FOCS.2016.52

Haris Aziz and Simon Mackenzie. 2020. A Bounded and Envy-Free Cake Cutting Algorithm. *Commun. ACM* 63, 4 (2020), 119–126. https://doi.org/10.1145/3382129

Steven J. Brams and Alan D. Taylor. 1995. An Envy-Free Cake Division Protocol. *The American Mathematical Monthly* 102, 1 (1995), 9–18. https://doi.org/10.1080/00029890.1995.11990526

Sina Dehghani, Alireza Farhadi, MohammadTaghi HajiAghayi, and Hadi Yami. 2018. Envy-free Chore Division for An Arbitrary Number of Agents. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2564–2583. https://doi.org/10.1137/1.9781611975031.164

Alireza Farhadi and Mohammadtaghi Hajiaghayi. 2026. The Query Complexity of Envy-Free Division: Cake, Chores, and Mixed Resources. https://doi.org/10.48550/arXiv.2611.11424 arXiv:2611.11424 [cs.GT]

Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. 2004. On Approximately Fair Allocations of Indivisible Goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*. Association for Computing Machinery, 125–131. https://doi.org/10.1145/988772.988792

Simon Mackenzie and Mashbat Suzuki. 2026. Exact Envy-Free Cake Cutting with Polynomial Expected Queries. arXiv:2610.28535 [cs.GT] https://arxiv.org/abs/2610.28535

Elisha Peterson and Francis Edward Su. 2009. N-Person Envy-Free Chore Division. https://doi.org/10.48550/arXiv.0909.0303 arXiv:0909.0303 [math.CO]

Alexandros Psomas. 2026. Cutting a Cake For Human Consumption: An Elementary n-agent Cake Cutting Protocol. arXiv:2610.30370 [cs.GT] https://arxiv.org/abs/2610.30370

Jack Robertson and William Webb. 1998. *Cake-Cutting Algorithms: Be Fair If You Can*. A K Peters. https://www.routledge.com/Cake-Cutting-Algorithms-Be-Fair-If-You-Can/Robertson-Webb/p/book/9781568810768

Qilin Ye and Yannan Bai. 2026. Cutting Down the Tower: Single-Exponential Envy-Free Cake Cutting. arXiv:2609.05191 [cs.GT] https://arxiv.org/abs/2609.05191

A Existence of certified suffix allocations

Assume positive costs on every interval of positive length and nonempty input pieces. We show that a certified suffix allocation exists. The argument uses envy-cycle exchanges [Lipton et al., 2004] and a limit; the trimming protocol itself uses only finitely many queries.

For $\eta > 0$, choose a suffix allocation with envy at most η maximizing total allocated length, breaking ties by minimizing the sum of the holders' own costs. Such an allocation exists: the empty suffixes are feasible, the suffix lengths range over a compact box, costs are continuous, and there are finitely many assignments.

Draw an arc from a holder to anyone she strictly envies. This graph is acyclic. Exchanging bundles along a cycle would lower every member's own cost while preserving both the set of suffixes and the envy bound, contradicting the tie-break. Moreover, every sink holds a whole piece. A sink envies nobody, so slightly lengthening a trimmed suffix raises her cost by at most η and keeps her envy within the bound. To everyone else the suffix becomes weakly less attractive. This would increase total allocated length, a contradiction.

As $\eta \downarrow 0$, take a subsequence with fixed assignment and reverse topological order and convergent suffix lengths. In this order, anyone whom a holder envies comes earlier. The limit is envy-free. Each holder of a trimmed limit suffix eventually held a trimmed suffix and hence strictly envied an earlier holder. Some such earlier holder recurs infinitely often. The envy lies in $(0, \eta]$, so it becomes an equality in the limit, giving the required certificate.

B Zero costs and coincident marks

Algorithm 2 was described for positive costs. We show how to run it on arbitrary nonnegative costs, including empty pieces. The simulation is local to a trimming call, whose output only compares

suffixes and whose certificate only requires whole pieces and equalities, so marks with the same true cost can be separated there. A related device, ordered marks with tie-breaking keys, appears in Ye and Bai [2026, Appendix F.3].

In this appendix, s denotes a virtual length and t a virtual cost. Represent each original label j by a virtual interval of unit length. Maintain two kinds of finite nondecreasing maps: one per label from virtual suffix lengths s to physical suffix positions x, and one per agent from virtual costs t to true costs u. Each agent–label pair also has a table of evaluations (s, t), strictly increasing in both coordinates. If s maps to x and t to u, a recorded evaluation means that the physical suffix at x costs the agent u.

Initialize virtual lengths $0, 1$ at the empty and whole physical suffixes, which may coincide. For each agent map virtual zero to true zero. Give her distinct whole-piece costs increasing positive integer ranks, allowing a positive rank to map to zero. Enter those ranks as her virtual whole-piece costs. All pair tables now have both endpoints.

