DISCUSSION PAPER SERIES

DP14976
(v. 2)

Solving Strong-Substitutes Product-Mix Auctions

Elizabeth Baldwin, Paul Goldberg, Paul Klemperer and Edwin Lock

FINANCIAL ECONOMICS

INDUSTRIAL ORGANIZATION

ISSN 0265-8003

Solving Strong-Substitutes Product-Mix Auctions

Elizabeth Baldwin, Paul Goldberg, Paul Klemperer and Edwin Lock

Discussion Paper DP14976
First Published 30 June 2020
This Revision 19 March 2021

Centre for Economic Policy Research
33 Great Sutton Street, London EC1V 0DX, UK
Tel: +44 (0)20 7183 8801
www.cepr.org

Solving Strong-Substitutes Product-Mix Auctions

Abstract

This paper develops algorithms to solve strong-substitutes product-mix auctions: it finds competitive equilibrium prices and quantities for agents who use this auction's bidding language to truthfully express their strong-substitutes preferences over an arbitrary number of goods, each of which is available in multiple discrete units. Our use of the bidding language, and the information it provides, contrasts with existing algorithms that rely on access to a valuation or demand oracle. We compute market-clearing prices using algorithms that apply existing submodular minimisation methods. Allocating the supply among the bidders at these prices then requires solving a novel constrained matching problem. Our algorithm iteratively simplifies the allocation problem, perturbing bids and prices in a way that resolves tie-breaking choices created by bids that can be accepted on more than one good. We provide practical running time bounds on both price-finding and allocation, and illustrate experimentally that our allocation mechanism is practical.

JEL Classification: D44

Keywords: bidding language, product-mix auction, Competitive Equilibrium, Walrasian Equilibrium, convex optimisation, strong substitutes, submodular minimisation

Elizabeth Baldwin - elizabeth.baldwin@economics.ox.ac.uk
Hertford College, Oxford University, Department of Economics, Oxford University and CEPR

Paul Goldberg - paul.goldberg@cs.ox.ac.uk
Dept. of Computer Science, Oxford University

Paul Klemperer - paul.klemperer@economics.ox.ac.uk
Oxford and CEPR

Edwin Lock - edwin.lock@cs.ox.ac.uk
Dept. of Computer Science, Oxford University

Solving Strong-Substitutes Product-Mix Auctions

Elizabeth Baldwin* Paul W. Goldberg† Paul Klemperer‡ Edwin Lock§

24th February 2021

Abstract

This paper develops algorithms to solve strong-substitutes product-mix auctions: it finds competitive equilibrium prices and quantities for agents who use this auction's bidding language to truthfully express their strong-substitutes preferences over an arbitrary number of goods, each of which is available in multiple discrete units. Our use of the bidding language, and the information it provides, contrasts with existing algorithms that rely on access to a valuation or demand oracle.

We compute market-clearing prices using algorithms that apply existing submodular minimisation methods. Allocating the supply among the bidders at these prices then requires solving a novel constrained matching problem. Our algorithm iteratively simplifies the allocation problem, perturbing bids and prices in a way that resolves tie-breaking choices created by bids that can be accepted on more than one good. We provide practical running time bounds on both price-finding and allocation, and illustrate experimentally that our allocation mechanism is practical.

Keywords: bidding language, product-mix auction, competitive equilibrium, Walrasian equilibrium, convex optimisation, strong substitutes, submodular minimisation

1 Introduction.

This paper develops algorithms that solve product-mix auctions in which participants can make bids that represent any strong-substitutes preferences for an arbitrary number of distinct goods. (These preferences are also known, in other literatures, as $M^\natural$-concave, matroidal and well-layered maps, and valuated matroids). It thus allows bidders to express more general preferences than could previously be permitted in these auctions, and finds competitive equilibrium prices and quantities consistent with these, and the auctioneer's preferences.

Importantly, our algorithms for finding equilibrium differ from existing ones in that they directly use the information that the product-mix auction 'language' provides. This information is in a very different form from the information provided by a valuation or demand oracle. This creates additional complexities, as well as simplifications which we can exploit. However, the language is conceptually simple, and easy for bidders to use in a (product-mix) auction.

The *product-mix auction* was developed in 2007-8 for the Bank of England to provide liquidity to financial institutions by auctioning loans to them [Klemperer, 2008]. It is now used at least monthly by the Bank, and more often when institutions are more likely to be under stress. (After the 2016 vote for 'Brexit', and starting again in March 2019, for instance, the auction was run weekly.)

*Dept. of Economics, Oxford University, UK, elizabeth.baldwin@economics.o x.ac.uk.
†Dept. of Computer Science, Oxford University, UK, paul.goldberg@cs.ox.ac.uk.
‡Dept. of Economics, Oxford University, UK, paul.klemperer@nuffield.ox.ac.uk .
§Dept. of Computer Science, Oxford University, UK, edwin.lock@cs.ox.ac.uk.

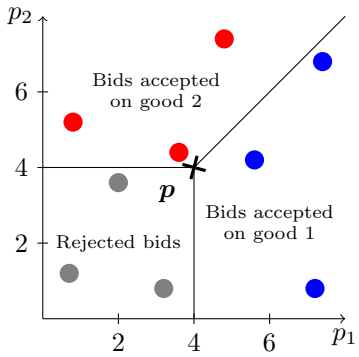


Figure 1: Price vectors divide $\mathbb{R}^n$ into regions that specify which good is allocated to bids lying in these region. This example works in the setting with two goods. The price vector $\boldsymbol{p} = (4, 4)$ (marked by a cross) divides $\mathbb{R}^2$ into three regions, each of which labelled according to the good that bids in this region are allocated.

The original implementation of the auction allowed bidders to each submit a list of bids, where each bid has $n + 1$ elements: a price for each of the n goods available, and a total quantity of goods sought by that bid.[1] The auction sets a uniform price for each good (i.e. every recipient of any particular good pays the same price per unit for that good), and gives each bid an allocation that maximises that bid's 'utility', assuming quasilinear preferences. That is, each bid is allocated its desired quantity of the good on which its price strictly exceeds the auction's price by most, if there is a unique such good, and is allocated nothing if all its prices are strictly below the auction's corresponding prices. (So if a bid only wants one particular good, it simply sets prices of zero for all the other $(n - 1)$ goods.) A bid that creates a tie (i.e. two or more of its prices exceed the auction's corresponding prices by most, or none of its prices exceed, but at least one equals, the auction's corresponding prices) can be allocated in any way consistent with equilibrium (see below). Figure 1 illustrates how the good that a bid is allocated depends on the bid's relative position to the price vector. In particular, we note that individual bids $(b_1, \ldots, b_n; +1)$ that seek exactly one item of a good can be interpreted as a unit-demand bidder with valuation vector $\boldsymbol{b} = (b_1, \ldots, b_n)$ and quasi-linear utilities. We motivate the bidding language in the following example.

Example 1 (Figure 2a). *Alice participates in an auction held with two goods, say apples (good 1) and bananas (good 2). For her breakfast, she requires a single item of fruit and is willing to pay up to £6 for either an apple or a banana. This is expressed by the bid* $\boldsymbol{b} := (6, 6; +1)$*; the last element in the bid denotes the quantity of goods sought. Suppose the auction's prices are set at* $\boldsymbol{p} = (p_1, p_2)$*. If at least one good is priced lower than the respective bid entry, she will demand the good* $i \in \{1, 2\}$ *that maximises utility* $b_i - p_i$*. Otherwise, if both apples and bananas are priced higher than her valuation of £6, she demands neither. In addition, as she is fond of bananas, Alice wishes to pick up a banana as a* second *fruit item, but only if the price of a banana does not exceed £4. This is expressed by the bid* $\boldsymbol{b}' := (0, 4; +1)$*.*

To illustrate Alice's demand, consider the following two possible auction prices. If the auction prices are set at $\boldsymbol{p} = (1, 3)$*, then Alice demands one apple and one banana:* $\boldsymbol{b}$ *demands an apple, as its utility for an apple or a banana is 5 and 3, respectively, while* $\boldsymbol{b}'$ *demands a banana, as its price is less than £4. On other hand, if auction prices are* $\boldsymbol{p} = (6, 5)$*, Alice demands only one banana:* $\boldsymbol{b}$ *demands a banana and* $\boldsymbol{b}'$ *demands nothing, as the price of a banana is too high. Alice's two bids* $\boldsymbol{b}$ *and* $\boldsymbol{b}'$*, together with the demand correspondence they induce, are shown in Figure 2a.*

[1] In the Bank of England's auction, the bidders are commercial banks, etc., each good is a loan secured against one of n different specified qualities of collateral (so the prices are interest rates), and the quantity is the amount of the loan (in £).

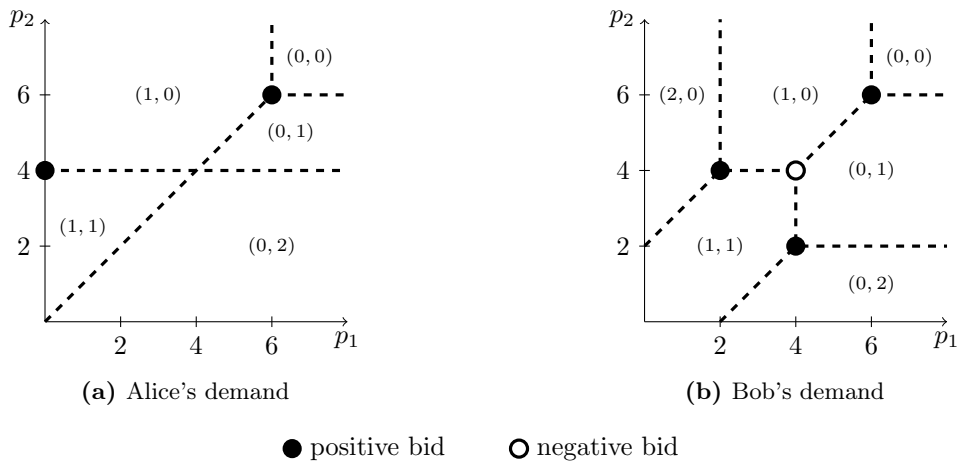


Figure 2: Examples of strong-substitutes demand correspondences on two goods {1,2} belonging to two bidders, Alice and Bob. Alice's demand is specified by bid list $\mathcal{B} = \{(6, 6; +1), (0, 4; +1)\}$ and Bob's bid list is $\mathcal{B}' = \{(2, 4; +1), (4, 2; +1), (4, 4; -1), (6, 6; +1)\}$. Positive and negative bids are depicted as solid and hollow circles, respectively. Price space is divided into regions corresponding to demanded bundles (x_1, x_2), with x_i denoting the number of items of good i. At $\boldsymbol{p} = (4, 4)$, Alice demands bundles $\{(1, 0), (0, 1), (1, 1), (0, 2)\}$ and Bob demands bundles $\{(1, 0), (0, 1), (1, 1)\}$, the discrete convex hulls of bundles demanded in the regions surrounding $\boldsymbol{p}$.

The auction's prices are chosen to maximise the sum of the bids' and auctioneer's welfare – that is, it finds competitive equilibrium prices and quantities.[2] Moreover, the auctioneer expresses her preferences about which goods to allocate in the form of supply functions which can themselves be equivalently represented as (and thus converted into) a list of bids of the kind described above.[3]

Positive and Negative bids. In versions of the product-mix auction thus far implemented, all bids are for positive quantities of goods. In the quasi-linear preferences case, the market-clearing prices can then be found by solving straightforward linear programs, and finding an allocation of the auctioneer's supply to the individual bidders is similarly straightforward. However, some strong-substitutes[4] preferences (over bundles of non-negative quantities of goods) can only be

[2]Specifically, the auction finds the lowest such price vector. Since the bids automatically express "strong substitutes" preferences for all bidders (see Baldwin and Klemperer [2021]), there exist equilibrium price vectors, and also a unique one among them at which every good's price is lowest.

[3]See Appendix 1E of Klemperer [2018] for how to convert supply functions into bid lists. (The bids that the auctioneer's supply functions would be converted into would ensure that it *sells more* units on a good when prices are high, analogous to a buyer *buying fewer* units in this case.) Although the auctioneer's preferences *could* equivalently have been represented as a list of bids of the kind made by the bidders, describing them simply as two-dimensional graphs of 'supply schedules' was an important feature of the auction design: participants' ability to express their preferences in the ways that are most natural for them is crucial to getting an auction accepted for practical use, to getting bidders to participate, and to the auction working efficiently.

The Bank of England's original program restricted to $n = 2$. Since 2014 its program permits much larger n (it is currently being run with $n = 3$), and it also allows richer forms of preferences to be expressed by the auctioneer (but not by the bidders, whose preference expression has, by contrast, been restricted in recent auctions). A variant that allowed for bidders' budget constraints (hence non-quasilinear preferences) was programmed for the Government of Iceland in 2015-16. See Klemperer [2008, 2010, 2018] and Baldwin and Klemperer [2021] for full discussion.

[4] Strong-substitutes preferences are those that would be ordinary substitutes preferences if we treated every unit of every good as a separate good. Such preferences have many attractive properties; they mean, for example, that if the price of any one good increases, and the demand for it decreases, then the demand for all other goods can increase by at most the amount of that decrease. Strong substitutability is equivalent to $M^\natural$-concavity [Shioura and Tamura, 2015]. See Baldwin and Klemperer [2019] for a discussion of the relationship between 'strong' and 'ordinary' substitutes.

expressed by using lists of bids for both positive *and negative* quantities,[5] and it has been shown that *any* strong-substitutes preferences can be represented using lists of positive and negative bids.[6] The rules for accepting, or not accepting, negative bids are identical to those for positive bids: if a negative bid is accepted at some prices, it is allocated a negative quantity of the good it demands, (partially) cancelling the allocated quantities of positive bids that are accepted at the same prices.

To illustrate why negative bids are useful, we consider the setting with two distinct goods 1 and 2 in which an agent has strong substitutes valuation u and is interested in at most 2 units. In other words, the agent only has a positive value for bundles $(0,1),(1,0),(1,1),(0,2)$ and $(2,0)$, where the 2-dimensional vectors (x_1,x_2) denote bundles containing x_1 units of good 1 and x_2 units of good 2. It is known [Shioura, 2017] that the strong substitutes condition is equivalent to $\mathrm{M}^\natural$-concavity which, in our setting, is equivalent to discrete concavity together with the following three additional conditions:

(i) $u((1,1)) \leq u((0,1)) + u((1,0))$,

(ii) $u((2,0)) - u((1,0)) \leq u((1,1)) - u((0,1))$.

(iii) $u((0,2)) - u((0,1)) \leq u((1,1)) - u((1,0))$.

It is easy to see (by infinitesimally and independently perturbing bundle values) that the generic case has strict inequalities in all three conditions. Intuitively, a strict inequality in the first condition implies that a single unit of good 1 is valued strictly less if the agent also has a unit of 2 than if he does not, reflecting strict substitutability as distinct from complementarity or independence between the two goods. A strict inequality in the second or third condition implies that the marginal utility of a good is reduced more by receiving an additional unit of the same good than by receiving a unit of a different good; 'more variety' is valuable. In contrast, it is easy to check that any demand correspondence generated only by positive bids is non-generic, in that at least one of the three conditions must be an equality. Thus a demand correspondence such as the one illustrated in Example 2 and Figure 2b, in which all three conditions are strict inequalities, cannot be represented by positive bids only. We refer to Klemperer [2018, Appendix IC] and Klemperer [2008, p. 15] for further details and examples of the usefulness of negative bids.

Example 2 (Figure 2b). *A combination of positive and negative bids is required to express Bob's demand correspondence depicted in Figure 2b, which divides price space into six demand regions for bundles with at most two units. We have* $u((0,1)) = u((1,0)) = 6$, $u((0,2)) = u((2,0)) = 8$ *and* $u((1,1)) = 10$. *Bob's bid list is* $\mathcal{B}' = \{(2,4;+1),(4,2;+1),(4,4;-1),(6,6;+1)\}$.

Our contributions. This paper addresses the computational challenges of determining uniform component-wise minimal market-clearing prices (at which total market demands equal the quantities of each good that are available) and allocating a fixed collection of the goods at these prices, to bidders whose demands are defined by lists of positive and negative bids that express strong-substitute preferences. Our algorithms exploit the fact that the use of the product-mix bidding language allows for the efficient computation of a demanded bundle as well as of the indirect utility derived at any given prices.

Section 2 introduces the product-mix auction's strong-substitutes bidding language in more detail, and develops some of its properties. A first contribution of our paper is to show that it is coNP-complete to determine whether a given list of positive and negative bids constitute a valid demand correspondence. However, when the number of goods, or the number of negative bids, is bounded by a constant, we present a polynomial-time algorithm for checking validity.

[5] The ability to express such preferences was not thought necessary in the Bank of England's application, but might be useful in closely related environments (see Klemperer [2018]).

[6] See [Baldwin and Klemperer, 2021]. This work was also described in Baldwin et al. [2016]; Klemperer [2010] noted the result for $n = 2$.

In Section 3, we consider algorithms for finding component-wise minimal market-clearing prices that have practical running time bounds. We adopt a discrete steepest descent method from the discrete optimisation literature [Murota, 2003, Shioura, 2017] that employs submodular minimisation to find the steepest descent directions and present two techniques that reduce the number of iterations required by taking long steps in the steepest descent direction. Full details are provided in Appendix B. The two long-step techniques we describe yield the first algorithms for price finding that are fully polynomial in the input size of the bid lists and, since submodular minimisation is rapid in practice [Chakrabarty et al., 2017], we expect our approach to have a fast running time. Indeed, preliminary experiments suggest our method performs well in practice.

Our main contribution, given in Section 4, is an efficient polynomial-time algorithm that allocates the auctioneer's chosen supply among the bidders at given market-clearing prices. The difficulty in developing this lies in handling bids whose utility is maximised on more than one alternative good (or whose utility from its most-preferred good is exactly zero), as tie-breaking choices interact with each other. This gives rise to a novel matching problem. Our algorithm proceeds by iteratively simplifying the allocation problem at hand, allocating unambiguous bids and perturbing bids and prices in a way that resolves a subset of the tie-breaks and yields a simplified allocation problem. Progress is measured in terms of reductions in the number of edges of a multigraph associated with the allocation problems.

