

The Complexity of Exact MMS Allocations

Anonymous author

Abstract

An allocation of indivisible goods is MMS-fair if every agent receives at least her maximin share, and such allocations need not exist. We classify the complexity of exact maximin shares for additive valuations, with the values encoded in binary or in unary. With values in binary, computing a maximin share is OptP-complete and computing an MMS-fair allocation is FP^{NP} -complete, already for two agents with identical valuations. With values in binary, deciding whether an MMS-fair allocation exists is P^{NP} -complete, already for four agents holding three distinct valuations. With values in unary and the number of agents part of the input, computing a maximin share is $\text{OptP}[O(\log n)]$ -complete and computing an MMS-fair allocation is complete for Chen and Toda’s class $\text{FNP}/\text{OptP}[O(\log n)]$, again with identical valuations. In the same setting, deciding existence is $\text{P}^{\text{NP}}[O(\log n)]$ -complete. An MMS-fair allocation always exists if all agents but one share a valuation, so the two existence results need agents with differing valuations, and the two computation results do not. All four reductions share one framework, in which a common weight fixes the admissible partitions and small charges encode a formula, and each uses one of two devices for satisfiability, one for each encoding. The proofs were found, and this text was written, in interaction with Anthropic’s Claude models. The human author has checked the proofs in the main text, but not yet those in the appendix.

Keywords

fair allocation, maximin share, indivisible goods, computational complexity, satisfiability

1 Introduction

The maximin share of an agent is the value she can guarantee herself by dividing the goods into as many bundles as there are agents and receiving the worst of them [8]. An allocation is MMS-fair if every agent receives at least her maximin share. Such allocations need not exist [18], and most of the literature therefore studies approximate guarantees (see the survey of Amanatidis et al. [1]). The complementary question is to recognise the instances in which an exact MMS-fair allocation does exist. The complexity of this problem has been open since Bouveret and Lemaître [6] raised it, and Heinen et al. [16] and the survey of Nguyen and Rothe [21, Open problem 2] list it as open.

Existence. Our first main result is the following (Theorem 6.14): *deciding whether an MMS-fair allocation exists is P^{NP} -complete for additive valuations with values in binary, and hardness holds already for $n = 4$ agents with three distinct valuations, held by one, one and two agents, respectively.* The class P^{NP} is natural here: fairness asks whether some allocation reaches the maximin shares, which are

Table 1: Our results, for additive valuations. All entries are completeness results. With values in unary and a fixed number of agents, all problems are in P.

	binary	unary, n free
MMS-VALUE	$\text{OptP}(n = 2, \text{identical})$	$\text{OptP}[O(\log n)]$ (identical)
MMS-ALLOC	$\text{FP}^{\text{NP}}(n = 2, \text{identical})$	$\text{FNP}/\text{OptP}[O(\log n)]$ (identical)
MMS-EXISTS	$\text{P}^{\text{NP}}(n = 4)$	$\text{P}^{\text{NP}}[O(\log n)]$
where	Sections 5 and 6	Sections 7 and 8

themselves optima over all partitions. This gives the exact complexity of the problem under additive valuations. The strongest lower bound known so far is DP-hardness [22], from concurrent work (see below). Agents with differing valuations are needed for a reason: if all agents but one share a valuation, then an MMS-fair allocation always exists (Remark 3.6). So every reduction based on identical valuations, and every reduction to two agents, produces only yes-instances.

Computation. Two further problems ask for a number or an allocation rather than a yes or no: MMS-VALUE asks for the maximin share of a given agent, and MMS-ALLOC for an MMS-fair allocation, or the answer that none exists. With values in binary, MMS-VALUE is OptP-complete, and MMS-ALLOC is FP^{NP} -complete (Theorem 5.2), already for $n = 2$ agents with identical valuations. Identical valuations are no obstacle here: an MMS-fair allocation then always exists, but finding one is as hard as computing the optimum of an NP optimization problem.

Encodings. With values in unary, all numbers are polynomial in the input size, and for every *fixed* number of agents all problems studied here are solvable in polynomial time (Proposition 3.7). So hardness in unary needs the number of agents to grow. With the number of agents part of the input, MMS-VALUE is $\text{OptP}[O(\log n)]$ -complete, MMS-ALLOC is complete for the class $\text{FNP}/\text{OptP}[O(\log n)]$ of Chen and Toda [9] (Theorems 7.2 and 7.4), both again for identical valuations, and deciding existence is $\text{P}^{\text{NP}}[O(\log n)]$ -complete (Theorem 8.8). Table 1 summarizes the results.

Concurrent work. In independent, concurrent work, Gupta and Seetharaman [15] show that deciding whether an MMS-fair allocation exists is DP-hard for additive valuations with values in binary, and leave its exact complexity open. Theorem 6.14 shows that it is P^{NP} -complete, which strengthens their lower bound, as $\text{DP} \subseteq \text{P}^{\text{NP}}$. They also show that the problem is P^{NP} -complete for 2-additive valuations, and that for additive valuations it is coNP -hard already for three agents with values in binary, and strongly coNP -hard for agents of two types. Our $\text{P}^{\text{NP}}[O(\log n)]$ -hardness with values in unary (Theorem 8.8) implies strong coNP -hardness as well, but with $r + 2$ distinct valuations rather than two. Their three-agent result shows that our four agents are not a lower bound for hardness.



This work is licensed under a Creative Commons Attribution International 4.0 License.

Technique. All our instances give agent i the valuation $u_i(g) = K \cdot w(g) + \delta_i(g)$ for a common weight w and small charges δ_i , so that only allocations in which every bundle has the same weight matter: the weights are a hard constraint, and the charges are soft preferences among such allocations. Two devices encode satisfiability into the weights. The *halving gadget*, the standard reduction from SAT to SUBSET-SUM, splits a set of goods into two halves of equal weight exactly along the models of a formula. Its weights have polynomially many bits, so it serves the binary encoding. The *gear block* splits a set of goods into many bundles of equal weight along the models of an instance of positive 1-in-3 satisfiability. Its weights are polynomially bounded, so it also serves the unary encoding. In both, one bundle, the *ledger*, records the model, and a charge on the ledger records the model’s value.

For the computational problems, all agents share one valuation, and the maximin share is the value of the best model, shifted by a known constant. For existence, a *core* of twelve goods has exactly two partitions into bundles of equal weight, and in each of them one of four agents falls short of her maximin share, so the core alone is a switch with two positions, neither of them fair. Two *gates*, each a halving gadget, repair this conditionally (see Figure 2 for an overview): one sets the maximin share of the agent who falls short from the lexicographically largest model of a formula β , and the other sets the best bundle she can be given from that of a formula α (Section 6.1 makes this precise). An MMS-fair allocation then exists if and only if the second amount covers the gap, which is a comparison between the lexicographic maxima of α and β , and taking $\beta = \varphi$ and $\alpha = \varphi \wedge z_k$ yields ODD-LEX-MAX-SAT. With values in unary a gate can carry only one bit. Instead, a core with $2m$ agents carries many agents of this kind, each with two gates on one shared gear block, and existence becomes a disjunction over them of the conditions “ α_e is satisfiable and β_e is not”. By a criterion of Wagner [26], this is $P^{NP}[O(\log n)]$ -hard.

Roadmap. Section 2 discusses related work. Section 3 defines the problems and the complexity classes, proves membership for all of them at once, and collects the source problems of our reductions, with the known lemmas about them that we use. Section 4 sets up the common framework: weights, charges and digit blocks, graded values, the two devices (the halving gadget and the gear block), and the two endgames. Sections 5–8 prove the four hardness results, in the order binary computation, binary existence, unary computation, unary existence. This order uses each device first in its simplest form: the halving gadget on its own in Section 5 before it is attached to the core in Section 6, and the gear block on its own in Section 7 before it is attached to the core in Section 8. Each of these sections opens with a proof strategy that introduces its vocabulary and says where each part is proved. Proofs not given in the main text are in Appendix A, and Appendix B lists the notation. Section 9 describes how AI tools were used to obtain these results and to write this paper, and how far the proofs have been checked.

2 Related Work

Fair allocation of indivisible goods was surveyed by Bouveret, Chevaleyre and Maudet [4] and, for approximate fairness, by Amnatidis et al. [1]. Nguyen and Rothe [21] surveyed its complexity.

Maximin shares. Bouveret and Lemaître [6] showed that computing the maximin share is NP-hard already for two agents with identical additive valuations, that deciding whether an MMS-fair allocation exists is NP-hard for 2-additive valuations, and that it lies in Σ_2^P . Aziz, Rauchecker, Schryen and Walsh [3] showed that computing an MMS-fair allocation of goods is strongly NP-hard, and NP-hard already for two agents, for identical additive valuations, for which one always exists. For MMS-VALUE and MMS-ALLOC, our results strengthen these NP-hardness results to completeness for OptP and FP^{NP} with values in binary, and for OptP[$O(\log n)$] and FNP//OptP[$O(\log n)$] with values in unary. Heinen, Nguyen, Nguyen and Rothe [16] studied maximin, proportional and minimax share allocations, and showed, among other results, that deciding whether a minimax share allocation exists is NP-hard for additive valuations. They left the complexity of MMS existence open. Lonn and Truszczynski [20] showed that MMS existence lies in P^{NP} , also for goods on a graph whose bundles must be connected, and that it lies in NP for unicyclic graphs. Theorem 6.14 shows that their upper bound is tight. For identical agents, the maximin share is the optimum of max-min machine scheduling, which has a polynomial-time approximation scheme [27]. Our results on MMS-VALUE concern its exact value. Kurokawa, Procaccia and Wang [18] showed that MMS-fair allocations need not exist for three or more agents with additive valuations, and Feige, Sapir and Tauber [12] gave a smallest counterexample, with three agents and nine goods. Such counterexamples are a precondition for the hardness of MMS-EXISTS: if an MMS-fair allocation always existed, every instance would be a yes-instance and the problem would be trivial. For two agents, and more generally whenever all agents but one share a valuation, one always exists (Remark 3.6), so every no-instance, and with it every hard instance, needs at least three agents and no valuation held by all agents but one.

Envy-freeness and Pareto-optimality. The other fairness question known to lie above NP combines envy-freeness with Pareto-optimality. Bouveret and Lang [5] showed that deciding whether an envy-free and Pareto-optimal allocation exists is Σ_2^P -complete under logical preferences, de Keijzer, Bouveret, Klos and Zhang [11] showed the same for additive valuations, and in concurrent work Bu and Tao [7] showed that this persists for a constant number of agents with additive valuations. By Theorem 6.14, MMS existence is P^{NP} -complete for additive valuations, so it is not Σ_2^P -hard unless $\Sigma_2^P = P^{NP}$. The concurrent work of Gupta and Seetharaman [15] on maximin shares is discussed in the introduction.

3 Preliminaries

A *fair allocation instance* is a triple $(N, \mathcal{G}, \mathcal{U})$ where $N = \{1, \dots, n\}$ is a set of *agents*, $\mathcal{G} = \{g_1, \dots, g_m\}$ is a set of (indivisible) *goods*, and $\mathcal{U} = \{u_1, \dots, u_n\}$ is a set of *utility functions*. For each agent $i \in N$, their utility function $u_i: 2^{\mathcal{G}} \rightarrow \mathbb{Z}_{\geq 0}$ maps a *bundle* $S \subseteq \mathcal{G}$ of goods to a nonnegative integer $u_i(S)$ called the *value* of bundle S . Utility functions are *additive*: for every $i \in N$ and $S \subseteq \mathcal{G}$ it holds that $u_i(S) = \sum_{g \in S} u_i(\{g\})$, and we abbreviate $u_i(\{g\}) = u_i(g)$. The agents have *identical valuations* if $u_i = u_j$ for all $i, j \in N$.

Encodings. An instance is given by the numbers $u_i(g)$. It is given *in binary* if these numbers are written in binary, and *in unary* if they

are written in unary, so that the length of an instance given in unary is at least the sum of all values $u_i(g)$. The number of agents is *fixed* if a problem is restricted to instances with a constant number n of agents, and *free* if n is part of the input.

An *allocation* π is a partition of goods among the agents, that is, a mapping $\pi: \mathcal{N} \rightarrow 2^{\mathcal{G}}$ from agents to bundles with $\pi(i) \cap \pi(j) = \emptyset$ for all $i, j \in \mathcal{N}, i \neq j$. An allocation π is *complete* if $\bigcup_{i \in \mathcal{N}} \pi(i) = \mathcal{G}$, that is, each good is allocated to some agent. In the remainder, allocations are assumed to be complete.

We define the maximin share of an agent $i \in \mathcal{N}$ as

$$\mu_i = \max_{\pi \in \Pi_{\mathcal{N}, \mathcal{G}}} \min_{j=1}^n u_i(\pi(j))$$

where $\Pi_{\mathcal{N}, \mathcal{G}}$ is the set of all allocations of $\mathcal{G}$ among $\mathcal{N}$. As this does not depend on which agent receives which bundle, we also call an allocation a *partition* (into n bundles) when we compute maximin shares, and write partitions as π as well. If the agents have identical valuations, we write μ for their common maximin share, and call a partition μ -*optimal* if all its bundles are worth at least μ . An allocation π is *MMS-fair* if for all agents $i \in \mathcal{N}$ it holds that $u_i(\pi(i)) \geq \mu_i$. Since some bundle of every partition is worth at most the average, $\mu_i \leq u_i(\mathcal{G})/n$. If the valuations are identical, the MMS-fair allocations are exactly the μ -optimal partitions.

3.1 Problems

In MMS-VALUE, one is given an instance and an agent i , and the task is to output μ_i . In MMS-ALLOC, one is given an instance, and the task is to output an MMS-fair allocation, or the symbol $\perp$ if none exists. In the decision problem MMS-EXISTS, one is given an instance, and the task is to decide whether it admits an MMS-fair allocation. In general, a *search problem* is a relation $\Pi \subseteq \Sigma^* \times \Sigma^*$. For an input x , we let $\text{Sol}_{\Pi}(x) = \{y : (x, y) \in \Pi\}$ if this set is non-empty, and $\text{Sol}_{\Pi}(x) = \{\perp\}$ otherwise, and we write $\text{Sol}(x)$ when Π is clear. An algorithm *solves* Π if on every input x it outputs some element of $\text{Sol}_{\Pi}(x)$. We identify a function f with the search problem $\{(x, f(x))\}$, so that $\text{Sol}_f(x) = \{f(x)\}$. For MMS-ALLOC, $\text{Sol}(I)$ is the set of correct outputs defined above. Each problem comes in two versions, with the instances given in binary or in unary.

3.2 Complexity classes and reductions

We assume the reader to be familiar with standard notions from computational complexity theory (e.g., the classes P and NP, and polynomial-time many-one reductions) [2]. The class P^{NP} , also known as Δ_2^{P} [25], consists of all decision problems that can be solved by a polynomial-time algorithm with access to an NP oracle, and $\text{P}^{\text{NP}}[O(\log n)]$, also known as Θ_2^{P} [26], of those that can be solved in this way with $O(\log n)$ oracle queries on inputs of length n . The class DP [22] consists of the intersections $L_1 \cap L_2$ of a language $L_1 \in \text{NP}$ and a language $L_2 \in \text{coNP}$, and Σ_2^{P} [25] of the decision problems solvable in nondeterministic polynomial time with access to an NP oracle. We have $\text{NP} \cup \text{coNP} \subseteq \text{DP} \subseteq \text{P}^{\text{NP}}[O(\log n)] \subseteq \text{P}^{\text{NP}} \subseteq \Sigma_2^{\text{P}}$. The class FP^{NP} consists of all functions that can be computed by a polynomial-time algorithm with access to an NP oracle, and $\text{FP}^{\text{NP}}[O(\log n)]$ of those computed in this way with

$O(\log n)$ oracle queries. For optimization we follow Krentel [17] and Gasarch, Krentel and Rappoport [14].

Definition 3.1. An *NP metric machine* is a nondeterministic machine that runs in polynomial time and on every branch halts and writes a natural number in binary. A function f is in OptP if $f(x)$ is, for every x , the largest number written on a branch of some NP metric machine on input x , and it is in $\text{OptP}[O(\log n)]$ if in addition $f(x)$ has $O(\log |x|)$ bits.

Definition 3.2. A search problem Π *reduces* to a search problem Π' if there are polynomial-time functions T_1, T_2 such that $T_2(x, y) \in \text{Sol}_{\Pi}(x)$ for every x and every $y \in \text{Sol}_{\Pi'}(T_1(x))$.

By the convention for Sol , this says that (A) T_2 maps every solution of $T_1(x)$ to a solution of x , and (B) $T_1(x)$ has a solution whenever x has one, as otherwise $T_2(x, \perp)$ would have to be a solution of x . Concretely, if $T_1(x)$ had no solution, a solver for Π' would answer $\perp$ on it, and T_2 would have nothing to decode. So (B) is what lets the reductions of Theorems 5.2 and 7.4 say that $\perp$ is never a correct output on their instances. Hence every algorithm solving Π' gives one solving Π , and reductions compose. The quantifier “every” matters: a solver for Π' may return any element of $\text{Sol}_{\Pi'}(T_1(x))$, so T_2 must decode each of them. Unless said otherwise, $T_2(x, \perp) = \perp$. A search problem is in FP^{NP} if some polynomial-time algorithm with access to an NP oracle solves it.

Convention. Throughout the paper, hardness and completeness for a class of functions or search problems (OptP , $\text{OptP}[O(\log n)]$, FP^{NP} , $\text{FP}^{\text{NP}}[O(\log n)]$ and $\text{FNP}/\text{OptP}[O(\log n)]$ below) are with respect to the reductions of Definition 3.2, unless stated otherwise. For a class of decision problems (NP , P^{NP} and $\text{P}^{\text{NP}}[O(\log n)]$), they are with respect to polynomial-time many-one reductions.

Metric and Levin reductions. Two familiar notions are special cases of Definition 3.2. If $\Pi = f$ and $\Pi' = g$ are functions, then $\text{Sol}_{\Pi'}(T_1(x)) = \{g(T_1(x))\}$, and the definition says that $f(x) = T_2(x, g(T_1(x)))$ for all x . This is Krentel’s *metric reduction* [17]. If only $\Pi = f$ is a function, it says that $f(x) = T_2(x, y)$ for every $y \in \text{Sol}_{\Pi'}(T_1(x))$. A *Levin reduction* [2, 19] between search problems with polynomial-time checkable solutions is a triple (T_1, T_2, T_3) of polynomial-time functions such that T_1 is a many-one reduction between their decision versions, $T_2(x, y)$ is a solution of x for every solution y of $T_1(x)$, and $T_3(x, y)$ is a solution of $T_1(x)$ for every solution y of x . So T_2 gives (A), and T_3 witnesses (B). With $T_2(x, \perp) = \perp$, the pair (T_1, T_2) is a reduction in the sense of Definition 3.2. The converse fails, as (B) asks only that $T_1(x)$ has a solution, not that one can be computed from a solution of x . Hence completeness results under metric reductions, such as Krentel’s for OptP below, and under Levin reductions, such as the Cook–Levin theorem for SAT [10, 19], carry over to Definition 3.2. Our gadgets follow the pattern of a Levin reduction: the Cook–Levin construction (Lemma 3.10), Lemma 3.11, the halving gadget (Lemma 4.4) and the gear block (Lemma 4.5) come with polynomial-time maps in both directions between the models of the source formula and the solutions of the gadget, as the proofs of their converse parts are constructive. Our main reductions are not Levin reductions, and need not be: their sources are functions in OptP or $\text{OptP}[O(\log n)]$, search problems in $\text{FNP}/\text{OptP}[O(\log n)]$, or decision problems in P^{NP}

or $P^{NP}[O(\log n)]$, none of which comes with solutions that can be checked in polynomial time, and Definition 3.2 asks only for (B), not for a map T_3 .

THEOREM 3.3 ([14, 17]). *A function that is complete for OptP under metric reductions is FP^{NP} -complete, and a function that is complete for OptP[$O(\log n)$] under metric reductions is $FP^{NP}[O(\log n)]$ -complete.*

Both completeness notions in Theorem 3.3 are under metric reductions, a special case of Definition 3.2, so a function that is OptP-complete under metric reductions is complete for OptP and for FP^{NP} in the sense of our convention. By composition, a search problem to which every function in OptP reduces is FP^{NP} -hard. With values in unary, MMS-ALLOC is not captured by these classes, but by a class of Chen and Toda [9].

Definition 3.4 ([9]). The class $FNP//OptP[O(\log n)]$ consists of the partial multi-valued functions F for which there are a polynomial-time nondeterministic transducer T and a function $g \in OptP[O(\log n)]$ such that, for every x , $F(x)$ is the set of strings that T outputs on its accepting branches on input $\langle x, \text{bin}(g(x)) \rangle$. We view F as the search problem $\{(x, y) : y \in F(x)\}$, so that $Sol_F(x) = F(x)$ if $F(x) \neq \emptyset$, and $Sol_F(x) = \{\perp\}$ otherwise. A search problem Π is in this class if some such F solves it: for every x , $F(x) = \emptyset$ if x has no solution in Π , and $\emptyset \neq F(x) \subseteq Sol_\Pi(x)$ otherwise.

In Definition 3.4, a string y belongs to $F(x)$ if *some* accepting branch of T outputs it. So even when $g(x)$ is known, checking whether $y \in F(x)$ is an NP question: it asks for a branch. (This matters only in Section 7.4.) Our hardness proof for MMS-ALLOC with values in unary (Theorem 7.4) needs solutions that can be checked in polynomial time once $g(x)$ is known. It writes the check as a formula by the Cook–Levin construction (Lemma 3.10 below). Its reduction obtains such solutions by putting the branch into the solution: it works with pairs (y, ω) of an output y and an accepting branch ω that outputs it, and its second map discards ω (Section 7.4). Chen and Toda’s reductions [9, Def. 2.2] ask in addition that $T_1(x)$ always has a solution, and ours in Theorem 7.4 does.

In words, $FNP//OptP[O(\log n)]$ consists of the search problems whose solutions can only be checked once a small optimum is known: first compute a number of $O(\log n)$ bits, itself the optimum of an NP metric machine, and then the solutions become verifiable. MMS-ALLOC with values in unary is the model example: an allocation cannot be checked against the maximin shares until the maximin shares are known. Membership is read in the same way as for FP^{NP} : the problem must be solved, not its solution relation decided. Problems in the class need not have solutions for every x . On such x , $Sol_F(x) = \{\perp\}$, so a reduction from F must output $\perp$.

3.3 Membership

Membership is proved once, for all problems and both encodings.

PROPOSITION 3.5. *For arbitrary n and arbitrary additive valuations:*

- (i) *MMS-VALUE is in OptP, and in OptP[$O(\log n)$] with values in unary,*
- (ii) *MMS-ALLOC is in FP^{NP} , and with values in unary it is in $FNP//OptP[O(\log n)]$,*

- (iii) *MMS-EXISTS is in P^{NP} , and with values in unary they are in $P^{NP}[O(\log n)]$.*

The proof is in the appendix. With values in binary, it computes each μ_i by binary search. With values in unary, that would cost too many queries, as there are n agents. Instead, an NP metric machine guesses n partitions and an allocation, and writes $2 \sum_i \ell_i + [\text{the allocation gives every agent } i \text{ at least } \ell_i]$, where ℓ_i is the value to agent i of the worst bundle of the i -th partition. The maximum is attained only when every ℓ_i equals μ_i , so its parity says whether an MMS-fair allocation exists. With values in unary, this number has $O(\log |I|)$ bits. Lemma 7.3 uses the same device: a low bit guarding a witness under a high-order optimum.

Hardness requires agents with differing valuations, for the following reason.

Remark 3.6. If all agents but one have the same valuation u , then an MMS-fair allocation exists [12, Prop. 7]: take a partition maximizing $\min_j u(\pi(j))$, let the remaining agent i pick her favourite bundle, which is worth at least $u_i(\mathcal{G})/n \geq \mu_i$ to her, and give the other bundles to the other agents. In particular, this holds for identical valuations and for $n = 2$ agents. So a hardness proof for MMS-EXISTS must not give any valuation to $n - 1$ of the n agents, and every reduction based on identical valuations or on two agents produces only yes-instances. Our construction in binary uses $n = 4$ agents with three distinct valuations, held by one, one and two agents, respectively (Section 6), and the one in unary uses $2m$ agents of two types, each type held by $m \geq 2$ agents (Section 8). The computational problems are not affected: for them, Sections 5 and 7 use identical valuations.

For a fixed number of agents and values in unary, all these problems are easy. We prove this in the appendix by a dynamic programme over the vectors of bundle values.

PROPOSITION 3.7. *For every fixed n , with values in unary, MMS-VALUE, MMS-ALLOC and MMS-EXISTS are solvable in polynomial time.*

So in Sections 7 and 8 the number of agents must grow with the input.

3.4 Source problems and interface lemmas

Nothing in this section is new. Its lemmas are known, or follow directly from standard constructions. We state them only to fix our notation and the exact form in which Sections 5–8 use them. Where a proof is needed to obtain that form, it is standard, and we include it in the appendix for completeness.

Let φ be a CNF formula over the variables $z_1, \dots, z_k$ with clauses $C_1, \dots, C_q$, where $k, q \geq 1$, none of the clauses empty, and where we view each clause as a set of literals, so that its width ℓ_c is its number of distinct literals. For an assignment σ , we write $\text{val}(\sigma) = \sum_{i: \sigma(z_i)=1} 2^{k-i}$, $\Delta = 2^k - 1$, and $\text{lex}(\varphi) = \max_{\sigma \models \varphi} \text{val}(\sigma)$ if φ is satisfiable. Consequently, $0 \leq \text{val}(\sigma) \leq \Delta$ and $\text{lex}(\varphi)$ is the value of φ ’s lexicographically largest model, where z_1 is the most significant variable. In the decision problem ODD-LEX-MAX-SAT, one is given a satisfiable CNF formula φ as above, and the task is to decide whether φ ’s lexicographically largest model sets z_k to 1, that is, whether $\text{lex}(\varphi)$ is odd. In SAT, one is given a CNF formula and asked whether it is satisfiable.