For a virtual evaluation at s, reuse a recorded pair if present. Otherwise obtain its physical position x from the label map, interpolating monotonically between neighboring entries and recording s with this image when s is new. Evaluate the true cost u at x. The two pair entries bracketing s have virtual costs $t_- < t_+$ whose true images bracket u. Insert a fresh $t \in (t_-, t_+)$ with image u in the agent's map and record (s, t). The finitely many intervening entries have ordered images. Insert t at the midpoint of a gap consistent with u, using the last such gap to settle ties.

For a virtual inverse query at t, reuse the recorded length s if the pair's table already contains (s, t). Otherwise interchange the roles of the maps: interpolate the requested virtual cost t in the agent's map, recording t with this image u when t is new, then cut true cost u on the physical label. Clip the answer to the physical positions of the bracketing pair entries. Monotonicity preserves its cost. Between the bracketing virtual lengths, the intervening entries of the label map have ordered physical images. Insert a fresh virtual length s at the midpoint of a gap consistent with the clipped position, again using the last such gap to settle ties, and record (s, t). In either operation, a consistent insertion gap exists because the bracketing images enclose the required image and all intervening images are nondecreasing, including when images coincide. All interpolation and ties use fixed rules. Each operation costs at most one physical query given the component records.

Register the returned suffixes and their costs to all agents, at most n^2 more evaluations. Linear interpolation of each strictly increasing pair table extends every finite transcript to positive nonatomic costs on the virtual pieces. Thus the positive protocol terminates within its bound, and its output has a certificate on such a completion. For each agent, her single nondecreasing cost map preserves every weak comparison and equality on projection. Each virtual whole piece maps to its physical whole, so the same ordering certifies the physical allocation. This also covers arbitrary choices of valid physical cut answers.

C An example of a rotation

This appendix illustrates case 2 of Reassign (Algorithm 3 and Figure 4), in which the new suffixes are passed around a cycle.

Example 2 (A rotation). The numbers below illustrate case 2; they need not arise from a run of Divide. Let $n = 3$, so $K = 36$, and let agent 1 be the cutter. Suppose the costs are as follows.

	X_1	X_2	X_3	Y_1	Y_2	Y_3	U
agent 1	**2**	3	5	**1**	1	3	4
agent 2	2	**2**	2	2	**2**	2	3
agent 3	5	2	**2**	4	3	**3**	3

Both X and Y are envy-free, and Y has the shape that Core produces: agent 3 holds a whole piece, costing the cutter 3, and $c_1(U) = 4$. In $(X \oplus Y, U)$, agent 1 dominates agent 3 and agent 3 dominates agent 1, with advantages $5 \geq 4$ and $4 \geq 3$, and there are no other dominations. So Γ has the arcs $1 \to 2$, $2 \to 1$, $2 \to 3$ and $3 \to 2$, and is strongly connected. Every arc has gap 0, so case 1 does not apply. The large gaps $g_{13} = 1/2$ and $g_{31} = 1/3$ lie on pairs that are already dominated, so they are not arcs and cannot be used.

Here $h = 3$, with $g_{13} = 1/2$. The path $1 \to 2 \to 3$ gives $g_{23} = 0 \leq K^{-2}$ at $s = 1$, so $a = 1$, $b = 2$ and $\alpha = K^{-2}$. The return path from 3 to 2 is the single arc $3 \to 2$: agent 3 takes Y_2 and agent 2 takes Y_3. Both are indifferent, so no envy appears. Agent 1 now finds $D_2 = X_2 \cup Y_3$ costlier than $D_1 = X_1 \cup Y_1$ by $6 - 3 = 3$, where before the rotation her advantage over agent 2 was 1. This is not yet a domination, since $c_1(U) = 4$; it becomes one once Repair shrinks the residue.

D Recording costs

For the pieces used across subroutines we keep each active agent's cost of every component interval. Initial totals require at most n evaluations. One evaluation per agent at a new physical cut determines both parts by subtraction, so recording a cut for everyone costs at most $n + 1$ queries. Trimming also considers trial marks. During that call we keep the input component records fixed, and obtain an agent's cost for a new suffix by evaluating just the part in its boundary component. Only the returned boundaries need to enter the records passed to later calls.

E Proofs for the Trimming Reductions

We prove the results of Section 8 under its positivity assumptions. The cake facts needed below are included so that the reductions do not depend on a separate cake protocol.

E.1 Certification and greatest suffixes for cake

For a cake suffix vector s, let $\mathcal{F}_i(s)$ be the labels maximizing agent i's value. Recall that a cake certificate orders the holders so that each proper suffix is tied by an earlier holder with that holder's own suffix.