Software implementations. Our price-finding and allocation algorithms are conceptually simple. They are also straightforward to implement, which has allowed us to develop two practical implementations in Haskell and Python; these can be found at `http://pma.nuff.ox.ac.uk` and `https://github.com/edwinlock/product-mix`, respectively. Some effort was made to optimise for speed in the Python implementation by exploiting fast matrix operations provided by the NumPy package [van der Walt et al., 2011]. Furthermore, in an effort to implement submodular minimisation efficiently, the Fujishige-Wolfe algorithm [Chakrabarty et al., 2014] was implemented in combination with a memoization technique to reduce the number of submodular function queries.

In order to evaluate the practical running time of our allocation algorithm, we used our Python implementation to run experiments on auctions with various numbers of goods, bidders and bids. The results of these experiments, given in Appendix A to this paper, demonstrate that our allocation algorithm is efficient in practice for realistic auction sizes.

Related work. Our work continues a long literature, spanning economics and discrete convex analysis, on 'gross' and 'strong substitutes' [Kelso and Crawford, 1982, Milgrom and Strulovici, 2009] and '$M^\natural$-concavity' [Murota and Shioura, 1999]. While the discrete convex analysis literature generally allows for multiple units of each good, much focus in economics has been on the case in which there is only one unit of each good; Milgrom and Strulovici [2009] showed that 'strong substitutes' provide the suitable generalisation of gross substitutes to the multi-unit case, retaining existence of equilibrium while insisting that any two units of the same good should have the same price.

Algorithms to compute equilibrium prices in these contexts go back to Kelso and Crawford [1982], Murota et al. [2013, 2016] and Ausubel [2006]; see Shioura and Tamura [2015], Murota [2016] and Paes Leme [2017] for recent surveys. Market-clearing prices are commonly found either by performing a discrete steepest-descent search or by solving a convex optimisation problem by means of an improved cutting plane method of Lee et al. [2015] (cf. Paes Leme and Wong [2017]). We note that to the best of our knowledge, the cutting plane method has not yet been implemented and may be computationally expensive in practice; moreover, solving the convex optimisation problem is not guaranteed to find component-wise minimal prices. While the steepest-descent methods described in the literature run in pseudo-polynomial time in the valuation and demand oracle settings, as compared to the fully polynomial algorithm of

Paes Leme and Wong [2017], our steepest descent algorithm uses long steps and exploits the bid representation of bidder demand to close this gap, yielding a competitive fully polynomial algorithm in the bidding-language setting.

Significantly, whereas previous literature has developed algorithms for finding market-clearing prices, few have addressed the (harder) problem of finding a valid allocation of goods, given those prices. One such work is Murota [2003], which presents an algorithm that works in the multi-unit case by reducing the allocation problem to a network flow problem and relies on oracle access to the valuation function of each bidder. Paes Leme and Wong [2017] provide a different algorithm, also in the valuation oracle setting, but this is only applicable in the case in which there is only one unit of each good. We elaborate on this in Section 2.4. As the computation of a bidder's valuation of a given bundle is expensive in our bidding-language setting, running the algorithm of Murota [2003] to solve the allocation problem for product-mix auctions would incur a significant cost for each query to the valuation oracle. In contrast, our algorithms directly exploit the specific representation provided by the product-mix auction's bidding language to find both prices and allocations.

We note that the valuation oracle setting of Murota et al. [2013, 2016] and others provides no straightforward way to compute a demanded bundle or the indirect utility function at given prices. In contrast, our bidding-language setting is somewhat analogous to but more informative than the demand oracle setting of Ausubel [2006] and Paes Leme and Wong [2017], which presupposes access to an oracle returning a demanded bundle at given prices.

This paper assumes that bidders are able to construct and submit bid lists that represent their strong-substitutes preferences. When the number of goods is large or the preferences are otherwise complex, this may present difficulties for bidders. Goldberg et al. [2020] address this issue by presenting algorithms that generate bid lists on behalf of bidders using only access to a demand or valuation oracle to elicit bidders' preferences.

2 Preliminaries.

We denote $[n] \coloneqq \{1, \ldots, n\}$ and $[n]_0 \coloneqq \{0, \ldots, n\}$. In our auction model, there are n distinct goods $[n]$; a single copy of a good is an *item*. A *bundle* of goods, typically denoted by $\boldsymbol{x}, \boldsymbol{y}$ or $\boldsymbol{z}$ in this paper, is a vector in $\mathbb{Z}^n$ whose i-th entry denotes the number of items of good i. The *target bundle* $\boldsymbol{t}$ is a bundle the auctioneer wants to allocate amongst the bidders. Vectors $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n$ typically denote vectors of prices, with a price entry for each of the n goods. We write $\boldsymbol{p} \leq \boldsymbol{q}$ when the inequality holds component-wise. It is often convenient to regard a rejected bid as being accepted on a notional *reject good* for which bids and prices are always zero. Letting the reject good be 0, the set of goods is then $[n]_0$. In this setting, we identify bundles and prices with the $n+1$-dimensional vectors obtained by adding a 0-th entry of value 0.

A *valuation* u is a function that maps bundles to non-negative real numbers. We assume that bidders have quasi-linear utilities, i.e. the utility derived from bundle $\boldsymbol{x}$ at price $\boldsymbol{p}$ by a bidder with valuation u is given by

$$u(\boldsymbol{x}) - \boldsymbol{p} \cdot \boldsymbol{x}. \tag{1}$$

Any valuation u is associated with a demand correspondence D_u that maps $\boldsymbol{p}$ to the set of bundles that maximise (1). We omit the subscript u when it is clear from context.

For any subset $X \subseteq [n]$, $\boldsymbol{e}^X$ denotes the characteristic vector of X, i.e. an n-dimensional vector whose i-th entry is 1 if $i \in X$, 0 otherwise. Furthermore, $\boldsymbol{e}^i$ denotes the vector whose i-th entry is 1 and other entries are 0.

A set function $f : 2^{[n]} \to \mathbb{Z}$ is *submodular* if it satisfies $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for all $S, T \subseteq V$. Submodular function minimisation (SFM) is the task of finding a minimiser of a submodular function. It is well-known that the minimisers of submodular functions form a lattice; that is, if S and T are minimisers of f, then so are $S \cup T$ and $S \cap T$. SFM can be solved in polynomial time using the improved cutting plane method of Lee et al. [2015]. For SFM that is

efficient in practice, we refer to two SFM algorithms from the literature. The subgradient descent approach by Chakrabarty et al. [2017] determines a minimiser in time $O(nF^3\gamma \log n)$, while the Fujishige-Wolfe algorithm described by Chakrabarty et al. [2014] takes time $O(F^2(n^2\gamma + n^3))$, where F is an upper bound on the absolute value of f and γ denotes the time it takes to query f. Experimental results [Chakrabarty et al., 2014] indicate that the running time of the Fujishige-Wolfe algorithm depends less on F than suggested by the above bound. For the price-finding algorithms described in Section B, we require a subroutine that finds the *inclusion-wise minimal minimiser*. Note that such a subroutine can be obtained by calling any SFM algorithm $n+1$ times. Indeed, let S be a minimiser of f and, for every $v \in S$, let S_v be a submodular minimiser of the function f restricted to $[n] \setminus \{v\}$. Then if S_0 denotes the minimal minimiser of f, we have $v \in S_0$ if and only if $f(S_v) > f(S)$, as the minimisers of a submodular function form a lattice. Hence, we obtain $S_0 \coloneqq \{v \in S \mid f(S_v) > f(S)\}$.

2.1 Strong substitutes valuation functions.

In this paper, we assume that bidders have *strong substitutes* valuations. We review some basic properties of strong-substitutes (SS) valuations. A SS valuation divides price space into regions corresponding to bundles: any bundle $\boldsymbol{x}$ has a price region where $\boldsymbol{x}$ is demanded, possibly along with other bundles (see Figure 2). It is known [Murota, 2003, Theorem 11.16] that each such region is a convex lattice. When a demand region for some bundle $\boldsymbol{x}$ has full dimensionality, in its interior $\boldsymbol{x}$ is the only bundle demanded; we call this interior a *unique demand region* (UDR).

Definition 1. A valuation u is *ordinary substitutes* if, for any prices $\boldsymbol{p}' \geq \boldsymbol{p}$ with $D_u(\boldsymbol{p}) = \{\boldsymbol{x}\}$ and $D_u(\boldsymbol{p}') = \{\boldsymbol{x}'\}$, we have $x'_k \geq x_k$ for all k such that $p_k = p'_k$. A valuation u is *strong substitutes (SS)* if, when we consider every unit of every good to be a separate good, u is ordinary substitutes.

Definition 1 for strong substitutes is equivalent to the definition of Milgrom and Strulovici [2009] and also to $M^\natural$-concavity [Murota and Shioura, 1999]: see footnote 4. It is equivalent to gross substitutes (GS) [Kelso and Crawford, 1982] if there is only one item of each good. While GS guarantees that a competitive equilibrium exists in single-unit auction markets, the condition is not sufficient for the existence of such an equilibrium in the multi-unit case (Shioura and Tamura [2015] give an example). SS represents a generalisation of single-unit GS that provides a general sufficient condition for the existence of an equilibrium. We refer to Shioura and Tamura [2015] for a detailed discussion on the distinction between GS and SS.

Definition 2. The *indirect utility function* f_u of valuation u maps a price vector $\boldsymbol{p}$ to the utility that a bidder with demand D_u has for receiving her preferred bundle at a given price vector $\boldsymbol{p}$ in the following way.

$$f_u(\boldsymbol{p}) \coloneqq u(\boldsymbol{x}) - \boldsymbol{p} \cdot \boldsymbol{x}, \text{ where } \boldsymbol{x} \in D_u(\boldsymbol{p}). \tag{2}$$

We note that $f_u(\boldsymbol{p})$ is convex, piecewise-linear and continuous for SS valuations u.

2.2 Representing strong substitutes valuations with weighted bids.

We describe how every strong substitutes valuation function can be represented by a finite list $\mathcal{B}$ of positive and negative bids. A *bid* $\boldsymbol{b}$ consists of an $(n+1)$-dimensional vector $(b_1, \ldots, b_n; b_{n+1})$, where the first n components $b_1, \ldots, b_n$ of the vector denote the bid's 'valuation' for the goods $[n]$ and the $(n+1)$-th component b_{n+1} is an integer corresponding to the bid's *weight* (the quantity of goods sought by the bid, which may be positive or negative). For the reader's convenience, we also define the alternative notation $w(\boldsymbol{b}) \coloneqq b_{n+1}$ to denote $\boldsymbol{b}$'s weight. Moreover, we restrict ourselves to positive and negative unit weights ± 1. This is without loss of generality, as any bid with a weight of $w(\boldsymbol{b}) \in \mathbb{Z}$ can be represented by $w(\boldsymbol{b})$ unit bids with the same vector and of the

same sign. Finally, when working with the notional reject good 0 as described in Section 2, we identify the bid $\boldsymbol{b}$ with the $(n+2)$-dimensional vector $(0, b_1, \ldots, b_n; b_{n+1})$ obtained by prepending a 0-th entry of value 0 (and thus indexing from 0 instead of 1). This allows us to define a demand correspondence $D_{\mathcal{B}}$ that maps each price vector $\boldsymbol{p}$ to a set of bundles demanded at $\boldsymbol{p}$, and an indirect utility function $f_{\mathcal{B}}$ associated with $\mathcal{B}$.

A bid $\boldsymbol{b}$ *demands good* $i \in [n]_0$ *at* $\boldsymbol{p}$ if the surplus $b_i - p_i$ at price $\boldsymbol{p}$ is maximal, that is if we have $i \in \arg\max_{i \in [n]_0}(b_i - p_i)$; recall that good 0 is the notional 'reject' good and $b_0 = p_0 = 0$ by definition. We say that $\arg\max_{i \in [n]_0}(b_i - p_i)$ is the set of *demanded goods of* $\boldsymbol{b}$ *at* $\boldsymbol{p}$. A bid $\boldsymbol{b}$ is *marginal* (on the set of its demanded goods) at $\boldsymbol{p}$ if $\boldsymbol{b}$ demands more than one good at $\boldsymbol{p}$. Moreover, we say that price $\boldsymbol{p}$ is marginal with respect to a given bid list $\mathcal{B}$ if there are bids in $\mathcal{B}$ that are marginal at $\boldsymbol{p}$, and non-marginal otherwise. We illustrate these definitions using Example 1 (Figure 2a). Recall that Alice's bid list $\mathcal{B}$ consists of two positively-weighted bids $\boldsymbol{b} \coloneqq (6, 6; 1)$ and $\boldsymbol{b}' \coloneqq (0, 4; 1)$. At prices $\boldsymbol{p} = (2, 4)$, the bid $\boldsymbol{b}$ is not marginal as it demands only good 1 while $\boldsymbol{b}'$ is marginal between the reject good and good 2. At prices $\boldsymbol{p} = (6, 6)$, the bid $\boldsymbol{b}$ is marginal between the reject good, good 1, and 2 while $\boldsymbol{b}'$ is rejected (it only demands the reject good). Finally, at prices $\boldsymbol{p} = (6, 2)$, both bids are non-marginal, as they each demand only good 2. More generally, all prices that lie on facets of the polyhedral complex in Figure 2a are marginal, while all prices in the interior of the polyhedra are non-marginal.

For any bid list $\mathcal{B}$, we define the *demand correspondence* $D_{\mathcal{B}}(\boldsymbol{p})$ at prices $\boldsymbol{p}$ as follows, distinguishing between the cases that $\boldsymbol{p}$ is marginal and non-marginal. If $\boldsymbol{p}$ is non-marginal, every bid $\boldsymbol{b} \in \mathcal{B}$ uniquely demands some good $i(\boldsymbol{b})$. In this case, $D_{\mathcal{B}}(\boldsymbol{p})$ is a singleton set consisting of the bundle $\boldsymbol{x}$ that is obtained by adding up an amount $w(\boldsymbol{b})$ of good $i(\boldsymbol{b})$ for each $\boldsymbol{b} \in \mathcal{B}$, i.e. $D_{\mathcal{B}}(\boldsymbol{p}) = \{\boldsymbol{x}\}$ with $\boldsymbol{x} \coloneqq \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \boldsymbol{e}^{i(\boldsymbol{b})}$. In Example 1, the demand correspondence at non-marginal prices $\boldsymbol{p} = (6, 2)$ is $D_{\mathcal{B}}(6, 2) = \{(0, 2)\}$, as both bids demand good 2. At $\boldsymbol{p} = (1, 2)$, the demand correspondence is $D_{\mathcal{B}}(1, 2) = \{(1, 1)\}$, as $\boldsymbol{b}$ uniquely demands good 2 and $\boldsymbol{b}'$ uniquely demands good 1. Figure 2a shows the demand in the different polyhedron interiors corresponding to non-marginal prices.

If $\boldsymbol{p}$ is marginal, $D_{\mathcal{B}}(\boldsymbol{p})$ consists of the discrete convex hull of the bundles demanded at non-marginal prices arbitrarily close to $\boldsymbol{p}$, where the discrete convex hull of a set of bundles X is defined as $\mathrm{conv}(X) \cap \mathbb{Z}^n$. In Figure 2a, we see that bundles demanded in the local neighbourhood of $\boldsymbol{p} = (0, 4)$ are $(1, 0)$ and $(1, 1)$, which implies $D_{\mathcal{B}}(\boldsymbol{p}) = \{(1, 0), (1, 1)\}$. At $\boldsymbol{p} = (4, 4)$, we have $D_{\mathcal{B}}(\boldsymbol{p}) = \{(1, 0), (0, 1), (1, 1), (0, 2)\}$.

For any bid list $\mathcal{B}$, we can define the *indirect utility function*

$$f_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \max_{i \in [n]_0} (b_i - p_i). \tag{3}$$

From (3) it is clear that we can compute $f_{\mathcal{B}}(\boldsymbol{p})$ efficiently. In our setting, we can also efficiently compute a bundle demanded at a given price $\boldsymbol{p}$. This *demand oracle problem* is noted in Paes Leme [2017] as an algorithmic primitive needed to implement the Walrasian tâtonnement procedure. If $\boldsymbol{p}$ is non-marginal, simply allocate each bid $\boldsymbol{b}$ a positive or negative item of the good $i(\boldsymbol{b})$ it uniquely demands and add up these items to obtain the demanded bundle $\boldsymbol{x} = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \boldsymbol{e}^{i(\boldsymbol{b})}$. However, if $\boldsymbol{p}$ is marginal and the bid list contains both positive and negative bids, we emphasise that independently allocating to each bid one of the goods it demands may not result in a bundle that lies in $D_{\mathcal{B}}(\boldsymbol{p})$. In Example 2 (depicted in Figure 2b), for instance, we see that at prices $\boldsymbol{p} = (4, 4)$, allocating a negative item of good 1 to bid $(4, 4; -1)$, a positive item of good 1 to bid $(4, 2; 1)$ and a positive item of good 2 each to bids $(2, 4; 1)$ and $(6, 6; 1)$ leads to a total bundle of $(0, 2) \notin D_{\mathcal{B}}(\boldsymbol{p})$. Instead, care must be taken to accept bids in a consistent way; one way to do this is to perturb entries of the price vector slightly so as to break ties, then note that the resulting bundle is demanded at the unperturbed prices. Note that a demand oracle does not provide us with a way to tell whether a given bundle of interest is demanded at $\boldsymbol{p}$.

A result by Baldwin and Klemperer [2021] shows that any SS demand correspondence can be represented as a finite list of positive and negative bids, and this representation is essentially unique (up to redundancies). Conversely, however, not all lists of positive and negative bids induce a strong substitutes valuation function. We call a bid list *valid* if the indirect utility function $f_{\mathcal{B}}$ defined in (3) is convex; Theorem 1 gives two further equivalent characterisations of validity. The proof is given in Appendix D.1. In Section 4, we will also introduce a weaker local notion of validity by introducing local validity in the ε-neighbourhood of a price $\boldsymbol{p}$.