LEMMA 3.8 ([17]). *ODD-LEX-MAX-SAT is P^{NP} -complete.*

Krentel [17, Thm. 3.4] states this without proof and without the restriction to satisfiable formulas, which our reduction needs. The standard proof produces satisfiable formulas, and we include it in the appendix for completeness.

Our reduction compares two lexicographic maxima.

LEMMA 3.9. *Let φ be a satisfiable CNF formula over $z_1, \dots, z_k$, and let $\beta = \varphi$ and $\alpha = \varphi \wedge z_k$. Then $\varphi \in \text{ODD-LEX-MAX-SAT}$ if and only if α is satisfiable and $\text{lex}(\alpha) \geq \text{lex}(\beta)$.*

The computational problems start from an NP metric machine instead. We call a CNF formula a 3-CNF if every clause has exactly three literal occurrences, where a literal may occur more than once. The next lemma is the Cook–Levin construction [10], applied to metric machines as by Krentel [17].

LEMMA 3.10 ([10, 17]). *Let M be an NP metric machine that writes exactly $p = p(|x|)$ bits on every branch on input x . There is a polynomial-time map from x to a 3-CNF formula φ over variables $z_1, \dots, z_k$ with $k \geq p$, the output bits $z_1, \dots, z_p$ being the first, such that the output bits of every model of φ , read with z_1 most significant, are the number written on some branch of M on input x , and every number written on a branch arises in this way. Moreover, for every polynomial-time relation R with polynomially bounded third argument, there is a polynomial-time map from x to a 3-CNF formula η_b over $z_1, \dots, z_p$, variables $y_1, \dots, y_{p'}$, a variable b and further variables, such that an assignment to the z_i , the y_j and b extends to a model of η_b if and only if $b = 0$ or $R(x, c, y)$ holds, where c is the number with bits $z_1, \dots, z_p$ and y the string $y_1 \dots y_{p'}$.*

For the second part, the Cook–Levin construction gives a 3-CNF formula η with the same property without b . Adding $\neg b$ to each clause $(l_1 \vee l_2 \vee l_3)$ of η and splitting the result into $(\neg b \vee l_1 \vee u) \wedge (\neg u \vee l_2 \vee l_3)$ with a fresh variable u gives η_b , as the two clauses have a common model extending an assignment to the other variables if and only if $b = 0$ or one of l_1, l_2, l_3 is true.

As every branch of M writes a number, φ is satisfiable. The unary constructions start from positive 1-in-3 satisfiability. A *positive exactly-1 instance* ψ is a finite family of clauses, each a set of at least two distinct variables, and an assignment that makes exactly one variable of every clause true is an *exactly-1 model* of ψ , and ψ is satisfiable if it has one. In POS-1-IN-3-SAT, one is given a positive exactly-1 instance whose clauses have three variables each, and the task is to decide whether it is satisfiable. The problem is NP-complete [13, 23]. Clauses of other sizes arise only in Section 8, where the wildcard (Definition 8.3) adds a fourth variable to every clause. We use it instead of SAT because the gear block of Section 4.5 needs clauses that are equations, not inequalities: exactly one variable of every clause is true. The standard reduction from SAT via Schaefer’s gadgets gives more than NP-hardness. We state it in the form we use, in which models translate in both directions and the variables whose values we grade keep their names.

LEMMA 3.11 (CF. [23]). *There is a polynomial-time map from a 3-CNF formula φ over $z_1, \dots, z_k$ to a POS-1-IN-3-SAT instance ψ whose variables include $z_1, \dots, z_k$, such that (a) every variable of ψ occurs in some clause of ψ , (b) the restriction to $z_1, \dots, z_k$ of every exactly-1 model of ψ is a model of φ , and (c) every model of φ arises as such a restriction.*

For completeness, we give the proof with Schaefer’s gadgets [23] in the appendix. The unary existence result uses a criterion of Wagner for $P^{NP}[O(\log n)]$ -hardness.

LEMMA 3.12 ([26]). *Let A be an NP-complete language, B a language, and f a polynomial-time function such that for every $k \geq 1$ and all $x_1, \dots, x_{2k}$ with $x_{j+1} \in A \Rightarrow x_j \in A$ for each $j < 2k$, the number $|\{j : x_j \in A\}|$ is odd if and only if $f(x_1, \dots, x_{2k}) \in B$. Then B is $P^{NP}[O(\log n)]$ -hard.*

Remark 3.13. An optimum of $O(\log n)$ bits can be located by asking, for each j , whether some branch writes at least j . Once it is located, one more NP question at that level asks for a solution. This is the standard way to use an optimum of $O(\log n)$ bits, and Section 7 realises both steps in a single number with one extra bit (Lemma 7.3).

4 The Framework

This section sets up what all four constructions share. We first describe it informally, and Sections 4.2–4.6 give the precise statements.

4.1 Overview

Weights and charges. In all our instances, an agent values a good by a common *weight*, scaled by a large factor K , plus a small *charge* of her own. The weights act as a hard constraint: a bundle that is one unit of weight short loses more than any charge can make up, so only the *balanced* partitions, in which all bundles have the same weight, matter, both for the maximin shares and for the allocations that meet them. The charges act as soft preferences among the balanced partitions. An agent’s maximin share is then a common value minus her *drop*, which is how far below this value her worst bundle lies in the balanced partition that is best for her. So the weights decide which partitions are possible, and the charges decide who is happy with them.

Weights from digit blocks. Each constraint on the weights is a *digit block*: a number for each good, such that every bundle must receive the same share of the block’s total. Several blocks are combined into one weight by writing them as the digits of a number in a large enough base, so that one test on the weight checks every block at once. A block ignores the goods on which it is zero, so we build each gadget from its own blocks, and two gadgets constrain each other only through the goods that both of them see. We will make each such shared good explicit, and a balanced partition is then glued together from one partition of each gadget.

Recording a model. To encode a formula, we need a gadget whose balanced partitions correspond to its models. The *halving gadget* is the standard reduction from SAT to SUBSET-SUM, read as the problem of splitting a set of goods into two halves of equal weight. The half that does not contain a designated good, the *anchor*, is a *ledger*, which records a model of the formula. A charge on the literal goods then reads off the value of that model, and a *grading* of the variables says how much each variable contributes to it. The halving gadget is the device for the binary encoding. Its weights are exponential, so the unary encoding needs a second device, the *gear block*, which has polynomial weights but many bundles: one

ledger, and many small bundles that each hold goods of a single gadget.

Two endgames. Every construction ends in one of two ways. For the computational problems, all agents share one valuation, and the maximin share is the value of the best model, shifted by a known constant. The *graded ledger* (Lemma 4.7) states what a construction must provide for this, and turns it into reductions for all three computational problems at once. For existence, the agents differ, and a *core* of goods on which no allocation is fair is repaired conditionally. Its contract, the *calculus*, is stated in Section 6.2 and sketched in Section 4.6.

Where to find what. Section 4.2 defines weights, charges and digit blocks, and shows that only balanced partitions matter (Lemma 4.3). Section 4.3 defines graded values, Section 4.4 the halving gadget and its charges (Lemma 4.4), Section 4.5 the gear block (Lemma 4.5), and Section 4.6 the graded ledger (Lemma 4.7 and Corollary 4.8) and an overview of the calculus.

4.2 Weights, charges and balance

All our instances are built from integer weights, and the agents' valuations add small charges to these weights. For a function δ on goods and a bundle B , we write $\delta(B) = \sum_{g \in B} \delta(g)$. Throughout, n is the number of agents of the instance under construction, and a partition of the goods has n bundles. We combine several constraints into one weight by the following lemma. A *digit block* on a set of goods $\mathcal{G}$ assigns each good g a digit $c(g) \in \mathbb{Z}_{\geq 0}$. Its *total* is $T = c(\mathcal{G})$, and a bundle S *meets* the block if $c(S) = T/n$.

LEMMA 4.1. *Let $c_1, \dots, c_r$ be digit blocks on $\mathcal{G}$ whose totals $T_1, \dots, T_r$ are divisible by n . Let $\rho_j = T_j + 1$, $W_r = 1$ and $W_j = \rho_{j+1}W_{j+1}$ for $j < r$, and let $w(g) = \sum_j W_j c_j(g)$ and $\tau = \sum_j W_j T_j/n$. Then $w(\mathcal{G}) = n\tau$, and $w(S) = \tau$ if and only if S meets every block.*

Each digit block is an independent constraint, *every bundle receives an n -th of this block*, and the radices are large enough that no digit sum carries into the next digit, so the single test $w(S) = \tau$ checks all constraints at once. For example, with $n = 4$ and two blocks of totals 4 and 12, we get $\rho_2 = 13$, $w(g) = 13c_1(g) + c_2(g)$ and $\tau = 13 \cdot 1 + 3$, and $w(S)$ is the two-digit number $(c_1(S), c_2(S))$ in base 13, which equals $(1, 3)$ if and only if S meets both blocks. This lets us design the gear block, the core and the gates below block by block.

We call the set of goods on which a block has a positive digit its *support*, and we say that the block *sees* these goods. Two blocks constrain each other only through goods that both see. To make this precise, for a set F of blocks with union of supports $\mathcal{G}_F$, we call a partition of $\mathcal{G}_F$ into n bundles that all meet every block of F a *configuration* of F .

LEMMA 4.2. *Let $F_1, \dots, F_h$ be a partition of the digit blocks of Lemma 4.1, with unions of supports $\mathcal{G}_1, \dots, \mathcal{G}_h$ covering $\mathcal{G}$. A partition π of $\mathcal{G}$ is balanced if and only if its restriction to each $\mathcal{G}_l$ is a configuration of F_l . Conversely, given a configuration π_l of each F_l and bijections f_l from the bundles of π_l to $\{1, \dots, n\}$, the sets $\pi(j) = \bigcup_l f_l^{-1}(j)$ form a balanced partition if and only if, for every good $g \in \mathcal{G}_l \cap \mathcal{G}_{l'}$, the bundles of π_l and $\pi_{l'}$ containing g are mapped to the same*

index. In particular, if $\mathcal{G}_l$ and $\mathcal{G}_{l'}$ are disjoint, the two configurations can be combined by any bijection.

So, as a block ignores goods outside its support, gadgets built from disjoint families of blocks interact only through the goods they share, and we will make each such shared good explicit. The existence proofs use this to attach gadgets to a core (Sections 6.4 and 8), while the computational proofs need only one gadget. When a balanced partition π is combined in this way, we say that a bundle $f_l^{-1}(j)$ of π_l *lies in* the bundle $\pi(j)$, and that bundles of π_l and $\pi_{l'}$ *lie together* if they lie in the same bundle of π . So the second part of Lemma 4.2 says that two bundles containing the same shared good must lie together. The weights of Lemma 4.1 are less than $\prod_j \rho_j$, so they have at most $\sum_j \log_2 \rho_j$ bits. In particular, if there are a constant number of blocks and every total is polynomial in some parameter, then so are all weights. Our instances have the following form.

CONSTRUCTION. *Let $\mathcal{G}$ be a set of goods with positive integer weights w and $w(\mathcal{G}) = n\tau$, and let $\delta_1, \dots, \delta_n: \mathcal{G} \rightarrow \mathbb{Z}$ be charges with $\delta_i(\mathcal{G}) = 0$ and $|\delta_i(S)| \leq M$ for all i and all $S \subseteq \mathcal{G}$. Let $I(\delta_1, \dots, \delta_n)$ be the instance with agents $\{1, \dots, n\}$ and goods $\mathcal{G}$ in which agent i has the valuation $u_i(g) = K \cdot w(g) + \delta_i(g)$, for $K = 2M + 1$.*

We call δ_i the *charge* of agent i : all agents value the goods by the common weight w , scaled by K , and the charge is the small, agent-specific part on top of it, and M is a bound on how much a charge can add to or take from a bundle. Valuations are positive, as $K > M \geq |\delta_i(g)|$. We call a partition of $\mathcal{G}$ into n bundles *balanced* if every bundle has weight τ . In a balanced partition, bundle S is worth $K\tau + \delta_i(S)$ to agent i . We call $\delta_i(S)$ its *offset* for i , as it measures how far the bundle is from the common value $K\tau$ for her. The *drop* of agent i is $d_i = -\max_{\pi} \min_j \delta_i(\pi(j))$, where π ranges over the balanced partitions: if she divides the goods herself into a balanced partition, her worst bundle lies at least d_i below $K\tau$, and she can achieve exactly d_i . As the n offsets sum to $\delta_i(\mathcal{G}) = 0$, we have $d_i \geq 0$.

The valuations thus work on two levels. The weights are a hard constraint: a bundle that is one unit of weight short loses $K > 2M$, which no charge can make up. The charges are soft preferences, which only decide between balanced partitions. The next lemma makes this precise: both the maximin shares and the allocations that meet them live on balanced partitions, and the maximin share of agent i is $K\tau - d_i$, where $-d_i$ is the smallest offset, that is, the offset of her worst bundle, in the balanced partition that is best for her, so the drop is exactly how far her maximin share drops below $K\tau$.

LEMMA 4.3. *Let π be a partition of the goods of $I(\delta_1, \dots, \delta_n)$.*

- (i) *If $u_i(\pi(j)) \geq K\tau - M$ for some agent i and all j , or if $u_i(\pi(i)) \geq K\tau - M$ for all i , then π is balanced.*
- (ii) *If balanced partitions exist and $d_i \leq M$, then $\mu_i = K\tau - d_i$.*

4.3 Graded values

The value $\text{val}(\sigma)$ of Section 3.4 weighs the variable z_i by 2^{k-i} . This is what the existence proofs need, as they compare two lexicographic maxima, but the computational problems need other weights. A function in OptP is the largest number written by a branch of a machine, and this number sits on the few variables that

encode the machine's output bits: these must be weighed as the bits of a number, and all other variables not at all (Section 5). With values in unary, the total weight must moreover be polynomial (Section 7). We call a vector $\zeta = (\zeta_1, \dots, \zeta_k) \in \mathbb{Z}_{\geq 0}^k$ a *grading* of the variables $z_1, \dots, z_k$, and write

$$\text{val}_\zeta(\sigma) = \sum_{i=1}^k \zeta_i \sigma(z_i), \quad \Delta_\zeta = \sum_{i=1}^k \zeta_i, \quad \text{lex}_\zeta(\varphi) = \max_{\sigma \models \varphi} \text{val}_\zeta(\sigma)$$

for the *graded value* of an assignment σ , its largest possible value, and the largest graded value of a model of a satisfiable formula φ . So $0 \leq \text{val}_\zeta(\sigma) \leq \Delta_\zeta$. A grading may be 0 on some variables, which then do not count. In particular, $\text{val}_\zeta(\sigma)$ is also defined for an assignment σ to more variables than $z_1, \dots, z_k$, and only depends on the restriction of σ to $z_1, \dots, z_k$. We call $\zeta_i = 2^{k-i}$ the *default grading*, and leave out ζ for it, here and in all notation below that depends on a grading, so the default grading gives back val , Δ and lex .

4.4 The halving gadget

The halving gadget is the reduction from SAT to SUBSET-SUM, read as a halving problem: to split a set of goods X_φ into two parts of equal weight, one picks one literal good per variable and tops up every clause with slack goods, which is possible exactly when the picked literals satisfy every clause. An extra good, the anchor, tells the two parts apart, and the assignment is read off the part without it, the ledger, so called because it records the assignment. Formally, we call the goods X_φ of the next lemma the *halving gadget* of φ . For X_φ with weights w and $w(X_\varphi) = 2\tau_\varphi$, we call a subset $S \subseteq X_\varphi$ a *half* if $w(S) = \tau_\varphi$. We call a the *anchor*, a half not containing a a *ledger*, and P_i and N_i the goods of the literals z_i and $\neg z_i$, respectively. For $S \subseteq X_\varphi$, we let σ_S be the assignment with $\sigma_S(z_i) = 1$ if and only if $P_i \in S$.

LEMMA 4.4. *There is a polynomial-time computable function mapping a CNF formula φ as in Section 3.4 to a set of goods $X_\varphi = \{P_i, N_i : i \leq k\} \cup \{s_j^c : c \leq q, j < \ell_c\} \cup \{a\}$ with positive integer weights w such that $w(X_\varphi) = 2\tau_\varphi$ for an integer τ_φ , and:*

- (a) *every ledger S contains exactly one of P_i, N_i for each i , and σ_S is a model of φ ,*
- (b) *every model of φ equals σ_S for some ledger S .*

Clauses of width 1 are allowed, and such a clause has no slack goods.

Complements of halves are halves, as $w(X_\varphi) = 2\tau_\varphi$. As exactly one of every two complementary halves contains a , every half is either a ledger or the complement of one, which we call its *anchor side*, as it is the half that holds the anchor. In particular, X_φ has a half if and only if φ is satisfiable. We record two properties of the construction in the proof.

- (W1) All weights, and τ_φ , are multiples of $b = 2\ell + 1 \geq 3$, where $\ell = \max_c \ell_c$. The weights are written in base b , and $b > 2\ell$ ensures that adding weights never carries from one digit into the next.
- (W2) All weights, and τ_φ , are less than b^{k+q+1} , so they have $O((k+q) \log b)$ bits.

The weights are thus exponential in the size of φ , but their bit length is polynomial.

Next, we define the charges that let an agent read off which assignment a half encodes. For a grading ζ , we define two charges $X_\varphi \rightarrow \mathbb{Z}$, the *lex charge* and its *negative form*, by

$$\begin{aligned} \chi_{\text{lex}}^\zeta(P_i) &= \zeta_i, & \chi_{\text{lex}}^\zeta(a) &= -\Delta_\zeta, & \text{and 0 on all other goods,} \\ \chi_{\text{lex}}^{-,\zeta}(N_i) &= -\zeta_i, & \chi_{\text{lex}}^{-,\zeta}(a) &= \Delta_\zeta, & \text{and 0 on all other goods.} \end{aligned}$$

Following our convention, χ_{lex} and χ_{lex}^- are the charges for the default grading. Both sum to 0 over X_φ , so the anchor side of a ledger always carries the opposite charge of the ledger. By Lemma 4.4(a), a ledger S contains P_i exactly for the variables that σ_S sets to 1, and N_i exactly for those it sets to 0, so the charges are:

	ledger S	anchor side $X_\varphi \setminus S$
χ_{lex}^ζ	$\text{val}_\zeta(\sigma_S)$	$-\text{val}_\zeta(\sigma_S)$
$\chi_{\text{lex}}^{-,\zeta}$	$-(\Delta_\zeta - \text{val}_\zeta(\sigma_S))$	$\Delta_\zeta - \text{val}_\zeta(\sigma_S)$

Under either charge, the best ledger for the agent is that of a model of largest graded value. So, by Lemma 4.4(b), if φ is satisfiable, the largest charge of a ledger is $\text{lex}_\zeta(\varphi)$ under the lex charge and $-(\Delta_\zeta - \text{lex}_\zeta(\varphi))$ under its negative form.

The two forms differ in which half is the worse one. Under the negative form, the ledger carries at most 0 and its anchor side at least 0, so the ledger is the worse half, and a maximin share, which is determined by the worst bundle of a partition, reads off the best model. Section 5 and the drop gate of Section 6 use it in this way. Under the lex charge, the ledger is the better half, so it can be handed to an agent on top of another bundle, to raise its value, and the relief gate of Section 6 uses it in this way. The grading changes only the charges, and never the weights.

Interface: the rest of the paper uses the halving gadget only through Lemma 4.4, properties (W1) and (W2), and this table.

4.5 The gear block

With values in unary, the halving gadget is too long: by (W2), its weights have $\Theta((k+q) \log b)$ bits, so written in unary they are exponentially long. Sections 7 and 8 therefore need a device built from a bounded number of digit blocks with polynomial totals, whose weights are then polynomial by Lemma 4.1. The gear block is such a device. It pays for its small weights with many bundles: instead of one ledger and its complement, it has one ledger and many small bundles, each of which holds goods of a single gadget.

The gear block's source problem is positive exactly-1 satisfiability (Section 3.4), with clauses of any size at least two. We use it instead of SAT because of how the block checks a clause. Every occurrence of a variable in a clause C leaves one edge good over, of one kind if the variable is true and of another if it is false, and C has a fixed set of bundles to absorb these goods: one that accepts only the kind left by a true variable, and $|C| - 1$ that accept only the kind left by a false one (Figure 1). So the block fixes the *number* of true variables in every clause: a clause is an equation, not an inequality. A clause of SAT can be satisfied by one, two or three true literals, and the halving gadget accommodates this with slack goods, one digit per clause. Positive exactly-1 clauses need no slack, and no negated literals either. Lemma 3.11 translates a 3-CNF formula into such clauses, keeping its models.

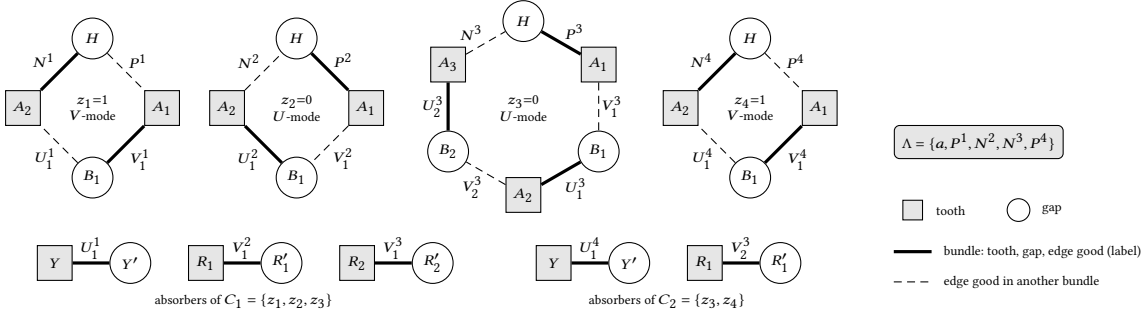


Figure 1: The gear block on the single gadget $\psi = \{C_1, C_2\}$ with $C_1 = \{z_1, z_2, z_3\}$ and $C_2 = \{z_3, z_4\}$, at $n = 15$ bundles, in the configuration in which z_1 and z_4 are true. The gadget index is suppressed throughout. Squares are teeth, circles are gaps, and the labels on the edges are edge goods. Each variable owns a gear, drawn as the cycle $H - A_1 - B_1 - \dots - A_{k+1} - H$ that alternates between gaps and teeth, with an edge good riding on each of its edges: a 4-cycle for z_1, z_2 and z_4 , which occur once, and a 6-cycle for z_3 , which occurs in both clauses. A thick edge is a bundle: part (i) of Lemma 4.5 says that every bundle but the ledger is one tooth, one gap and one edge good of a single gadget, and an even cycle has exactly two perfect matchings (the triples T1-T4 of Lemma A.1), which are the two truth values: U -mode, which takes P and the U -copies, for false, and V -mode, which takes the V -copies and N , for true. Part (iv) puts the terminal not in the ledger with the hub H , so the matching decides which terminal is left over, and part (ii) collects those, one per gear, into the ledger Λ beside the anchor a . The dashed edge goods also have to go somewhere, and the absorbers are the only bundles left: each clause C has one U -absorber, the edge $Y - Y'$, which accepts only a U -copy of C , and $|C| - 1$ V -absorbers, drawn as the edges $R_i - R'_i$, which accept only V -copies of C . A true variable leaves a U -copy of each of its clauses, and a false one a V -copy, so exactly one variable of each clause is true, which is part (iii). The two occurrences of z_3 cannot disagree: its gear turns as a whole, so it leaves V -copies of both clauses, as here, or U -copies of both. The digits are in Table 7 in the appendix.

Vocabulary. The gear block is built from a list $\psi_1, \dots, \psi_\Gamma$ of positive exactly-1 instances, in each of which every variable occurs in some clause. We call each ψ_g a *gadget*, and take their variable sets to be disjoint. We write n_v, n_c and n_o for the total numbers of variables, clauses and variable occurrences of all gadgets. A *gear* is a pair (g, z) of a gadget g and a variable z of ψ_g , so there are n_v gears, and we say that the gear *belongs to* gadget g . The block occupies a set of goods G_{gear} consisting of a designated good a , the *anchor*, and fresh goods. Every fresh good has a *kind*, which is *tooth*, *gap* or *edge*, and belongs to exactly one gadget. Among these goods, the rest of the paper refers by name only to three per gear (g, z) : the *hub* $H^{g,z}$, a gap, and the *terminals* $P^{g,z}$ and $N^{g,z}$, two edges, all three belonging to gadget g . In a configuration of the block with n bundles, we call the bundle containing the anchor the *ledger* Λ . Unlike the ledger of the halving gadget, which is the half *without* the anchor, the ledger of the gear block is the bundle *with* it: there, the anchor only tells two complementary halves apart, while here it marks the one bundle, among many, that collects a terminal of every gear. The letters are mnemonic: the ledger holds the *positive* terminal $P^{g,z}$ if z is decoded as true, and the *negative* terminal $N^{g,z}$ if it is decoded as false, just as a ledger of the halving gadget holds P_i or N_i .

Roles. The goods are best read as a graph, as in Figure 1. The teeth and the gaps are its vertices, on the two sides of a bipartite graph, and the edge goods lie on its edges. A bundle other than the ledger holds one tooth, one gap and one edge good, so it is an edge of the graph together with an edge good on it, and outside the ledger a configuration is a perfect matching. The goods belong to *units*, which are the gears and the absorbers of the clauses (below).