Lemma 10 (Cake certification and maximality). *Every positive cake instance has a certified suffix allocation. Its suffix vector is coordinatewise at least every envy-free suffix vector on the same agents and pieces. Consequently the certified vector is unique, and every envy-free matching of it is certified.*

Proof. *Existence.* For $\eta > 0$, maximize total retained length among suffix allocations with envy at most η, breaking ties by maximizing the sum of the holders' own values. Compactness, continuity and finitely many matchings give a maximizer. Draw an arc from each holder to anyone she strictly envies. A directed cycle could be exchanged to raise all its members' own values while preserving the envy bound, contradicting the tie-break. Every source holds a whole piece: nobody envies her, so a small extension of a proper suffix would keep all envy within η and increase total length. As $\eta \downarrow 0$, take a subsequence with fixed matching and topological order and convergent suffix lengths. The limit is envy-free. Each holder of a proper limit suffix eventually holds a proper suffix and hence has an incoming envy arc. One earlier envious holder recurs along a subsequence; her envy, bounded by η, becomes the equality required for a cake certificate.

A counting test. An envy-free cake allocation is certified exactly when every nonempty set H of trimmed labels satisfies

$$\left|\{i : \mathcal{F}_i(s) \cap H \neq \varnothing\}\right| > |H|. \tag{8}$$

For a certified allocation, its $|H|$ holders all count, and an earlier holder outside H ties the first H-label in the certificate, giving one more. Conversely, draw an arc from a holder to each holder

whose suffix she ties with her own. If some holders cannot be reached from a whole piece, their labels form a nonempty trimmed set H with no incoming arc. Exactly its $|H|$ holders can have a favorite in H, contradicting (8). Ordering by distance from a whole piece gives a certificate. This test depends only on the suffixes.

Maximality. Let s be certified and y envy-free. If $H = \{j : y_j > s_j\}$ is nonempty, it consists of trimmed labels of s. Every agent with a favorite in H at s strictly prefers a label in H to every label outside H at y: a favorite in H has grown strictly, whereas all outside labels have weakly shrunk. She must therefore hold an H-label at y. At most $|H|$ agents can do so, contradicting (8). Thus $y \le s$. Two certified vectors consequently coincide, and the counting test certifies every envy-free matching of that vector. □

E.2 Two-way reductions for certified trimming

Proof of Proposition 2. *From chores to cake.* Use f_{ij}, b_i and G_{ji} from (7). The inverse is defined because $b_i \le f_{ij}(1)$. Each G_{ji} is continuous, strictly increasing and zero at zero, so it specifies a valid nonatomic valuation on the new piece.

The following maps convert envy-free suffix vectors in both directions, reversing their matchings:

$$x_j = \max_i G_{ji}(s_i), \qquad s_i = \frac{\min_j f_{ij}(x_j)}{b_i}. \tag{9}$$

Suppose cake agent j holds cake piece i. Envy-freeness gives $G_{ji}(s_i) = x_j$, hence $f_{ij}(x_j) = b_i s_i$. For every other chore label k, the definition of x_k gives $f_{ik}(x_k) \ge b_i s_i$. Giving chore piece j to chore agent i is therefore envy-free. Conversely, in an envy-free chore allocation set $d_i = \min_j f_{ij}(x_j)$. Since $x_j \le 1$, we have $0 \le d_i \le b_i$. With $s_i = d_i/b_i$, each $G_{ji}(s_i) \le x_j$, with equality when agent i holds label j. The reversed matching is therefore cake-envy-free.

The matched equalities show that the two maps in (9) are mutual inverses on envy-free vectors. Both maps are coordinatewise nondecreasing, so they carry the greatest vector of either instance to the greatest vector of the other. Proposition 1, Lemma 6 and Lemma 10 therefore ensure that certified outputs recover certified allocations.

From cake to chores: normalize whole-piece values. Let $v_{ij}(s)$ be the input cake suffix values. Set $M = 2\max_{i,j} v_{ij}(1)$ and $h_{ij} = v_{ij}/M$; this common scaling preserves all comparisons and certificates. Add a dummy agent 0 and piece 0. With $\delta_i = \frac{1}{2}\min_j h_{ij}(1) > 0$, define

$$\begin{aligned} H_{ij}(s) &= h_{ij}(s) && (i, j \ne 0),\\ H_{i0}(s) &= \delta_i s && (i \ne 0),\\ H_{0j}(s) &= s && (\text{all } j). \end{aligned} \tag{10}$$

Then $\max_i H_{ij}(1) = 1$ for every piece j.