Theorem 1. *Let $\mathcal{B}$ be a list of positive and negative bids and $f_{\mathcal{B}}$ be the associated indirect utility function as defined in* (3)*. The following conditions are equivalent.*

1. *$\mathcal{B}$ is valid.*
2. *There is no price vector $\boldsymbol{p}$ and pair of distinct goods $i, i' \in [n]_0$ at which the weights of the bids marginal on i and i' sum to a negative number.*
3. *$f_{\mathcal{B}}$ is the indirect utility function of a strong substitutes valuation u with quasi-linear utilities so that $D_{\mathcal{B}}(\boldsymbol{p}) = D_u(\boldsymbol{p})$ for all $\boldsymbol{p} \in \mathbb{R}^n$.*

2.3 Deciding validity of bid lists.

In our auction, bidders submit their lists of positive and negative bids to the auctioneer prior to the auction. We show in Theorem 2 that checking the validity of a given list of bids is computationally hard. The proof, given in Appendix C, exploits definition (2) of validity in Theorem 1. However, in Appendix C we provide a straightforward algorithm to verify the validity of a given bid list in polynomial time if the number of goods, or the number of negative bids, is bounded by a constant.

Theorem 2. *Deciding the validity of a given list of positive and negative bids is* coNP*-complete.*

Theorem 3. *Let $\mathcal{B}$ be a list of positive and negative bids. If the number of goods or the number of negative bids is bounded by a constant, there exists a polynomial-time algorithm for checking the validity of $\mathcal{B}$.*

Furthermore, many useful subclasses of bid lists (of arbitrary size) are easily checkable for validity in practice. Hence, while the coNP-completeness result is disappointing, it does not seem to be an important limitation on the auction, as sensible restrictions on the permitted bids can allow us to check validity efficiently in practice.

2.4 The computational challenges.

In this subsection we state the computational problems to be solved. For any bidder $j \in J$, where J is the set of bidders, $\mathcal{B}^j$ denotes the bids of bidder j. We assume each $\mathcal{B}^j$ is *valid*, as defined in Section 2.2, and provided as a list of vectors encoded in binary. Let $\mathcal{B}$ be the aggregate list of bids obtained by aggregating the lists of all bidders. As the aggregation of strong substitutes valuation functions is strong substitutes, $\mathcal{B}$ is valid. Figure 3a depicts the aggregate bid list of Alice and Bob from Figure 2. The running times of our algorithms are given in terms of n, $|J|$, $|\mathcal{B}|$ and $M := \max_{\boldsymbol{b} \in \mathcal{B}} \|\boldsymbol{b}\|_\infty$, the maximum bid vector entry.

Suppose the auctioneer intends to sell target bundle $\boldsymbol{t}$. Our aim is to compute a competitive equilibrium: a market-clearing price $\boldsymbol{p}$ at which $\boldsymbol{t}$ is demanded and an allocation of $\boldsymbol{t}$ to the various bidders so that every bidder receives a bundle they demand at $\boldsymbol{p}$. In the event that not enough bids are made for the target bundle, some items can go unsold, which is equivalent to the auctioneer buying them back from the market. To reflect this, the auctioneer places a total

of $\|\boldsymbol{t}\|_1 + 1$ positive bids at $\mathbf{0}$.[7] The computation of a competitive equilibrium separates into two problems.

The price-finding problem. Given the aggregate bid list $\mathcal{B}$ and target bundle $\boldsymbol{t}$, find the coordinate-wise minimal *market-clearing* price $\boldsymbol{p}$ at which $\boldsymbol{t}$ is demanded, that is $\boldsymbol{t} \in D_{\mathcal{B}}(\boldsymbol{p})$.

The allocation problem. Given a valid bid list $\mathcal{B}^j$ for each bidder $j \in J$, a target bundle $\boldsymbol{t}$ and market-clearing price $\boldsymbol{p}$, allocate $\boldsymbol{t}$ to the bidders so that each bidder receives a bundle they demand at $\boldsymbol{p}$, that is, a partition $(\boldsymbol{t}^j)_{j\in J}$ of $\boldsymbol{t}$ with $\boldsymbol{t}^j \in D_{\mathcal{B}^j}(\boldsymbol{p})$ for all $j \in J$.

Note that we do not ask for a breakdown of which of j's bids are accepted on which goods.[8] Indeed, we show that a solution to the allocation problem can be obtained without this knowledge.

Theorem 4. *Suppose that each bidder $j \in J$ has SS demand correspondence D^j. Given a target allocation $\boldsymbol{t}$ and a market-clearing price $\boldsymbol{p}$ for $\boldsymbol{t}$, there exists a partition $\boldsymbol{t} = \sum_{j\in J} \boldsymbol{t}^j$ such that $\boldsymbol{t}^j \in D^j(\boldsymbol{p})$.*

Theorem 4 (cf. Danilov et al. [2001], Murota [2003], Milgrom and Strulovici [2009]) ensures that if the target bundle $\boldsymbol{t}$ is aggregately demanded at market-clearing price $\boldsymbol{p}$, then there exists an allocation of $\boldsymbol{t}$ among the bidders so that every bidder receives a bundle she demands. However, care must be taken in finding such an allocation, as bidders cannot simply be allocated an arbitrary bundle they demand at the market-clearing price. (Indeed, in Figure 2 the lowest prices at which Alice and Bob aggregately demand $(1, 1)$ is given by $\boldsymbol{p} = (4, 4)$. If we allocate Alice the whole bundle $(1, 1)$, Bob does not demand the empty remainder $(0, 0)$ at $\boldsymbol{p}$.)

Working in the valuation oracle setting, Paes Leme and Wong [2017] present an algorithm that solves the allocation problem if there exist prices at which the target bundle is uniquely demanded. Such prices are guaranteed in the single-unit case (i.e. when there is only one unit of every good) but need not exist in the multi-unit case. (Indeed, the bundle $(1, 1)$ is not uniquely demanded at any prices for the aggregate demand correspondence of Alice and Bob shown in Figure 3a.)

If we had (oracle) access to the valuations of the bidders, we could perturb the valuations such that the target bundle is uniquely demanded at some price by subtracting carefully constructed functions that are discrete-convex in one variable.[9] However, recall that our bidding language does not give straightforward access to bidder valuations.[10] Moreover, it is not clear how to perturb the bids individually to achieve a suitable 'indirect' perturbation of the valuations. Instead, our allocation algorithm takes the approach of perturbing bids bidder by bidder.

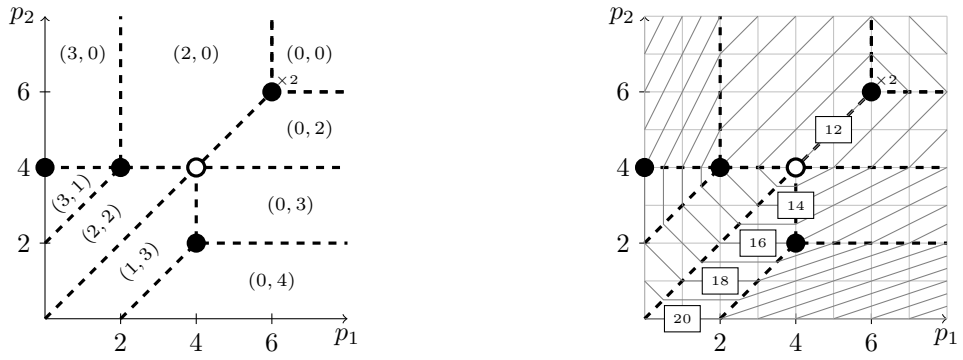
Some intuition can be gained by considering the addition of a small 'random' perturbation vector $\boldsymbol{v}^j$ to all the bids in bid set $\mathcal{B}^j$ (a different $\boldsymbol{v}^j$ for each $j \in J$) and recomputing the new market-clearing prices. This has the effect of breaking ties between bids in two different bid sets. (This also uses the plausible fact, established in Proposition 14, that small perturbations make only small changes to the market clearing price(s).) However, the perturbations need

[7] Of course, any positive reserve price can be reflected by appropriately locating the auctioneer's bids.

[8] Once a partition of $\boldsymbol{t}$ has been obtained, it is easy to compute such a breakdown since the task can be formulated as a maximum network flow problem, in which each positive bid $\boldsymbol{b}$ is a source node with outgoing flow given by its weight $w(\boldsymbol{b})$. There are $n + 1$ intermediate nodes, one node v_i for each good i, including the 'reject' good 0. Connect each positive bid node to the goods on which it may be accepted. Each v_i is connected to a sink node whose incoming flow is the i-th element of $\boldsymbol{t}^j$. (For $i = 0$ the incoming flow is given by the total weight of bids minus the number of elements of $\boldsymbol{t}^j$.) Each negative bid $\boldsymbol{b}$ is a sink node whose flow is its weight $w(\boldsymbol{b})$, and is connected to any v_i for which $\boldsymbol{b}$ demands good i.

[9] See Murota [2003, Theorem 6.13 (4)].

[10] To the best of our knowledge, the most efficient way to compute the valuation of bundle $\boldsymbol{x}$ is to determine a price $\boldsymbol{p}$ at which $\boldsymbol{x}$ is demanded using our price-finding algorithm from Section B, computing the indirect utility $f_{\mathcal{B}}(\boldsymbol{p})$ at this price and then solving (2) for $u(\boldsymbol{x})$.



(a) Aggregate demand of Alice and Bob

(b) Lyapunov function g for target bundle $\boldsymbol{t} = (1, 1)$

Figure 3: Figure (a) shows the demand of the aggregate bid list obtained by aggregating Alice's and Bob's bids from Figure 2, while (b) depicts a contour plot of the Lyapunov function g for target bundle $\boldsymbol{t} = (1, 1)$. Note that $\boldsymbol{p} = (4, 4)$ is the minimal market-clearing price for $\boldsymbol{t}$. Starting at $\mathbf{0}$, MinUp repeatedly moves in direction $d = (1, 1)$ until it reaches $\boldsymbol{p}$, whereas LongStepMinUp makes a single long step from $\mathbf{0}$ to $\boldsymbol{p}$.

to be exponentially small in the number of goods in order to ensure that all possible ties are broken. (One can check, noting the pigeonhole principle, that if perturbations are multiples of some inverse-polynomial, then they will have distinct subsets having the same sum, losing the guarantee that all possible ties are broken.) Our algorithm of Section 4 avoids this by systematically perturbing and un-perturbing bid sets, making decisions on marginal bids as it proceeds.

A desideratum that we do not address here is fairness of allocation, in the sense that equal bidders should be treated equally. Note that this is in conflict with our requirement that bids should be allocated entirely or not at all: in the simplest configuration where two bidders offer the same price for a single available item, only one of those bidders will have their bid allocated.

3 Finding market-clearing prices.

In order to determine component-wise minimal prices at which the market is cleared, we apply an iterative steepest-descent approach from the discrete convex optimisation literature [Shioura, 2017]. This approach generalises an algorithm used by Ausubel's ascending auction design [Ausubel, 2006] and Gul and Stacchetti [2000] for the task of finding equilibrium prices in single-unit markets. Shioura [2017] and others define the *Lyapunov function* with regard to indirect utility function f_u and target bundle $\boldsymbol{t}$ as $g_{\boldsymbol{t}}(\boldsymbol{p}) \coloneqq f_u(\boldsymbol{p}) + \boldsymbol{t} \cdot \boldsymbol{p}$. Given a starting point $\boldsymbol{p} \coloneqq \boldsymbol{p}^0$ that is known to be dominated by the market-clearing price $\boldsymbol{p}^*$, the steepest-descent method repeatedly applies submodular minimisation (cf. Section 2) to find the component-wise minimal subset $S \subseteq [n]$ minimising $g_{\boldsymbol{t}}(\boldsymbol{p}+\boldsymbol{e}^S) - g_{\boldsymbol{t}}(\boldsymbol{p})$ and moves a unit step in this direction by updating $\boldsymbol{p} \coloneqq \boldsymbol{p}+\boldsymbol{e}^S$. The method terminates once a local maximum is reached, upon which it has found component-wise minimal market clearing prices. Adapting this method to our bidding-language setting, we can use our direct knowledge of the bids and (3) to express $g_{\boldsymbol{t}}$ for any list $\mathcal{B}$ as

$$g_{\boldsymbol{t}}(\boldsymbol{p}) \coloneqq f_{\mathcal{B}}(\boldsymbol{p}) + \boldsymbol{t} \cdot \boldsymbol{p} = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \max_{i \in [n]_0} (b_i - p_i) + \boldsymbol{t} \cdot \boldsymbol{p}. \tag{4}$$

As $\mathcal{B}$ is a valid bid list by assumption, Theorem 1 guarantees that the function $f_{\mathcal{B}}$ defined in (3) is the indirect utility function f_u of some strong substitutes valuation u, so (4) is a well-defined Lyapunov function. (Figure 3b depicts a contour plot of the Lyapunov function defined for the aggregate demand of Alice and Bob as given in Examples 1 and 2 and shown in Figure 3a.) Moreover, (4) implies that $g_{\boldsymbol{t}}$ can be evaluated at any $\boldsymbol{p}$ in time $O(n|\mathcal{B}|)$.

In order to improve the running time of the steepest-descent method, Shioura [2017] proposes aggregating several steps in the same direction into a single 'long' step. After finding the component-wise minimal direction $\boldsymbol{e}^S$, we move $\lambda(\boldsymbol{p}, S)$ unit steps in this direction at once by updating $\boldsymbol{p} \coloneqq \boldsymbol{p} + \lambda(\boldsymbol{p}, S)\boldsymbol{e}^S$, where $\lambda(\boldsymbol{p}, S)$ is defined as

$$\lambda(\boldsymbol{p}, S) \coloneqq \max\{\lambda \in \mathbb{Z}_+ \mid g'(\boldsymbol{p}; S) = g'(\boldsymbol{p} + (\lambda - 1)\boldsymbol{e}^S; S)\} \tag{5}$$

and $g'(\boldsymbol{p}; S) \coloneqq g_{\boldsymbol{t}}(\boldsymbol{p} + \boldsymbol{e}^S) - g_{\boldsymbol{t}}(\boldsymbol{p})$. In Appendix B, we present two methods for computing (5) at every iteration of the steepest-descent procedure. The first method uses binary search to find $\lambda(\boldsymbol{p}, S)$ in time $O(n|\mathcal{B}| \log M)$, while the second (the 'demand change' method) exploits our knowledge of the individual bids to determine the step length in time $O(n|\mathcal{B}|^2)$. Hence we see that as the running time guarantees differ, the best method in practice is context-specific. Overall, these improvements yield the first price-finding algorithms that are fully polynomial in the size of the bid lists.

Theorem 5. *The steepest descent approach takes time* $O(n^2|\mathcal{B}|^2 \log M + n|\mathcal{B}|T(n))$ *and* $O(n^2|\mathcal{B}|^3 + n|\mathcal{B}|T(n))$ *with the* binary search *and* demand change *long step methods, respectively.*

The computation time of the steepest descent method, even with long steps, is dominated by the task of finding a component-wise minimal submodular minimiser. In practice, we can exploit our knowledge of the bids to decrease the dimensionality of the submodular function to be minimised, which leads to performance improvements. For full details on these practical improvements, and on the 'unit-step' and 'long-step' steepest-descent methods for computing market-clearing prices in the bidding-language setting, we refer to Appendix B.

4 Allocations to the separate bidders.

Suppose we are given a market-clearing price $\boldsymbol{p}$ for target bundle $\boldsymbol{t}$. We now present an algorithm that solves the allocation problem, i.e. finds a partition $(\boldsymbol{t}^j)_{j \in J}$ of the target bundle $\boldsymbol{t}$ such that $\boldsymbol{t}^j$ is demanded by bidder j at price $\boldsymbol{p}$. We note that while the market-clearing price $\boldsymbol{p}$ returned by our steepest descent approach in Section 3 is component-wise minimal, our allocation algorithm works for any integral market-clearing price.

Algorithm 1 ALLOCATE

Input: Initial allocation problem $\mathcal{A}$.
Output: Target bundle allocation.
Allocate all non-marginal bids by applying NONMARGINALS (Algorithm 3).
while $\mathcal{A}$ has a non-empty bid list **do**
    Run the FINDPARAMS subroutine (Algorithm 2).
    if FINDPARAMS returns a demand cluster (I, j^*) with at most one link good i^* **then**
        Process unambiguous marginal bids with UNAMBIGUOUSMARGINALS (Algorithm 4).
    else
        Simplify the allocation problem with SHIFTPROJECTUNSHIFT (Algorithm 5).
    Allocate all (newly) non-marginal bids with NONMARGINALS (Algorithm 3).
return the partial allocation bundle $\boldsymbol{m}^j$ for each bidder $j \in J$.

Our algorithm ALLOCATE, stated in Algorithm 1, repeatedly simplifies the problem until it becomes vacuous. It generates a sequence of *allocation problems* by iteratively allocating parts of the target bundle to bidders and removing satisfied bids from the bid lists until the residual supply bundle (initially the target bundle) is empty. In order to capture this formally, we define a generalised allocation problem in Section 4.1; the definition introduces a weaker notion of local

validity for bid lists and also features a partial allocation bundle for each bidder as well as a residual target bundle. With every allocation problem we associate a corresponding *marginal bids multigraph* and *derived graph*. These graphs, and key terminology such as demand clusters and link goods, are defined in Section 4.2. The marginal bids multigraph is used to quantify progress and establish the running time of the algorithm, while the derived graph is required for Allocate to decide whether to apply the UnambiguousMarginals or ShiftProjectUnshift procedure.

The algorithm starts by removing all non-marginal bids and transferring their demanded items from the residual supply to the partial allocation bundle of the bid's owner. This is performed by the procedure NonMarginals, which is defined in Section 4.3. Next, the algorithm calls the FindParams subroutine (defined in Section 4.2), which constructs and works on the derived graph. If FindParams identifies a subset of marginal bids that can be allocated unambiguously, Allocate invokes UnambiguousMarginals (defined in Section 4.3) to process these bids. Otherwise FindParams identifies parameters that Allocate uses to call the ShiftProjectUnshift procedure. ShiftProjectUnshift is defined in Section 4.4; it does not allocate items and delete bids but instead simplifies the allocation problem by shifting and projecting the bids in order to strictly reduce the demand ties between bids. Finally, NonMarginals is invoked again to process any bids that may have become non-marginal. The algorithm repeats this process until the allocation problem become vacuous in the sense that all bid lists are empty, at which point the partial allocation bundles constitute a valid allocation of the target bundle $\boldsymbol{t}$ at prices $\boldsymbol{p}$. Section 4.5 gives the running time and a proof of correctness for Allocate.