By unit and kind, with the superscripts dropped, they are these:

unit	teeth	gaps	edges
gear (g, z)	$A_1, \dots, A_{k+1}$	$H, B_1, \dots, B_k$	P, N, U_j, V_j
U -absorber of C	Y	Y'	none
V -absorber of C	R_i	R'_i	none

for a gear with k occurrences, $j \leq k$, a clause C and $i < |C|$. The anchor counts as both a tooth and a gap. The absorbers have no edge goods of their own. They take the copies that the gears leave over. The construction is in the appendix, which shows that the digits admit exactly the triples of the roles below (Lemma A.1). Its parts play the following roles, which Figure 1 shows on two clauses that share a variable.

- *Teeth and gaps.* The digits put exactly one tooth and one gap into every bundle. As the anchor counts as both, the ledger holds no other tooth or gap. The blocks unit and edge let a tooth share a bundle only with a gap of the same unit, and with exactly one edge.
- *Gears.* A variable with k occurrences has $k+1$ teeth $A_1, \dots, A_{k+1}$ and $k+1$ gaps, its hub H and $B_1, \dots, B_k$, which may pair only along the cycle $H - A_1 - B_1 - A_2 - \dots - B_k - A_{k+1} - H$. As on a gear wheel, teeth and gaps alternate around the cycle. A cycle of even length has exactly two perfect matchings, and they are the two truth values of the variable: each seats every tooth in a neighbouring gap, so the gear turns as a whole, one step one way or the other. We call the matching that pairs A_1 with H , and hence puts P in the hub's bundle, U -mode, and the other one, which puts N in the hub's bundle,

V-mode. So *U-mode* leaves N for the ledger, and *V-mode* leaves P .

- *Edges*. Each pair takes the edge good on its edge of the cycle. At its j -th occurrence, in a clause C , the gear takes one of two copies of C , a *U*-copy or a *V*-copy according to the matching, and leaves the other copy over.
- *Absorbers*. Each clause C has $|C|$ tooth-gap pairs of its own for the copies left over: one *U*-absorber, which takes only a *U*-copy of C , and $|C| - 1$ *V*-absorbers, which take only *V*-copies. A true variable leaves a *U*-copy, so exactly one variable of C is true.
- *Hub and terminals*. The hub is the one gap of a gear that belongs to no occurrence. Its two edges of the cycle carry the terminals P and N , so its bundle keeps exactly one terminal, chosen by the matching, and the digits allow the other terminal only in the ledger. The hub thus turns the state of the gear into a good that leaves the gear.
- *Anchor*. The anchor marks the ledger, which collects these terminals, one per gear.

Each gear with k occurrences supplies $k + 1$ tooth-gap pairs, $n_o + n_v$ in all, each clause C supplies $|C|$ pairs for its absorbers, n_o in all, and the ledger is one more bundle. So there are exactly $2n_o + n_v + 1$ bundles, one per tooth.

How a clause is checked. A variable decoded as true has its gear in *V-mode*. A gear in *V-mode* takes the *V*-copy of each of its clauses and leaves the *U*-copy over, and a gear in *U-mode* does the opposite. Each clause C has exactly one *U*-absorber and $|C| - 1$ *V*-absorbers, so exactly one of its variables leaves a *U*-copy: exactly one variable of each clause is true. For example, in Figure 1, z_1 is true and leaves U_1^1 to the *U*-absorber $Y - Y'$ of $C_1 = \{z_1, z_2, z_3\}$, while z_2 and z_3 are false and leave V_1^2 and V_1^3 to its two *V*-absorbers. The letters are crossed: *V-mode* leaves *U*-copies and puts P in the ledger.

LEMMA 4.5. *Let $\psi_1, \dots, \psi_\Gamma$ be gadgets with n_v variables and n_o occurrences in all, let $n = 2n_o + n_v + 1$, and let a be a good. There are a set of goods G_{gear} as in the vocabulary above, of size $3(n - 1) + n_v + 1$, and seven digit blocks with digit 0 on all goods outside G_{gear} and with totals polynomial in n , all computable in polynomial time, with the following properties. Let π be a configuration of the seven blocks, with n bundles, and Λ its ledger.*

- Locality: every bundle other than Λ consists of exactly one tooth, one gap and one edge, and all three belong to the same gadget.
- Ledger: Λ consists of a and exactly one of $P^{g,z}, N^{g,z}$ for every gear (g, z) .
- Decoding: for every g , the assignment σ_g with $\sigma_g(z) = 1$ if and only if $P^{g,z} \in \Lambda$ is an exactly-1 model of ψ_g .
- The other terminal: for every gear (g, z) , the terminal of $P^{g,z}, N^{g,z}$ that is not in Λ lies in the bundle of the hub $H^{g,z}$. By (i), that bundle is not Λ , and all its goods belong to gadget g .
- Converse: for all exactly-1 models $\sigma_1, \dots, \sigma_\Gamma$ of the gadgets, some configuration decodes to them as in (iii).

In particular, the seven blocks have a configuration if and only if every gadget is satisfiable.

Remark 4.6. The seven blocks have digit 0 outside G_{gear} , so for every balanced partition of a larger instance that has the seven

blocks among its digit blocks, Lemma 4.5 describes the restriction of the partition to the goods G_{gear} : the bundle of $H^{g,z}$ is then the bundle of the partition containing $H^{g,z}$, and part (i) says what that bundle holds of G_{gear} , not what else it may hold. The anchor may be fresh, or it may be a good that the larger instance has already built, which the block borrows. The block does not distinguish the two cases. Section 7 uses a fresh anchor and one gadget, and Section 8 borrows a core good as the anchor and uses many gadgets.

Part (i), locality, is the one the later sections rely on most: no bundle other than the ledger holds goods of two different gadgets. So a charge placed on a good of gadget g that does not go to the ledger lands on a bundle all of whose other goods of G_{gear} belong to g . Charges attached to different gadgets meet only in the ledger, whose contents part (ii) fixes exactly. Part (iv) says where the other terminal goes.

Charges. The ledger is one bundle among many, and the terminal of a gear that is not in the ledger lies in that gear's hub bundle. If we put negative charges on terminals, as the negative form does for the halving gadget, the charges of the terminals outside the ledger would be scattered over many bundles, and the minimum over bundles would see only the largest of them, not their sum. So, to read a graded value off the ledger, we orient the charge the other way: for a grading ζ of the variables of the gadgets, we put $-\Delta_\zeta$ on the anchor and $+\zeta_z$ on the terminal $P^{g,z}$ of each gear (g, z) . As the anchor now lies in the ledger, the ledger carries its $-\Delta_\zeta$. Then, by (ii) and (iii), the ledger carries $-\Delta_\zeta + \text{val}_\zeta(\sigma) \leq 0$ for the decoded assignment σ , and every other bundle carries at least 0. Section 7 uses this charge. Section 8 instead puts charges on bundles other than the ledger, where, by locality, charges of different gadgets never meet.

Interface: the rest of the paper uses the gear block only through Lemma 4.5 and Remark 4.6.

4.6 Two endgames

Each construction ends in one of two ways. The computational problems use identical valuations and read a single number off the instance: the maximin share. The following lemma, the *graded ledger*, says when that number is the optimum of a source problem. It applies whenever the balanced partitions decode to assignments, for instance to the models of a formula, and the worst bundle of each balanced partition carries exactly the graded value of its assignment, shifted by a constant. For identical valuations $\delta_i = \chi$, we write $\chi(\pi) = \min_j \chi(\pi(j))$ for the charge of the worst bundle of a partition π .

LEMMA 4.7. *Let $I = I(\chi, \dots, \chi)$ have n agents with identical valuations, let ζ be a grading with $\Delta_\zeta \leq M$, and let Σ be a nonempty set of assignments. Let $\pi \mapsto \sigma_\pi$ be a polynomial-time map from the balanced partitions to Σ such that*

- (GL1) $\chi(\pi) = -\Delta_\zeta + \text{val}_\zeta(\sigma_\pi)$ for every balanced partition π , and
- (GL2) every $\sigma \in \Sigma$ equals σ_π for some balanced partition π .

Then $\mu = K\tau - \Delta_\zeta + \max_{\sigma \in \Sigma} \text{val}_\zeta(\sigma)$, MMS-fair allocations exist and are balanced, and $\text{val}_\zeta(\sigma_\pi) = \max_{\sigma \in \Sigma} \text{val}_\zeta(\sigma)$ for every MMS-fair allocation π .

PROOF. By (GL1) and (GL2), balanced partitions exist and the common drop is $d = \Delta_\zeta - \max_\sigma \text{val}_\zeta(\sigma)$, which lies between 0 and $\Delta_\zeta \leq M$, as $0 \leq \text{val}_\zeta \leq \Delta_\zeta$. So $\mu = K\tau - d$ by Lemma 4.3(ii). As the valuations are identical, the MMS-fair allocations are the μ -optimal partitions, and one exists, as μ is attained by some partition. All bundles of such a partition π are worth at least $\mu \geq K\tau - M$, so π is balanced by Lemma 4.3(i), and its worst bundle has offset at least $-d$. By (GL1), $\text{val}_\zeta(\sigma_\pi) \geq \Delta_\zeta - d = \max_\sigma \text{val}_\zeta(\sigma)$. $\square$

So a construction that wants the maximin share to encode an optimum owes two things: (GL1), a charge computation, and (GL2), the converse half of its device's interface lemma. The lemma then yields reductions for both computational problems at once.

COROLLARY 4.8. *Let $x \mapsto I_x$ be a polynomial-time map to instances as in Lemma 4.7, such that K , τ and Δ_ζ are computable from x in polynomial time, and let $V(x) = \max_{\sigma \in \Sigma} \text{val}_\zeta(\sigma)$. Then $T_1(x) = (I_x, 1)$ and $T_2(x, y) = y - K\tau + \Delta_\zeta$ form a metric reduction from V to MMS-VALUE. Moreover, the correct outputs of MMS-ALLOC on I_x do not include $\perp$, and every correct output π is balanced and has $\text{val}_\zeta(\sigma_\pi) = V(x)$.*

PROOF. The first claim is the formula for μ . The correct outputs of MMS-ALLOC are the MMS-fair allocations, and the rest is Lemma 4.7. $\square$

The second endgame decides existence rather than computing a number. Here the agents must differ, as with identical valuations an MMS-fair allocation always exists (Remark 3.6). Both existence proofs contain a *core*, a small set of goods on which no allocation is fair. The core has exactly two balanced partitions, its *phases*, and the agents come in two *types*, each of which loses a fixed *cost* s on some bundles of one phase and nothing in the other. So every allocation hurts some agent. Some agents, the *blockers*, carry no further charge and have maximin share $K\tau$. They force every fair allocation into one phase, and into giving one special agent, the *dial*, a bundle that costs her s . The rest of the instance, the *gates*, adds a charge of the dial that supplies two numbers: her *drop* dr , a maximum over partitions, which puts her maximin share at $K\tau - dr$, and her *relief* rl , a maximum over allocations, which says how much of the cost the bundle she receives can make up. The *calculus* of the core (Lemma 6.7 in Section 6.2, and Lemma 8.2 for several dials) lists two conditions on the dial's charge under which an MMS-fair allocation exists if and only if $-s + rl \geq -dr$ (the reader may take this on trust until Lemma 6.7). So a construction that wants existence to encode a comparison owes these two conditions, as it owes (GL1) and (GL2) to the graded ledger. The gates are built from the halving gadget in Section 6 and from the gear block in Section 8. Sections 5 and 7 use the graded ledger, and Sections 6 and 8 use the calculus.

5 Computation with Values in Binary

This section shows that MMS-VALUE is OptP-complete and MMS-ALLOC is FP^{NP} -complete, already for two agents with identical valuations (Theorem 5.2).

5.1 Proof strategy

The target. Fix a function $f \in \text{OptP}$. From an input x we build an instance I_x with two agents of identical valuations whose maximin

share is $K\tau - \Delta_\zeta + f(x)$, for numbers K , τ and Δ_ζ computed from x . With identical valuations, the MMS-fair allocations are exactly the μ -optimal partitions, and we arrange that each of them decodes to a branch of the machine that writes $f(x)$. We reduce from every $f \in \text{OptP}$ directly, via the Cook–Levin construction of Lemma 3.10, so that the output bits of the machine are the only graded variables.

The device. By Lemma 3.10, $f(x)$ is the largest graded value $\text{lex}_\zeta(\varphi)$ of a model of a formula φ , for the grading ζ that gives the output bits the weights $2^{p-1}, \dots, 2^0$ and every other variable weight 0. The goods of I_x are the halving gadget X_φ , and both agents carry the negative form $\chi_{\text{lex}}^{-\zeta}$ as their charge. With two agents, a balanced partition is a pair of halves: a ledger and its anchor side. The ledger carries $-(\Delta_\zeta - \text{val}_\zeta(\sigma))$ for the model σ it records, and the anchor side carries the opposite, which is at least 0, so the ledger is the worse bundle, and the best balanced partition for the agents is one whose ledger records a model of largest graded value.

The endgame. This is the shape of the graded ledger (Lemma 4.7): (GL1) is the charge computation just described, and (GL2) is part (b) of Lemma 4.4. The lemma gives $\mu(I_x) = K\tau - \Delta_\zeta + f(x)$.

The reduction. For MMS-VALUE, the second map of the reduction subtracts the known constant $K\tau - \Delta_\zeta$. For MMS-ALLOC, it reads the model off the ledger of the output allocation and returns the number on its output bits. Every branch of the machine writes a number, so φ is satisfiable. Hence MMS-fair allocations exist, and $\perp$ is never a correct output.

Where to find what. Section 5.2 builds I_x , and Lemma 5.1 verifies (GL1) and (GL2) and computes the maximin share. Theorem 5.2 concludes. The halving gadget is in Section 4.4 and the graded ledger in Section 4.6.

5.2 The instance

Let $f \in \text{OptP}$ be computed by an NP metric machine M that writes exactly $p = p(|x|)$ bits on every branch on input x (padding with leading zeros if necessary). Let φ be the 3-CNF formula of Lemma 3.10, over $z_1, \dots, z_k$ with q clauses, whose first p variables are the output bits. We grade the variables by

$$\zeta_i = 2^{p-i} \text{ for } i \leq p, \quad \zeta_i = 0 \text{ for } i > p, \quad \text{so that } \Delta_\zeta = 2^p - 1.$$

Then $\text{val}_\zeta(\sigma)$ is the number written on the branch described by a model σ , so by Lemma 3.10,

$$\text{lex}_\zeta(\varphi) = f(x). \quad (1)$$

CONSTRUCTION (THE INSTANCE I_x). Let X_φ be the halving gadget of Lemma 4.4 for φ , with weights w and $w(X_\varphi) = 2\tau$, and let $\chi = \chi_{\text{lex}}^{-\zeta}$ be its negative form for the grading ζ . Let $M = \Delta_\zeta = 2^p - 1$ and $K = 2\Delta_\zeta + 1$. Then $I_x = I(\chi, \chi)$ is the instance with two agents and goods X_φ in which both agents have the valuation $u = K \cdot w + \chi$.

The charge χ is Δ_ζ on the anchor, $-\zeta_i$ on N_i , and 0 elsewhere. Its positive values sum to Δ_ζ and its negative values to $-\Delta_\zeta$, so $|\chi(S)| \leq \Delta_\zeta = M$ for all S , as the construction requires.

LEMMA 5.1. *The balanced partitions of I_x are exactly the pairs of a ledger S of X_φ and its anchor side, and I_x satisfies (GL1) and (GL2) of Lemma 4.7 with Σ the set of models of φ and $\sigma_\pi = \sigma_S$ for the ledger S*

of π . Hence $\mu(I_x) = K\tau - \Delta_\zeta + f(x)$. Moreover, I_x is computable from x in polynomial time, and every value has $O(p + k + q)$ bits.

THEOREM 5.2. *MMS-VALUE is OptP-complete under metric reductions, and hence FNP^{NP} -complete. MMS-ALLOC is FNP^{NP} -complete. Hardness holds already for $n = 2$ agents with identical valuations, on instances that always admit an MMS-fair allocation.*

PROOF. Membership is Proposition 3.5(i) and (ii), as $\text{OptP} \subseteq \text{FNP}^{\text{NP}}$ by binary search. For hardness, let $f \in \text{OptP}$ and map x to I_x . By Lemma 5.1 and Corollary 4.8, $T_1(x) = (I_x, 1)$ and $T_2(x, y) = y - K\tau + \Delta_\zeta$ form a metric reduction from f to MMS-VALUE, as K, τ and Δ_ζ are computable from x in polynomial time. With Theorem 3.3, MMS-VALUE is FNP^{NP} -complete. For MMS-ALLOC, let $T_1(x) = I_x$ and let $T_2(x, \pi) = \text{val}_\zeta(\sigma_S)$, for the ledger S of π , the bundle without the anchor. By Corollary 4.8, every correct output π is balanced, so its bundle without the anchor is a ledger, and $T_2(x, \pi) = f(x)$. Moreover, $\perp$ is not a correct output. So every $f \in \text{OptP}$ reduces to both problems, and by Theorem 3.3 and composition of reductions, so does every function in FNP^{NP} . $\square$

This section needs no core: with identical valuations an MMS-fair allocation always exists (Remark 3.6), and what is hard is to find one. What costs magnitude is the grading, which lives in the charge, with $p + 1$ bits, and not in the weights.

6 Existence with Values in Binary

This section shows that MMS-EXISTS is P^{NP} -complete, already for four agents with three distinct valuations (Theorem 6.14). For hardness we reduce from ODD-LEX-MAX-SAT (Lemmas 3.8 and 3.9) to instances $I_{\alpha, \beta}$.

6.1 Proof strategy

This section outlines the hardness proof and introduces its vocabulary informally. Sections 6.2–6.5 give the precise definitions.

The comparison. By Lemma 3.9, it suffices to decide, for formulas α and β with β satisfiable, whether α is satisfiable and $\text{lex}(\alpha) \geq \text{lex}(\beta)$. We build an instance $I_{\alpha, \beta}$ with four agents that has an MMS-fair allocation exactly in this case. One agent, the *dial* D , carries all information about the formulas. The other three, the *blockers*, only enforce the structure. Fairness for the dial comes down to the inequality $-s + \text{rl} \geq -\text{dr}$: the *cost* s is fixed, her *drop* $\text{dr} = \Delta - \text{lex}(\beta)$ is how far her maximin share lies below a common value, and her *relief* $\text{rl} = \text{lex}(\alpha)$ (or 0 if α is unsatisfiable) is how much of the cost can be made up. With $s = \Delta$, the inequality reads $\text{lex}(\alpha) \geq \text{lex}(\beta)$. So $\text{lex}(\beta)$ enters through a maximum over partitions, her maximin share, and $\text{lex}(\alpha)$ through a maximum over allocations, the best bundle she can be given. Two gadgets are needed for this. Anything the dial can be handed in an allocation, she can also place in a partition of her own. So with a single halving gadget on one formula φ , her maximin share and the best bundle she can receive are both determined by $\text{lex}(\varphi)$, and after cancelling the core's cost s , the fairness condition becomes $\text{lex}(\varphi) \geq c$ for a constant c computable from the instance. This is a single NP question, so it gives no P^{NP} -hardness unless $\text{NP} = \text{P}^{\text{NP}}$. With two gadgets, $\text{lex}(\beta)$ enters as a maximum over partitions, the dial's maximin share, and $\text{lex}(\alpha)$ as a maximum over allocations, the

bundle she receives. Their comparison is P^{NP} -hard by Lemmas 3.8 and 3.9.

Weights and charges. Every valuation is $K \cdot w + \delta_i$ as in Section 4.2: a common weight w , scaled by a large K , plus a small, agent-specific charge δ_i . Only balanced partitions, into four bundles of equal weight, matter: both the maximin shares and the fair allocations live on them. The weight is assembled from digit blocks, and blocks interact only through goods that both see (Lemma 4.2), so we build separate *gadgets* from separate blocks and join them at a few named shared goods. A balanced partition is then glued from one configuration of each gadget.

The core. The *core* consists of twelve goods with exactly two balanced partitions, the *phases* P^V and P^U . Two *perturbations* δ_V and δ_U , carried by the agents of *type* V (D and E^V) and of type U (E_1^U and E_2^U), are each *neutral* in one phase and cost s on three bundles of the other, the *deficit bundles*. We write Bad_V and Bad_U for the deficit bundles of P^U and of P^V . So a blocker's maximin share is computed in her neutral phase, where she loses nothing, while every allocation uses one phase and hurts some agent there. A bundle that costs a blocker s is worth less to her than her maximin share, so no MMS-fair allocation gives it to her. Hence every fair allocation must use phase P^U and give the dial a bundle of Bad_V . The fourth bundle, the *host bundle*, is *safe*: it has offset at least 0 for both perturbations in both phases, so no agent loses anything on it, and it is not a deficit bundle. In both phases, each bundle holds exactly one of the four *B-side goods* $B_1, B_2, \bar{B}_1, \bar{B}_2$, and is named after it. The host bundle is the bundle of $\bar{B}_2$, the *host*. These four goods are the only core goods that the rest of the instance touches. The core's *contract*, the *calculus* (Lemma 6.7), lists two conditions on the dial's charge under which fairness is exactly $-s + \text{rl} \geq -\text{dr}$.

The halving gadget. For a formula φ , the halving gadget X_φ of Section 4.4 splits into two halves of equal weight exactly along the models of φ . The half without the anchor is a ledger, which records a model, and the other is its anchor side. The lex charge χ_{lex} and its negative form χ_{lex}^- let the dial value a ledger by its model. The anchor side always carries the opposite charge.

The gates. A *gate* is a digit block that places a halving gadget X_φ next to the core, with a charge of the dial on X_φ . Next to the core means that the gate is a separate block, which sees no core good apart from some of the *B-side goods*, but whose goods are divided over the same four bundles as the core goods: each bundle of a balanced partition holds one bundle of the phase and one class (see below) of each gate. The bundles of a configuration of a gate are its *classes*. In both phases each bundle holds exactly one *B-side good*, so a gate block can pin a class to a named bundle by giving that good a digit: it *labels* the class. Besides X_φ , each gate has two *carriers*, each of which carries one half of X_φ into a bundle, and two goods that each fill a bundle on their own. *Light* carriers force the split. *Heavy* carriers also let the gate be *switched off*, keeping X_φ whole. The *drop gate*, light on X_β with χ_{lex}^- , always gives the dial a *drop ledger class* of charge at most $-\text{dr}$, and a *drop anchor class* of charge at least 0. It sets her drop. Its carriers are B_2 and $\bar{B}_2$, and $\bar{B}_2$ is also the anchor of X_β . The *relief gate*, heavy on X_α with χ_{lex} , has two *relief classes*, each of charge at most rl , and one of them can

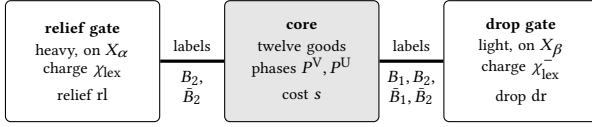


Figure 2: The three gadgets of $I_{\alpha, \beta}$ and the goods they share. Each gadget has its own interface lemma. The drop gate sees all four B -side goods of the core and the relief gate sees B_2 and $\bar{B}_2$. These are the only goods seen by the blocks of two gadgets. The last line of each box is what the gadget contributes to the dial's inequality $-s + rl \geq -dr$, with $s = \Delta$, $dr = \Delta - \text{lex}(\beta)$ and $rl = \text{lex}(\alpha)$.

carry exactly rl . It supplies her relief. Its carriers are fresh goods, and B_2 and $\bar{B}_2$ fill a bundle each. The relief gate must be able to be switched off because α may be unsatisfiable: then X_α has no half, and a light gate would leave no balanced partition at all, whereas the switched-off gate gives relief $rl = 0$. The drop gate must not be switchable, or the dial could avoid her ledger. It can be light because β is always satisfiable.

Coupling. The labels join the gadgets. The drop gate names $\bar{B}_2$ as the anchor of X_β and B_2 as its second carrier, so in every balanced partition the drop anchor class lies in the host bundle, where it cannot help the dial, and the drop ledger class in the bundle of B_2 . The relief gate lets B_2 and $\bar{B}_2$ fill a bundle each, so the relief classes lie in the bundles of B_1 and $\bar{B}_1$. No relief class can meet the drop ledger class or the host, and the dial's drop depends on β alone. All other goods are fresh. The goods outside the core, together with the B -side goods, are the *gate goods*. The coupling lemma (Lemma 6.12) shows that a balanced partition is a phase together with one configuration of each gate, glued at the B -side goods. From it, Lemma 6.13 checks the core's contract, which proves the theorem. Figure 2 shows the three gadgets and the goods they share.

Where to find what. Section 6.2 builds the core (Lemmas 6.1–6.4), says how gates may attach to it (Definition 6.5 and Lemma 6.6), and states its contract (Lemma 6.7). Section 6.3 builds the two gates in isolation on the halving gadget of Section 4.4 (Lemmas 6.8–6.11), and Section 6.4 couples them to the core (Lemma 6.12). Section 6.5 checks the contract (Lemma 6.13) and proves Theorem 6.14.

6.2 The core

The *core* is a set of twelve goods for four agents with exactly two balanced partitions. We will perturb it so that, in each of them, some agent falls short of her maximin share. It is the part of the instance that makes fairness fail, and the gates attach to it. We need a weight system with exactly two balanced partitions, such that each of two types of agents is harmed in exactly one of them, and with a bundle that is safe for everybody in both. Section 8.2 generalises the core to $2m$ agents, which Section 8 needs. Here, four agents suffice.