This padding preserves the certified solution. Let s^* be the original certified vector, whose existence follows from Lemma 10. Some original piece k is whole. Every real agent's own value is at least $h_{ik}(1) \ge 2\delta_i$, so she does not envy the whole dummy piece. The dummy agent can hold that piece without envy, since all suffix lengths are at most one. Placing her first in the original certificate certifies $(s^*, 1)$ in the enlarged instance. Uniqueness forces every certified output there to have those lengths. No real agent can hold piece 0, because she would envy the whole original piece k. Removing the dummy agent and piece leaves an envy-free matching of s^*, which is certified by Lemma 10.

Extend H_{ij} beyond 1 by $H_{ij}(s) = H_{ij}(1) + s - 1$. Define chore costs, again exchanging agents and labels, by

$$F_{ji}(x) = H_{ij}^{-1}(x), \qquad 0 \le x \le 1. \tag{11}$$

Here $\min_i F_{ji}(1) = 1$, since all whole-piece cake values are at most one and the dummy agent's are exactly one. Applying (7) to this chore instance returns precisely the enlarged cake instance on $[0, 1]$. The correspondence already proved converts a certified chore output to its certified cake output. Deleting the dummy agent and piece solves the original instance.

Oracle cost. Evaluating G_{ji} uses one original suffix cut; evaluating its inverse uses $G_{ji}^{-1}(z) = f_{ij}(z)/b_i$ and one suffix evaluation. The reverse construction adds only explicit scaling and linear functions. Whole-piece records and recovery cost $O(n^2)$ queries. An evaluation of a virtual interval is a difference of two suffix values; a virtual cut uses a suffix evaluation and inversion. Across pieces, sum their recorded whole values and query only the boundary pieces. Thus each simulated Robertson–Webb query uses $O(1)$ original queries. Searching the component records, reversing the matching, and recovering a certificate by reachability take polynomial additional work. □

E.3 Minimum-price equilibrium and the shared transformation

A competitive equilibrium in the market of Section 8.3 consists of nonnegative prices and a matching giving each buyer a utility-maximizing choice, including the outside option; every unmatched good has price zero.

Proof of Proposition 3. Every equilibrium matches all buyers and goods. Otherwise an unmatched buyer could obtain positive utility from an unmatched, zero-price good. Also $q_i \le b_i$, since a higher price gives every buyer negative utility. Set $x_j = \max_i u_{ji}(q_i) \in [0, 1]$. If buyer j receives good i, then $f_{ij}(x_j) = b_i - q_i$, whereas $f_{ik}(x_k) \ge b_i - q_i$ on every label k. Giving the size-x_j suffix to chore agent i is therefore envy-free. Conversely, an envy-free suffix allocation with holder costs $d_i = \min_j f_{ij}(x_j)$ induces nonnegative equilibrium prices $q_i = b_i - d_i$ and the reversed matching, by the same equalities and inequalities.

Let x^* be the certified chore vector and $d_i^* = \min_j f_{ij}(x_j^*)$. Proposition 1 gives $x^* \ge x$ for every envy-free vector, hence $d_i^* \ge d_i$. Thus $q_i^* = b_i - d_i^*$ is the componentwise minimum equilibrium price. At these prices the recovered lengths equal x^*, and every equilibrium matching is certified by Lemma 6.

Given the component records, evaluating $u_{ji}(q)$ uses at most one cut query, while its inverse at utility z is $q = b_i - f_{ij}(z)$ and uses at most one evaluation. The linear extensions require only arithmetic. Computing the b_i, recovering all suffix lengths and the demand graph, and finding a perfect matching use $O(n^2)$ additional queries and polynomial work. □

For a cake instance with suffix values v_{ij}, the corresponding market has one buyer per agent and one good per piece, outside-option utility zero, and utilities

$$U_{ij}(p) = \begin{cases} v_{ij}(1-p), & 0 \le p \le 1, \\ 1-p, & p \ge 1. \end{cases}$$

Every equilibrium is fully matched and has $p_j \le 1$, by the same zero-price and outside-option arguments. It corresponds exactly to an envy-free cake suffix allocation through $s_j = 1 - p_j$, with the same matching. Lemma 10 identifies the certified vector with componentwise minimum prices. Conversely, a market supplied in this form defines cake suffix values by $v_{ij}(s) = U_{ij}(1-s)$. Evaluations and inverses translate with constant oracle overhead, and $O(n^2)$ queries recover a matching when only prices are supplied. This is a two-way oracle equivalence for the specified market class.

Applying this cake interpretation to G_{ji} from (7) gives utilities $G_{ji}(1 - p_i) = f_{ij}^{-1}(b_i - b_i p_i)$. Rescaling by $q_i = b_i p_i$ recovers the direct chore-market formula on the entire equilibrium price range. The negative-utility extensions beyond that range have no effect on equilibria. Thus the

two trimming reductions and the market reduction express the same correspondence. They do not assert a polynomial algorithm for unrestricted nonquasilinear markets, nor extend the inverse formulas to flat suffix functions. Appendix B handles zero costs for the trimming protocol itself.