4.1 The generalised allocation problem.

We now introduce allocation problems formally and define the terminology and technical tools required to reason about how Allocate successively reduces the initial allocation problem using the three procedures NonMarginals, UnambiguousMarginals and ShiftProjectUnshift. In general, we note that applying these procedures may result in bid lists that are no longer valid in the sense of Theorem 1. (Figure 5 in Section 4.3 gives an example of how applying NonMarginals can result in an invalid bid list.) Instead, we introduce the notion of local validity that requires bid lists to be valid only in the neighbourhood of the market-clearing price $\boldsymbol{p}$. For prices $\boldsymbol{p}$ in this neighbourhood, we can define the demand correspondence $D_{\mathcal{B}}(\boldsymbol{p})$ in the same way as for globally valid bid lists. Let $B(\boldsymbol{p}, \varepsilon) \coloneqq \{\boldsymbol{y} \in \mathbb{R}^n \mid \|\boldsymbol{y} - \boldsymbol{p}\|_\infty < \varepsilon\}$ denote the open ball with regard to the L_∞-norm.

Definition 3. We say that a set of bids $\mathcal{B}$ is *(locally) valid at* $\boldsymbol{p}$ if there exists an open, convex neighbourhood P of $\boldsymbol{p}$ that satisfies the following equivalent criteria.

1. The indirect utility function $f_{\mathcal{B}}$ restricted to P is convex.
2. There is no price vector $\boldsymbol{p} \in P$ and pair of goods $i, i' \in [n]_0$ at which the weights of the bids marginal on i and i' sum to a negative number.

For a set of bids $\mathcal{B}$ that is valid at $\boldsymbol{p}$, define $D_{\mathcal{B}}(\boldsymbol{p})$ to be the discrete convex hull of bundles demanded at UDRs adjacent to $\boldsymbol{p}$. If $P = B(\boldsymbol{p}, \varepsilon)$ for some $\varepsilon > 0$, we say that $\mathcal{B}$ is *ε-valid at* $\boldsymbol{p}$.

Proposition 6. *The two criteria in Definition 3 are equivalent.*

The proof is given in Appendix D.2.

Definition 4. An *allocation problem* is a 4-tuple $\mathcal{A} = [\boldsymbol{p}, (\mathcal{B}^j)_{j \in J}, (\boldsymbol{m}^j)_{j \in J}, \boldsymbol{r}]$, where

1. $\boldsymbol{p} \in \mathbb{R}^{n+1}$ is a price vector with $p_0 = 0$,

2. for each bidder $j \in J$, $\mathcal{B}^j$ is a list of bids in $\{0\} \times \left(\mathbb{Z} + \{0, \frac{1}{10}\}\right)^n \times \mathbb{Z}$ that is locally valid at $\boldsymbol{p}$,

3. $\boldsymbol{m}^j \in \mathbb{Z}_+^{n+1}$ for each $j \in J$ is the partial allocation,

4. $\boldsymbol{r} \in \mathbb{Z}_+^{n+1}$ is the residual supply,

and there exists a valid allocation $(\boldsymbol{r}^j)_{j \in J}$ of the residual supply $\boldsymbol{r}$ to bidders, i.e. $\boldsymbol{r}^j \in D_{\mathcal{B}^j}(\boldsymbol{p})$ and $\sum_{j \in J} \boldsymbol{r}^j = \boldsymbol{r}$. For any such valid allocation, we say that $\boldsymbol{t}^j := \boldsymbol{r}^j + \boldsymbol{m}^j$ is a *solution* to $\mathcal{A}$.

Note that the bids need not be integral and the bid lists for the bidders are only required to be locally valid at $\boldsymbol{p}$. The intuition behind this definition is to capture the action of successively allocating items of the target bundle $\boldsymbol{t}$ to bidders, which may break global validity but preserves local validity at $\boldsymbol{p}$. For each bidder $j \in J$, the partial allocation $\boldsymbol{m}^j$ is a bundle denoting the subset of $\boldsymbol{t}$ already allocated to j, while $\boldsymbol{r}$ denotes the remaining part of $\boldsymbol{t}$ not yet allocated to any bidder. The 0-th coordinates of $\boldsymbol{m}^j$ and $\boldsymbol{r}$ denote the number of reject goods (not yet) allocated.

Definition 5. We say that $\mathcal{A}'$ is a *valid reduction from* $\mathcal{A}$ if $\mathcal{A}'$ is an allocation problem and all solutions to $\mathcal{A}'$ are also solutions to $\mathcal{A}$.

In the *initial problem*, the residual supply is given by the target bundle and the partial allocation vectors are $\mathbf{0}$. Note that t_0 denotes the total number of rejected bids and can be computed as the total weight of bids minus the total number of items in $\boldsymbol{t}$. In the *vacuous problem*, the bid lists are empty and the residual supply is $\mathbf{0}$.

Technical tools. The following observations and lemma will be used in Section 4.4 to prove that applying the procedure ShiftProjectUnshift to an allocation problem $\mathcal{A}$ produces an allocation problem that is a valid reduction from $\mathcal{A}$. We define the *surplus gap* of a bid $\boldsymbol{b}$ at $\boldsymbol{p}$ as the difference between the utility derived from a demanded good and the maximum utility derived from a non-demanded good. This allows us to describe the goods that $\boldsymbol{b}$ demands when we perturb the price or the bid by a small amount. Moreover, we see in Lemma 9 that the size of the neighbourhood around $\boldsymbol{p}$ in which a bid list is valid can be lower bounded by the surplus gaps of the bids in $\mathcal{B}$. Formally, let $I = \arg\max_{i \in [n]_0} b_i - p_i$ denote the goods demanded by $\boldsymbol{b}$ at $\boldsymbol{p}$. Then if $I \neq [n]_0$, we define the *surplus gap* of $\boldsymbol{b}$ at $\boldsymbol{p}$ as

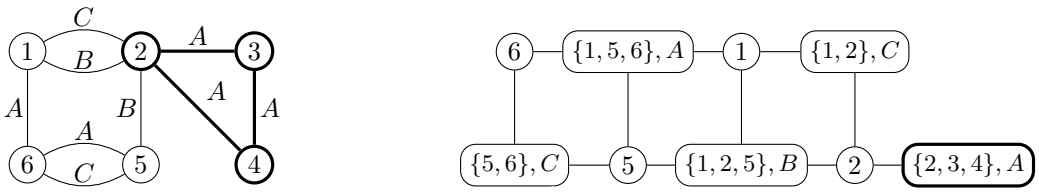
$$\max_{i \in [n]_0} (b_i - p_i) - \max_{i \in [n]_0 \setminus I} (b_i - p_i).$$

Observation 7. *If bid $\boldsymbol{b}$ demands goods I at $\boldsymbol{p}$ with a surplus gap of at least ε, then for any price $\boldsymbol{p}' \in B(\boldsymbol{p}, \varepsilon/2)$, the goods I' demanded by $\boldsymbol{b}$ at $\boldsymbol{p}'$ form a subset of I.*

Observation 8. *If bid $\boldsymbol{b}$ demands goods I at $\boldsymbol{p}$ with a surplus gap of at least ε, then for any bid $\boldsymbol{b}' \in B(\boldsymbol{b}, \varepsilon/4)$ and price $\boldsymbol{p}' \in B(\boldsymbol{p}, \varepsilon/4)$, the goods I' demanded by $\boldsymbol{b}'$ at $\boldsymbol{p}'$ form a subset of I.*

Lemma 9. *If $\mathcal{B}$ is δ-valid at $\boldsymbol{p}$ for some $\delta > 0$ and all bids in $\mathcal{B}$ have a surplus gap of at least ε at $\boldsymbol{p}$, then $\mathcal{B}$ is $\varepsilon/2$-valid at $\boldsymbol{p}$.*

Proof. We show that $f_{\mathcal{B}}$ is convex on $B(\boldsymbol{p}, \varepsilon/2)$ by verifying that it satisfies midpoint convexity. Note that $f_{\mathcal{B}}$ is linear on the line segment connecting $\boldsymbol{p}$ to $\boldsymbol{q}$, for any $\boldsymbol{q} \in B(p, \varepsilon/2)$. Indeed, $f_{\mathcal{B}}$ is the sum of terms $w(\boldsymbol{b}) \max_{i \in [n]_0}(b_i - p_i)$ and it suffices to show that each term is linear. Hence fix $\boldsymbol{b}$ and define $h(\boldsymbol{p}) = w(\boldsymbol{b}) \max_{i \in [n]_0}(b_i - p_i)$ as well as $\boldsymbol{r} = \theta\boldsymbol{p} + (1-\theta)\boldsymbol{q}$. By Observation 7, the goods demanded by $\boldsymbol{b}$ at $\boldsymbol{q}, \boldsymbol{r}$ and $\boldsymbol{p}$ satisfy $I_q \subseteq I_r \subseteq I_{\boldsymbol{p}}$. Hence, for any $i^* \in I_q$, we have $h(\boldsymbol{p}) = w(\boldsymbol{b})(b_{i^*} - p_{i^*}) = \theta h(\boldsymbol{p}) + (1-\theta)h(\boldsymbol{q})$. Now fix $\boldsymbol{q}, \boldsymbol{q}' \in B(\boldsymbol{p}, \varepsilon/2)$ and choose $\theta > 0$ so that $\boldsymbol{r} = \theta\boldsymbol{p} + (1-\theta)\boldsymbol{q}$, $\boldsymbol{r}' = \theta\boldsymbol{p} + (1-\theta)\boldsymbol{q}'$ and $(\boldsymbol{r} + \boldsymbol{r}')/2$ are in $B(\boldsymbol{p}, \delta)$. As $f_{\mathcal{B}}$ is convex



(a) A marginal bids multigraph. **(b)** A derived graph.

Figure 4: Example of a marginal bids multigraph (a) and derived graph (b) with three bidders A, B, C and six goods $1, \ldots, 6$. Goods $1, 2, 5, 6$ are link goods. Goods $2, 3, 4$ form a demand cluster $(\{2, 3, 4\}, A)$ with one link good, represented by a leaf demand cluster in the derived graph (bolded in both graphs).

on $B(\boldsymbol{p}, \delta)$, by assumption, we have $f\left(\frac{1}{2}(\boldsymbol{r} + \boldsymbol{r}')\right) \leq \frac{1}{2}f(\boldsymbol{r}) + \frac{1}{2}f(\boldsymbol{r}')$. Secondly, we have $f(\boldsymbol{r}) = \theta f(\boldsymbol{p}) + (1-\theta)f(\boldsymbol{q})$, $f(\boldsymbol{r}') = \theta f(\boldsymbol{p}) + (1-\theta)f(\boldsymbol{q}')$ and $f(\frac{1}{2}(\boldsymbol{r}+\boldsymbol{r}')) = \theta f(\boldsymbol{p}) + (1-\theta)f(\frac{1}{2}(\boldsymbol{q}+\boldsymbol{q}'))$ due to the linearity of $f_{\mathcal{B}}$ on the line segments connecting $\boldsymbol{p}$ to $\boldsymbol{q}, \boldsymbol{q}'$ and $(\boldsymbol{q}+\boldsymbol{q}')/2$. This implies midpoint convexity, $f\left(\frac{1}{2}(\boldsymbol{q} + \boldsymbol{q}')\right) \leq \frac{1}{2}f(\boldsymbol{q}) + \frac{1}{2}f(\boldsymbol{q}')$. □

4.2 The marginal bids graph and the derived graph.

With every allocation problem $\mathcal{A}$ we associate a marginal bids multigraph and a derived graph. As mentioned above, the marginal bids multigraph allows us to provide upper bounds on the running time of our allocation algorithm. Moreover, in order to decide whether to apply UnambiguousMarginals or ShiftProjectUnshift and to determine the input parameters for these procedures, our Allocate algorithm calls the subroutine FindParams, defined in this section, which first constructs the derived graph and then attempts to find a maximal open or closed path in this graph.

Definition 6. The *marginal bids graph* $G_{\mathcal{A}}$ associated with allocation problem $\mathcal{A}$ is an undirected edge-labelled multigraph whose vertices are the goods $[n]_0$ (including the 'reject' good). $G_{\mathcal{A}}$ has an edge (i, i') labelled with j if bidder j has a bid that is marginal at $\boldsymbol{p}$ between i and i'.

For any bidder j, let $G_{\mathcal{A}}[j]$ denote the subgraph of $G_{\mathcal{A}}$ induced by all j-labelled edges; note that this graph is simple, as there is at most one edge labelled j connecting two goods i and i' in $G_{\mathcal{A}}$. A pair (I, j) with $I \subseteq [n]_0$ and $j \in J$ is a *demand cluster* if I is the set of vertices of some connected component of $G_{\mathcal{A}}[j]$. Intuitively, this captures the dependencies between the demands of (and thus the possible allocations to) bidder j's bids. A vertex is a *link good* if its incident edges are labelled with more than one bidder. (Effectively, a link good expresses a dependency between demand clusters of different bidders.) We call a cycle in $G_{\mathcal{A}}$ a *multi-bidder cycle* if it contains edges labelled by different bidders. A vertex in such a cycle is called a *cycle-link good* if its two edges in the cycle are labelled differently. Note that any multi-bidder cycle has at least two cycle-link goods. Like link goods, multi-bidder cycles and their cycle-link goods capture the dependencies between demands of (and thus possible allocations to) different bidders.

Definition 7. The *derived graph* $D_{\mathcal{A}}$ associated with allocation problem $\mathcal{A}$ is a simple bipartite graph whose two disjoint and independent vertex sets are the set of link goods and the set of demand clusters of $G_{\mathcal{A}}$. There is an edge between link good i and demand cluster (I, j) if $i \in I$.

Note that by the definition of link goods, every link good is adjacent to at least two demand clusters in the derived graph. This is consistent with our observation that a link good can be interpreted as a link between connected components of $G_{\mathcal{A}}[j]$ for different bidders j. Figure 4 gives an example of a marginal bids multigraph and its corresponding derived graph.

We describe a procedure to construct the derived graph in near-linear time $O(\alpha(n)n|\mathcal{B}|)$ using a disjoint-union data structure. Here $\alpha(\cdot)$ is the inverse Ackermann function, which grows

extremely slowly and is near-constant in our context, as $\alpha(n) \leq 4$ for any $n \leq 2^{2^{2^{2^{16}}}}$. The disjoint-union data structure (cf. Tarjan and van Leeuwen [1984]) maintains a representation of a set partition and admits a $\alpha(n)$-time operation to merge two subsets of the partition. Internally, it assigns a distinct label to each subset of the partition and provides $\alpha(n)$-time access to the label of the subset in which a given element lies.

First we show how to compute the demand clusters for each bidder. Fix a bidder j and initialise the disjoint-union data structure. For each bid $\boldsymbol{b} \in \mathcal{B}^j$, compute the marginal goods S at prices $\boldsymbol{p}$ and, fixing any $i \in S$, merge the sets containing i and i' for all $i' \in S \setminus \{i\}$. Now the data structure has learnt the vertex partition induced by the connected components of $G_{\mathcal{A}}$ in time $O(\alpha(n)n|\mathcal{B}^j|)$. In order to recover this partition and express it as a family of sets, we initialise an empty family $\mathcal{K}$ of sets, where each set in $\mathcal{K}$ will have an associated label. For each good in $i \in [n]$, determine the label l of i's subset in the data structure. If there already is a set in $\mathcal{K}$ with label l, add i to this set. Otherwise, add the new singleton set $\{i\}$ with label l to $\mathcal{K}$. Finally, iterate through $\mathcal{K}$ and delete all singleton sets. This takes time $O(\alpha(n)n)$. Now $\mathcal{K}$ represents the family of demand clusters of bidder j. Hence in total it takes time $O(\alpha(n)n|\mathcal{B}|)$ to compute the demand clusters for all bidders.

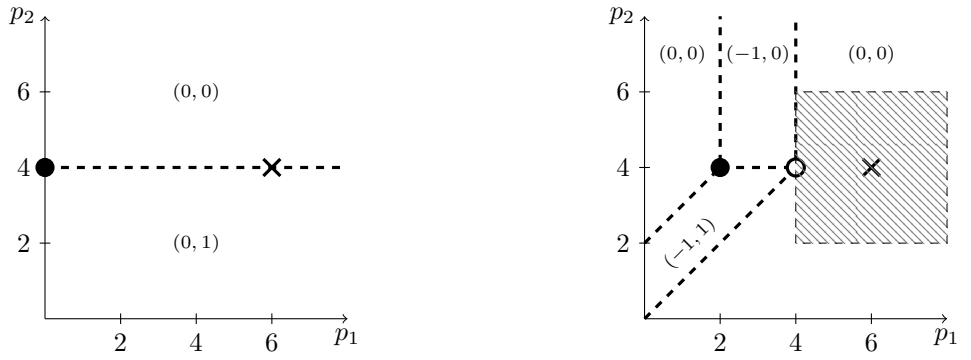
In order to compute the link goods, iterate through the demand clusters of all bidders and count the number of times each good appears. If a good appears at least twice, it is a link good. Once we know which good is a link good, we can compute the edges of the derived graph: for each demand cluster (I, j), add an edge between (I, j) and each link good in I. Iterating through the demand clusters of all bidders takes $O(nm) = O(n|\mathcal{B}|)$ time.

The FindParams subroutine. In every iteration, the Allocate algorithm employs a subroutine to decide, using the derived graph, whether to invoke UnambiguousMarginals or ShiftProjectUnshift, and to compute the input parameters for the respective procedure. This subroutine FindParams, given in Algorithm 2, takes as input an allocation problem and returns one of three possible outputs: a demand cluster (I, j) with no link goods, a demand cluster (I, j) with one link good denoted i^*, or a cycle-link good i^* and the label j^* of one of its incident edges in the multi-bidder cycle. The Allocate algorithm invokes UnambiguousMarginals if FindParams returns a demand cluster and ShiftProjectUnshift if it returns a cycle-link good and edge label.