The picture behind the core is the cycle $A_1B_1A_2B_2$ of Figure 3, in which V_1 joins A_1 to B_1 , U_1 joins B_1 to A_2 , V_2 joins A_2 to B_2 , and U_2 joins B_2 to A_1 . Every bundle will hold an edge together with its two endpoints, either the vertices themselves or copies of them, their shadows. The bundles of vertices form a perfect matching of the cycle, and the bundles of shadows take the edges that it leaves

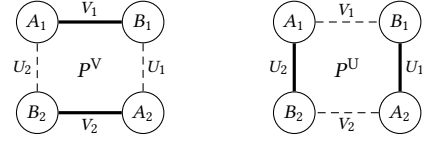


Figure 3: The cycle of the core and its two perfect matchings, the phases. Each thick edge forms a bundle with its endpoints, and each dashed edge forms a bundle with the shadows of its endpoints.

unused. The cycle has exactly two perfect matchings, $\{V_1, V_2\}$ and $\{U_1, U_2\}$, so the core is a switch with two positions.

CONSTRUCTION (THE CORE Σ_2). The core goods are the four vertices A_1, A_2, B_1, B_2 and the four edges V_1, V_2, U_1, U_2 of the cycle, and a shadow $\bar{A}_i, \bar{B}_i$ of each vertex. The goods $A_i, \bar{A}_i$ are the A -side goods, and $B_i, \bar{B}_i$ the B -side goods, as each edge of the cycle has one endpoint of each kind. They carry six digit blocks $\text{side}_A, \text{side}_B, \text{edge}, \text{rol}, L_1, L_2$, whose roles we explain below. Their digits are in Table 8 in the appendix. Their totals are 4 for the first four blocks and 12 for the link blocks L_1 and L_2 , so that, with $n = 4$ bundles, a bundle meets them if its digit sums are $(1, 1, 1, 1, 3, 3)$.

The blocks $\text{side}_A, \text{side}_B$ and edge give each bundle one A -side good, one B -side good and one edge. The *role* block rol gives each bundle exactly one of an A -vertex and a B -shadow, so a bundle holds either two vertices or two shadows, which fixes the role of the bundle. The *link* blocks L_1 and L_2 , through their digit sums 3, link the indices of these two goods to the edge, so that the edge joins its own endpoints. The two partitions of the core goods that meet all six blocks, P^V and P^U , are listed in Table 2. We call them the two *phases*, named after the edges of their matchings. In both phases, each bundle contains exactly one of the goods $B_1, B_2, \bar{B}_1$ and $\bar{B}_2$, and we name the bundle after it, which lets us refer to the same bundle in both phases, as the rows of Table 2 do.

LEMMA 6.1. *The partitions of the core goods into four bundles that meet all six blocks are exactly P^V and P^U .*

Next, we define two *perturbations*, charges on the core goods, with a *scale* $s \geq 1$ that we fix later, the amount by which a perturbation can hurt an agent. Each perturbation is *neutral* in one phase, that is, 0 on each of its bundles, and hurts in the other, and we name it after the phase in which it is neutral: δ_V is 0 on every bundle of P^V , and on P^U it costs s on three bundles and pays on the fourth, and δ_U is its mirror image, neutral on P^U .

$$\delta_V : \bar{A}_2 \mapsto 2s, \bar{B}_1 \mapsto -s, \bar{B}_2 \mapsto s, U_1, U_2 \mapsto -s,$$

$$\delta_U : \bar{A}_2 \mapsto -2s, \bar{B}_1 \mapsto s, \bar{B}_2 \mapsto 3s, V_1, V_2 \mapsto -s,$$

and 0 on all other goods (including all goods outside the core, later on). Reading off the bundles gives the following.

LEMMA 6.2. *Both δ_V and δ_U sum to 0. The perturbation δ_V has offset 0 on every bundle of P^V , and on P^U it has offset $-s$ on the bundles of B_1, B_2 and $\bar{B}_1$, and $3s$ on the bundle of $\bar{B}_2$. Symmetrically, δ_U has offset 0 on every bundle of P^U , and on P^V the same offsets as δ_V on P^U .*

Table 2: The two phases of the core, bundle by bundle, with the offsets of δ_V and δ_U .

bundle	P^V	P^U	$\delta_V (P^V / P^U)$	$\delta_U (P^V / P^U)$
deficit, $\ni B_1$	$A_1 B_1 V_1$	$A_2 B_1 U_1$	$0 / -s$	$-s / 0$
deficit, $\ni B_2$	$A_2 B_2 V_2$	$A_1 B_2 U_2$	$0 / -s$	$-s / 0$
deficit, $\ni \bar{B}_1$	$\bar{A}_2 \bar{B}_1 U_1$	$A_1 \bar{B}_1 V_1$	$0 / -s$	$-s / 0$
host, $\ni \bar{B}_2$	$\bar{A}_1 \bar{B}_2 U_2$	$A_2 \bar{B}_2 V_2$	$0 / 3s$	$3s / 0$

We call a bundle of the core *safe* if it has offset at least 0 for both perturbations in both phases. An agent of either type who receives a safe bundle thus loses nothing on the core goods, whichever phase is used. By Lemma 6.2, exactly one bundle is safe, the bundle of $\bar{B}_2$, which we call the *host bundle*, and $\bar{B}_2$ itself the *host*. The other three bundles, those of B_1 , B_2 and $\bar{B}_1$, are the *deficit bundles*: each of them costs s to one of the two types, in each phase. We let Bad_V be the set of deficit bundles of P^U , on which δ_V has offset $-s$, and Bad_U the set of deficit bundles of P^V , on which δ_U has offset $-s$.

The B -side goods are where the gates will attach to the core: in both phases each bundle holds exactly one of B_1 , B_2 , $\bar{B}_1$ and $\bar{B}_2$, and we name the bundle after it, so a gate may pin one of its classes to a named bundle by giving the good of that name a digit. A gate may share with the core only B -side goods, and every other core good will be seen by the core blocks alone. A charge that is 0 on all core goods other than the host therefore puts on each core bundle exactly the charge of its goods outside the core, together with the host if it holds it. So the rest of the paper refers to core bundles only by their names, as the deficit bundles and the host bundle, and by the sets Bad_V and Bad_U . In a balanced partition of a larger instance, we name each bundle after the core bundle that lies in it. For example, the host bundle is the bundle that contains $\bar{B}_2$.

Agents. The four agents are two agents of *type V*, who carry δ_V , and two agents of *type U*, who carry δ_U . One of the type- V agents is the *dial D*, who carries an additional charge χ , which is how the formulas enter: it turns her maximin share, and her value for some bundles, like a dial. The other three agents are *blockers*, who carry nothing else: E^V of type V , and E_1^U and E_2^U of type U . Each blocker is neutral in the phase that her type is named after, so her maximin share is the full $K\tau$. A bundle of offset $-s$ is worth $K\tau - s$ to her, less than her maximin share, so no MMS-fair allocation can give it to her. The blockers exist only to block every way of passing the core's deficit to someone other than the dial. A fair allocation must use the phase that hurts the dial's type, P^U , because only the dial can absorb a deficit: the blockers of type U force this phase, and the blocker of type V forces the deficit onto the dial. In the final instance (Lemma 6.13), the agents are:

agent	charge	neutral phase	maximin share
D	$\delta_V + \chi$	P^V (for δ_V alone)	$K\tau - \text{dr}$
E^V	δ_V	P^V	$K\tau$
E_1^U, E_2^U	δ_U	P^U	$K\tau$

LEMMA 6.3. (a) If the bundles of P^U are handed to the four agents, one each, some type- V agent receives a bundle of Bad_V . The same holds for P^V , type U and Bad_U . (b) For each type- V agent i and every $S \in \text{Bad}_V$, the bundles of P^U can be handed out so that i receives S

and every other agent receives a bundle of offset at least 0 for her perturbation.

PROOF. (a) The three bundles of Bad_V cannot all go to the two type- U agents. (b) Give S to i , the host bundle to the other type- V agent, and the two remaining bundles to the type- U agents, on which δ_U is 0. $\square$

Already the core alone, a small and structured instance, has no MMS-fair allocation, which is what the gates will repair. Every agent has a phase that is neutral for her, so all maximin shares are $K\tau$. But an allocation uses a single phase, and in it the two agents of the harmed type have only one safe bundle between them.

PROPOSITION 6.4. Let w be given by Lemma 4.1 for the six blocks of Σ_2 , and let $I_0 = I(\delta_V, \delta_V, \delta_U, \delta_U)$ be the instance with four agents on the twelve core goods, with $M = 4s$. Then $\mu_i = K\tau$ for every agent i , and I_0 admits no MMS-fair allocation.

Gated instances. The gates are built by different means than the core, and must not disturb its rigidity. A *gated instance* fixes the only goods through which they may interact with it, the B -side goods.

Definition 6.5. Let $s \geq 1$. A *gated instance* is an instance $I(\delta_V + \chi, \delta_V, \delta_U, \delta_U)$ of the construction of Section 4.2, with agents D, E^V, E_1^U, E_2^U , in which

- (G1) the goods are the core goods and further goods,
- (G2) the weights are given by Lemma 4.1 for the six core blocks, which have digit 0 on all goods outside the core, and further *gate blocks*, which have digit 0 on all core goods other than the B -side goods $B_1, B_2, \bar{B}_1, \bar{B}_2$,
- (G3) the dial's charge χ sums to 0 and is 0 on all core goods other than $\bar{B}_2$, and
- (G4) $M \geq s$.

The goods outside the core, together with the B -side goods that the gate blocks see, are the *gate goods*.

So the B -side goods seen by the gate blocks are both core goods and gate goods, and they are the only channel between the core and the gates. The definition is syntactic: it is checked on the blocks and the charge, not on the balanced partitions. The next lemma then shows, once for all gated instances, that every balanced partition has a phase, which the calculus needs. Section 8 reuses both for the core Σ_m , with the gear block as its gate blocks. A balanced partition then factorises into a phase, a configuration of the gate blocks, and a pinning condition on the B -side goods.

LEMMA 6.6. A partition π of the goods of a gated instance is balanced if and only if (i) its restriction to the core goods is P^V or P^U , its phase, (ii) its restriction to the gate goods is a configuration of the gate blocks, and (iii) for each B -side good seen by a gate block, the bundle of the configuration containing it lies in the bundle of the phase named after it. Conversely, if the B -side goods seen by the gate blocks lie in distinct bundles of a configuration of the gate blocks, then every phase and this configuration combine to a balanced partition by every bijection between their bundles that satisfies (iii).

PROOF. Apply Lemma 4.2 to the core blocks and the gate blocks, whose supports share exactly the B -side goods seen by the gate

blocks. The configurations of the core blocks are the phases by Lemma 6.1, and each B -side good lies in the bundle named after it in both phases. $\square$

Condition (iii) holds automatically for the restrictions of a balanced partition, as each good lies in one bundle. It is the condition of Lemma 4.2 that a phase and a configuration must satisfy to be glued together. The gate block of Section 8 sees only the host $\bar{B}_2$, and then (iii) says that the bundle of $\bar{B}_2$ lies in the host bundle, so the lemma covers it verbatim.

The core's contract. The following lemma, the *calculus*, reduces the existence of an MMS-fair allocation to two properties of the dial's charge on balanced partitions. It turns fairness into arithmetic with two numbers: her drop dr , by which χ lowers her maximin share, and her *relief* rl , by which χ can raise her value for a bundle of Bad_V , as it relieves part of the deficit s that the core puts on that bundle. It applies to every gated instance. For a partition π , we write $\chi(\pi) = \min_j \chi(\pi(j))$. The lemma has two hypotheses on the charge, and we refer to them by the names (drop) and (relief). They are conditions that must both hold, not cases. They pull in different directions, and this is the heart of the construction. Her maximin share is determined by her worst bundle, in *either* phase, so (drop) asks that χ is small on *some* bundle other than the host bundle in every partition. An allocation can hand her whichever bundle it likes, so (relief) asks that χ can be large on *one* bundle of Bad_V in some partition. The host bundle is excluded from (drop) because it is the one bundle where the core pays the dial: in phase P^U it has δ_V -offset $3s$, while every other bundle lies in Bad_V and has offset $-s$, so a small charge there bounds her offset in both phases. A fair allocation must use phase P^U and give the dial a bundle of Bad_V , so fairness comes down to $-s + rl \geq -dr$.

LEMMA 6.7. *Let I be a gated instance, and let dr and rl be integers with $0 \leq dr \leq M$ such that the following two conditions hold.*

- (drop) *Balanced partitions of phase P^V exist, and some balanced partition of phase P^V has $\chi \geq -dr$ on every bundle. Every balanced partition, of either phase, has $\chi \leq -dr$ on some bundle other than the host bundle.*
- (relief) *Every balanced partition of phase P^U has $\chi \leq rl$ on every bundle in Bad_V , and some balanced partition of phase P^U has $\chi(S) = rl$ for some $S \in \text{Bad}_V$.*

Then every blocker has maximin share $K\tau$, the dial has maximin share $K\tau - dr$, and I admits an MMS-fair allocation if and only if $rl \geq s - dr$.

PROOF. Every balanced partition has a phase by Lemma 6.6. *Blockers.* A balanced partition of phase P^V gives the type- V blocker offset 0 on every bundle (Lemma 6.2), and one of phase P^U , which exists by (relief), does so for the type- U blockers. So their drops are 0, and their maximin shares are $K\tau$ by Lemma 4.3(ii). *The dial.* In phase P^V , the offsets of D are her charges, so the partition of (drop) has all offsets at least $-dr$, and her drop is at most dr . Conversely, let S be the bundle of (drop), other than the host bundle, with $\chi(S) \leq -dr$. In phase P^V its offset for her is $\chi(S) \leq -dr$, and in phase P^U it lies in Bad_V , so its offset is $-s + \chi(S) \leq -dr$. So her drop is exactly dr , and her maximin share is $K\tau - dr$ by Lemma 4.3(ii), as $dr \leq M$.

($\Rightarrow$) An MMS-fair allocation gives every agent at least $K\tau - dr \geq K\tau - M$, so it is balanced by Lemma 4.3(i). In phase P^V , by Lemma 6.3(a), some type- U agent would receive a bundle of offset $-s$. Both type- U agents are blockers, so this is below her maximin share $K\tau$. So the phase is P^U , and by Lemma 6.3(a) some type- V agent receives a bundle $S \in \text{Bad}_V$. This is not the blocker E^V : her maximin share is $K\tau$, and S is worth $K\tau - s$ to her, as she carries no charge. So the dial receives S . Her offset on S is the sum $\delta_V(S) + \chi(S) = -s + \chi(S)$, which is at most $-s + rl$ by (relief), and at least $-dr$ as the allocation is fair. So $rl \geq s - dr$. ($\Leftarrow$) Let $rl \geq s - dr$, take a balanced partition π of phase P^U and a bundle $S \in \text{Bad}_V$ with $\chi(S) = rl$ as in (relief), and hand it out as in Lemma 6.3(b), with D receiving S . Then D gets offset $-s + rl \geq -dr$, and every blocker gets offset at least 0, so every agent receives at least her maximin share. $\square$

Interface: the rest of the paper uses the core only through its phases (Lemma 6.1), the deficit bundles, the host bundle and the host, the fact that each bundle holds exactly one B -side good, the factorisation (Lemma 6.6), and its contract (Lemma 6.7).

6.3 Gates

Each gate is built on the halving gadget X_ϕ of Section 4.4, with the lex charge χ_{lex} or its negative form χ_{lex}^- for the grading $\zeta_i = 2^{k-i}$, so that ledgers carry $\text{val}(\sigma_S)$, respectively $-(\Delta - \text{val}(\sigma_S))$.

Gate blocks. A *gate* is a digit block that places a halving gadget X_ϕ next to the core, together with a charge of the dial on X_ϕ . It is the channel through which a formula ϕ reaches the dial's valuation. Next to the core means, as in Section 6.1, that the gate is a block of its own, which sees no core good except some of the B -side goods, but whose goods are divided over the same four bundles as the core goods: by Lemma 6.6, each bundle of a balanced partition holds one bundle of the phase together with one bundle of a configuration of each gate. In both phases each bundle holds exactly one B -side good, so a gate block can pin one of its classes to a named bundle by giving that good a digit. Each gate is the halving problem of X_ϕ spread over four bundles: two goods whose digit is the target of a bundle fill a bundle each and take no part in the split, and the other two bundles share X_ϕ and two *carriers* equally. Normally each of these two bundles holds one carrier together with one half of X_ϕ , so the carrier carries that half into its bundle, hence its name. A carrier of digit 1 cannot stand in for part of a half, as the weights in X_ϕ come in multiples of b by (W1), so each of these bundles takes one carrier and one half. A carrier that weighs as much as a half lets the two carriers sit together instead, while X_ϕ stays whole. So with *light* carriers the block must split X_ϕ into two halves, and with *heavy* carriers the block can also be *switched off*: the carriers share one bundle, carrying no half, and X_ϕ fills the other.

We use one gate of each kind, and name each after what it gives the dial. The *drop gate* is light, on a halving gadget X_β whose anchor is the host $\bar{B}_2$. Its carriers are B_2 and the anchor $\bar{B}_2$, and B_1 and $\bar{B}_1$ fill a bundle each. The *relief gate* is heavy, on a halving gadget X_α with a fresh anchor a_α . Its carriers are fresh goods γ_3, γ_4 , and B_2 and $\bar{B}_2$ fill a bundle each. With $w_\beta(X_\beta) = 2\tau_\beta$ and $w_\alpha(X_\alpha) = 2\tau_\alpha$, their digits are as follows. A digit equal to the target fills a bundle on its own. Digit 1, on top of the weight in X_β if any, is a light

carrier, and digit τ_α a heavy one. In the drop block, $\bar{B}_2$ plays two roles at once: it is the anchor of X_β , with weight $w_\beta(\bar{B}_2)$, and a light carrier, with the extra 1, and its digit is the sum of the two. So the carrier $\bar{B}_2$ always takes the half that holds the anchor, the anchor side. This is how the drop block pins the anchor side to the bundle of $\bar{B}_2$. In the relief block, $\bar{B}_2$ only fills a bundle: every good of X_β has digit 0 there, so its role as anchor plays no part. Roles are per block, and the weight of a good adds its digits over all blocks (Lemma 4.1). The last column gives the role of each good in the drop block, and then in the relief block.

good	drop block	relief block	role (drop / relief)
$x \in X_\beta, x \neq \bar{B}_2$	$w_\beta(x)$	0	gadget good
$\bar{B}_2$	$w_\beta(\bar{B}_2) + 1$	$2\tau_\alpha$	anchor of X_β and light carrier / fills a bundle
B_2	1	$2\tau_\alpha$	light carrier / fills a bundle
$B_1, \bar{B}_1$	$\tau_\beta + 1$	0	fills a bundle
$x \in X_\alpha$	0	$w_\alpha(x)$	gadget good
γ_3, γ_4	0	τ_α	heavy carrier
target	$\tau_\beta + 1$	$2\tau_\alpha$	—

and 0 on all other goods. The totals are $2\tau_\beta + 1 + 1 + 2(\tau_\beta + 1) = 4(\tau_\beta + 1)$ and $2\tau_\alpha + 2\tau_\alpha + 4\tau_\alpha = 8\tau_\alpha$, so a bundle meets a block if its digit sum is the target. The next two lemmas describe each block on its own, without the core. Their proofs use only $w(X_\varphi) = 2\tau_\varphi$, (W1) and these digits.

LEMMA 6.8. *In every partition into four bundles that all meet the drop block: the bundles of B_1 and of $\bar{B}_1$ hold no other good of the block, the bundle of B_2 holds, besides B_2 , exactly a ledger S of X_β , the bundle of $\bar{B}_2$ holds exactly the anchor side $X_\beta \setminus S$, and B_2 and $\bar{B}_2$ lie in different bundles. Every ledger S gives such a partition.*

LEMMA 6.9. *In every partition into four bundles that all meet the relief block: the bundles of B_2 and of $\bar{B}_2$ hold no other good of the block, and the other two bundles hold, as goods of the block, either*

(S) $\{\gamma_l\} \cup T$ and $\{\gamma_{7-l}\} \cup (X_\alpha \setminus T)$, for a half T and some $l \in \{3, 4\}$,
or

(U) $\{\gamma_3, \gamma_4\}$ and X_α .

Every half T gives an instance of (S), and (U) always meets the block.

The drop gate always hands the dial a ledger of β , and so sets her drop, and with it her maximin share. It must be light, so that the dial cannot avoid a bundle carrying her ledger's charge. Otherwise she could switch it off and put all of X_β , with total charge 0, into one bundle. A light block on X_β has a configuration only if β is satisfiable (Lemma 6.8). This is why our source problem supplies a satisfiable β . The relief gate can provide her relief, by raising her value for one bundle. It must be heavy, that is, it must be possible to switch it off, for two reasons. First, $\alpha = \varphi \wedge z_k$ may be unsatisfiable. Then X_α has no half, so a light block on X_α would have no configuration, and by Lemma 4.2 the instance would have no balanced partition at all. Lemma 4.3(ii) and the calculus would not apply, and we would have no control over the maximin shares. With heavy carriers, case (U) always gives a configuration, in which both relief classes carry charge 0, so an unsatisfiable α gives relief $rl = 0$, and Lemma 6.13 shows that the instance is then a no-instance. Second, even for a satisfiable α , the dial switches the relief gate off in her own best partition (the proof of (drop) in Lemma 6.13): in case (S) one of its classes may carry a negative charge, which could push her worst bundle below $-\text{dr}$ and make her drop depend on α .

For a gate, we call the bundles of a configuration of its block (Lemma 4.2) its *classes*, and name each class that contains a B -side good after that good. In a balanced partition of the whole instance, each class lies in one bundle, in the sense of Section 4.2. We first describe each gate in isolation, with its own charge, which is χ_{lex}^- on X_β for the drop gate and χ_{lex} on X_α for the relief gate, and 0 on the other goods of the block. The charge of the dial is their sum.

LEMMA 6.10. *Let β be satisfiable. In every configuration of the drop block, the classes of B_1 and $\bar{B}_1$ are $\{B_1\}$ and $\{\bar{B}_1\}$, and carry charge 0. The class of B_2 , the drop ledger class, is $\{B_2\} \cup S$ for a ledger S , and carries $-(\Delta - \text{val}(\sigma)) \leq -(\Delta - \text{lex}(\beta))$, where $\sigma = \sigma_S$ is the model of β encoded by S . The class of $\bar{B}_2$, the drop anchor class, is $X_\beta \setminus S$, and carries $\Delta - \text{val}(\sigma) \geq 0$. Some configuration has a drop ledger class of charge exactly $-(\Delta - \text{lex}(\beta))$.*

LEMMA 6.11. *Let $rl = \text{lex}(\alpha)$ if α is satisfiable and $rl = 0$ otherwise. In every configuration of the relief block, the classes of B_2 and $\bar{B}_2$ are $\{B_2\}$ and $\{\bar{B}_2\}$, and carry charge 0, and the two classes not containing B_2 or $\bar{B}_2$ are the relief classes. In case (S), they are $\{\gamma_l\} \cup T$ and $\{\gamma_{7-l}\} \cup (X_\alpha \setminus T)$ for a half T and some $l \in \{3, 4\}$, and carry $\chi_{\text{lex}}(T)$ and $-\chi_{\text{lex}}(T)$. In case (U), they are $\{\gamma_3, \gamma_4\}$ and X_α , and both carry 0. Every relief class carries at most rl . Case (U) gives a configuration, and some configuration has a relief class of charge exactly rl .*

Interface: the rest of the paper uses the gates only through Lemmas 6.10 and 6.11, and through Lemma 6.8 for where B_2 and $\bar{B}_2$ lie.

6.4 Coupling the gates to the core

The drop block names $\bar{B}_2$ as the anchor of X_β and B_2 as its second carrier, so in every balanced partition the drop anchor class lies in the host bundle, which is not a deficit bundle, and the drop ledger class lies in the bundle of B_2 . The relief block gives B_2 and $\bar{B}_2$ the target as their digit, so the relief classes lie in the bundles of B_1 and $\bar{B}_1$. So no relief class can meet the drop ledger class or the host, and the dial's drop depends on β alone.

CONSTRUCTION (THE INSTANCE $I_{\alpha,\beta}$). *Let α and β be CNF formulas over $z_1, \dots, z_k$ as in Section 3.4, and let $s = \Delta$ (this choice is explained in Section 6.5). The goods of $I_{\alpha,\beta}$ are*

- the twelve goods of the core Σ_2 ,
- the goods X_β of Lemma 4.4 for β , with weights w_β and $w_\beta(X_\beta) = 2\tau_\beta$, where the anchor of X_β is the core good $\bar{B}_2$ and all other goods are fresh (the conclusions of Lemma 4.4 concern weights and do not depend on names),
- the goods X_α of Lemma 4.4 for α , with weights w_α and $w_\alpha(X_\alpha) = 2\tau_\alpha$, all of them fresh, with anchor a_α ,
- fresh goods γ_3, γ_4 .

The two halving gadgets are built independently, each with its own base b . The weights w of the goods are given by Lemma 4.1 for eight digit blocks: the six core blocks, with digit 0 on every good outside the core, and the drop block and the relief block of Section 6.3. The dial's charge χ is the negative form χ_{lex}^- on X_β (so $\chi(\bar{B}_2) = \Delta$), the lex charge χ_{lex} on X_α , and 0 on all other goods. Then $I_{\alpha,\beta} = I(\delta_V + \chi, \delta_V, \delta_U, \delta_U)$ with agents D, E^V, E_1^U, E_2^U and $M = 5\Delta$.

So $I_{\alpha,\beta}$ is a gated instance, with the drop block and the relief block as its gate blocks. Table 3 summarizes the two gates. Figure 4

Table 3: The two gates of $I_{\alpha,\beta}$.

	drop gate	relief gate
halving gadget	X_β	X_α
carriers	$B_2, \bar{B}_2$ (light, digit 1)	γ_3, γ_4 (heavy, digit τ_α)
fill a bundle each	$B_1, \bar{B}_1$	$B_2, \bar{B}_2$
anchor	$\bar{B}_2$ (label)	a_α (fresh)
dial's charge	χ_{lex}^-	χ_{lex}
interface	Lemma 6.10	Lemma 6.11
gives the dial	her drop dr	her relief rl

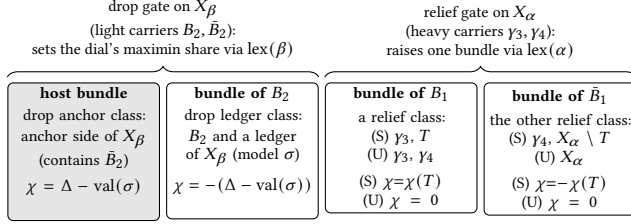


Figure 4: The gate goods of a balanced partition of $I_{\alpha,\beta}$ (Lemma 6.12), with the dial's charge χ on each bundle. Each box also holds the core goods of the bundle of the phase named in its heading. The two relief classes may also be matched with the bundles of B_1 and $\bar{B}_1$ the other way round.

shows the resulting shape of a balanced partition. Lemma 6.12 states it.

The block totals are 4 and 12 for the core, $4(\tau_\beta + 1)$ for the drop block and $8\tau_\alpha$ for the relief block, all divisible by 4. The charges δ_V, δ_U and χ each sum to 0, so the largest $|\delta_i(S)|$ is the sum of the positive values of δ_i : $4s$ for type U and $3s + 2\Delta = 5\Delta = M$ for the dial, from δ_V on $\bar{A}_2$ and $\bar{B}_2, \chi_{\text{lex}}^-$ on the host $\bar{B}_2$, and χ_{lex} on the goods P_i of X_α . Moreover, $M = 5\Delta \geq 4s$. Every good has a positive digit in some block, so its weight is positive: the core goods in the core blocks, the goods of X_β other than $\bar{B}_2$ in the drop block, and the goods of X_α and γ_3, γ_4 in the relief block. All digits have polynomially many bits by (W2), hence so do the weights by Lemma 4.1, and $K = 2M + 1$ has $O(k)$ bits. So $I_{\alpha,\beta}$ is computable in polynomial time.

The gate goods are the goods of the two gate blocks: X_β , which contains $\bar{B}_2, X_\alpha, \gamma_3, \gamma_4$, and $B_1, B_2, \bar{B}_1$. The next lemma puts the core and the two gates together.

LEMMA 6.12. *Let β be satisfiable, and let rl be as in Lemma 6.11 for α .*

- The balanced partitions of $I_{\alpha,\beta}$ are exactly the combinations, in the sense of Lemma 4.2, of a phase of the core, a configuration of the drop block and a configuration of the relief block, glued at the B-side goods: each class that contains a B-side good lies in the bundle named after it, and the two relief classes lie in the bundles of B_1 and $\bar{B}_1$, in either order. Every such combination is balanced.*
- Consequently, in every balanced partition, the host bundle holds the drop anchor class, with $\chi \geq 0$. The bundle of B_2 holds the drop ledger class, with $\chi \leq -(\Delta - \text{lex}(\beta))$. The*

bundles of B_1 and $\bar{B}_1$ hold one relief class each, with $\chi \leq \text{rl}$. In phase P^U , these three bundles are exactly Bad_V . For each phase, some balanced partition puts the drop ledger class of charge exactly $-(\Delta - \text{lex}(\beta))$ on the bundle of B_2 and a relief class of charge rl on the bundle of B_1 , and some balanced partition puts the drop ledger class of charge exactly $-(\Delta - \text{lex}(\beta))$ on the bundle of B_2 with the relief block in case (U).

6.5 Instantiation and the theorem

Our choice of the scale s is made for the calculus. With $s = \Delta$, $\text{dr} = \Delta - \text{lex}(\beta)$ and $\text{rl} = \text{lex}(\alpha)$, the core's penalty s and the Δ in the drop cancel, and only $\text{lex}(\alpha) \geq \text{lex}(\beta)$ remains.

LEMMA 6.13. *Let β be satisfiable with $\text{lex}(\beta) \geq 1$. Then $I_{\alpha,\beta}$ satisfies the hypotheses of Lemma 6.7 with $\text{dr} = \Delta - \text{lex}(\beta)$ and with rl as in Lemma 6.11 for α . Hence $\mu_{EV} = \mu_{E_1^U} = \mu_{E_2^U} = K\tau$, $\mu_D = K\tau - \Delta + \text{lex}(\beta)$, and $I_{\alpha,\beta}$ admits an MMS-fair allocation if and only if α is satisfiable and $\text{lex}(\alpha) \geq \text{lex}(\beta)$.*

PROOF. We have $0 \leq \text{dr} \leq \Delta \leq M$ and $\text{rl} \geq 0$. (drop) By Lemma 6.12(b), every balanced partition has $\chi \leq -\text{dr}$ on the bundle of B_2 , which holds the drop ledger class and is never the host bundle. Some balanced partition of phase P^V has charges $\text{dr}, -\text{dr}, 0, 0$, with the drop ledger class of charge $-\text{dr}$ and the relief gate switched off, in case (U), so it has $\chi \geq -\text{dr}$ on every bundle. (relief) By Lemma 6.12(b), in phase P^U the bundle of B_2 carries at most $-\text{dr} \leq 0 \leq \text{rl}$, and the bundles of B_1 and $\bar{B}_1$ carry at most rl. Some balanced partition of phase P^U has $\chi = \text{rl}$ on the bundle of B_1 , which lies in Bad_V .

By Lemma 6.7, an MMS-fair allocation exists if and only if $\text{rl} \geq s - \text{dr} = \text{lex}(\beta)$. If α is satisfiable, this says $\text{lex}(\alpha) \geq \text{lex}(\beta)$. Otherwise it says $0 \geq \text{lex}(\beta)$, which is false as $\text{lex}(\beta) \geq 1$. $\square$

So the two lexicographic maxima enter in different places: $\text{lex}(\beta)$ through the dial's maximin share, a maximum over partitions, and $\text{lex}(\alpha)$ through the best bundle she can be given.

THEOREM 6.14. *MMS-EXISTS is P^{NP} -complete. Hardness holds already for $n = 4$ agents with additive valuations, of which exactly three are distinct: agent E^V has $u_{EV} = K \cdot w + \delta_V$, agents E_1^U and E_2^U have $u_{E_1^U} = u_{E_2^U} = K \cdot w + \delta_U$, and agent D has $u_D = u_{EV} + \chi$.*

PROOF. Membership is Proposition 3.5(iii). For hardness, we reduce from ODD-LEX-MAX-SAT, which is P^{NP} -complete by Lemma 3.8. Given a satisfiable CNF formula φ over $z_1, \dots, z_k$, rename each z_i to z_{i+1} and add the unit clause (z_1) for a fresh most significant variable z_1 . The resulting formula φ^+ is satisfiable, $\text{lex}(\varphi^+) = 2^k + \text{lex}(\varphi) \geq 1$, and $\varphi^+ \in \text{ODD-LEX-MAX-SAT}$ if and only if $\varphi \in \text{ODD-LEX-MAX-SAT}$, as 2^k is even. Let $\beta = \varphi^+$ and $\alpha = \varphi^+ \wedge z_{k+1}$, whose unit clauses are allowed by Lemma 4.4, and map φ to $I_{\alpha,\beta}$, which is computable in polynomial time. As β is satisfiable with $\text{lex}(\beta) \geq 1$, by Lemmas 6.13 and 3.9, $I_{\alpha,\beta}$ admits an MMS-fair allocation if and only if $\varphi^+ \in \text{ODD-LEX-MAX-SAT}$, that is, if and only if $\varphi \in \text{ODD-LEX-MAX-SAT}$. The three valuations are distinct, as $\delta_V \neq \delta_U$ and $\chi(\bar{B}_2) = \Delta \geq 1$, so no valuation is held by three agents, as Remark 3.6 requires. $\square$

7 Computation with Values in Unary

This section shows that, with values in unary and the number of agents free, MMS-VALUE is $\text{OptP}[O(\log n)]$ -complete and MMS-ALLOC is $\text{FNP}/\text{OptP}[O(\log n)]$ -complete, already for identical valuations (Theorems 7.2 and 7.4).

7.1 Proof strategy

Why the setting changes. Two things change with values in unary. First, for a fixed number of agents every problem is in P (Proposition 3.7), so the number of agents must grow with the input. Second, the budget changes. The weights of the halving gadget have $\Theta((k+q)\log b)$ bits by (W2), so written in unary they are exponentially long. A charge of size 2^p written in unary is too long unless $p = O(\log n)$. So the device must be built from a bounded number of digit blocks with polynomial totals, whose weights are then polynomial by Lemma 4.1, and the graded quantity must have polynomial range, which is exactly what the value of a function in $\text{OptP}[O(\log n)]$ has.

The device. The gear block of Section 4.5 is such a device. We recall its shape. It takes a list of *gadgets*, instances of positive exactly-1 satisfiability, and has one *gear* per variable of each gadget. Its goods include an *anchor* a and, per gear, a *hub* and two *terminals*, P and N . In every configuration, the bundle containing the anchor is the *ledger*. The whole section rests on one sentence: *the ledger reads off a satisfying assignment of every gadget at once, one terminal per variable, and every other bundle holds goods of a single gadget*. The terminal of a gear that is not in the ledger lies with the gear's hub.

Charge orientation. All agents carry the charge of Section 4.5: $-\Delta_\zeta$ on the anchor and $+\zeta_i$ on the terminal P of each graded variable, the opposite orientation of the negative form χ_{lex}^- of the halving gadget. Then the ledger carries $-\Delta_\zeta + \text{val}_\zeta(\sigma)$, and every other bundle carries at least 0.

The class. Membership in $\text{FNP}/\text{OptP}[O(\log n)]$ (Proposition 3.5) is an optimum of $O(\log n)$ bits together with one NP witness. So hardness must encode a witness y , and not only a number, and must return $\perp$ when the source problem has no solution. Two devices do this. The witness is carried by the formula $\varphi \wedge \eta_b$, where η_b checks the solution relation at the level written by the machine unless $b = 0$, and the guard variable b gets the lowest graded weight. Then the optimum is $2g(x) + [F(x) \neq \emptyset]$ (Lemma 7.3), which carries both steps of Remark 3.13 in one number.

The endgame. The endgame is the graded ledger again (Lemma 4.7): (GL1) is the charge computation above, and (GL2) is the converse, part (v) of Lemma 4.5.

Where to find what. The gear block and its interface (Lemma 4.5 and Remark 4.6) are in Section 4.5, with its construction and proof in the appendix. Section 7.2 builds the instance and verifies (GL1) and (GL2) (Lemma 7.1), Section 7.3 proves Theorem 7.2, and Section 7.4 adds the witness bit and proves Theorem 7.4.

7.2 The instance

The instance is built once, for any 3-CNF formula and grading. Theorems 7.2 and 7.4 use it for two different formulas.

CONSTRUCTION (THE INSTANCE $J_{\varphi, \zeta}$). Let φ be a satisfiable 3-CNF formula and ζ a grading of its variables. Let ψ be the POS-1-IN-3-SAT instance of Lemma 3.11 for φ , with n_v variables, n_c clauses and $n_o = 3n_c$ occurrences. Apply Lemma 4.5 to the single gadget ψ , a fresh anchor a , and $n = 2n_o + n_v + 1$, and let w be given by Lemma 4.1 for its seven blocks, with n bundles. The charge χ is $\chi(a) = -\Delta_\zeta$, $\chi(P^z) = \zeta_z$ for each variable z of φ , and 0 on all other goods. Let $M = \Delta_\zeta$ and $K = 2\Delta_\zeta + 1$. Then $J_{\varphi, \zeta} = I(\chi, \dots, \chi)$ is the instance with n agents and goods G_{gear} in which all agents have the valuation $u = K \cdot w + \chi$.

The charge sums to 0, its positive values sum to Δ_ζ and its negative values to $-\Delta_\zeta$, so $|\chi(S)| \leq M$ for all S . We write σ_π for the assignment decoded by a balanced partition π as in Lemma 4.5(iii), and $\sigma_\pi|_\varphi$ for its restriction to the variables of φ .

LEMMA 7.1. *The instance $J_{\varphi, \zeta}$ satisfies (GL1) and (GL2) of Lemma 4.7 with Σ the set of exactly-1 models of ψ . Hence $\mu(J_{\varphi, \zeta}) = K\tau - \Delta_\zeta + \text{lex}_\zeta(\varphi)$, and $\sigma_\pi|_\varphi$ is a model of φ of graded value $\text{lex}_\zeta(\varphi)$ for every MMS-fair allocation π . Moreover, $J_{\varphi, \zeta}$ is computable in polynomial time, and all its values are polynomial in $|\varphi| + \Delta_\zeta$.*

7.3 The value

THEOREM 7.2. *With values in unary and the number of agents free, MMS-VALUE is $\text{OptP}[O(\log n)]$ -complete under metric reductions, and hence $\text{FNP}^{\text{NP}}[O(\log n)]$ -complete, already for identical valuations.*

PROOF. By Proposition 3.5(i), MMS-VALUE is in $\text{OptP}[O(\log n)]$, which is contained in $\text{FNP}^{\text{NP}}[O(\log n)]$ by binary search. For hardness, let f be a function in $\text{OptP}[O(\log n)]$, computed by an NP metric machine that writes exactly $p = O(\log |x|)$ bits on every branch, and let φ and ζ be as in Section 5.2, so that $\text{lex}_\zeta(\varphi) = f(x)$ by (1). Now $\Delta_\zeta = 2^p - 1$ is polynomial in $|x|$, so by Lemma 7.1, the instance $J_x = J_{\varphi, \zeta}$ is computable in polynomial time even in unary, and $\mu(J_x) = K\tau - \Delta_\zeta + f(x)$. By Corollary 4.8, the maps $T_1(x) = (J_x, 1)$ and $T_2(x, y) = y - K\tau + \Delta_\zeta$ form a metric reduction. The second claim follows by Theorem 3.3. $\square$

7.4 The witness bit, and the allocation

For MMS-ALLOC, the source is a search problem $F \in \text{FNP}/\text{OptP}[O(\log n)]$, given by a transducer T and a function $g \in \text{OptP}[O(\log n)]$ as in Definition 3.4, and the reduction must return some $y \in F(x)$, or $\perp$ if there is none. Let $R(x, c, w)$ hold if $w = (y, \omega)$ for an accepting branch ω of T on $\langle x, \text{bin}(c) \rangle$ that outputs y . Then R is a polynomial-time relation with $|w|$ polynomially bounded, and $F(x) \neq \emptyset$ if and only if some w satisfies $R(x, g(x), w)$. We reduce with these pairs, and at the end discard ω . Let g be computed by an NP metric machine that writes exactly $p = O(\log |x|)$ bits on every branch. Let φ and η_b be as in Lemma 3.10 for this machine and R , with a fresh guard variable b , where the variables $y_1, \dots, y_{p'}$ of Lemma 3.10, the witness variables, encode w , and with the variables of η_b other than $z_1, \dots, z_p$, the witness variables and b fresh. Let $\varphi' = \varphi \wedge \eta_b$, which is a 3-CNF formula. We grade $z_1, \dots, z_p$ and b by

$$\zeta'_i = 2^{p+1-i} \text{ for } i \leq p, \quad \zeta'_b = 1, \quad \text{so that } \Delta_{\zeta'} = 2^{p+1} - 1,$$

and all other variables by 0, so $\text{val}_{\zeta'}(\sigma) = 2c + \sigma(b)$, where c is the number with bits $\sigma(z_1), \dots, \sigma(z_p)$.

LEMMA 7.3. $\text{lex}_{\zeta'}(\varphi') = 2g(x) + [F(x) \neq \emptyset]$. Every model σ of φ' of this graded value has $\sigma(b) = [F(x) \neq \emptyset]$, and if $\sigma(b) = 1$, then the witness variables of σ form a pair $w = (y, \omega)$ with $R(x, g(x), w)$, and $y \in F(x)$.

PROOF. Setting $b = 0$ leaves the z_i and y_j unconstrained by η_b , and $b = 1$ is possible together with the output bits of a number c exactly if $R(x, c, w)$ holds for some w . So the largest graded value of a model with output bits c is $2c + [\exists w R(x, c, w)]$, for every number c written on a branch. The largest such c is $g(x)$, and $2c + 1 < 2g(x)$ for every smaller c , so the maximum is $2g(x) + [\exists w R(x, g(x), w)] = 2g(x) + [F(x) \neq \emptyset]$, and it is attained only with output bits $g(x)$ and $b = [F(x) \neq \emptyset]$. If $b = 1$, then η_b forces $R(x, g(x), w)$ for the string w of witness variables, so $w = (y, \omega)$ with $y \in F(x)$. $\square$

The guard is needed because F need not have solutions for every x . Without it, that is, with the unguarded formula η of Section 3.4 in place of η_b , the formula $\varphi \wedge \eta$ might still be satisfiable at a level c below $g(x)$, and the decoding would return a y certified at the wrong level instead of $\perp$. It might also be unsatisfiable, so that no balanced partition exists at all. The high bits locate $g(x)$ whatever F is, and the low bit says whether a solution exists at that level.

THEOREM 7.4. *With values in unary and the number of agents free, MMS-ALLOC is FNP//OptP[$O(\log n)$]-complete, already for identical valuations.*

PROOF. Membership is Proposition 3.5(ii). For hardness, let F be given by a transducer T and g as in Definition 3.4, and let $R(x, c, w)$ hold if $w = (y, \omega)$ for an accepting branch ω of T on $\langle x, \text{bin}(c) \rangle$ that outputs y . Then R is a polynomial-time relation with $|w|$ polynomially bounded, and $F(x) \neq \emptyset$ if and only if some w satisfies $R(x, g(x), w)$. We reduce with the construction of Section 7.4 for this R , and at the end discard ω . Let $T_1(x) = J_{\varphi', \zeta'}$, which is computable in polynomial time in unary by Lemma 7.1, as $\Delta_{\zeta'}$ is polynomial. On it, $\perp$ is not a correct output of MMS-ALLOC (Corollary 4.8). For a correct output π , let $\sigma = \sigma_\pi|_{\varphi'}$, read off the ledger. By Lemma 7.1, σ is a model of φ' of graded value $\text{lex}_{\zeta'}(\varphi')$. Let $T_2(x, \pi) = \perp$ if $\sigma(b) = 0$, and otherwise let it be the y -part of the pair $w = (y, \omega)$ formed by the witness variables of σ . By Lemma 7.3, $T_2(x, \pi) \in F(x)$ if $F(x) \neq \emptyset$, and $T_2(x, \pi) = \perp$ otherwise. That is, $T_2(x, \pi) \in \text{Sol}_F(x)$ for every correct output π , as Definition 3.2 requires. As $\perp$ is never a correct output on $T_1(x)$, this is also a reduction in the sense of Chen and Toda [9, Def. 2.2]. $\square$

Table 4 sets the two computational constructions side by side. Only two differences are choices: the device changes, because a halving gadget written in unary is too long, and the range of Δ_{ζ} changes, because a charge of size 2^p written in unary is too long as well. The rest follows from the device: the gear block buys its rigidity by counting bundles rather than by the size of its weights, so it needs $2n_o + n_v + 1$ agents where the halving gadget needs two.

8 Existence with Values in Unary

This section shows that, with values in unary and the number of agents free, MMS-EXISTS is $\text{P}^{\text{NP}}[O(\log n)]$ -complete (Theorem 8.8).

Table 4: The two computational constructions side by side.

	binary (Sec. 5)	unary (Sec. 7)
endgame	graded ledger, $\mu = K\tau - \Delta_{\zeta} + \max_{\sigma} \text{val}_{\zeta}(\sigma)$	
valuations	identical (no core, no dial, no gate)	
constants	$M = \Delta_{\zeta}, K = 2\Delta_{\zeta} + 1$	
device	halving gadget	gear block
blocks	1, base $b \leq 7$	7, polynomial totals
source	3-CNF φ	Pos-1-IN-3-SAT instance ψ
Δ_{ζ}	$2^p - 1$, exponential	$2^p - 1, p = O(\log n)$
agents	2	$2n_o + n_v + 1$
ledger	the half without a	the bundle of a
charge	$-\zeta_i$ on $N_i, +\Delta_{\zeta}$ on a	$+\zeta_i$ on $P^{z_i}, -\Delta_{\zeta}$ on a
size	$O(p + k + q)$ bits	polynomial values
value	OptP	OptP[$O(\log n)$]
allocation	FP^{NP}	$\text{FNP}/\text{OptP}[O(\log n)]$

8.1 Proof strategy

Same contract, one bit per gate. The calculus (Lemma 8.2) says that a gated instance admits an MMS-fair allocation if and only if $\text{rl}_e \geq s - \text{dr}_e$ for some dial D_e . Section 6 made drop and relief lexicographic maxima, but the budget of Section 7.1 forbids this with values in unary: a charge of size 2^k written in unary is too long. Instead, each gate carries one bit. With $A = \text{Pos-1-IN-3-SAT}$, we take

$$s = 4, \quad \text{dr}_e = 2 \cdot [\beta_e \notin A], \quad \text{rl}_e = 2 \cdot [\alpha_e \in A]$$

for instances α_e, β_e of A . As $\text{dr}_e, \text{rl}_e \in \{0, 2\}$, the inequality $\text{rl}_e \geq 4 - \text{dr}_e$ holds exactly if both are 2, so the condition reads: $\alpha_e \in A$ and $\beta_e \notin A$ for some e . This is the idea of Section 6 in another encoding: there, a large $\text{lex}(\beta)$ gave a small drop and made fairness harder, and here $\beta_e \in A$ likewise gives drop 0 and makes fairness impossible for that dial. In both constructions, fairness asks that her relief covers the part of the cost s that her drop does not. This is a disjunction over polynomially many dials, and on chains this is exactly the parity condition of Wagner's criterion (Lemma 3.12), as the members of A form a prefix.

Why the number of agents grows. For a fixed number of agents the problem is in P (Proposition 3.7). Here the core Σ_m of Section 6.2 has m set by the size of the formulas, with $n = 2m$ agents, of which $r \leq m - 1$ are dials and the others blockers. The blockers let the number of bundles exceed the number of dials, which the gates need, as each gate occupies many bundles.

The device. The gates are built on the gear block of Section 4.5, now with $2r$ gadgets, two per dial, all sharing one ledger. The anchor is the host $\bar{B}_2$, a core good that the block borrows (Remark 4.6), so the ledger Λ is the host bundle, which is safe. Locality replaces the pinning of the relief classes, and the host replaces the pinning of the drop anchor class (Section 6.4). Locality (Lemma 4.5(ii)) says that charges on different gadgets never meet in a bundle other than the ledger. The ledger, where charges do meet, is pinned to the host bundle, outside Bad_V .

The wildcard. The gear block has a configuration only if every gadget is satisfiable, and without a configuration there is no balanced partition at all. So we make every gadget satisfiable by adding

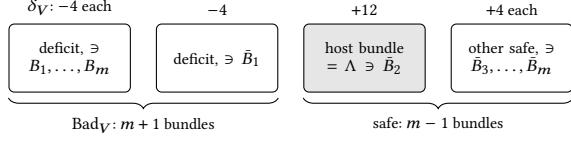


Figure 5: The $2m$ bundles of J in phase P^U , with the offsets of δ_V at $s = 4$. The host bundle is the ledger Λ of the gear block, as its anchor is the host $\bar{B}_2$. Dial D_e carries two charges: -2 on N_e^β and $+2$ on P_e^α . Each lies in Λ or in a bundle of its own gadget (Lemma 8.6). The -2 must leave Λ if $\beta_e \notin \text{Pos-1-IN-3-SAT}$, and the $+2$ can leave it only if $\alpha_e \in \text{Pos-1-IN-3-SAT}$. The existence question is whether some dial's $+2$ can reach a bundle of Bad_V while her -2 has left the ledger.

a fresh variable w , the *wildcard*, to each of its clauses: $w = 1$ always gives a model, and $w = 0$ gives one if and only if the original instance is satisfiable. This plays the role of the heavy carriers of Section 6.3, which let the relief gate be switched off.

The gates as two charges. Dial D_e carries -2 on the terminal N_w of the wildcard of her β_e -gadget, and $+2$ on the terminal P_w of the wildcard of her α_e -gadget, and nothing else. The -2 lies in the ledger if w is decoded as false, which is possible exactly if $\beta_e \in A$, and otherwise in the hub bundle of that gear. The $+2$ lies in the ledger if w is decoded as true, which is always possible, and otherwise, which is possible exactly if $\alpha_e \in A$, in the hub bundle of that gear, which may be placed on any bundle other than the ledger. So the $+2$ can reach a bad bundle only if $\alpha_e \in A$, and the -2 can join the $+2$ in the ledger, where they cancel, only if $\beta_e \in A$. Figure 5 shows the bundles of phase P^U .

Where to find what. Section 8.2 generalises the core, gated instances and the calculus to $2m$ agents and many dials (Lemmas 8.1 and 8.2). Section 8.3 adds the wildcard (Lemma 8.4). Section 8.4 builds the instance J and bounds its constants (Lemma 8.5). Section 8.5 states the two gates (Lemma 8.6) and checks the calculus (Lemma 8.7), and Section 8.6 proves Theorem 8.8.

8.2 The core for general m

The core of Section 6.2 has four agents, one of them a dial. Here we need $2m$ agents, with m growing with the input, and many dials. We generalise each ingredient in turn, and the gates will again attach at the single host $\bar{B}_2$. For $i \leq m$, let $i^+ = i + 1$ if $i < m$ and $m^+ = 1$, the cyclic successor.