FindParams works by walking through the graph to find a maximal path. A *path* in a graph is a sequence of distinct vertices $v_1, \ldots, v_k$ such that v_i and v_{i+1} are connected by an edge for all $i \in [k-1]$; it is *maximal* if we cannot extend the path with any vertex v_{k+1}. We say that the path is *closed* if there is an edge between v_1 and v_k, and *open* otherwise. (Hence a closed path is a cycle.)

Algorithm 2 FindParams

Compute the derived graph $D_{\mathcal{A}}$.
if $D_{\mathcal{A}}$ contains an isolated demand cluster **then**
    return this demand cluster (without a link good).
Starting from any link good, take a walk in $D_{\mathcal{A}}$ to find a maximal path that alternates between link good and demand cluster vertices (without revisiting edges).
Let i^* and (I, j^*) be the last link good and demand cluster visited, respectively.
if the path is open **then**
    return (I, j^*) and i^*.
else
    return i^* and j^*.



(a) Alice's bid list after applying NonMarginals. **(b)** Bob's list after applying NonMarginals.

Figure 5: The resulting bid lists after applying NonMarginals (Algorithm 3) to Alice's and Bob's bid lists from Figure 2 at prices $\boldsymbol{p} = (6, 4)$ (marked by a cross). Note that Alice's bid list remains globally valid, whereas Bob's bid list is only locally 2-valid at $\boldsymbol{p}$ (indicated by the hatched square).

Lemma 10. *In time $O(\alpha(n)n|\mathcal{B}|)$,* FindParams *returns a demand cluster (I, j^*) with no link goods, a demand cluster (I, j^*) with one link good i^*, or a cycle-link good i^* and the label j^* of one of its incident edges in the multi-bidder cycle.*

Proof. Note that any path has length at most $2n$. Hence constructing the derived graph, which takes time $O(\alpha(n)n|\mathcal{B}|)$, dominates the running time. By construction of the derived graph, an isolated demand cluster in $D_{\mathcal{A}}$ contains no link goods, while a leaf demand cluster contains exactly one link good. Hence if there is an isolated demand cluster, the subroutine returns a demand cluster without a link good. If the path found is open, the last vertex must be a leaf demand cluster and the subroutine returns a demand cluster with single link good. Now suppose the path is closed and consider only the cycle formed by alternating link good and demand cluster vertices. Firstly, note that there is a path between consecutive link goods in the marginal bids graph $G_{\mathcal{A}}$ using only j-labelled edges. Secondly, consecutive demand clusters have different bidder labels. These two observations imply that $G_{\mathcal{A}}$ contains a simple multi-bidder cycle with cycle-link good i^* and an incident edge labelled with j^*. □

4.3 Allocating unambiguous bids.

Non-marginal bids. Suppose bidder j has a non-marginal bid $\boldsymbol{b}$ at market-clearing prices $\boldsymbol{p}$ that demands good $i \in [n]_0$ in the allocation problem $\mathcal{A}$. Then this bid contributes exactly $w(\boldsymbol{b}) \in \{-1, 1\}$ items of good i to any solution of $\mathcal{A}$. Hence we can unambiguously allocate these items to $\boldsymbol{m}^j$ and remove them from the residual supply $\boldsymbol{r}$, thus accepting the non-marginal bid on the appropriate good. NonMarginals, given in Algorithm 3, processes all non-marginal bids in this way. Note that while this operation may not preserve global validity of bid lists, the resulting lists remain locally valid at $\boldsymbol{p}$, so that the result is a valid allocation problem. Figure 5 gives an example. Allocate calls NonMarginals in every iteration in order to process non-marginal bids that can be allocated unambiguously.

Lemma 11. *Given a valid allocation problem $\mathcal{A}$,* NonMarginals *(Algorithm 3) outputs a reduction $\mathcal{A}'$ of $\mathcal{A}$ in linear time. Moreover, we have $G_{\mathcal{A}} = G_{\mathcal{A}'}$.*

Proof. Let $\mathcal{A}' = [\boldsymbol{p}, (\mathcal{B}^j)'_{j\in J}, (\boldsymbol{m}^j)'_{j\in J}, \boldsymbol{r}']$ be the output of NonMarginals. First we show that $\mathcal{A}'$ is a valid allocation problem. Fix some $j \in J$. Since $\mathcal{B}^j$ is locally valid by assumption, the indirect utility function $f_{\mathcal{B}^j}$ is convex in some small neighbourhood of $\boldsymbol{p}$. Allocating a non-marginal bid to bidder j corresponds to subtracting from the utility function $f_{\mathcal{B}^j}(q)$ the term $\max_{i\in[n]_0}(b_i - q_i)$. In a sufficiently small neighbourhood of $\boldsymbol{p}$, this term is linear, so the resulting

Algorithm 3 NonMarginals (accept non-marginal bids)

Input: Allocation problem $\mathcal{A} = [\boldsymbol{p}, (\mathcal{B}^j)_{j\in J}, (\boldsymbol{m}^j)_{j\in J}, \boldsymbol{r}]$.
Output: Reduced allocation problem $\mathcal{A}'$ without non-marginal bids.
for all bidders $j \in J$ **do**
    for all non-marginal bids $\boldsymbol{b} \in \mathcal{B}^j$ **do**
        Determine unique good i demanded by $\boldsymbol{b}$ and remove $\boldsymbol{b}$ from $\mathcal{B}^j$.
        Increment m_i^j by $w(\boldsymbol{b})$.
        Decrement r_i by $w(\boldsymbol{b})$.

utility function is also convex in some open neighbourhood of $\boldsymbol{p}$. It is straightforward to see that $(\boldsymbol{t}^j)_{j\in J}$ is a solution to $\mathcal{A}'$ if and only if it is a solution to $\mathcal{A}$, so $\mathcal{A}'$ is a reduction of $\mathcal{A}$. To see that $G_{\mathcal{A}} = G_{\mathcal{A}'}$, note that the marginal bids are unchanged. □

Unambiguous marginal bids. Let $\mathcal{A}$ be an allocation problem without non-marginal bids and suppose FindParams returns a demand cluster (I, j) with at most one link good. UnambiguousMarginals is invoked by Allocate to unambiguously allocate the bids specified by this demand cluster. By the definition of a demand cluster, none of bidder j's bids are marginal between items in both I and $[n]_0 \setminus I$. Let $\mathcal{B}_I^j$ denote the bids marginal on goods in I. All bids $\boldsymbol{b} \in \mathcal{B}_I^j$ contribute an item of a good in I to bidder j's final allocation bundle, while all other bids contribute an item of a good not in I.

If the demand cluster (I, j) has no link goods, none of the bids by bidders other than j are marginal on I, so all items of goods $i \in I$ in the residual supply must be allocated to j. Hence we reduce our allocation problem by transferring r_i items from the residual supply $\boldsymbol{r}$ to bidder j's partial allocation bundle $\boldsymbol{m}^j$ for each $i \in I$ and removing all bids marginal on items in I from $\mathcal{B}^j$.

If the demand cluster (I, j) has a single link good i^*, all items of goods in $I \setminus \{i^*\}$ must be allocated to j by the same argument as above. Secondly, the bids in $\mathcal{B}_I^j$ must be allocated a total of $\sum_{\boldsymbol{b}\in\mathcal{B}_I^j} w(\boldsymbol{b})$ units of goods from I. Hence the difference $\sum_{\boldsymbol{b}\in\mathcal{B}_I^j} w(\boldsymbol{b}) - \sum_{i\in I\setminus\{i^*\}} r_i$ gives us the number of items of i^* that must be allocated to bidder j. This yields Algorithm 4.

Algorithm 4 UnambiguousMarginals (process all unambiguous marginal bids)

Input: Allocation problem $\mathcal{A} = [\boldsymbol{p}, (\mathcal{B}^j)_{j\in J}, (\boldsymbol{m}^j)_{j\in J}, \boldsymbol{r}]$ and demand cluster (I, j) with at most one link good i^*.
Output: Valid reduction $\mathcal{A}'$ of $\mathcal{A}$ with no bids marginal on goods in I.
if (I, j) has no link goods **then**
    for all $i \in I$ **do**
        Increment m_i^j by r_i and set r_i to 0.
else
    Compute $d = \sum_{\boldsymbol{b}\in\mathcal{B}_I^j} w(\boldsymbol{b}) - \sum_{i\in I\setminus\{i^*\}} r_i$.
    Increment $m_{i^*}^j$ by d and decrement r_{i^*} by d.
    for all $i \in I \setminus \{i^*\}$ **do**
        increment m_i^j by r_i and set r_i to 0.
Remove all bids marginal on I from $\mathcal{B}^j$.

Lemma 12. UnambiguousMarginals*, given in Algorithm 4, returns a valid reduction of the input allocation problem in time* $O(n|\mathcal{B}^j|)$*. Moreover, the marginal bids graph* $G_{\mathcal{A}'}$ *of* $\mathcal{A}'$ *has strictly fewer edges than* $G_{\mathcal{A}}$*.*

Proof. In order to see that $\mathcal{A}'$ is a valid allocation problem, we verify that the new bid list $(\mathcal{B}^j)'$ of bidder j after applying UnambiguousMarginals is locally valid by checking criterion (2) of Definition 3. Fix a price $\boldsymbol{p}$ and goods i, i'. Recall that bids in $\mathcal{B}^j$ cannot be marginal on goods in I and $[n]_0 \setminus I$, by the definition of key lists. If $i \in I$ and $i' \in [n]_0 \setminus I$, then $\mathcal{B}^j$ has no bids marginal on i and i'. If both i, i' are goods in I, all bids marginal on i and i' are removed by UnambiguousMarginals. If neither i nor i' is a good in I, then none of the bids marginal on i and i' are removed. As $\mathcal{B}^j$ is locally valid by assumption, this implies that $(\mathcal{B}^j)'$ is also locally valid. It is straightforward to see that a solution to $\mathcal{A}'$ is a solution to $\mathcal{A}$, as the partial allocation performed by UnambiguousMarginals is unambiguous. Finally, let $i, i' \in I$ and note that the marginal bids graph $G_{\mathcal{A}}$ contains an edge between i and i' that is not present in $G_{\mathcal{A}'}$. As removing bids does not introduce new edges to the graph, $G_{\mathcal{A}'}$ has strictly fewer edges. □

4.4 The shift-project-unshift reduction.

Suppose that FindParams returns a cycle-link good i^* and the label j^* of one of its adjacent edges in the cycle. In this case Allocate invokes the ShiftProjectUnshift procedure to obtain a reduction. Unlike NonMarginals and UnambiguousMarginals, this procedure does not make progress by allocating items and deleting the satisfied bids. Instead, it temporarily shifts the bids of bidder j^* and perturbs the market-clearing price $\boldsymbol{p}$ using SFM. We show that the marginal bids graph for this new allocation problem has strictly fewer edges, reducing the overall dependencies between the demands of bids. Finally, the procedure projects the bids of bidder j^* in such a way that it can unperturb the market-clearing price while retaining the dependency structure of the new marginal bids graph. We now introduce the shift and project operations.

Shifting bids. Suppose all bids are integral and we shift some bidder's bids by a small quantity $\varepsilon < 1/4$ (we use $\varepsilon = 1/10$ for concreteness) in the direction of i by adding $\varepsilon \boldsymbol{e}^i$ to each bid vector. Then Lemma 13 and Proposition 14 (proved in Appendix D.3) together show that we can use SFM to find a price $\boldsymbol{p}^\varepsilon \in \{\boldsymbol{p} + \varepsilon \boldsymbol{e}^S, S \subseteq [n]\}$, at which the new allocation problem $\mathcal{A}'$ with shifted bids and price vector $\boldsymbol{p}^\varepsilon$ is a valid reduction.

Lemma 13. *Fix ε with $|\varepsilon| < 1/4$, as well as $i \in [n]$ and $j \in J$. Let $\mathcal{A}$ be an allocation problem with integral bids and prices, and let $\mathcal{A}'$ be obtained by replacing $\mathcal{B}^j$ with $(\mathcal{B}^j)' \coloneqq \{\boldsymbol{b} + \varepsilon \boldsymbol{e}^i \mid \boldsymbol{b} \in \mathcal{B}^j\}$. Then the bid lists of all bidders in $\mathcal{A}'$ are locally valid at any price $\boldsymbol{p}^\varepsilon \in B(\boldsymbol{p}, \varepsilon)$. Moreover, the bundles demanded by any bidder at $\boldsymbol{p}^\varepsilon$ in $\mathcal{A}'$ form a subset of the bundles they demand at $\boldsymbol{p}$ in $\mathcal{A}$.*

Proof. As the bids and prices in $\mathcal{A}$ are integral, each bid either demands all goods $[n]_0$ or has a surplus gap of at least 1 at $\boldsymbol{p}$. By Lemma 9, all bid lists are $1/2$-valid at $\boldsymbol{p}$. This implies that every unshifted bid list is $(1/2 - \varepsilon)$-valid at $\boldsymbol{p}^\varepsilon$ and the shifted bid list is $(1/2 - 2\varepsilon)$-valid at $\boldsymbol{p}^\varepsilon$. By Observations 7 and 8, a non-marginal bid demands the same good for all $\boldsymbol{q} \in B(\boldsymbol{p}^\varepsilon, \delta)$ and sufficiently small $\delta > 0$. Hence a bidder will demand the same set of bundles at all non-marginal prices in $B(\boldsymbol{p}^\varepsilon, \delta)$. As the bundles a bidder demands at $\boldsymbol{p}^\varepsilon$ are by definition the discrete convex hull of bundles they demand at non-marginal prices infinitesimally close to $\boldsymbol{p}^\varepsilon$, we are done. □

Proposition 14. *Let $\mathcal{A}$ be an allocation problem with price vector $\boldsymbol{p}$ and fix ε with $|\varepsilon| < 1/4$. If we shift bidder j's bids by ε, there exists a price $\boldsymbol{p}^\varepsilon \in \{\boldsymbol{p} \pm \varepsilon \boldsymbol{e}^S, S \subseteq [n]\}$, at which the residual supply $\boldsymbol{r}$ is demanded. We can determine $\boldsymbol{p}^\varepsilon$ using submodular minimisation of the Lyapunov function g with regard to the aggregate bid list and the residual supply $\boldsymbol{r}$.*

Projecting bids. Next we define a projection operation on bids. The idea of this operation is to modify a bid $\boldsymbol{b}$ so that its surplus gap at $\boldsymbol{p}$ is increased. As a consequence, if that bid or the price is perturbed slightly, it 'remembers' its preferred goods. Let $\boldsymbol{b}$ be a bid and $I = \arg\max_{i\in[n]_0}(b_i - p_i)$ be the set of demanded goods at price $\boldsymbol{p}$. The projection $\boldsymbol{b}'$ of $\boldsymbol{b}$ w.r.t. $\boldsymbol{p}$ is defined as follows.

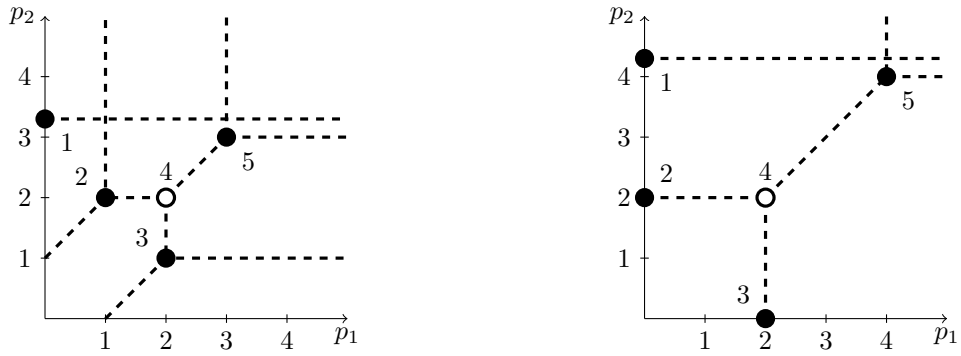
$$\boldsymbol{b}' \coloneqq \begin{cases} \boldsymbol{b} - \boldsymbol{e}^{[n]_0 \setminus I} & \text{if } 0 \in I, \\ \boldsymbol{b} + \boldsymbol{e}^{I} & \text{otherwise.} \end{cases}$$

Note that we allow for bid vector entries to be negative. Figure 6 illustrates the projection operation.

Observation 15. *If bid $\boldsymbol{b}$ demands all goods $[n]_0$, we have $\boldsymbol{b}' = \boldsymbol{b}$. Otherwise, the projection operation on $\boldsymbol{b}$ w.r.t. $\boldsymbol{p}$ increases the bid's surplus gap at $\boldsymbol{p}$ by 1.*

Lemma 16. *Projecting all bids in a bidder's bid list $\mathcal{B}$ w.r.t. $\boldsymbol{p}$ does not change the set of bundles demanded at $\boldsymbol{p}$. Moreover, if $\mathcal{B}$ is locally valid at $\boldsymbol{p}$, the projected bid list is locally $1/2$-valid.*

Proof. Suppose all bids in $\mathcal{B}$ have a surplus gap of at least ε at $\boldsymbol{p}$ and $\mathcal{B}$ is locally δ-valid at $\boldsymbol{p}$. Note that at any price $\boldsymbol{q}$ in the open ball $B(\boldsymbol{p}, \varepsilon)$ centred at $\boldsymbol{p}$, every bid demands the same set of goods before and after projecting. Hence the projected bid list is locally $\min\{\delta, \varepsilon\}$-valid at $\boldsymbol{p}$, and the set of bundles demanded at $\boldsymbol{p}$, consisting of the discrete convex hull of bundles demanded in UDRs bordering $\boldsymbol{p}$ remains the same. The second statement follows from Observations 9 and 15. □



(a) Five numbered bids before projecting.

(b) The result of projecting w.r.t. prices $\boldsymbol{p} = (4, 4)$.