The cycle is now $A_1 B_1 A_2 B_2 \cdots A_m B_m$, in which V_i joins A_i to B_i and U_i joins B_i to A_{i^+} . It still has exactly two perfect matchings, $\{V_1, \dots, V_m\}$ and $\{U_1, \dots, U_m\}$.

CONSTRUCTION (THE CORE Σ_m). The core goods are the $2m$ vertices A_i, B_i and the $2m$ edges V_i, U_i of the cycle, for $i \leq m$, and a shadow $\bar{A}_i, \bar{B}_i$ of each vertex, $6m$ goods in all. They carry six digit blocks, with digit 0 on all goods not listed: side_A is 1 on the A -side goods $A_i, \bar{A}_i$, side_B is 1 on the B -side goods $B_i, \bar{B}_i$, edg is 1 on the

edges, and rol is 1 on A_i and $\bar{B}_i$, and the link blocks are

$$\begin{aligned} L_1 : A_i, \bar{A}_i &\mapsto i, & V_i &\mapsto m+1-i, & U_i &\mapsto m+1-i^+, \\ L_2 : B_i, \bar{B}_i &\mapsto i, & V_i, U_i &\mapsto m+1-i. \end{aligned}$$

Their totals are $2m$ for the first four blocks and $2m(m+1)$ for the link blocks, so that, with $n = 2m$ bundles, a bundle meets them if its digit sums are $(1, 1, 1, 1, m+1, m+1)$. For $m = 2$, this is the core Σ_2 of Section 6.2 (Table 8 in the appendix).

The phases are

$$P^V : \text{bundles } \{A_i, B_i, V_i\} \text{ and } \{\bar{A}_{i^+}, \bar{B}_i, U_i\} \text{ for } i \leq m,$$

$$P^U : \text{bundles } \{A_{i^+}, B_i, U_i\} \text{ and } \{\bar{A}_i, \bar{B}_i, V_i\} \text{ for } i \leq m,$$

and, as before, each bundle contains exactly one of the goods B_i and $\bar{B}_i$, and we name it after it. The perturbations are

$$\delta_V : \bar{A}_2 \mapsto 2s, \bar{B}_1 \mapsto -s, \bar{B}_i \mapsto s \ (i \geq 2), U_i \mapsto -s,$$

$$\delta_U : \bar{A}_2 \mapsto -2s, \bar{B}_2 \mapsto 3s, \bar{B}_i \mapsto s \ (i \neq 2), V_i \mapsto -s,$$

and 0 on all other goods. For $m = 2$ they are those of Section 6.2. The deficit bundles are now the $m+1$ bundles of $B_1, \dots, B_m$ and $\bar{B}_1$, and the safe bundles, in the sense of Section 6.2, are the $m-1$ bundles of $\bar{B}_2, \dots, \bar{B}_m$. The host bundle is still the bundle of the host $\bar{B}_2$. As before, Bad_V and Bad_U are the sets of deficit bundles of P^U and of P^V .

LEMMA 8.1. Let $m \geq 2$.

- The partitions of the core goods of Σ_m into $2m$ bundles that meet all six blocks are exactly P^V and P^U .
- Both δ_V and δ_U sum to 0. The perturbation δ_V has offset 0 on every bundle of P^V , and on P^U it has offset $-s$ on every deficit bundle, $3s$ on the host bundle, and s on the other safe bundles. Symmetrically, δ_U has offset 0 on every bundle of P^U , and on P^V the same offsets as δ_V on P^U . In particular, the bundles of $\bar{B}_2, \dots, \bar{B}_m$ are safe.
- Let there be m agents of each type. If the bundles of P^U are handed to the $2m$ agents, one each, some type- V agent receives a bundle of Bad_V . The same holds for P^V , type U and Bad_U . For each type- V agent i and every $S \in \text{Bad}_V$, the bundles of P^U can be handed out so that i receives S , the other type- V agents receive safe bundles, and every agent other than i receives a bundle of offset at least 0 for her perturbation.
- With m agents of each type and $M = (m+2)s$, the instance $I(\delta_V, \dots, \delta_V, \delta_U, \dots, \delta_U)$ on the core goods has $\mu_i = K\tau$ for every agent i , and admits no MMS-fair allocation.

Parts (a), (b) and (d) are proved in the appendix, and for $m = 2$ they are Lemmas 6.1 and 6.2 and Proposition 6.4. Part (c) is proved as Lemma 6.3: the $m+1$ bundles of Bad_V cannot all go to the m type- U agents. For the second claim, give S to i , the $m-1$ safe bundles to the other type- V agents, and the m remaining bundles to the type- U agents, on which δ_U is 0.

Agents and gated instances. The $n = 2m$ agents are m agents of type V and m of type U . Among the type- V agents, $r \leq m-1$ are dials $D_1, \dots, D_r$, and dial D_e carries an additional charge χ_e on top of δ_V . All other agents are blockers. A gated instance over Σ_m is defined as in Definition 6.5, with the core Σ_m , these $2m$ agents, and condition (G3) for each of the charges $\chi_1, \dots, \chi_r$. For $m =$

2 and $r = 1$, it is a gated instance as before. Lemma 6.6 holds for gated instances over Σ_m verbatim, with the same proof, using Lemma 8.1(a) in place of Lemma 6.1.

The contract. With several dials, the calculus gains one condition: handing a bundle of Bad_V to one dial must leave every other dial at or above her maximin share. For a partition π , we write $\chi_e(\pi) = \min_j \chi_e(\pi(j))$. The (drop) condition changes in phase P^U for the following reason. For $m \geq 3$, the safe bundles other than the host bundle have δ_V -offset $+s$ in phase P^U , so a small charge on them does not bound the dial's offset, and (drop) speaks about Bad_V instead of about the bundles other than the host bundle. For $m = 2$, the new (drop) follows from (drop) of Lemma 6.7, as every bundle of P^U other than the host bundle lies in Bad_V .

LEMMA 8.2. *Let I be a gated instance over Σ_m with dials $D_1, \dots, D_r$. For each dial D_e , let dr_e and rl_e be integers with $0 \leq \text{dr}_e \leq M$ such that the following two conditions hold.*

- (drop) *Balanced partitions of phase P^V exist, and some balanced partition of phase P^V has $\chi_e \geq -\text{dr}_e$ on every bundle. Every balanced partition of phase P^V has $\chi_e \leq -\text{dr}_e$ on some bundle, and every balanced partition of phase P^U has $\chi_e \leq s - \text{dr}_e$ on some bundle in Bad_V .*
- (relief) *As in Lemma 6.7 for χ_e and rl_e , where the balanced partition π with $\chi_e(S) = \text{rl}_e$ has in addition $\chi_{e'}(\pi) \geq -\text{dr}_{e'}$ for every other dial $D_{e'}$.*

Then every blocker has maximin share $K\tau$, every dial D_e has maximin share $K\tau - \text{dr}_e$, and I admits an MMS-fair allocation if and only if $\text{rl}_e \geq s - \text{dr}_e$ for some e .

PROOF. The proof of Lemma 6.7 applies, with Lemma 8.1 in place of Lemmas 6.2 and 6.3. The maximin shares are computed for each dial separately. For the drop of D_e : in phase P^V the bundle of (drop) has offset $\chi_e \leq -\text{dr}_e$, and in phase P^U the bundle of (drop) in Bad_V has offset $-s + \chi_e \leq -\text{dr}_e$. In $(\Rightarrow)$, the type- V agent who receives a bundle $S \in \text{Bad}_V$ is not a blocker, so she is some dial D_e , and the same computation gives $\text{rl}_e \geq s - \text{dr}_e$. In $(\Leftarrow)$, let $\text{rl}_e \geq s - \text{dr}_e$ and $S \in \text{Bad}_V$, take the balanced partition π of phase P^U given by (relief), and hand it out as in Lemma 8.1(c), with D_e receiving S . Then D_e gets offset $-s + \text{rl}_e \geq -\text{dr}_e$ and every blocker gets offset at least 0, as before. The only new case is another dial $D_{e'}$, who receives a safe bundle, of offset at least $0 + \chi_{e'}(\pi) \geq -\text{dr}_{e'}$. $\square$

8.3 The wildcard

Definition 8.3. For a positive exactly-1 instance ψ and a fresh variable w , let $\psi^w = \{C \cup \{w\} : C \text{ a clause of } \psi\}$. We call w the *wildcard* of ψ^w .

LEMMA 8.4. (a) ψ^w is satisfiable. (b) ψ^w has an exactly-1 model with w false if and only if ψ is satisfiable.

PROOF. (a) Set w true and every other variable false: each clause then contains exactly the true variable w . (b) With w false, each clause $C \cup \{w\}$ has exactly one true variable if and only if C has. $\square$

We apply this to POS-1-IN-3-SAT instances, after two harmless normalisations: we delete the variables that occur in no clause, and we replace an instance without clauses, which is a yes-instance, by the yes-instance $\{\{a, b, c\}\}$. Then every instance has a clause, every

Table 5: The two gates of dial D_e in J , with $A = \text{Pos-1-IN-3-SAT}$ and w the decoded value of the wildcard, that is, the wildcard mode of the gadget.

	threshold gate	relief gate
gadget	β_e^w	α_e^w
dial's charge	-2 on N_e^β	$+2$ on P_e^α
in Λ if	$w = 0$ (iff $\beta_e \in A$)	$w = 1$ (always)
off Λ if	$w = 1$ (always)	$w = 0$ (iff $\alpha_e \in A$)
interface	Lemma 8.6	Lemma 8.6
gives the dial	drop $\text{dr}_e = 2[\beta_e \notin A]$	relief $\text{rl}_e = 2[\alpha_e \in A]$

variable occurs in a clause, and ψ^w is a positive exactly-1 instance with clauses of four variables in which every variable occurs, as the gear block requires.

8.4 The instance

CONSTRUCTION (THE INSTANCE J). Let $(\alpha_1, \beta_1), \dots, (\alpha_r, \beta_r)$ be normalised POS-1-IN-3-SAT instances. For each e , let the β_e -gadget be $\psi_{\beta_e}^w = \beta_e^w$ and the α_e -gadget be $\psi_{\alpha_e}^w = \alpha_e^w$, each with its own fresh wildcard w , and write P_e^β, N_e^β and H_e^β for the terminals and the hub of the wildcard's gear in the β_e -gadget, and P_e^α, N_e^α and H_e^α likewise for α_e . Let $N = 2n_o + n_v + 1$, where n_v and n_o are the total numbers of variables and occurrences of the $2r$ gadgets. A gadget with v variables and o occurrences contributes $2o + v$ to N , so N has the parity of $n_v + 1$. If N is odd, append one further gadget, the padding gadget, the normalised instance $\psi_{\text{pad}} = \{\{a, b, c\}, \{a, b, d\}\}$ on four fresh variables with its own fresh wildcard. It is satisfiable ($c = d = 1$ and all else 0), every variable occurs, and with its wildcard it has 5 variables and 8 occurrences, so it adds 21 to N and makes N even. From now on, n_v and n_o count all gadgets, the padding gadget included, and $N = 2n_o + n_v + 1$ is even. Let $m = N/2$. Each of the $2r \geq 2$ gadgets has, with its wildcard, at least 4 variables and 4 occurrences, so $N \geq 2 \cdot 8 + 8 + 1 = 25$, and $m \geq 13 \geq 2$. Then J is the gated instance over Σ_m with

- scale $s = 4$ and r dials,
- as gate blocks, the seven blocks of the gear block of Lemma 4.5 for the $2r$ gadgets and the padding gadget, if any, with $n = 2m = 2n_o + n_v + 1$ bundles and anchor $a = \bar{B}_2$,
- for each dial D_e the charge χ_e with $\chi_e(N_e^\beta) = -2$, $\chi_e(P_e^\alpha) = +2$, and 0 elsewhere,
- $M = (m + 2)s$ and $K = 2M + 1$.

The padding gadget is attached to no dial and carries no charge. Lemma 8.6 applies to it like to any other gadget, and in the proof of Lemma 8.7 it is put in wildcard mode $w = 1$ (Section 8.5). The number of dials is within the budget of Section 8.2: each gadget has a variable, so $n_v \geq 2r$ and $2m \geq n_v + 1 \geq 2r + 1$, which gives $r \leq m - 1$. Table 5 summarizes the two gates of a dial.

LEMMA 8.5. J is a gated instance over Σ_m : in particular, $|\delta_i(S)| \leq M$ for every agent i and every bundle S . Its values are polynomial in the total size of the instances α_e, β_e , so J is computable in polynomial time with values in unary.

8.5 The gates and the calculus

The gear block is built once, over all dials, with the host $\bar{B}_2$ as its anchor. By Lemma 6.6, its ledger Λ , the bundle of the host, lies in the host bundle, so Λ is the host bundle, which is safe and not in Bad_V . By Lemma 4.5(ii)–(iv), for each gadget the wildcard’s terminal P_w lies in Λ exactly if the decoded assignment sets w true, and N_w lies in Λ exactly if it sets w false. The other terminal lies in the bundle of the hub H_w of that gadget, a bundle other than Λ . Reading this off with Lemma 8.4 gives the following lemma, which contains both gates. We say that a gadget is in *wildcard mode* $w = 1$ or $w = 0$ according to the decoded value of its wildcard. So wildcard mode $w = 1$ is V -mode of the wildcard’s gear, and $w = 0$ is U -mode (Section 4.5).

LEMMA 8.6. *Let ψ^w be a gadget of J with wildcard w , hub H^w and terminals P^w, N^w , for a normalised Pos-1-IN-3-SAT instance ψ . In every balanced partition of J , exactly one of P^w, N^w lies in Λ , and the other lies in the bundle of H^w , which is not Λ and holds no other terminal. The gadget can be in wildcard mode $w = 1$ (with $P^w \in \Lambda$) in some balanced partition always, and in wildcard mode $w = 0$ (with $N^w \in \Lambda$) in some balanced partition if and only if $\psi \in \text{Pos-1-IN-3-SAT}$. The wildcard modes of all gadgets, the phase, and the matching of the bundles of the gear block other than Λ to the bundles of the phase other than the host bundle can be chosen independently.*

The lemma follows from Lemma 4.5(i)–(v), Lemma 8.4 and Lemma 6.6. The bundle of H^w holds no other terminal as it holds exactly one edge by locality, and deleting unused variables changes no satisfiability. Read off for the two charges of dial D_e , it gives the two gates of Table 5. The *threshold gate*: the -2 of D_e , on N_e^β , lies in Λ if and only if the β_e -gadget is in wildcard mode $w = 0$, which is possible if and only if $\beta_e \in \text{Pos-1-IN-3-SAT}$, and otherwise in the bundle of the hub H_e^β . The *relief gate*: the $+2$ of D_e , on P_e^α , lies in Λ if and only if the α_e -gadget is in wildcard mode $w = 1$, which is always possible, and otherwise, which is possible if and only if $\alpha_e \in \text{Pos-1-IN-3-SAT}$, in the bundle of the hub H_e^α . *Interface*: the rest of this section uses the gates only through Lemma 8.6 and the locality of Lemma 4.5(i).

LEMMA 8.7. *J satisfies the hypotheses of Lemma 8.2 with $s = 4$, $\text{dr}_e = 2 \cdot [\beta_e \notin \text{Pos-1-IN-3-SAT}]$ and $\text{rl}_e = 2 \cdot [\alpha_e \in \text{Pos-1-IN-3-SAT}]$. Hence every blocker has maximin share $K\tau$, every dial D_e has maximin share $K\tau - \text{dr}_e$, and J admits an MMS-fair allocation if and only if $\alpha_e \in \text{Pos-1-IN-3-SAT}$ and $\beta_e \notin \text{Pos-1-IN-3-SAT}$ for some e .*

Locality is used twice in the proof of Lemma 8.7: in (drop), so that the relief cannot cancel the drop outside the ledger, and in (relief), so that the two charges of a dial never meet in a bad bundle. The host is used once, in (relief): the ledger, where the always-possible wildcard mode of the relief gate puts its $+2$, is the host bundle, which is never a deficit bundle.

8.6 The theorem

THEOREM 8.8. *With values in unary and the number of agents free, MMS-EXISTS is $\text{P}^{\text{NP}}[O(\log n)]$ -complete. Hardness holds for instances with $2m$ agents holding $r + 2$ distinct valuations, for $r \leq m - 1$: m agents of type U share one valuation, $m - r$ blockers of type V share*

Table 6: The two existence constructions side by side, with $A = \text{Pos-1-IN-3-SAT}$.

	binary (Sec. 6)	unary (Sec. 8)
core	Σ_m : six blocks, $6m$ goods, $2m$ agents, two phases	
types	m agents each, $ \text{Bad}_V = m + 1$	
contract	Lemmas 6.7 and 8.2: fair iff $\text{rl}_e \geq s - \text{dr}_e$ for some e	
m , agents	$m = 2$, four agents	set by the formula size
dials	one	$r \leq m - 1$
scale s	Δ	4
device	two halving gadgets	one gear block
gate blocks	2, one per formula	7, shared by all gadgets
coupling	B -side labels	host $\bar{B}_2$ and locality
switching off	heavy carriers	the wildcard
drop dr_e	$\Delta - \text{lex}(\beta)$	$2 \cdot [\beta_e \notin A]$
relief rl_e	$\text{lex}(\alpha)$, or 0	$2 \cdot [\alpha_e \in A]$
condition	$\text{lex}(\alpha) \geq \text{lex}(\beta)$	$\alpha_e \in A, \beta_e \notin A$
source	ODD-LEX-MAX-SAT	Wagner’s criterion
valuations	three	$r + 2$
result	P^{NP} -complete	$\text{P}^{\text{NP}}[O(\log n)]$ -complete

another, and each of the r dials has her own. For every fixed number of agents, the problems are in P .

PROOF. Membership is Proposition 3.5(iii), and the last claim is Proposition 3.7. For hardness, we apply Lemma 3.12 with $A = \text{Pos-1-IN-3-SAT}$, which is NP-complete, and $B = \text{MMS-EXISTS}$. Given Pos-1-IN-3-SAT instances $x_1, \dots, x_{2k}$, normalise them as in Section 8.3, let $(\alpha_i, \beta_i) = (x_{2i-1}, x_{2i})$ for $i \leq k$ and $r = k$, and let $f(x_1, \dots, x_{2k})$ be the instance J , which is computable in polynomial time with values in unary by Lemma 8.5. By Lemma 8.7, J admits an MMS-fair allocation if and only if $x_{2i-1} \in \text{Pos-1-IN-3-SAT}$ and $x_{2i} \notin \text{Pos-1-IN-3-SAT}$ for some i , which for chains is the case if and only if $|\{j : x_j \in \text{Pos-1-IN-3-SAT}\}|$ is odd: on a chain, $x_{j+1} \in \text{Pos-1-IN-3-SAT}$ implies $x_j \in \text{Pos-1-IN-3-SAT}$, so the members of Pos-1-IN-3-SAT are a prefix $x_1, \dots, x_h$ with $h = |\{j : x_j \in \text{Pos-1-IN-3-SAT}\}|$. Hence $x_{2i-1} \in \text{Pos-1-IN-3-SAT}$ and $x_{2i} \notin \text{Pos-1-IN-3-SAT}$ holds for some i if and only if $h = 2i - 1$ is odd. So MMS-EXISTS is $\text{P}^{\text{NP}}[O(\log n)]$ -hard by Lemma 3.12. The dials’ valuations are pairwise distinct and differ from the blockers’, as their charges are nonzero and lie on different goods. Each of the two blocker valuations is held by at most $m \leq n - 2$ agents, as Remark 3.6 requires. $\square$

Table 6 sets the two existence constructions side by side. Above the rule, both use the same objects. Below it, the gates differ in how much information they carry.

9 AI Tool Disclosure

This section applies to the first version of this paper, posted on projectdiderot.com on 2 October 2026. The itemized AI Tool Disclosure certificate is on the paper’s page there.

Tools and timeline. The results were obtained in several long interactive sessions between 9 and 13 September 2026 with Claude Fable 5.1 and Claude Opus 5 (Anthropic), through the claude.ai chat interface and through Claude Code, with web search enabled. A complete proof was first obtained on 13 September 2026. The

many sessions after that, with Claude Opus 5 and Opus 5.5, up to 2 October 2026, reshaped its presentation.

Division of work. The human author chose the problems studied and the source problems of the reductions, and provided the overall proof strategies and the ideas for which kinds of gadgets could be used. The AI models searched for gadgets that satisfy these requirements, and worked out how to combine them into the reductions. The results were reached in steps: hardness for NP, then for coNP, for DP, for the levels of the Boolean hierarchy, and finally for P^{NP} and $P^{NP}[O(\log n)]$. This paper contains only the final results, not the intermediate ones. All text in this paper was written by Claude Opus 5 and 5.5, following a long sequence of feedback and detailed requests for changes by the human author. The presentation is human-curated and AI-written. The concurrent work of Gupta and Seetharaman [15] was seen only after the proofs had been found.

Verification status. The human author has checked every step of the proofs in the main text (Sections 3–8). The proofs in Appendix A have not yet been checked by the human author. All four completeness results rely on statements proved there, among them membership (Proposition 3.5) and the gear block (Lemma 4.5). Independent verification is welcome, in particular of the proofs in the appendix.

Collaboration. The human author aims to produce a fully human-written version of these results and their proofs, for submission to a traditional venue, and is open to working with others on understanding, checking and communicating them. The author can be reached through the paper’s page on projectdiderot.com.

References

- [1] Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu. 2023. Fair division of indivisible goods: Recent progress and open questions. *Artificial Intelligence* 322 (2023), 103965.
- [2] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity – A Modern Approach*. Cambridge University Press. I–XXIV, 1–579 pages.
- [3] Haris Aziz, Gerhard Rauchecker, Guido Schryen, and Toby Walsh. 2017. Algorithms for Max-Min Share Fair Allocation of Indivisible Chores. In *AAAI 2017*. AAAI Press, 335–341.
- [4] Sylvain Bouveret, Yann Chevaleyre, and Nicolas Maudet. 2016. Fair Allocation of Indivisible Goods. In *Handbook of Computational Social Choice*. Cambridge University Press, 284–310.
- [5] Sylvain Bouveret and Jérôme Lang. 2008. Efficiency and Envy-freeness in Fair Division of Indivisible Goods: Logical Representation and Complexity. *Journal of Artificial Intelligence Research* 32 (2008), 525–564.
- [6] Sylvain Bouveret and Michel Lemaître. 2016. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems* 30, 2 (2016), 259–290.
- [7] Xiaolin Bu and Biaoshuai Tao. 2026. Fair and Efficient Allocations: Decision Problems in the Gap of Polynomial Hierarchy. arXiv:2609.31849 [cs.GT]
- [8] Eric Budish. 2011. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy* 119, 6 (2011), 1061–1103.
- [9] Zhi-Zhong Chen and Seinosuke Toda. 1995. The Complexity of Selecting Maximal Solutions. *Information and Computation* 119, 2 (1995), 231–239.
- [10] Stephen A. Cook. 1971. The Complexity of Theorem-Proving Procedures. In *STOC 1971*. ACM, 151–158.
- [11] Bart de Keijzer, Sylvain Bouveret, Tomas Klos, and Yingqian Zhang. 2009. On the Complexity of Efficiency and Envy-Freeness in Fair Division of Indivisible Goods with Additive Preferences. In *ADT 2009 (LNCS, Vol. 5783)*. Springer, 98–110.
- [12] Uriel Feige, Ariel Sapir, and Laliv Tauber. 2021. A Tight Negative Example for MMS Fair Allocations. In *WINE 2021 (LNCS, Vol. 13112)*. Springer, 355–372. Full version: arXiv:2104.04977.
- [13] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

- [14] William I. Gasarch, Mark W. Krentel, and Kevin J. Rappoport. 1995. OptP as the Normal Behavior of NP-Complete Problems. *Mathematical Systems Theory* 28, 6 (1995), 487–514.
- [15] Sushmita Gupta and Sanjay Seetharaman. 2026. On the Hardness of Maximin Share Allocations. arXiv:2609.15841 [cs.GT]
- [16] Tobias Heinen, Nhan-Tam Nguyen, Trung Thanh Nguyen, and Jörg Rothe. 2018. Approximation and complexity of the optimization and existence problems for maximin share, proportional share, and minimax share allocation of indivisible goods. *Autonomous Agents and Multi-Agent Systems* 32, 6 (2018), 741–778.
- [17] Mark W. Krentel. 1988. The Complexity of Optimization Problems. *J. Comput. System Sci.* 36, 3 (1988), 490–509.
- [18] David Kurokawa, Ariel D. Procaccia, and Junxing Wang. 2018. Fair Enough: Guaranteeing Approximate Maximin Shares. *J. ACM* 65, 2 (2018), 8:1–8:27.
- [19] Leonid A. Levin. 1973. Universal Sequential Search Problems. *Problems of Information Transmission* 9, 3 (1973), 265–266.
- [20] Zbigniew Lonc and Mirosław Truszczynski. 2020. Maximin Share Allocations on Cycles. *Journal of Artificial Intelligence Research* 69 (2020), 613–655.
- [21] Trung Thanh Nguyen and Jörg Rothe. 2023. Complexity Results and Exact Algorithms for Fair Division of Indivisible Items: A Survey. In *IJCAI 2023*. ijcai.org, 6732–6740.
- [22] Christos H. Papadimitriou and Mihalis Yannakakis. 1984. The Complexity of Facets (and Some Facets of Complexity). *J. Comput. System Sci.* 28, 2 (1984), 244–259.
- [23] Thomas J. Schaefer. 1978. The Complexity of Satisfiability Problems. In *STOC 1978*. ACM, 216–226.
- [24] Michael Sipser. 2012. *Introduction to the Theory of Computation* (3rd ed.). Cengage Learning, Boston, MA.
- [25] Larry J. Stockmeyer. 1976. The Polynomial-Time Hierarchy. *Theoretical Computer Science* 3, 1 (1976), 1–22.
- [26] Klaus W. Wagner. 1990. Bounded Query Classes. *SIAM J. Comput.* 19, 5 (1990), 833–846.
- [27] Gerhard J. Woeginger. 1997. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters* 20, 4 (1997), 149–154.