Figure 6: Example of the project operation w.r.t. prices $\boldsymbol{p} = (4, 4)$ on five bids numbered from 1 to 5 shown in (a). The projected bids are shown in (b). Note that only the negative bid is unchanged, as it demands all goods, including the reject good.

The procedure. The procedure ShiftProjectUnshift is stated in Algorithm 5. It shifts and projects to reduce the number of edges in marginal goods graph. This effectively reduces the number of demand ties among the bids in our allocation problem. Lemmas 17 (proved in Appendix D.4) and 18 establish that the reduction obtained by ShiftProjectUnshift is valid and makes progress.

Lemma 17. *The marginal bids graph $G_{\mathcal{A}'}$ of $\mathcal{A}'$ has strictly fewer edges than $G_{\mathcal{A}}$.*

Lemma 18. ShiftProjectUnshift*, given in Algorithm 5, returns a reduction $\mathcal{A}'$ of $\mathcal{A}$ in time $O(T(n) + n|\mathcal{B}|)$, where $T(n)$ is the time it takes to minimise an n-dimensional submodular function.*

Algorithm 5 ShiftProjectUnshift

1: **Input:** Allocation problem $\mathcal{A} = [\boldsymbol{p}, (\mathcal{B}^j)_{j\in J}, (\boldsymbol{m}^j)_{j\in J}, \boldsymbol{r}]$, cycle-link good i^* and edge label j^*.
2: **Output:** Allocation problem $\mathcal{A}'$.
3: Add $\frac{1}{10}\boldsymbol{e}^{i^*}$ to each of j^*'s bids and compute a new market-clearing price $\boldsymbol{p}^\varepsilon = \boldsymbol{p} + \frac{1}{10}\boldsymbol{e}^{S^*}$ by solving $S^* = \arg\min_{S\subseteq[n]} g_{\boldsymbol{r}}(\boldsymbol{p} \pm \frac{1}{10}\boldsymbol{e}^S)$ using SFM.
4: Project every bid set w.r.t. $\boldsymbol{p}^\varepsilon$.
5: Subtract $\frac{1}{10}\boldsymbol{e}^{i^*}$ from each of j^*'s bids and reset price to $\boldsymbol{p}$.

Proof. Let $\mathcal{A}'$ and $\mathcal{A}''$ denote the allocation problems after executing lines 3 and 4, respectively. First we show that $\mathcal{A}'$ is a reduction of $\mathcal{A}$. Indeed, by Proposition 14 and Step 1, the residual bundle $\boldsymbol{r}$ is aggregately demanded at $\boldsymbol{p}^\varepsilon$ in $\mathcal{A}'$. Furthermore, for any allocation $(\boldsymbol{r}^j)_{j\in J} = (\boldsymbol{t}^j - \boldsymbol{m}^j)_{j\in J}$ of the residual bundle $\boldsymbol{r}$ at price $\boldsymbol{p}^\varepsilon$ in $\mathcal{A}'$, Lemma 13 implies that $(\boldsymbol{m}^j + \boldsymbol{r}^j)_{j\in J}$ is a solution of $\mathcal{A}$. Secondly, Lemma 16 implies that $\mathcal{A}''$ is a reduction of $\mathcal{A}'$. Finally, after line 5, the bid lists are locally valid at $\boldsymbol{p}$ by Lemma 13. Proposition 14 implies that the residual bundle $\boldsymbol{r}$ is demanded at some price $\boldsymbol{p}' \in \{\boldsymbol{p}^\varepsilon \pm \varepsilon\boldsymbol{e}^S \mid S \subseteq [n]\}$. Note that if a bundle is demanded at $\boldsymbol{p}'$, it is also demanded at $[\boldsymbol{p}']$ due to the structure of our integral bid vectors, where $[\cdot]$ denotes the operation of rounding each component to the nearest integer. As $[\boldsymbol{p}'] = \boldsymbol{p}$, the result follows. □

4.5 The main algorithm.

The algorithm Allocate, stated in Algorithm 1 above, combines the procedures NonMarginals, UnambiguousMarginals and ShiftProjectUnshift in order to solve the allocation problem described in Section 2.4. Recall that it uses FindParams as a subroutine to decide whether to call UnambiguousMarginals or ShiftProjectUnshift in each iteration of the loop. Theorem 19 proves correctness and gives a running time bound for Allocate. We note that this bound is likely to be pessimistic.

Theorem 19. Allocate *solves the allocation problem in time* $O\left(n^2|J|(\alpha(n)n|\mathcal{B}| + T(n))\right)$, *where* $T(n)$ *is the time required to minimise an* n*-dimensional submodular set function.*

Proof. By construction, the marginal bids graph of the initial allocation problem has at most $|J|\binom{n+1}{2}$ edges. Every call to UnambiguousMarginals or ShiftProjectUnshift strictly reduces the number of edges (by Lemmas 12 and 17). Hence after at most $|J|\binom{n+1}{2}$ iterations of Step 2, the marginal bids graph of the current allocation problem is an empty graph, implying that there are no more marginal bids. In particular, at this point a single call to NonMarginals allocates all remaining non-marginal bids and returns a vacuous allocation problem. As all three procedures NonMarginals, UnambiguousMarginals and ShiftProjectUnshift return a reduction in the sense of Definition 5, the solution to the final vacuous allocation problem is also a solution to the original allocation problem. To see the running time guarantee, note that FindParams and the procedures NonMarginals, UnambiguousMarginals and ShiftProjectUnshift are each called at most $|J|\binom{n+1}{2} = O(n^2|J|)$ times, and FindParams dominates NonMarginals and UnambiguousMarginals. □

Incorporating priorities. The FindParams subroutine as stated in Section 4.2 is not fully specified, and different implementations may lead to different inputs for UnambiguousMarginals or ShiftProjectUnshift when given the same allocation problem. If the input for ShiftProjectUnshift depends on the implementation used, ties may be broken differently and thus the target bundle is allocated differently among the bidders.

In order to control which input for ShiftProjectUnshift is returned, we propose the use of a priority list consisting of a permutation of all good-bidder pairs $(i, j) \in [n]_0 \times J$. The priority

list can be chosen to favor certain types of bidder; for example, if it is considered desirable to bring 'small' bidders (having low demand) into the market, we can prioritise their bids, making it slightly more likely that they will be allocated. Moreover, the list may be given as an additional input parameter at the start of Allocate or be generated and updated dynamically as Allocate runs. The subroutine FindParams is replaced by PriorityParams, which returns the highest pair (i, j) in the priority list that constitutes a valid input to ShiftProjectUnshift, or an input to UnambiguousMarginals if no such pair exists. Recall that a pair (i, j) is a valid input for ShiftProjectUnshift if i is a cycle-link good in some multi-bidder cycle and j is the label of one of its edges in this cycle. This is the case if the derived graph $D_{\mathcal{A}}$ has a cycle containing the edge from i to the demand cluster (I, j) satisfying $i \in I$. For any (undirected) graph G and edge vw, one can check whether G has a cycle containing vw by determining whether its endpoints v and w are connected in the graph $G - vw$ obtained by deleting vw, using breadth first search (BFS). This implies the following subroutine.

Algorithm 6 PriorityParams

1: Compute the derived graph $D_{\mathcal{A}}$.
2: **for all** pairs (i, j) in the priority list **do**
3: **if** i is a link good and $i \in I$ for some demand cluster (I, j) **then**
4: Check whether the edge from i to (I, j) lies in a cycle by temporarily removing the edge and verifying (using BFS) whether the two endpoints are still connected in the graph.
5: **return** (i, j) if the edge lies in a cycle.
6: **return** some leaf demand cluster (I, j) and the adjacent link good i.

To see that PriorityParams is well-defined, note that the priority list is a permutation of all possible pairs (i, j). Hence if the derived graph contains a cycle, the subroutine will return it, otherwise there exist at least two leaves for Step 2 to choose from. In practice, we can prune the priority list dynamically by removing pairs (i, j) once i is no longer a link good. This is possible because UnambiguousMarginals and ShiftProjectUnshift do not add edges to the derived graph, and so a good cannot become a link good again at some later point.

5 Conclusions and further work.

This paper has provided a practical process for running auctions in which bidders can express strong substitutes preferences using positive and negative bids. Previous work has shown [Baldwin and Klemperer, 2021] that *all* strong substitutes preferences can be represented using appropriate combinations of these bids. Although the coNP-completeness result we showed in Section 2.3 indicates that, at least in high dimensions, some restrictions may be needed on the allowed sets of bids, useful sub-classes of collections of bids are easily checkable for validity in practice, and can be checked offline, bidder by bidder, prior to the auction.

Our use of the product-mix auction's bidding language (by contrast with previous authors' usage of an abstract oracle) facilitates computing allocations as well as prices. One question for future research is whether we can also exploit the information provided by the bidding language to improve the computational efficiency of the pre-existing submodular function minimisation subroutine that both our price-finding and allocation algorithms use. Another obvious question is the extent to which our methods can be extended to broader classes of valuations.

Acknowledgments.

Baldwin and Klemperer were supported by ESRC Grant ES/L003058/1. We are grateful for reviewer feedback.

References

L. M. Ausubel. An efficient dynamic auction for heterogeneous commodities. *American Economic Review*, 96(3):602–629, June 2006.

E. Baldwin and P. Klemperer. Understanding preferences: "demand types", and the existence of equilibrium with indivisibilities. *Econometrica*, 87(3):867–932, May 2019.

E. Baldwin and P. Klemperer. Proof that the strong substitutes product-mix auction bidding language can represent any strong substitutes preferences. preprint, 2021. URL `http://elizabeth-baldwin.me.uk/papers/strongsubsproof.pdf`.

E. Baldwin, P. W. Goldberg, and P. Klemperer. Tropical intersections and equilibrium (day 2 slides). Hausdorff School on Tropical Geometry and Economics, 2016. URL `http://people.math.gatech.edu/~jyu67/HCM/Baldwin2.pdf`.

D. Chakrabarty, P. Jain, and P. Kothari. Provable submodular minimization using Wolfe's algorithm. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 802–809. Curran Associates, Inc., Cambridge, MA, USA, 2014.

D. Chakrabarty, Y. T. Lee, A. Sidford, and S. C.-W. Wong. Subquadratic submodular function minimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 1220–1231, New York, NY, USA, 2017. ACM.

V. Danilov, G. Koshevoy, and K. Murota. Discrete convexity and equilibria in economies with indivisible goods and money. *Mathematical Social Sciences*, 41(3):251 – 273, 2001.

P. W. Goldberg, E. Lock, and F. Marmolejo-Cossío. Learning strong substitutes demand via queries. In X. Chen, N. Gravin, M. Hoefer, and R. Mehta, editors, *Procs. of the 16th Conference on Web and Internet Economics*, pages 401–415. Springer, 2020.

F. Gul and E. Stacchetti. The English auction with differentiated commodities. *Journal of Economic theory*, 92(1):66–95, 2000.

A. S. Kelso and V. P. Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica*, 50:1483–1504, 1982.

P. Klemperer. A new auction for substitutes: Central bank liquidity auctions, the U.S. TARP, and variable product-mix auctions. working paper, Nuffield College, 2008. URL `https://www.nuffield.ox.ac.uk/economics/Papers/2008/substsauc.pdf`.

P. Klemperer. The product-mix auction: A new auction design for differentiated goods. *Journal of the European Economic Association*, 8(2–3):526–536, 2010.

P. Klemperer. Product-mix auctions. working paper 2018-W07, Nuffield College, 2018. URL `https://www.nuffield.ox.ac.uk/economics/Papers/2018/2018W07productmix.pdf`.

Y. T. Lee, A. Sidford, and S. C.-W. Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 1049–1065, Washington, DC, USA, 2015. IEEE Computer Society. doi: 10.1109/FOCS.2015.68. URL `http://dx.doi.org/10.1109/FOCS.2015.68`.

A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*, volume 1. Oxford University Press, New York, 1995.

P. Milgrom and B. Strulovici. Substitute goods, auctions, and equilibrium. *Journal of Economic Theory*, 144(1):212–247, 2009.

K. Murota. *Discrete Convex Analysis*, volume 10. SIAM Monographs on Discrete Mathematics and Applications, 2003.

K. Murota. Discrete convex analysis: A tool for economics and game theory. *Journal of Mechanism and Institution Design*, 1(1):151–273, 2016.

K. Murota and A. Shioura. M-convex function on generalized polymatroid. *Mathematics of Operations Research*, 24(1):95–105, 1999.

K. Murota and A. Shioura. Exact bounds for steepest descent algorithms of L-convex function minimization. *Operations Research Letters*, 42(5):361–366, 2014.

K. Murota, A. Shioura, and Z. Yang. Computing a walrasian equilibrium in iterative auctions with multiple differentiated items. In L. Cai, S.-W. Cheng, and T.-W. Lam, editors, *Algorithms and Computation*, pages 468–478, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-45030-3.

K. Murota, A. Shioura, and Z. Yang. Time bounds for iterative auctions: A unified approach by discrete convex analysis. *Discrete Optimization*, 19:36–62, 2016. doi: 10.1016/j.disopt.2016.01.001.

R. Paes Leme. Gross substitutability: An algorithmic survey. *Games and Economic Behavior*, 106:294–316, 2017.

R. Paes Leme and S. C.-W. Wong. Computing Walrasian equilibria: Fast algorithms and structural properties. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 632–651, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics.

A. Schrijver. *Theory of Linear and Integer Programming*. Interscience series in discrete mathematics and optimisation. John Wiley & Sons, 1998.

A. Shioura. Algorithms for L-convex function minimization: Connection between discrete convex analysis and other research fields. *Journal of the Operations Research Society of Japan*, 60(3):216–243, 2017.

A. Shioura and A. Tamura. Gross substitutes condition and discrete concavity for multi-unit valuations: a survey. *Journal of the Operations Research Society of Japan*, 58(1):61–103, 2015.

R. E. Tarjan and J. van Leeuwen. Worst-case analysis of set union algorithms. *Journal of the ACM*, 31(2):245–281, Mar. 1984. doi: 10.1145/62.2160.

S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, March 2011.

A Experiments.

In order to evaluate the practical running time of our allocation algorithm, we run experiments on various numbers of goods, bidders and bids. We use our own Python implementation of the product-mix auction, available at `https://github.com/edwinlock/product-mix`. Some effort was made to optimise for speed by exploiting fast matrix operations provided by the NumPy package [van der Walt et al., 2011]. Furthermore, in an effort to implement submodular minimisation efficiently, the Fujishige-Wolfe algorithm was implemented in combination with a memoization technique to reduce the number of submodular function queries.

A.1 Generating test data.

We describe a procedure to generate a valid list of positive and negative bids at points within the lattice $[M]^n$ and a bundle $\boldsymbol{x}$ that is demanded in aggregate by these bids at prices $\boldsymbol{p} = \frac{1}{2}M\boldsymbol{e}^{[n]}$ in the centre of the lattice. In our experiments, we fix $M = 100$. The bids are generated in such a way that for parameter q at least q bids are marginal between two or more goods at $\boldsymbol{p}$ and we pick $\boldsymbol{x}$ such that $\boldsymbol{p}$ is the component-wise minimal market-clearing price vector. For any permutation π of $[n]$, let $\boldsymbol{p}_\pi = \boldsymbol{p} + \sum_{i\in[n]} \frac{\varepsilon}{2^i}\boldsymbol{e}^{\pi(i)}$ for some $\varepsilon < 0.1$. Note that a unique bundle is demanded at $\boldsymbol{p}_\pi$ for any permutation π.

Algorithm 7 GenerateList(n,M,q)

Initialise: Empty bid list $\boldsymbol{b}$ and bundle $\boldsymbol{x} = \boldsymbol{0}$.
Repeat the following q times:
Pick a subset S of goods with $|S| \geq 2$ from $[n]_0$ and flip a fair coin.
if the coin lands on heads **then**
    Pick a positive bid that is marginal on goods S at $\boldsymbol{p}$ and add it to $\mathcal{B}$.
    Pick any good $i \in S$ uniformly at random and increment x_i by one.
else
    Generate a negative bid $\boldsymbol{b}$ that is marginal on goods S at $\boldsymbol{p}$, as well as the following three positive bids.
    Pick two goods $i, j \in [n]$ and add a bid at points $\boldsymbol{b} - \lambda_i\boldsymbol{e}^i$ and $\boldsymbol{b} - \lambda_j\boldsymbol{e}^j$ for some $1 \leq \lambda_i \leq b_i - 1$ and $1 \leq \lambda_j \leq b_j - 1$.
    Add a bid at $\boldsymbol{b} + \lambda\boldsymbol{1}$ for some $1 \leq \lambda \leq \min_{i\in[n]} M - b_i$.
    Pick a permutation π of $[n]$ and increment $\boldsymbol{x}$ by the bundle aggregately demanded at $\boldsymbol{p}_\pi$ by the four bids just generated.
if $\boldsymbol{p}$ is the component-wise minimal market-clearing price vector of $\boldsymbol{x}$ **then**
    return $\mathcal{B}$.
else
    Repeat the algorithm.

Generated in this way, every bid list has $2.5q$ bids in expectation. The procedure GenerateList is repeated for each bidder, so the total number of bids generated is $B = 2.5qm$.

Lemma 20. *The bid list generated by* GenerateList *is valid.*

Proof. Note that the union of finitely many valid bid lists is again a valid bid list. Hence it suffices to show that the list consisting of one negative bid $\boldsymbol{b}$ and three positive bids $\boldsymbol{b} - \lambda_i\boldsymbol{e}^i, \boldsymbol{b} - \lambda_j\boldsymbol{e}^j$ and $\boldsymbol{b} + \lambda\boldsymbol{1}$ is valid. To see this, apply Theorem 19 from the main paper. □

A.2 Testing the algorithm.

We run Allocate on allocation problems with bid lists generated by GenerateList for different numbers of goods n, bidders m and bids B. It should be noted that, as expected, the value of M has no discernable impact on the running time of the allocation algorithm and thus can be fixed to $M = 100$. We perform three pairs of experiments in which we vary one of the parameters n, m, q and fix the other two to realistic values.

1. For the first pair of experiments, we vary the average number of bids by running GenerateList with values $q = 20, 40, 60, \ldots, 500$. We fix the number of bidders to $m = 5$ and the number of goods to $n = 2$ and $n = 10$, respectively.

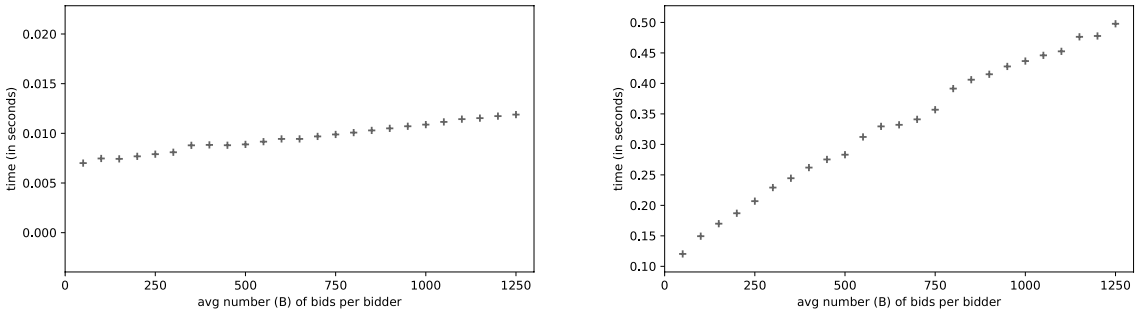
2. For the second pair of experiments, we vary the number of goods from $n = 10$ to 50 in steps of 5 and fix q to 50 and 100, respectively.

3. Finally, the last pair of experiments varies the number of bidders from $m = 2$ to $m = 20$ in steps of 1 with two goods $n = 2$, while the bid numbers are fixed by setting q to 50 and 100, respectively.

For each data point (n, m, q), 50 allocation problems with n goods, m bidders and $2.5q$ bids per bidder (in expectation) are generated. The ALLOCATE algorithm is then timed on each allocation problem and the average over all 50 times is recorded.

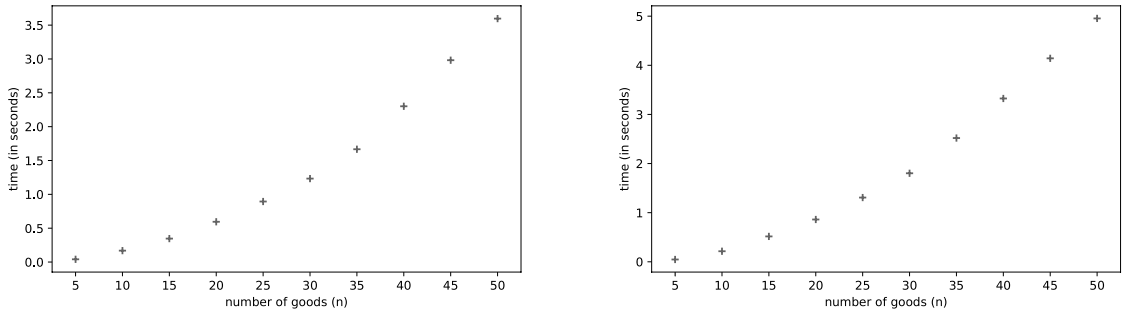
A.3 Results.

The outcomes of the three pairs of experiments are shown in Figures 7 to 9. The experimental data corroborates the running time bounds for ALLOCATE given in Theorem 9: the algorithms is linear in the number of bids and quadratic in the number of goods. Figure 9 suggests that our algorithm runs in quadratic time on our generated allocation problems, which is in line with our theoretical bound, as the total number of bids $B = 2.5qm$ is linear in m. Overall, we see that our allocation algorithm runs quickly even when presented with a large number of bids.



(a) Parameters: $n = 2, m = 5, M = 100$. **(b)** Parameters: $n = 10, m = 5, M = 100$.

Figure 7: Testing ALLOCATE by varying the number B of bids for each bidder while keeping all other parameters fixed. We increase q from 20 to 500 in steps of 20, fix the number of bidders at $m = 5$ and set the number of goods to $n = 2$ (a) and $n = 10$ (b), respectively.

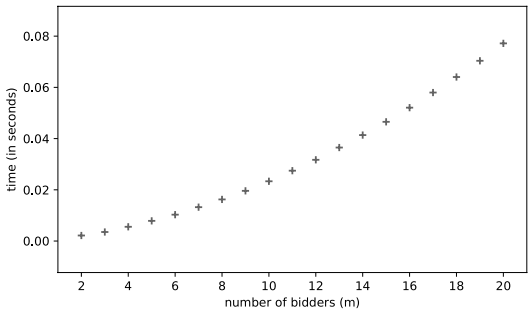


(a) Parameters: $m = 5, M = 100, q = 50$. **(b)** Parameters: $m = 5, M = 100, q = 100$.

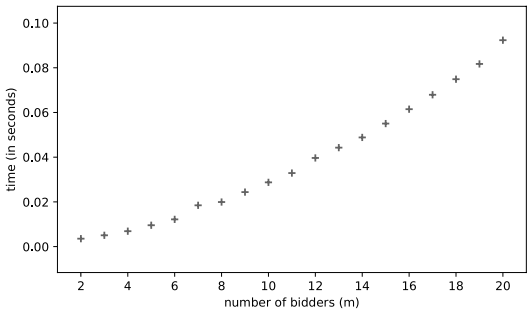
Figure 8: Testing ALLOCATE by varying the number n of goods. We increase n from 5 to 50 in steps of 5, fix the number of bidders at $m = 5$ and $M = 100$ and set the number of bids to $q = 50$ (a) and $q = 100$ (b), respectively.

B Finding market-clearing prices.

We discuss two iterative steepest descent algorithms, MINUP and LONGSTEPMINUP, from Shioura [2017] that determine a minimal minimiser of an $L^\natural$-convex function. Both algorithms



(a) Parameters: $n = 2, M = 100, q = 50$.



(b) Parameters: $n = 2, M = 100, q = 100$.

Figure 9: Testing ALLOCATE by varying the number m of bidders. We increase m from 2 to 20 in steps of 1, fix the number of goods at $n = 2$ and set q to 50 (a) and 100 (b), respectively.

use the SFM subroutine described in Section 2 to find the component-wise minimal discrete steepest descent direction. For the second algorithm, we present two methods of computing step lengths and show that both methods yield a polynomial running time in our bidding-language setting.

In order to apply the steepest descent method to our price-finding problem, we define a Lyapunov function g and note in Proposition 21 that its restriction to $\mathbb{Z}^n_+$ is $L^\natural$-convex. Moreover, Lemma 22 states that the lowest market-clearing price is integral and finding it reduces to determining the minimal minimiser of g. This approach generalises an algorithm used by Ausubel's ascending auction design [Ausubel, 2006] and Gul and Stacchetti [2000] for the task of finding equilibrium prices in single-unit markets.

Let $\mathcal{B}$ be a valid bid list. By Theorem 1, the function $f_\mathcal{B}$ defined in (3) is the indirect utility function f_u of some strong substitutes valuation u. For any strong substitutes valuation u, the *Lyapunov function* with regard to indirect utility function f_u and target bundle $\boldsymbol{t}$ is defined as $g_{\boldsymbol{t}}(\boldsymbol{p}) \coloneqq f_u(\boldsymbol{p}) + \boldsymbol{t} \cdot \boldsymbol{p}$. We suppress the subscript $\boldsymbol{t}$ if it is clear from context. We can use our knowledge of the bids in $\mathcal{B}$ and (3) to express $g_{\boldsymbol{t}}$ for the list $\mathcal{B}$ as

$$g_{\boldsymbol{t}}(\boldsymbol{p}) \coloneqq f_\mathcal{B}(\boldsymbol{p}) + \boldsymbol{t} \cdot \boldsymbol{p} = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \max_{i \in [n]_0} (b_i - p_i) + \boldsymbol{t} \cdot \boldsymbol{p}. \tag{6}$$

From (6) it is clear that we can evaluate g at any price $\boldsymbol{p}$ in time $O(n|\mathcal{B}|)$.

A function $f : \mathbb{Z}^n \to \mathbb{R}$ is $L^\natural$-convex if it satisfies the *translation submodularity* property,

$$f(\boldsymbol{p}) + f(\boldsymbol{q}) \geq f((\boldsymbol{p} - \alpha\mathbf{1}) \vee \boldsymbol{q}) + f(\boldsymbol{p} \wedge (\boldsymbol{q} + \alpha\mathbf{1})) \quad (\forall \boldsymbol{p}, \boldsymbol{q} \in \mathbb{Z}^n_+, \forall \alpha \in \mathbb{Z}_+). \tag{7}$$

Here, $\vee$ and $\wedge$ denote the component-wise maximum and minimum, respectively.

The following proposition and lemma demonstrate that we can use algorithms for minimising $L^\natural$-convex functions to find the minimal price $\boldsymbol{p}^*$ at which $\boldsymbol{t}$ is demanded.

Proposition 21 (Murota et al. [2013, 2016]). *The Lyapunov function $g_{\boldsymbol{t}}$ restricted to $\mathbb{Z}^n_+$ is $L^\natural$-convex.*

Proof. It is known that the indirect utility function f_u restricted to $\mathbb{Z}^n$ is $L^\natural$-convex if and only if the valuation function u is strong substitutes (cf. Shioura and Tamura [2015, Theorem 7.1]). Secondly, it is easy to verify that adding a linear term to an $L^\natural$-convex function preserves $L^\natural$-convexity. As the bid list $\mathcal{B}$ is valid, we have $f_\mathcal{B} = f_u$ for some strong substitutes valuation u and it follows that the function $f_\mathcal{B}(\boldsymbol{p}) + \boldsymbol{t} \cdot \boldsymbol{p}$ is $L^\natural$-convex. □

Lemma 22. *The Lyapunov function $g_{\boldsymbol{t}}$ with regard to a valid (integral) bid list and any target bundle $\boldsymbol{t}$ is convex, and $\boldsymbol{p}$ is a minimiser of g if and only if it is a market-clearing price for target bundle $\boldsymbol{t}$. Moreover, the minimal minimiser of g is integral.*

Proof. The first statement is immediate from the fact that $f_{\mathcal{B}}$ is convex and $\boldsymbol{t} \cdot \boldsymbol{p}$ is a linear term. To see the second statement, note that for any market-clearing price $\boldsymbol{p}$ of target bundle $\boldsymbol{t}$, we have $g(\boldsymbol{p}) = u(\boldsymbol{t})$, whereas for any price $\boldsymbol{p}$ at which $\boldsymbol{t}$ is not demanded, we have $g(\boldsymbol{p}) = \max_{\boldsymbol{x} \in D(\boldsymbol{p})}(u(\boldsymbol{p}) - \boldsymbol{x} \cdot \boldsymbol{p}) + \boldsymbol{t} \cdot \boldsymbol{p} > u(\boldsymbol{t})$. Here u is the valuation function as defined in Section 2 and we use (2).

Finally, the integrality of the minimal minimiser of g follows from the fact that if $\boldsymbol{t}$ is demanded at $\boldsymbol{p}$, then $\boldsymbol{t}$ is also demanded at $\lfloor \boldsymbol{p} \rfloor$. To see this, fix a bid $\boldsymbol{b}$ and note that if $\boldsymbol{b}$ demands good i at $\boldsymbol{p}$, it still demands i at $\lfloor \boldsymbol{p} \rfloor$. Indeed, as $\boldsymbol{b}$ demands i at $\boldsymbol{p}$, we have $b_j - p_j \leq b_i - p_i$ for all goods j, which implies

$$b_j - \lfloor p_j \rfloor < b_j - p_j + 1 \leq b_i - p_i + 1 \leq b_i - \lfloor p_i \rfloor + 1.$$

Due to the integrality of the bids and prices in $\lfloor p \rfloor$, this implies $b_j - \lfloor p_j \rfloor \leq b_i - \lfloor p_i \rfloor$. □

Let $\boldsymbol{p}$ be a point that is dominated by some minimiser of g. Shioura [2017] noted that $\boldsymbol{p}$ minimises g if and only if $g(\boldsymbol{p}) \leq g(\boldsymbol{p} + \boldsymbol{e}^S)$ for every $S \subseteq [n]$. For any integral point $\boldsymbol{p} \in \mathbb{Z}^n_+$ and $S \subseteq [n]$, let the *slope function* $g'(\boldsymbol{p}; S) \coloneqq g(\boldsymbol{p} + \boldsymbol{e}^S) - g(\boldsymbol{p})$ denote the amount by which $\boldsymbol{p}$ decreases when moving in the direction of $\boldsymbol{e}^S$. If S minimises $g'(\boldsymbol{p}; S)$, we call $\boldsymbol{e}^S$ a *steepest descent direction*. For any integral vector $\boldsymbol{p}$, the $L^\natural$-convexity of g implies that $g'(\boldsymbol{p}; S)$ is an integral submodular function [Shioura and Tamura, 2015, Theorem 7.2] and hence there exists a unique component-wise *minimal steepest descent direction* $\boldsymbol{e}^{S_0}$.

Let $\boldsymbol{p}^*$ be the minimal minimiser of g. If $\boldsymbol{p}$ is dominated by $\boldsymbol{p}^*$ and we move in the minimal steepest descent direction, the point $\boldsymbol{p} + \boldsymbol{e}^{S_0}$ is also dominated by $\boldsymbol{p}^*$. This suggests Algorithm 8 (called GreedyUpMinimal in Shioura [2017]), which iterates a point $\boldsymbol{p}$ by moving by some step $\boldsymbol{e}^{S_0}$, all the while remaining dominated by $\boldsymbol{p}^*$, until $\boldsymbol{p} = \boldsymbol{p}^*$.

Algorithm 8 MinUp

1: Pick a point $\boldsymbol{p} \leq \boldsymbol{p}^*$ (e.g. $\boldsymbol{p} = \boldsymbol{0}$).
2: Find the inclusion-wise minimal set $S_0 \in [n]$ minimising $g'(\boldsymbol{p}; S)$.
3: **if** $S_0 = \emptyset$ **then**
4: **return** $\boldsymbol{p}$.
5: **else**
6: Set $\boldsymbol{p} = \boldsymbol{p} + \boldsymbol{e}^{S_0}$ and go to line 2.

By the existence of the auctioneer's reserve bids, we can initialise $\boldsymbol{p}$ to $\boldsymbol{0}$. For this starting point, running time analysis by Murota and Shioura [2014] implies that 8 iterates exactly $\|\boldsymbol{p}^*\|_\infty$ times. As we know that $\boldsymbol{p}^*$ is bounded from above by the component-wise maximum over all bids $\mathcal{B}$, the number of iterations is at most $M \coloneqq \max_{\boldsymbol{b} \in \mathcal{B}} \|\boldsymbol{b}\|_\infty$. In each iteration of Step 2, we can determine S_0 using the SFM subroutine described in Section 2 that finds a minimal minimiser of a given function in time $T(n)$. This leads to the following running time for MinUp.

Theorem 23 (cf. Murota and Shioura [2014]). *The algorithm* MinUp *finds the component-wise minimal market-clearing price in time $O(MnT(n))$, where $T(n)$ denotes the time it takes to find a minimiser of g'.*

We note that the running times of the two practical SFM algorithms mentioned in Section 2 are given with respect to an upper bound on the absolute value of the objective function. In order to provide such an upper bound for our slope function g', observe that the two points $\boldsymbol{p}$ and $\boldsymbol{p} + \boldsymbol{e}^S$ share a demanded bundle $\boldsymbol{x}$, for any $\boldsymbol{p}$ and $S \subseteq [n]$. As every bid contributes at most one item to $\boldsymbol{t}$ and $\boldsymbol{x}$, this implies $g'(\boldsymbol{p}; S) = (\boldsymbol{t} - \boldsymbol{x}) \cdot \boldsymbol{e}^S \leq |\mathcal{B}|$.

B.1 Longer step sizes.

The analysis by Murota and Shioura [2014] shows that MinUp performs optimally for an iterative algorithm that is constrained to steps with an L_∞-size of at most 1. As described by Shioura [2017], we can, however, exploit monotonicity properties of the function $g'(\boldsymbol{p}; S_0)$ in order to increase step sizes without changing the trajectory of $\boldsymbol{p}$ as the algorithm runs. This reduces the number of SFM subroutine calls, the most expensive part of the algorithm. In particular, if $\boldsymbol{e}^{S_0}$ denotes the minimal steepest descent direction at $\boldsymbol{p}$, we take a single long step $\lambda \boldsymbol{e}^{S_0}$ for some $\lambda \in \mathbb{Z}_+$ that is equivalent to several consecutive steps of MinUp in the same direction $\boldsymbol{e}^{S_0}$.

We follow Shioura [2017] in choosing step length

$$\lambda(\boldsymbol{p}, S_0) = \max\{\lambda \in \mathbb{Z}_+ \mid g'(\boldsymbol{p}; S_0) = g'(p + (\lambda - 1)\boldsymbol{e}^{S_0}; S_0)\}, \tag{8}$$

that is, the farthest distance we can move before the slope

$$g'(\boldsymbol{p} + (\lambda - 1)\boldsymbol{e}^{S_0}; S_0) = g(\boldsymbol{p} + \lambda \boldsymbol{e}^{S_0}) - g(\boldsymbol{p} + (\lambda - 1)\boldsymbol{e}^{S_0})$$

in the direction of $\boldsymbol{e}^{S_0}$ changes. We give two methods to compute (8) in Section B.1.1. This leads to the following algorithm (referred to as GreedyUp-LS in Shioura [2017]).

Algorithm 9 LongStepMinUp

1: Pick a point $\boldsymbol{p} \leq \boldsymbol{p}^*$ (e.g. $\boldsymbol{p} = \boldsymbol{0}$).
2: Compute inclusion-wise minimal minimiser $S_0 \subseteq [n]$ of $g'(\boldsymbol{p}; S_0)$ using SFM.
3: Determine $\lambda(\boldsymbol{p}, S_0)$, as defined by (8), using a method from Section B.1.1.
4: **if** $S_0 \neq \emptyset$ **then**
5: Set $\boldsymbol{p} = \boldsymbol{p} + \boldsymbol{e}^{S_0}$ and go to line 2.
6: **return** $\boldsymbol{p}$.