A Omitted Proofs

The proofs are given in the order of the sections.

A.1 Proofs for Section 3

The proofs of Lemmas 3.8 and 3.11 are standard. We include them for completeness, and to show that they give the exact form stated in Section 3.4.

PROOF OF PROPOSITION 3.5. (i) An NP metric machine guesses a partition π and writes $\min_j u_i(\pi(j))$. The largest number written is μ_i . With values in unary, $\mu_i \leq u_i(\mathcal{G})$ is polynomial in the input size, so it has $O(\log |I|)$ bits.

(ii) and (iii), binary. Deciding whether $\mu_i \geq k$ for a given k is in NP, and $0 \leq \mu_i \leq u_i(\mathcal{G})$, so each μ_i can be computed by binary search with polynomially many oracle calls. Given the μ_i , the question whether some MMS-fair allocation assigns given goods to given agents is in NP. One such query, with no goods assigned, decides existence, which gives (iii). For (ii), if an MMS-fair allocation exists, the goods are assigned one at a time, each to the first agent for which the query still answers yes, and the result is an MMS-fair allocation. Otherwise the output is $\perp$.

(ii) and (iii), unary. An NP metric machine guesses n partitions $\pi^1, \dots, \pi^n$ and an allocation A , lets $\ell_i = \min_j u_i(\pi^i(j))$, which is at most μ_i , and writes

$$2 \sum_i \ell_i + [u_i(A(i)) \geq \ell_i \text{ for all } i],$$

where $[P]$ is 1 if P holds and 0 otherwise. A branch writes the maximum $g(I)$ only if $\ell_i = \mu_i$ for every i . Then the bracket says that A is MMS-fair. So $g(I) = 2 \sum_i \mu_i + 1$ if an MMS-fair allocation exists, and $g(I) = 2 \sum_i \mu_i$ otherwise. All numbers written are polynomial, so $g \in \text{OptP}[O(\log n)]$. For (ii), let T be the transducer that, on input $\langle I, \text{bin}(c) \rangle$, guesses a branch $(\pi^1, \dots, \pi^n, A)$ of this machine,

accepts if the branch writes c and c is odd, and then outputs A , and let $F(I)$ be its set of outputs on $\langle I, \text{bin}(g(I)) \rangle$. If an MMS-fair allocation exists, then $g(I)$ is odd, and $F(I)$ is the non-empty set of allocations A on branches that write $g(I)$, each of them MMS-fair. Otherwise $F(I) = \emptyset$. So F solves MMS-ALLOC. For (iii), $g(I)$ has $O(\log |I|)$ bits, so binary search computes it with $O(\log |I|)$ queries to the NP oracle $\{(I, k) : g(I) \geq k\}$, and an MMS-fair allocation exists if and only if $g(I)$ is odd. $\square$

PROOF OF PROPOSITION 3.7. Let $\Omega = \max_i u_i(\mathcal{G})$, which is polynomial in the input size, as the values are in unary. Given valuations $v_1, \dots, v_n$ for the n coordinates, process the goods one at a time, and maintain the set of *profiles* $(v_1(\pi(1)), \dots, v_n(\pi(n)))$ of the partitions π of the goods seen so far, with one such partition for each profile. Each coordinate lies in $\{0, \dots, \Omega\}$, so there are at most $(\Omega + 1)^n$ profiles, which is polynomial for fixed n , and each good extends each profile in n ways. Running this with $v_j = u_i$ for all j gives all profiles of partitions for agent i , and μ_i is the largest least coordinate among them. This solves MMS-VALUE. Running it with $v_j = u_j$ gives the profiles of all allocations. The MMS-fair allocations are those whose profile has j -th coordinate at least μ_j for all j . This decides MMS-EXISTS and, with the stored partition, solves MMS-ALLOC. $\square$

PROOF OF LEMMA 3.8. Membership. Determine the lexicographically largest model one variable at a time: the i -th oracle query asks whether some model of φ extends the values already fixed for $z_1, \dots, z_{i-1}$ and sets z_i to 1. After k queries, read off the value of z_k .

Hardness. Let $L \in \text{P}^{\text{NP}}$ be decided by a polynomial-time oracle machine $\mathcal{M}$ that makes $p = p(|x|)$ queries to SAT on input x . By the Cook-Levin construction [10], we build in polynomial time a CNF formula Λ_x over variables $y_1, \dots, y_p$, then a block Y of witness and computation variables, then a last variable d , which asserts that (i) if $y_j = 1$, then the part of Y reserved for j is a model of the j -th query formula, which is computed in polynomial time from x and $y_1, \dots, y_{j-1}$ by simulating $\mathcal{M}$ with the j' -th answer taken to be $y_{j'}$ for all $j' < j$, and (ii) $d = 1$ if and only if $\mathcal{M}$ accepts x with these answers. Setting all y_j to 0 makes (i) vacuous, so Λ_x is satisfiable. In the lexicographically largest model, by induction on j , $y_j = 1$ if and only if the j -th query formula (for the true answers $y_1, \dots, y_{j-1}$) is satisfiable: if it is, some model extends the prefix with $y_j = 1$, and if it is not, (i) rules out $y_j = 1$. So this model carries $\mathcal{M}$'s true oracle answers, and by (ii) its value of d is 1 if and only if $x \in L$. As d is the last variable, $x \in L$ if and only if $\Lambda_x \in \text{ODD-LEX-MAX-SAT}$. $\square$

PROOF OF LEMMA 3.9. Every model of α is a model of β , so if α is satisfiable, then $\text{lex}(\alpha) \leq \text{lex}(\beta)$, with equality if and only if the lexicographically largest model of φ sets z_k to 1. If α is unsatisfiable, then no model of φ sets z_k to 1. $\square$

PROOF OF LEMMA 3.11. We use three gadgets, each a set of 1-in-3 clauses [23]. Negated literals are allowed until the second gadget removes them.

Clauses. For a clause of φ with literals l_1, l_2, l_3 , introduce four variables $d_0, \dots, d_3$ private to it and impose (l_1, d_0, d_2) , $(\neg l_2, d_0, d_3)$ and $(\neg l_3, \neg d_1, d_2)$. These admit an extension to $d_0, \dots, d_3$ if and only if some l_t is true. If $l_1 = l_2 = l_3 = 0$, the second clause forces

$d_0 = d_3 = 0$, the first then forces $d_2 = 1$, and the third then has two true literals. Each of the seven other patterns has an extension, as a direct check shows. This holds also if a literal occurs twice in the clause, as the check concerns only the truth values of l_1, l_2, l_3 and each of the three clauses contains just one of them.

Negation. For every variable v that occurs negated in the resulting clauses, introduce a companion $\bar{v}$ and three variables g_0, g_1, g_2 private to v , and impose $(v, \bar{v}, g_0)$, $(v, \bar{v}, g_1)$ and (g_0, g_1, g_2) . If $v = \bar{v} = 1$, the first clause fails, and if $v = \bar{v} = 0$, then $g_0 = g_1 = 1$ and the third fails. So $\bar{v} = \neg v$ is forced, and for either value of v the choice $g_0 = g_1 = 0, g_2 = 1$ extends. Replacing every $\neg v$ by $\bar{v}$ leaves each clause a set of three distinct variables without negation.

Padding. A variable z_i with no occurrence receives one clause (z_i, p_i, q_i) with p_i, q_i fresh. Either value of z_i extends, so this only buys the occurrence required by (a).

Each gadget contributes $O(1)$ variables and clauses per clause, negated variable and padded variable of φ , so the map is polynomial-time. Part (b) holds because a clause gadget has an extension only if its clause is satisfied, and (c) because every model of φ extends through all gadgets, as just verified. The variables $z_1, \dots, z_k$ are never renamed. $\square$

A.2 Proofs for Section 4

PROOF OF LEMMA 4.1. For every $S \subseteq \mathcal{G}$ we have $0 \leq c_j(S) \leq T_j < \rho_j$, so $w(S) = \sum_j W_j c_j(S)$ is written in the mixed radix $(\rho_1, \dots, \rho_r)$ with digits $c_j(S)$, and τ with digits T_j/n . Such representations are unique. $\square$

PROOF OF LEMMA 4.2. A block of F_l has digit 0 outside $\mathcal{G}_l$, so a bundle meets it if and only if the bundle's restriction to $\mathcal{G}_l$ does. The first claim follows by Lemma 4.1. The combination condition says exactly that the sets $\pi(j)$ are pairwise disjoint, and then the restriction of $\pi(j)$ to $\mathcal{G}_l$ is $f_l^{-1}(j)$. $\square$

PROOF OF LEMMA 4.3. (i) Under either hypothesis, each bundle S satisfies $u_i(S) \geq K\tau - M$ for some i , so $K \cdot w(S) = u_i(S) - \delta_i(S) \geq K\tau - 2M > K(\tau - 1)$, as $K = 2M + 1$, and hence $w(S) \geq \tau$ since weights are integers. Equality holds for all bundles since $w(\mathcal{G}) = n\tau$. (ii) A balanced partition attaining the drop gives $\mu_i \geq K\tau - d_i$. If a partition had all bundles worth at least $K\tau - d_i + 1 \geq K\tau - M$ to agent i , it would be balanced by (i), with smallest offset at least $-d_i + 1$, contradicting the definition of d_i . $\square$

PROOF OF LEMMA 4.4. The construction is a variant of the standard digit reduction from SAT to SUBSET-SUM [24] (see also [13]): weights are written in base b , with a *variable digit* for each z_i and a *clause digit* for each clause C_c , and a half picks one *literal good* per variable and tops up each clause digit with *slack* goods, which is possible if and only if the clause is satisfied. The *anchor* a adds a 1 to every clause digit, which makes the complement of a ledger a half as well, so that halves come in complementary pairs, told apart by the anchor.

For an assignment σ , let $\theta_c(\sigma)$ be the number of literals of C_c made true by σ . Let $\ell = \max_c \ell_c$ and $b = 2\ell + 1$. Each weight, and τ_φ , is written in base b with the variable digit of z_i at position $q + i$ and the clause digit of C_c at position c , that is, $w(x) = \sum_{i'} d_{q+i'}(x) b^{q+i'} + \sum_{c'} d_{c'}(x) b^{c'}$ for the digits $d_p(x)$ given in the

following table, where $[P]$ is 1 if P holds and 0 otherwise.

	variable digit $d_{q+i'}$	clause digit $d_{c'}$
P_i	$[i' = i]$	$[z_i \in C_{c'}]$
N_i	$[i' = i]$	$[\neg z_i \in C_{c'}]$
s_j^c	0	$[c' = c]$
a	0	1
τ_φ	1	$\ell_{c'}$

Every row has a nonzero digit (for a since $q \geq 1$), so all weights are positive, and X_φ is computable in polynomial time. As the lowest position is 1, all weights and τ_φ are multiples of b , which is (W1). As the highest position is $q+k$ and all digits are less than b , they are less than b^{q+k+1} , which is (W2). For $S \subseteq X_\varphi$, let $\theta_c(S)$ be the number of literals of C_c whose good is in S . The digit sums of S are $|S \cap \{P_i, N_i\}| \leq 2$ at position $q+i$ and $\theta_c(S) + |S \cap \{s_1^c, \dots, s_{\ell_c-1}^c\}| + [a \in S] \leq 2\ell_c$ at position c . As $2 \leq 2\ell < b$ and $2\ell_c \leq 2\ell < b$, no carries occur, and $w(S)$ is determined digitwise. For $S = X_\varphi$ the digit sums are 2 and $2\ell_c$, twice the digits of τ_φ , so $w(X_\varphi) = 2\tau_\varphi$. Moreover, S is a half if and only if

- (H1) S contains exactly one of P_i, N_i for each i , and
- (H2) $\theta_c(S) + |S \cap \{s_1^c, \dots, s_{\ell_c-1}^c\}| + [a \in S] = \ell_c$ for each c .

Now let $S \subseteq X_\varphi$ with $a \notin S$ satisfy (H1), and let $\sigma = \sigma_S$. Then S contains exactly the goods of the literals made true by σ , so $\theta_c(S) = \theta_c(\sigma)$, and (H2) states that S contains exactly $\ell_c - \theta_c(\sigma)$ slack goods of C_c . As C_c has only $\ell_c - 1$ slack goods, this is possible if and only if $\theta_c(\sigma) \geq 1$. Hence every ledger S satisfies (H1), and σ_S is a model of φ , which shows (a). Conversely, for every model σ , the set S_σ of the goods of the literals made true by σ and the goods $s_1^c, \dots, s_{\ell_c-1}^c$ for each c satisfies (H1) and (H2) and does not contain a , so it is a ledger with $\sigma_{S_\sigma} = \sigma$, which shows (b). For a clause of width $\ell_c = 1$ there are no slack goods, and (H2) for a ledger says $\theta_c(\sigma) = 1$, that is, the clause is satisfied. So the argument covers unit clauses, as the lemma states. $\square$

A.3 The gear block: construction and proof

This part proves Lemma 4.5. The vocabulary of Section 4.5 is assumed. All other names are local to this part. The goods of the block carry the gadget and the variable as superscripts, so they are not to be confused with the goods of the core. The picture is this: each gear is a cycle of $2(k+1)$ teeth and gaps, for a variable with k occurrences, whose two perfect matchings are the two truth values of the variable, and the blocks pos and cls make these matchings consistent across the clauses of the gadget.

CONSTRUCTION (THE GEAR BLOCK). Input: gadgets $\psi_1, \dots, \psi_\Gamma$, a number of bundles $n = 2n_o + n_v + 1$, and an anchor a .

Occurrences. An occurrence is a triple $o = (g, z, j)$, the j -th occurrence of the variable z in gadget g , and $C(o)$ is its clause. We write $k_{g,z}$ for the number of occurrences of z in ψ_g and $r_C = |C|$, so that $\sum_{g,z} k_{g,z} = \sum_C r_C = n_o$.

Units. A unit is a gear (g, z) , a U -absorber (g, C) , or a V -absorber (g, C) for a clause C of ψ_g . We number the units $1, \dots, L$ and write λ_v for the number of unit v , we number the gears $1, \dots, n_v$ and write $\iota_{g,z}$ for the number of gear (g, z) , and we number the clauses of all gadgets $1, \dots, n_c$ and write $\#C$ for the number of clause C . Finally, $h = L + 1$, $c = \max_{g,z} k_{g,z} + n_v + n_c + 3$, $c' = n_c + 1$ and $t_{tm} = \sum_{g,z} \iota_{g,z} + 1$.

Goods and digits. *There are seven blocks tooth, gap, unit, edge, pos, cls and tm, with targets 1, 1, h , n_v , c , c' and t_{tm} per bundle, on the goods of Table 7. These goods are G_{gear} . In the table, λ is the number of the unit of the row, $k = k_{g,z}$, and $C_j = C(g, z, j)$. The kind anch of the anchor counts as both a tooth and a gap.*

Each block enforces one condition. Blocks tooth and gap enforce one tooth and one gap per bundle. Block unit enforces that they belong to the same unit. Block edge enforces one edge per bundle and n_v in the ledger. Block pos enforces which edge of the gear's cycle a pair takes and whether it is a U - or a V -copy. Block cls enforces that a copy belongs to the right clause. Finally, block tm enforces that terminals pair with hubs.

Nonnegativity and size. Since $c \geq k_{g,z} + 3$, we get $c - (k+2) \geq 1$ and $c - 1 - j \geq 2$ for $j \leq k$, and $c - 2 \geq 1$. Since $c \geq n_v + n_c + 3$, the anchor's pos-digit is $c - n_v - n_c \geq 3$ (this is why c contains the term $n_v + n_c$, and not only $\max k_{g,z} + 3$). Since $\#C \leq n_c$, we get $c' - \#C \geq 1$. Finally, $h - \lambda_v \geq 1$, $n_v - 1 \geq 0$, $t_{tm} - \iota_{g,z} \geq 1$ and $t_{tm} - \sum \iota = 1$. All targets, and hence all totals, are polynomial in n . The number of goods is $3(n-1) + n_v + 1$, by the counts below.

The totals. **Teeth:** $\sum_{g,z} (k_{g,z} + 1) + \sum_C r_C + 1 = (n_o + n_v) + n_o + 1 = n$, as clause C contributes one Y and $r_C - 1$ goods R . **Gaps:** the same count, the anchor included. **Edges:** $2n_o + 2n_v = (n-1) + n_v$, which is one edge for each bundle other than the ledger and n_v for the ledger. **The unit total is nh :** the units of the teeth other than a are the units of the gaps other than a , with multiplicity, so the teeth and gaps other than a pair up unit for unit and contribute $(n-1)h$, and the anchor contributes h . **The edge total is nn_v :** the $n-1$ gaps other than a contribute $(n-1)(n_v-1)$, and the $n-1+n_v$ edges contribute $n-1+n_v$. **The cls total is nc' :** The gaps $B_j^{g,z}$ contribute $n_o c' - \sum_C r_C \#C$, as clause C has r_C occurrences. The copies contribute $2 \sum_C r_C \#C$, a U -copy and a V -copy per occurrence. The absorber gaps Y' and R' contribute $\sum_C r_C (c' - \#C) = n_o c' - \sum_C r_C \#C$. The n_v hubs and the anchor contribute c' each. The terms $\sum_C r_C \#C$ cancel, and the total is $c'(2n_o + n_v + 1) = nc'$. **The pos total is nc :** Gear (g, z) with k occurrences contributes $\sum_{j \leq k+1} j + (c-k-2) + \sum_{j \leq k} (c-1-j) + 0 \cdot k + 1 \cdot k + (k+1) + 1 = (k+1)(c+1)$, from its teeth, its hub, its gaps B_j , its U -copies, its V -copies and its two terminals. Clause C with $r = r_C$ variables contributes $1 + (c-1) + (r-1)(1+c-2) = rc - r + 1$ from its absorbers. The anchor contributes $c - n_v - n_c$. With $\sum_{g,z} (k_{g,z} + 1) = n_o + n_v$ and $\sum_C (r_C c - r_C + 1) = n_o c - n_o + n_c$, the total is $(n_o + n_v)(c+1) + n_o c - n_o + n_c + c - n_v - n_c = c(2n_o + n_v + 1) = nc$. **The tm total is nt_{tm} :** the $n-1$ gaps other than a contribute $(n-1)t_{tm} - \sum \iota$, the $2n_v$ terminals $2 \sum \iota$, and the anchor $t_{tm} - \sum \iota$.

Admissible triples. The proof has two layers. The first, the next lemma, consumes every digit: it lists the bundles that the seven blocks admit. The second, the proof of Lemma 4.5, never mentions a digit, except in one check of the converse, and can be followed on Figure 1 alone.

LEMMA A.1. *Let π be a configuration of the seven blocks with $n = 2n_o + n_v + 1$ bundles, and Λ the bundle of a . Then Λ consists of a and exactly n_v edges, and every other bundle is a triple of a tooth, a gap and an edge of one of the following six forms, for a gear (g, z) with $k = k_{g,z}$ occurrences and $C_j = C(g, z, j)$, or for a clause C of a*

Table 7: The digits of the gear block, block by block.

good	kind	gadget	unit	tooth	gap	unit	edge	pos	cls	tm
$A_j^{g,z}, 1 \leq j \leq k+1$	tooth	g	gear (g, z)	1	0	λ	0	j	0	0
$H^{g,z}$ (hub)	gap	g	gear (g, z)	0	1	$h - \lambda$	$n_v - 1$	$c - (k+2)$	c'	$t_{tm} - t_{g,z}$
$B_j^{g,z}, 1 \leq j \leq k$	gap	g	gear (g, z)	0	1	$h - \lambda$	$n_v - 1$	$c - 1 - j$	$c' - \#C_j$	t_{tm}
$U_j^{g,z}, 1 \leq j \leq k$	edge	g	U -copy of C_j	0	0	0	1	0	$\#C_j$	0
$V_j^{g,z}, 1 \leq j \leq k$	edge	g	V -copy of C_j	0	0	0	1	1	$\#C_j$	0
$P^{g,z}$ (terminal)	edge	g	terminal	0	0	0	1	$k+1$	0	$t_{g,z}$
$N^{g,z}$ (terminal)	edge	g	terminal	0	0	0	1	1	0	$t_{g,z}$
$Y^{g,C}$	tooth	g	U -absorber (g, C)	1	0	λ	0	1	0	0
$Y'^{g,C}$	gap	g	U -absorber (g, C)	0	1	$h - \lambda$	$n_v - 1$	$c - 1$	$c' - \#C$	t_{tm}
$R_i^{g,C}, 1 \leq i < r_C$	tooth	g	V -absorber (g, C)	1	0	λ	0	1	0	0
$R'_i{}^{g,C}, 1 \leq i < r_C$	gap	g	V -absorber (g, C)	0	1	$h - \lambda$	$n_v - 1$	$c - 2$	$c' - \#C$	t_{tm}
a (anchor)	anch	—	—	1	1	h	0	$c - n_v - n_c$	c'	$t_{tm} - \sum t$

gadget ψ_g with r_C variables:

form	tooth	gap	edge
T1	$A_j^{g,z}$	$H^{g,z}$	$P^{g,z}$
T2	$A_j^{g,z}$	$H^{g,z}$	$N^{g,z}$
T3 ($1 \leq j \leq k$)	$A_j^{g,z}$	$B_j^{g,z}$	a V -copy of C_j
T4 ($1 \leq j \leq k$)	$A_{j+1}^{g,z}$	$B_j^{g,z}$	a U -copy of C_j
T5	$Y^{g,C}$	$Y'^{g,C}$	a U -copy of C
T6 ($1 \leq i, i' < r_C$)	$R_i^{g,C}$	$R'_{i'}{}^{g,C}$	a V -copy of C

In particular, a terminal can lie outside Λ only in the bundle of the hub of its own gear. Conversely, every such triple meets the blocks tooth, gap, unit, edge, pos, cls and tm.

PROOF. *Step 1 (one tooth and one gap per bundle).* The block tooth has total n and target 1, so every bundle has exactly one tooth, counting a as one. The block gap has total n , target 1 and digit 1 on a , so Λ contains no tooth or gap besides a , and every other bundle has exactly one gap and one tooth.

Step 2 (one edge per bundle, and one unit). Let a bundle other than Λ have a tooth of unit v and a gap of unit v' , as it does by Step 1. Edges have unit-digit 0, so its unit-sum is $\lambda_v + h - \lambda_{v'}$, which equals h if and only if $v = v'$. Its one gap contributes $n_v - 1$ to edge, so the edge equation $n_v - 1 + e = n_v$, for its number e of edges, forces $e = 1$. The ledger holds no gap, and the anchor has edge-digit 0, so Λ holds exactly n_v edges. So every bundle other than Λ is a tooth, a gap and an edge, with tooth and gap of the same unit, and Λ is a together with n_v edges. Each of the two conditions has a block of its own, so no argument is needed that they do not interfere. It remains to determine the edge, unit by unit.

Step 3 (absorbers). Consider $\{Y^{g,C}, Y'^{g,C}, e\}$. The tm equation $t_{tm} + tm(e) = t_{tm}$ gives $tm(e) = 0$, so e is a clause copy, as the terminals have positive tm-digit. The cls equation $(c' - \#C) + cls(e) = c'$ gives $cls(e) = \#C$, so e is a copy of C . The pos equation $1 + (c - 1) + pos(e) = c$ gives $pos(e) = 0$, so e is a U -copy of C : form T5. Likewise, in $\{R_i^{g,C}, R'_{i'}{}^{g,C}, e\}$, the tm and cls equations make e a copy of C , and the pos equation $1 + (c - 2) + pos(e) = c$ gives $pos(e) = 1$, so e is a V -copy of C : form T6. Here cls names the clause and pos tells the two copies apart, so no argument is needed that the clause and the kind of copy, if packed into one digit, would not interfere.

Step 4 (gears). Fix a gear (g, z) with $k = k_{g,z}$ occurrences, and drop the superscripts. Consider a bundle $\{A_j, B_{j'}, e\}$. The tm equation forces $tm(e) = 0$, so e is a clause copy. The cls equation $(c' - \#C_{j'}) + cls(e) = c'$ gives $cls(e) = \#C_{j'}$, so e is a copy of $C_{j'}$. The pos equation $j + (c - 1 - j') + pos(e) = c$ gives $pos(e) = 1 + j' - j \in \{0, 1\}$, so e is a U -copy of $C_{j'}$ with $j = j' + 1$, form T4, or a V -copy of $C_{j'}$ with $j = j'$, form T3. Consider the bundle $\{A_j, H, e\}$ of the hub. The cls equation $c' + cls(e) = c'$ forces $cls(e) = 0$, so e is a terminal. The tm equation $(t_{tm} - t_{g,z}) + tm(e) = t_{tm}$ forces $tm(e) = t_{g,z}$, and the terminals of different gears have different numbers, so $e \in \{P^{g,z}, N^{g,z}\}$. The pos equation $j + (c - k - 2) + pos(e) = c$ gives $pos(e) = k + 2 - j$, so either $e = P^{g,z}$ and $j = 1$, form T1, or $e = N^{g,z}$ and $j = k + 1$, form T2. As in Step 3, the clause and the position on the cycle are read off two separate blocks, so they cannot interfere.

Terminals occur only in forms T1 and T2, together with the hub of their own gear, so a terminal outside Λ lies in that hub's bundle. The converse is read off Table 7, as each triple has one tooth and one gap of the same unit and one edge, and the pos, cls and tm sums are those computed in Steps 3 and 4. $\square$

PROOF OF LEMMA 4.5. Let π be a configuration of the seven blocks with n bundles, and Λ its ledger. By Lemma A.1, Λ is a and n_v edges, and every other bundle is a triple of one of the forms T1–T6.

Locality (i). Each of the forms T1–T6 consists of goods of a single gadget, as the copies of a clause of ψ_g belong to g .

Gears. Fix a gear (g, z) with k occurrences, and drop the superscripts. Its $k + 1$ teeth $A_1, \dots, A_{k+1}$ and its $k + 1$ gaps $H, B_1, \dots, B_k$ lie outside Λ , each in exactly one triple, and in forms T1–T4 they are paired with each other only, and exactly along the edges of the cycle $H - A_1 - B_1 - A_2 - \dots - B_k - A_{k+1} - H$ of length $2(k + 1)$. So they form a perfect matching of this cycle. An even cycle has exactly two perfect matchings (Figure 1), the two modes of Section 4.5: U -mode, the triples T1 and T4 for all $j \leq k$, which use P and a U -copy of each clause C_j , and V -mode, the triples T3 for all $j \leq k$ and T2, which use a V -copy of each C_j and N . So the hub's bundle holds exactly one terminal, P in U -mode and N in V -mode.

Ledger (ii). The other terminal of each gear lies in no triple, as a terminal outside Λ can only lie in the bundle of its own hub (Lemma A.1), which holds the first one. So it lies in Λ . These are n_v distinct edges, one per gear, and Λ has exactly n_v edges, which

Table 8: The six digit blocks of the core Σ_2 .

	A_1	A_2	$\bar{A}_1$	$\bar{A}_2$	B_1	B_2	$\bar{B}_1$	$\bar{B}_2$	V_1	V_2	U_1	U_2
side _A	1	1	1	1	0	0	0	0	0	0	0	0
side _B	0	0	0	0	1	1	1	1	0	0	0	0
edg	0	0	0	0	0	0	0	0	1	1	1	1
rol	1	1	0	0	0	0	1	1	0	0	0	0
L_1	1	2	1	2	0	0	0	0	2	1	1	2
L_2	0	0	0	0	1	2	1	2	2	1	2	1

proves (ii). Moreover, $P^{g,z} \in \Lambda$ exactly if the gear is in V -mode. With the gears, this also proves (iv).

Decoding (iii). Fix a clause C of gadget g with $r = r_C$ variables, and hence r U -copies and r V -copies. None of them lies in Λ , which holds only terminals besides a , so each lies in a triple of forms T3–T6. The occurrence of a variable of C takes a U -copy of C if its gear is in U -mode, and a V -copy if it is in V -mode. The remaining copies lie in the triples T5 and T6 of C , of which there are exactly one and $r - 1$, one for each tooth Y and R_i of C . Counting V -copies, the number of occurrences of C in gears in V -mode plus $r - 1$ is r , so exactly one variable of C has its gear in V -mode, that is, has $P^{g,z} \in \Lambda$. So σ_g is an exactly-1 model of ψ_g , which proves (iii).

Converse (v). Given exactly-1 models $\sigma_1, \dots, \sigma_T$, put each gear (g, z) in V -mode if $\sigma_g(z) = 1$ and in U -mode otherwise, and form the bundles of the gears by the corresponding matchings, each pair with its own copy $U_j^{g,z}$ or $V_j^{g,z}$. Send each unused U -copy of C to the U -absorber of C and each unused V -copy to a V -absorber of C , which the count above makes possible exactly because C has exactly one true variable, and put a with the n_v terminals not in hub bundles. Every bundle other than Λ is a triple of forms T1–T6, so it meets all seven blocks by Lemma A.1. The bundle Λ meets tooth, gap, unit, edge, cls and tm by inspection. It meets pos because exactly one variable of each clause is true: the anchor contributes $c - n_v - n_c$, and the terminal of gear (g, z) in Λ contributes $k_{g,z} + 1$ if it is $P^{g,z}$, that is, if z is true, and 1 if it is $N^{g,z}$. So the pos-sum of Λ is $c - n_v - n_c + n_v + \sum k_{g,z} = c$, where the sum ranges over the true variables and equals n_c , as each clause contains exactly one true variable. The configuration decodes to $\sigma_1, \dots, \sigma_T$, as $P^{g,z} \in \Lambda$ exactly for the gears in V -mode. This proves (v). $\square$

A.4 Proofs for Section 5

PROOF OF LEMMA 5.1. A partition into two bundles is balanced if and only if both bundles have weight τ , that is, if it is a pair of complementary halves. Exactly one of them contains the anchor, so it is a ledger S and its anchor side. (GL1) By the table of Section 4.4, the ledger carries $-(\Delta_\zeta - \text{val}_\zeta(\sigma_S)) \leq 0$ and the anchor side carries $\Delta_\zeta - \text{val}_\zeta(\sigma_S) \geq 0$, so $\chi(\pi) = -\Delta_\zeta + \text{val}_\zeta(\sigma_S)$. (GL2) Every model of φ is σ_S for some ledger S , by Lemma 4.4(b), and $\Sigma \neq \emptyset$ as φ is satisfiable. So Lemma 4.7 and (1) give the maximin share. As φ is a 3-CNF, every clause has width at most 3 and the base is $b \leq 7$, so by (W2) the weights have $O(k + q)$ bits, and $K = 2^{p+1} - 1$ has $p + 1$ bits. $\square$

A.5 Proofs for Section 6

The core lemmas of Section 6.2 are the case $m = 2$ of Lemma 8.1, so we prove parts (a), (b) and (d) of that lemma, for every $m \geq 2$,

which proves Lemmas 6.1 and 6.2 and Proposition 6.4 as well. For $m = 2$, the digits of the six blocks are those of Table 8.

In both phases, we call the bundle of B_i a *vertex bundle* and the bundle of $\bar{B}_i$ a *shadow bundle*.

PROOF OF LEMMA 8.1(A). Both P^V and P^U meet all blocks: side_A, side_B and edg by inspection, rol because each bundle holds exactly one of an A -vertex and a B -shadow, L_1 because $i + (m + 1 - i) = i^+ + (m + 1 - i^+) = m + 1$, and L_2 because $i + (m + 1 - i) = m + 1$. Conversely, by side_A, side_B and edg, each bundle holds exactly one A -side good, one B -side good and one edge. By L_1 , the A -side good of the bundle of V_i has index i , and the A -side good of the bundle of U_i has index i^+ . By L_2 , the B -side good of the bundle of V_i or U_i has index i . By rol, every bundle is of the form (an A -vertex, a B -vertex, an edge) or (an A -shadow, a B -shadow, an edge). So the two bundles holding B_i and $\bar{B}_i$ hold V_i and U_i , and either B_i is with V_i , giving the bundles $\{A_i B_i V_i\}$ and $\{\bar{A}_i \bar{B}_i U_i\}$, or B_i is with U_i , giving $\{A_i^+ B_i U_i\}$ and $\{\bar{A}_i \bar{B}_i V_i\}$. Let h be the number of indices i for which B_i is with U_i . The A -vertex in the bundle of B_i has index i or i^+ , that is, $i + [B_i \text{ is with } U_i] \text{ modulo } m$, and every A -vertex lies in exactly one such bundle, so these indices form a permutation of $\{1, \dots, m\}$. Summing them modulo m gives $h \equiv 0$, so $h = 0$, which gives P^V , or $h = m$, which gives P^U . $\square$

PROOF OF LEMMA 8.1(B). Both perturbations are 0 on all vertices, and $\bar{A}_2$ is the only A -shadow either of them charges, so in P^V the shadow bundle $\{\bar{A}_i^+, \bar{B}_i, U_i\}$ holds a charged A -shadow exactly for $i = 1$. In P^V , a vertex bundle $\{A_i, B_i, V_i\}$ has δ_V -offset 0 and δ_U -offset $-s$. The shadow bundle of $\bar{B}_1$ has δ_V -offset $2s - s - s = 0$ and δ_U -offset $-2s + s = -s$. That of $\bar{B}_2$ has δ_V -offset $s - s = 0$ and δ_U -offset $3s$. That of $\bar{B}_i$ for $i \geq 3$ has δ_V -offset $s - s = 0$ and δ_U -offset s . In P^U , a vertex bundle $\{A_i^+, B_i, U_i\}$ has δ_V -offset $-s$ and δ_U -offset 0. The shadow bundle $\{\bar{A}_i, \bar{B}_i, V_i\}$ has δ_V -offsets $-s$, $2s + s = 3s$ and s for $i = 1$, $i = 2$ and $i \geq 3$, and δ_U -offsets $s - s = 0$, $-2s + 3s - s = 0$ and $s - s = 0$. The sums are $2s - s + (m - 1)s - ms = 0$ for δ_V and $-2s + 3s + (m - 1)s - ms = 0$ for δ_U . $\square$

PROOF OF LEMMA 8.1(D). The positive values of δ_V and δ_U sum to $(m + 1)s$ and $(m + 2)s$, and their negative values to the negatives, so $|\delta_i(S)| \leq M$ for all S . By Lemma 4.1 and part (a), the balanced partitions are P^V and P^U . By part (b), P^V gives a type- V agent offset 0 on every bundle, and P^U does so for a type- U agent, so all drops are 0 and $\mu_i = K\tau$ by Lemma 4.3(ii). An MMS-fair allocation would give every agent at least $K\tau$, so it would be balanced by Lemma 4.3(i), and by part (c) some agent would receive a bundle of offset $-s$, worth $K\tau - s < \mu_i$ to her. $\square$

PROOF OF LEMMA 6.8. We write $w = w_\beta$ and $\tau = \tau_\beta$. All digits of the drop block are positive on its support. The digit of B_1 and of $\bar{B}_1$ equals the target $\tau + 1$, so their bundles hold no other good of the block, and they are two different bundles. The other two bundles hold the goods $X_\beta \cup \{B_2\}$. Let such a bundle hold the goods $S \subseteq X_\beta \setminus \{\bar{B}_2\}$, and possibly B_2 or $\bar{B}_2$. With neither, $w(S) = \tau + 1$. With both, $1 + w(\bar{B}_2) + 1 + w(S) = \tau + 1$, that is, $w(S \cup \{\bar{B}_2\}) = \tau - 1$. Both are impossible, as $w(S)$, $w(S \cup \{\bar{B}_2\})$ and τ are multiples of $b \geq 3$ by (W1). So each of the two bundles holds exactly one of $B_2, \bar{B}_2$. The bundle of B_2 has $1 + w(S) = \tau + 1$, so S is a half without the anchor, that is, a ledger, and the bundle of $\bar{B}_2$ holds the rest,

$X_\beta \setminus S$, the anchor side. Conversely, for a ledger S , the bundles $\{B_1\}$, $\{\bar{B}_1\}$, $\{B_2\} \cup S$ and $X_\beta \setminus S$ have digit sums $\tau + 1$, $\tau + 1$, $1 + \tau$ and $\tau + 1$. $\square$

PROOF OF LEMMA 6.9. We write $w = w_\alpha$ and $\tau = \tau_\alpha$. The digit of B_2 and of $\bar{B}_2$ equals the target 2τ , and all digits are positive on the support, so their bundles hold no other good of the block. A remaining bundle with $j \in \{0, 1, 2\}$ carriers and the goods $S \subseteq X_\alpha$ has $j\tau + w(S) = 2\tau$, that is, $w(S) = (2 - j)\tau$. So $j = 1$ gives a half, which is (S), and $j = 0$ and $j = 2$ give $S = X_\alpha$ and $S = \emptyset$, which is (U). Both (S), for every half T , and (U) meet the block by the same computation. $\square$

PROOF OF LEMMA 6.10. By Lemma 6.8, the classes are $\{B_1\}$, $\{\bar{B}_1\}$, $\{B_2\} \cup S$ and $X_\beta \setminus S$ for a ledger S . The charge is 0 on B_1 , $\bar{B}_1$, B_2 , so the charges of the classes are 0, 0, and those of the ledger and the anchor side in the χ_{lex}^- row of the table in Section 4.4, and $\text{val}(\sigma) \leq \text{lex}(\beta)$. By Lemma 4.4(b), the ledger of the lexicographically largest model exists, and by Lemma 6.8 it gives a configuration. $\square$

PROOF OF LEMMA 6.11. By Lemma 6.9, the classes of B_2 and $\bar{B}_2$ are $\{B_2\}$ and $\{\bar{B}_2\}$, and carry 0, and the other two classes are as in (S) or (U), where (U) gives a configuration. In case (U) the class $\{\gamma_3, \gamma_4\}$ carries 0, and the class X_α carries $\chi_{\text{lex}}(X_\alpha) = 0$. In case (S), which requires α to be satisfiable, let σ' be the model encoded by the ledger among T and $X_\alpha \setminus T$ (Lemma 4.4(a)). Then $\chi_{\text{lex}}(T) = \text{val}(\sigma')$ if T is a ledger and $\chi_{\text{lex}}(T) = -\text{val}(\sigma')$ if T is an anchor side, so the two relief classes carry $\text{val}(\sigma')$ and $-\text{val}(\sigma')$, in some order, each of which is at most $\text{val}(\sigma') \leq \text{lex}(\alpha)$, as $\text{val}(\sigma') \geq 0$. The charge rl is attained by the relief class $\{\gamma_1\} \cup T$ in case (S) with T the ledger of the lexicographically largest model of α , which exists by Lemma 4.4(b), and in case (U) if α is unsatisfiable. $\square$

PROOF OF LEMMA 6.12. (a) By Lemma 6.6, a balanced partition is a phase glued to a configuration of the gate blocks at the B -side goods, each of them lying in the bundle named after it. By Lemma 4.2 applied to the two gate blocks, whose supports $X_\beta \cup \{B_1, B_2, \bar{B}_1\}$ and $X_\alpha \cup \{\gamma_3, \gamma_4, B_2, \bar{B}_2\}$ share only B_2 and $\bar{B}_2$, a configuration of the gate blocks is a configuration of the drop block glued to a configuration of the relief block, such that the classes containing B_2 lie together, and so do the classes containing $\bar{B}_2$. Their configurations are described by Lemmas 6.8 and 6.9: the four B -side goods lie in four different classes of the drop block, and the classes $\{B_2\}$ and $\{\bar{B}_2\}$ of the relief block hold no other good of it, so the two relief classes lie together with the classes $\{B_1\}$ and $\{\bar{B}_1\}$ of the drop block, in either order. So the four B -side goods lie in four different bundles of every configuration of the gate blocks, and by the converse of Lemma 6.6 every phase and every such configuration combine, with each class containing a B -side good in the bundle named after it. No other condition applies. (b) The charge χ is the sum of the charges of the two gates of Lemmas 6.10 and 6.11, as it is 0 on the core goods other than $\bar{B}_2$ and on γ_3, γ_4 . So each bundle carries the charge of the class of the drop gate plus that of the class of the relief gate that it holds. The host bundle holds the drop anchor class and $\{\bar{B}_2\}$, the bundle of B_2 the drop ledger class and $\{B_2\}$, and the bundles of B_1 and $\bar{B}_1$ hold $\{B_1\}$ and $\{\bar{B}_1\}$ and one relief class each. The bounds are those of Lemmas 6.10 and 6.11. In phase P^U , the bundles other than the host bundle form Bad_V by definition. For the last claims, combine

by (a) the configurations of the two gates that attain the charges in Lemmas 6.10 and 6.11, matching the relief class of charge rl with the bundle of B_1 , or the drop configuration with a configuration of the relief block in case (U). $\square$

A.6 Proofs for Section 7

PROOF OF LEMMA 7.1. By Lemma 4.1, the balanced partitions are the configurations of the seven blocks. (GL1) By Lemma 4.5(ii), the ledger Λ consists of a and one terminal per variable, so it carries $-\Delta_\chi$ from the anchor and ζ_z for each variable z with $P^z \in \Lambda$, that is, $\chi(\Lambda) = -\Delta_\chi + \text{val}_\chi(\sigma_\pi) \leq 0$. Every other bundle holds only goods of charge at least 0, so it carries at least 0, and $\chi(\pi) = -\Delta_\chi + \text{val}_\chi(\sigma_\pi)$. (GL2) By Lemma 4.5(v), every exactly-1 model of ψ is σ_π for some configuration π , and ψ is satisfiable by Lemma 3.11(c), as φ is. The grading is 0 outside the variables of φ , so by Lemma 3.11(b) and (c), $\max_{\sigma \in \Sigma} \text{val}_\chi(\sigma) = \text{lex}_\chi(\varphi)$, and Lemma 4.7 gives the maximin share and the claim on MMS-fair allocations. By Lemma 4.5, the seven totals are polynomial in $n = O(|\varphi|)$, so by Lemma 4.1 so are all weights, and every value $K \cdot w(g) + \chi(g)$ is polynomial in $|\varphi| + \Delta_\chi$. $\square$

A.7 Proofs for Section 8

PROOF OF LEMMA 8.5. (G1) and (G2) hold, as the seven blocks of the gear block have digit 0 outside G_{gear} , and G_{gear} contains no core good other than the anchor $\bar{B}_2$. (G3) holds, as χ_e sums to 0 and is 0 on all core goods, and $r \leq m - 1$. (G4) holds, as $M = (m + 2)s \geq s$. The positive values of δ_U sum to $3s + (m - 1)s = (m + 2)s$, and its negative values to $-2s - ms$. Those of δ_V sum to $(m + 1)s$ and $-(m + 1)s$, and a dial adds $+2$ and -2 to these, and $(m + 1)s + 2 \leq (m + 2)s$. So $|\delta_i(S)| \leq M$ for every agent i and bundle S . The six core blocks have totals $O(m^2)$, and the seven blocks of the gear block have totals polynomial in n , with $n = 2m = 2n_o + n_v + 1$. So by Lemma 4.1 all weights are polynomial in $n_o + n_v$, and so are $K = 2M + 1 = O(m)$ and all values $K \cdot w(g) + \delta_i(g)$. Every good has a positive digit in some block, so all weights are positive, and all values are positive as $K > M$. The number of goods is $6m + 3(n - 1) + n_v + 1$, and $n_o + n_v$ is linear in the total size of the instances. $\square$

PROOF OF LEMMA 8.7. We have $0 \leq \text{dr}_e \leq 2 \leq M$. The conclusion follows from Lemma 8.2, as $\text{rl}_e \geq 4 - \text{dr}_e$ with $\text{rl}_e, \text{dr}_e \in \{0, 2\}$ holds exactly if $\text{rl}_e = \text{dr}_e = 2$. We check the two hypotheses in turn, for a dial D_e . The only charges of D_e are -2 and $+2$, so $\chi_e(\pi) \geq -2$ for every partition π , and $\chi_e(\pi) \leq 0$ as χ_e sums to 0. The padding gadget of Section 8.4, if present, is put in wildcard mode $w = 1$ throughout.

(drop) Balanced partitions of phase P^V exist, as every gadget is satisfiable (Lemmas 8.4, 4.5(v) and 6.6). If $\beta_e \in \text{Pos-1-IN-3-SAT}$, put the β_e -gadget in wildcard mode $w = 0$ and the α_e -gadget in wildcard mode $w = 1$: then both charges lie in Λ and cancel, so $\chi_e \geq 0 = -\text{dr}_e$ on every bundle, and in every balanced partition some bundle has $\chi_e \leq 0 = -\text{dr}_e$, as χ_e sums to 0. If $\beta_e \notin \text{Pos-1-IN-3-SAT}$, the β_e -gadget is in wildcard mode $w = 1$ in every balanced partition, so the -2 lies in the bundle of the hub H_e^β , which is not Λ and by locality holds no other terminal, and hence no other charge of D_e . So that bundle has $\chi_e = -2 = -\text{dr}_e$, in every balanced partition, and $\chi_e \geq -2$ on every bundle of every partition. Finally, in phase P^U ,

the set Bad_V has $m + 1 \geq 3$ bundles, while the charges of D_e lie in at most two bundles, so some bundle of Bad_V has $\chi_e = 0 \leq s - \text{dr}_e$.

(relief) The bundles of Bad_V are not Λ , which is the host bundle. By locality, every bundle other than Λ holds at most one terminal, so it carries at most one of the two charges of D_e . So every bundle of Bad_V carries at most +2, and carries +2 only if it holds P_e^α outside Λ , which by Lemma 8.6 is possible only if $\alpha_e \in \text{Pos-1-IN-3-SAT}$. So its charge is at most rl_e . Conversely, let $S \in \text{Bad}_V$. If $\alpha_e \in \text{Pos-1-IN-3-SAT}$, put the α_e -gadget in wildcard mode $w = 0$ and match the bundle of its hub H_e^α with S . The β_e -gadget is in any possible wildcard mode, and its charge lies in Λ or in another bundle. Then $\chi_e(S) = 2 = \text{rl}_e$. If $\alpha_e \notin \text{Pos-1-IN-3-SAT}$, put the α_e -gadget in wildcard mode $w = 1$, and if the -2 lies

outside Λ , match its bundle with a bundle other than S . Then $\chi_e(S) = 0 = \text{rl}_e$. In both cases, put the gadgets of every other dial $D_{e'}$ as follows. If $\beta_{e'} \in \text{Pos-1-IN-3-SAT}$, put its $\beta_{e'}$ -gadget in wildcard mode $w = 0$ and its $\alpha_{e'}$ -gadget in wildcard mode $w = 1$, so that both its charges lie in Λ and cancel, and $\chi_{e'}(\pi) = 0 = -\text{dr}_{e'}$. Otherwise, $\chi_{e'}(\pi) \geq -2 = -\text{dr}_{e'}$ in any case. These choices are compatible, as the wildcard modes of all gadgets and the matching are independent. $\square$

B Notation

Table 9 lists the notation of Sections 3–8, with the section in which each symbol is defined.

Table 9: Notation.

symbol	meaning	sec.	symbol	meaning	sec.
$\text{val}(\sigma), \text{lex}(\varphi)$	value of an assignment, and of the lexicographically largest model	3.4	X_φ, τ_φ	halving gadget of φ , whose ledger is a half without a , and weight of a half	4.4
Δ	$2^k - 1$, the largest value of an assignment	3.4	a, b	anchor of X_φ , and base of its weights	4.4
$\text{Sol}_\Pi(x)$	solutions of a search problem, or $\{\perp\}$	3.1	$\chi_{\text{lex}}^\zeta, \chi_{\text{lex}}^{-\zeta}$	lex charge and its negative form, written $\chi_{\text{lex}}, \chi_{\text{lex}}^-$ for the default grading	4.4
$\text{OptP}, \text{OptP}[O(\log n)]$	optima of NP metric machines	3.2	(GL1), (GL2)	calibration and surjectivity of the graded ledger	4.6
$\text{FNP} // \text{OptP}[O(\log n)]$	search with an optimum as advice	3.2	Σ_2, Σ_m	the core, with 12 goods for 4 agents, and with $6m$ goods for $2m$ agents	6.2, 8.2
φ, η_b	Cook–Levin formulas of a metric machine on input x , with η_b guarded by b	3.4	i^+	cyclic successor of $i \leq m$	8.2
$z_1, \dots, z_p$	output bits of φ	3.4	P^V, P^U	the two phases of the core	6.2
$K \cdot w + \delta_i$	valuation: weight scaled by $K = 2M + 1$, plus charge	4.2	δ_V, δ_U	perturbations, neutral in P^V and P^U	6.2
τ	weight of every bundle of a balanced partition	4.2	s	scale of the perturbations, the cost, with $s = \Delta$ in $I_{\alpha, \beta}$ and $s = 4$ in J	6.2
M	bound on $ \delta_i(S) $	4.2	$\text{Bad}_V, \text{Bad}_U$	the deficit bundles of P^U , with δ_V -offset $-s$, and those of P^V , with δ_U -offset $-s$	6.2
d_i	drop of agent i	4.2	$\tilde{B}_2$	the host, the only core good a gate sees, whose bundle is the host bundle	6.2
$\zeta, \text{val}_\zeta, \text{lex}_\zeta, \Delta_\zeta$	grading, graded value, its largest value over models, and its largest value, with ζ left out for the default grading $\zeta_i = 2^{k-i}$	4.3	deficit, safe	bundles that cost s to one type in each phase, and bundles of offset ≥ 0 for both types in both phases	6.2
$\chi(\pi)$	charge of the worst bundle of π	4.6	$D, D_1, \dots, D_r$	the dial, and the dials for general m	6.2, 8.2
χ, χ_e	the charge of the dial, of dial D_e , on top of δ_V	6.2	E^V, E_1^U, E_2^U	the blockers of $I_{\alpha, \beta}$	6.2
dr, rl and dr_e, rl_e	drop and relief of the dial, and of dial D_e	6.2	$B_1, B_2, \tilde{B}_1, \tilde{B}_2$	the B -side goods, which name the bundles of the core and label the gates' classes	6.2
(S), (U)	the relief gate split into two halves, and switched off	6.3	γ_3, γ_4	carriers of the relief gate	6.3
ψ_{pad}	padding gadget	8.4	T1–T6	admissible triples of the gear block	A.3
I_x	instance for OptP, two agents	5.2	G_{gear}	goods of the gear block	4.5
n_v, n_c, n_o	variables, clauses, occurrences of the gadgets	4.5	$H^{g,z}, P^{g,z}, N^{g,z}$	hub and terminals of gear (g, z)	4.5
Λ	ledger of the gear block (the bundle with a)	4.5	$J_{\varphi, \zeta}$	gear-block instance with identical valuations	7.2
$b(\text{guard}), \zeta'$	guard variable, and the grading with values 2^{p+1-i} and 1	7.4	w, ψ^w	wildcard, and gadget with wildcard	8.3
J	gated instance for $\text{P}^{\text{NP}}[O(\log n)]$	8.4	$P_e^\beta, N_e^\beta, H_e^\beta$	terminals and hub of the wildcard's gear	8.4