Note that the values of $\boldsymbol{p}$ follow the same trajectory for MinUp and LongStepMinUp. This is an immediate consequence of Lemma 25, which rests on the monotonicity properties of $g'(\boldsymbol{p}, S_0)$ stated in Proposition 24.

Proposition 24 (Shioura [2017], Theorem 4.16). *Let S_0 and S_0' be* minimal *steepest descent directions at $\boldsymbol{p}$ and $\boldsymbol{p} + \boldsymbol{e}^{S_0}$, respectively. Then we have*

1. *$g'(\boldsymbol{p} + \boldsymbol{e}^{S_0}; S_0') > g'(\boldsymbol{p}; S_0)$ or*
2. *$g'(\boldsymbol{p} + \boldsymbol{e}^{S_0}; S_0') = g'(\boldsymbol{p}; S_0)$ and $S_0 \subseteq S_0'$.*

Lemma 25. *Let S_0 denote the minimal steepest descent at $\boldsymbol{p}$ and let $\lambda(\boldsymbol{p}, S_0)$ be defined by (8). Then $\boldsymbol{e}^{S_0}$ is the minimal steepest descent at $\boldsymbol{p} + (\lambda - 1)\boldsymbol{e}^{S_0}$ for any $1 \leq \lambda \leq \lambda(\boldsymbol{p}, S_0)$.*

Shioura [2017] bounds the number of iterations of LongStepMinUp as follows.

Theorem 26 (Shioura [2017], Theorem 4.17). *The number of iterations of* LongStepMinUp *is at most $n \max\{-g'(\boldsymbol{0}; S) \mid S \subseteq [n]\}$.*

Note that $\boldsymbol{0}$ and $\boldsymbol{e}^{S_0}$ share a demanded bundle $\boldsymbol{x}$, so for any $S \subseteq [n]$ we have

$$g'(\boldsymbol{0}; S) = g(\boldsymbol{e}^S) - g(\boldsymbol{0}) = f_u(\boldsymbol{e}^S) + \boldsymbol{t} \cdot \boldsymbol{e}^S - f_u(\boldsymbol{0}) = (\boldsymbol{t} - \boldsymbol{x}) \cdot \boldsymbol{e}^S \geq -\sum_{i \in [n]} x_i \geq -|\mathcal{B}|,$$

as each bid contributes at most one unit to the demanded bundle. This implies that LongStepMinUp takes at most $n|\mathcal{B}|$ iterations to find component-wise minimal equilibrium prices.

B.1.1 Computing the step length.

Fix a price $\boldsymbol{p}$ and let $\boldsymbol{e}^{S_0}$ be the component-wise minimal steepest descent direction at $\boldsymbol{p}$. We describe two methods to compute the step length defined by (8). The first method uses binary search and is also suggested in Shioura [2017], while second method exploits our knowledge of the bids to determine $\lambda(\boldsymbol{p}; S_0)$. Note that we can evaluate $g'(\boldsymbol{p}; S_0)$ in time $O(n|\mathcal{B}|)$, as $g'(\boldsymbol{p}; S_0) = g(\boldsymbol{p} + \boldsymbol{e}^{S_0}) - g(\boldsymbol{p})$.

Theorem 27. *The* LongStepMinUp *algorithm in combination with the* binary search *and* demand change *methods has a respective running time of* $O(n^2|\mathcal{B}|^2 \log M + n|\mathcal{B}|T(n))$ *and* $O(n^2|\mathcal{B}|^3 + n|\mathcal{B}|T(n))$.

A description of the two methods, as well as a proof of this theorem, is provided below. Note that as the two methods have different running time guarantees, the best method in practice is context-specific.

Binary search. Note that $\lambda(\boldsymbol{p}; S_0)$ can be bounded by the total number of unit steps in MinUp, which in turn is bounded by M. By Proposition 24, we have that $g'(\boldsymbol{p}; S_0) < g'(\boldsymbol{p} + \lambda \boldsymbol{e}^{S_0}; S_0)$ implies $g'(\boldsymbol{p}; S_0) < g'(\boldsymbol{p} + \lambda' \boldsymbol{e}^{S_0}; S_0)$ for all $\lambda' > \lambda$. Hence we can apply binary search to find $\lambda(\boldsymbol{p}; S_0)$ in time $O(n|\mathcal{B}| \log M)$.

Demand change. Alternatively, we can exploit our knowledge of the individual bids to determine $\lambda(\boldsymbol{p}; S_0)$. We proceed by performing a demand-change procedure, which repeatedly determines the highest value μ for which every bid $\boldsymbol{b} \in \mathcal{B}$ demands the same goods (or a superset thereof) at prices $\boldsymbol{p} + \mu \boldsymbol{e}^{S_0}$ that it demands at $\boldsymbol{p} + \boldsymbol{e}^{S_0}$ and updates $\boldsymbol{p}$ to $\boldsymbol{p} + \mu \boldsymbol{e}^{S_0}$.

Fix a bid $\boldsymbol{b}$ and let I denote the goods it demands at $\boldsymbol{p}$. If $I \not\subseteq S_0$, then $\boldsymbol{b}$ demands the same set of goods $I_{\boldsymbol{b}} = I \setminus S_0$ at all prices $\boldsymbol{p} + \mu' \boldsymbol{e}^{S_0}$ with $\mu' \geq 1$. On the other hand, suppose $I \subseteq S_0$. We define $\mu_{\boldsymbol{b}} \coloneqq \min_{j \in [n]_0 \setminus S_0}((b_i - p_i) - (b_j - p_j))$, where i is any good in I, and consider the set of goods that $\boldsymbol{b}$ demands at prices $\boldsymbol{p} + \mu' \boldsymbol{e}^{S_0}$ with $\mu' \geq 1$. If $1 \leq \mu' < \mu_{\boldsymbol{b}}$, then $\boldsymbol{b}$ demands I, if $\mu' = \mu_{\boldsymbol{b}}$ then $\boldsymbol{b}$ demands a superset of I and if $\mu' > \mu_{\boldsymbol{b}}$, then $\boldsymbol{b}$ demands none of the goods in I. We define $\mathcal{C}$ to be the set of bids that demand a subset of S_0 at $\boldsymbol{p}$, and let $\mu(\boldsymbol{p}, S_0) \coloneqq \min_{\boldsymbol{b} \in \mathcal{C}} \mu_{\boldsymbol{b}}$. Note that we can determine the value of $\mu(\boldsymbol{p}, S_0)$ in time $O(n|\mathcal{B}|)$.

The demand-change procedure takes as input a price $\boldsymbol{p}^0 \coloneqq \boldsymbol{p}$ and direction $\boldsymbol{e}^{S_0}$, and consists of the following steps. Initially, we set λ to 0. Compute $\mu(\boldsymbol{p}, S_0)$, and increment λ by $\mu(\boldsymbol{p}, S_0)$. If we have $\lambda = \lambda(\boldsymbol{p}^0, S_0)$, return λ. Otherwise, increment $\boldsymbol{p}$ by $\mu \boldsymbol{e}^{S_0}$ and repeat the above with the same value for $\boldsymbol{e}^{S_0}$.

Note that in order to check whether $\lambda = \lambda(\boldsymbol{p}^0; S_0)$, we can compute $g'(\boldsymbol{p}^0 + \mu \boldsymbol{e}^{S_0}; S_0)$ and verify that $g'(\boldsymbol{p}^0 + \mu \boldsymbol{e}^{S_0}; S_0) = g'(\boldsymbol{p}^0; S_0)$, which takes time $O(n|\mathcal{B}|)$. Lemma 28 proves that the demand-change procedure correctly computes the value of $\lambda(\boldsymbol{p}; S_0)$ in time $O(n|\mathcal{B}|^2)$.

Lemma 28. *The demand-change procedure returns* $\lambda(\boldsymbol{p}; S_0)$ *in at most* $|\mathcal{B}|$ *iterations.*

Proof. Let K denote the number of iterations in the demand-change procedure and let $\boldsymbol{p}^k, \mu^k$ and C^k be the values of $\boldsymbol{p}, \mu(\boldsymbol{p}, S_0)$ and C after the k-th iteration. For notational convenience, let $\boldsymbol{p}^0 = \boldsymbol{p}$ and $\mu^0 = 0$. Proposition 24 implies that it suffices to show

$$g'(\boldsymbol{p}; S_0) = g'(\boldsymbol{p}^K - \boldsymbol{e}^S; S_0) \text{ and } g'(\boldsymbol{p}; S_0) < g'(\boldsymbol{p}^K; S_0) \tag{9}$$

in order to prove the first claim that $\lambda = \lambda(\boldsymbol{p}, S_0)$.

Firstly, note that there is a bundle $\boldsymbol{x}$ that is demanded at $\boldsymbol{p}^{k-1}$ and $\boldsymbol{p}^k$, as well as all prices in between these two points. Indeed, if a bid $\boldsymbol{b}$ demands good i at $\boldsymbol{p}^{k-1} + \boldsymbol{e}^{S_0}$, then $\boldsymbol{b}$ also demands i at the two prices $\boldsymbol{p}^{k-1}$ and $\boldsymbol{p}^k - \boldsymbol{e}^{S_0} = \boldsymbol{p}^{k-1} + (\mu^k - 1)\boldsymbol{e}^{S_0}$, by construction of μ^k. Hence, making use of (2), we get

$$g'(\boldsymbol{p}^{k-1}; S_0) = (\boldsymbol{t} - \boldsymbol{x}) \cdot \boldsymbol{e}^{S_0} = g'(\boldsymbol{p}^k - \boldsymbol{e}^{S_0}; S_0).$$

Secondly, note that for every $0 \leq k < K$, we have $g'(\boldsymbol{p}^k - \boldsymbol{e}^{S_0}; S_0) = g'(\boldsymbol{p}^k; S_0)$, and for the last iteration K we have $g'(\boldsymbol{p}^K - \boldsymbol{e}^{S_0}; S_0) < g'(\boldsymbol{p}^K; S_0)$. This implies (9).

Now we turn to the second claim, that $K \leq |\mathcal{B}|$. This follows from the fact that $C^{k-1} \supsetneq C^k$ and $|C^0| \leq |\mathcal{B}|$. Indeed, if $\boldsymbol{b} \notin C^{k-1}$, then $\boldsymbol{b}$ demands goods $I \not\subseteq S_0$ at $\boldsymbol{p}^{k-1}$ and $I_{\boldsymbol{b}} = I \setminus S_0$ at any prices $\boldsymbol{p}^{k-1} + \mu \boldsymbol{e}^{S_0}$ for any $\mu \geq 1$, so $b \notin C^k$. Now suppose $\boldsymbol{b} \in C^{k-1}$ is a bid for which $\mu^{k-1} = \mu_{\boldsymbol{b}}$ (at $\boldsymbol{p}^{k-1}$). Then we claim that $\boldsymbol{b} \notin C^k$. Indeed, the demanded goods I' of $\boldsymbol{b}$ at $\boldsymbol{p} + (\mu_{\boldsymbol{b}} + 1)\boldsymbol{e}^{S_0}$ satisfy $I' \not\subseteq S_0$ by construction of $\mu_{\boldsymbol{b}}$. By the same argument as above, $\boldsymbol{b}$ demands goods not in S_0 at all prices $\boldsymbol{p} + \mu \boldsymbol{e}^{S_0}$ with $\mu \geq \mu_{\boldsymbol{b}} + 1$. □

B.2 Some practical improvements.

The computation time of MinUp and LongStepMinUp is dominated by the task of finding a minimal set S_0 minimising $g'(\boldsymbol{p}; S)$ using SFM. In practice, we can exploit our direct access to the bids to speed up the computation of S_0 by decreasing the dimensionality of the submodular function to be minimised. The following observations can be seen as a special case of observations by Ausubel [2006]. Fix $\boldsymbol{p} \in \mathbb{Z}^n_+$ and let $\boldsymbol{e}^{S_0}$ be the minimal steepest descent direction at $\boldsymbol{p}$. Note that $\boldsymbol{p}$ and $\boldsymbol{p} + \boldsymbol{e}^{S_0}$ share a demanded bundle $\boldsymbol{x}$ due to the structure of our price space. Hence

$$g'(\boldsymbol{p}; S_0) = g(\boldsymbol{p} + \boldsymbol{e}^{S_0}) - g(\boldsymbol{p}) = (\boldsymbol{t} - \boldsymbol{x}) \cdot \boldsymbol{e}^{S_0},$$

and S_0 minimises this term if and only if S_0 contains all indices $i \in [n]$ with $t_i < x_i$ and no indices $j \in [n]$ with $t_j > x_j$. In particular, we have $S_0 \coloneqq \{i \in [n] \mid t_i < x_i\}$ due to the minimal minimiser property of S_0.

If $\boldsymbol{p}$ has a unique demanded bundle $\boldsymbol{x}$, which is easy to verify, it is straightforward to compute $\boldsymbol{x}$. Hence, in this case we can determine the minimal steepest descent direction $\boldsymbol{e}^{S_0}$ without performing SFM. In the case that $\boldsymbol{p}$ has at least two demanded bundles, the demanded bundle $\boldsymbol{x}$ shared by $\boldsymbol{p}$ and $\boldsymbol{p} + \boldsymbol{e}^{S_0}$ is unknown. However, if we define index sets $I_{\boldsymbol{p}}$ and $J_{\boldsymbol{p}}$ as

$$I_{\boldsymbol{p}} \coloneqq \{i \in [n] \mid x_i > t_i, \forall \boldsymbol{x} \in D(\boldsymbol{p})\},$$
$$J_{\boldsymbol{p}} \coloneqq \{i \in [n] \mid x_i \leq t_i, \forall \boldsymbol{x} \in D(\boldsymbol{p})\},$$

we have $I_{\boldsymbol{p}} \subseteq S_0$ and $J_{\boldsymbol{p}} \cap S_0 = \emptyset$. Hence we can restrict ourselves to minimising the submodular function $h \colon [n] \setminus (I_{\boldsymbol{p}} \cup J_{\boldsymbol{p}}) \to \mathbb{Z}$ defined by $h(T) = g'(\boldsymbol{p}; T \cup I_{\boldsymbol{p}})$ and reduce the dimensionality of the SFM problem from n to $n - |I_{\boldsymbol{p}} \cup J_{\boldsymbol{p}}|$. The following lemma shows that computing $I_{\boldsymbol{p}}$ and $J_{\boldsymbol{p}}$ is cheap.

Lemma 29. *$I_{\boldsymbol{p}}$ and $J_{\boldsymbol{p}}$ can be computed in time $O(n|\mathcal{B}|)$.*

Proof. Note that $I_{\boldsymbol{p}} = \{i \in [n] \mid \min_{\boldsymbol{x} \in D(\boldsymbol{p})} x_i > t_i\}$ and $J_{\boldsymbol{p}} = \{i \in [n] \mid \max_{\boldsymbol{x} \in D(\boldsymbol{p})} x_i \leq t_i\}$, where we compute the minimum and maximum component-wise. Fix $i \in [n]$. The minimum $\min_{x \in D(\boldsymbol{p})} x_i$ is attained if no marginal bid selects good i. Hence

$$\min x_i = \sum \{w(\boldsymbol{b}) \mid \boldsymbol{b} \in \mathcal{B} \text{ is non-marginal and demands } i\}.$$

Similarly, we claim that $\max_{\boldsymbol{x} \in D(\boldsymbol{p})} x_i$ is attained if good i is selected whenever possible and thus $\max_{\boldsymbol{x} \in D(\boldsymbol{p})} x_i$ is the sum of the weights of the bids for which i is demanded. Consider the price perturbation $\boldsymbol{p}'$ defined by $p'_i = p_i$ and $p'_j = p_j + \varepsilon$ for $j \neq i$. Here $\varepsilon > 0$ is chosen sufficiently small so that every bid that is marginal on i at $\boldsymbol{p}$ is non-marginal and demands i at $\boldsymbol{p}'$, while every non-marginal bid demanding good i at $\boldsymbol{p}$ is also non-marginal and demands i at $\boldsymbol{p}'$. This perturbation corresponds to our proposed rule to compute $\max x_i$ by selecting good i whenever possible.

Note that all bundles $\boldsymbol{x}'$ at price $\boldsymbol{p}'$ have the same number of items of i and are also demanded at $\boldsymbol{p}$. As $\boldsymbol{p} \leq \boldsymbol{p}'$ and $p_i = p'_i$, the strong substitutes property implies that there exists $\boldsymbol{x}' \in D(\boldsymbol{p}')$ such that $x_i \leq x'_i$ for all $\boldsymbol{x} \in D(\boldsymbol{p})$. In other words, any demanded bundle $\boldsymbol{x}'$ at $\boldsymbol{p}'$ maximises x_i.□

C Hardness of testing validity of unrestricted bid sets.

Here we show that the question of whether a set of bids is valid is coNP-complete. We also present a simple algorithm verifying the validity of a given list of positive and negative unit bids that runs in polynomial time if the number of goods, or the number of negative bids, is bounded by a constant. For any list of bids $\mathcal{B}$, let $\mathcal{B}^+$ and $\mathcal{B}^-$ denote the positive and negative bids.

C.1 Checking validity is coNP-complete.

Definition 8. Given a list of positive and negative bids $\mathcal{B}$, the problem Valid Bids is to decide whether $\mathcal{B}$ is valid.

Theorem 30 (Theorem 2). Valid Bids *is* coNP*-complete.*

The proof of Theorem 30 uses an equivalent definition of validity for the list of bids. Define regions in $\mathbb{R}^n_+$ that are generated by a point $\boldsymbol{p}$ and coordinates $i, j \in [n]$ as follows.

$$H_i^p \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n_+ \mid \boldsymbol{x} \leq \boldsymbol{p}, x_i = p_i\} \tag{10}$$

$$F_{ij}^p \coloneqq \{x + \beta \boldsymbol{e}^{[n]} \mid \beta \in \mathbb{R}_+, \boldsymbol{x} \leq \boldsymbol{p}, x_i = p_i \text{ and } x_j = p_j.\} \tag{11}$$

A bid $\boldsymbol{b} = (b_1, \ldots, b_n; b_{n+1})$ is *contained* in a region H_i^p (or F_{ij}^p) if the 'valuation' vector $(b_1, \ldots, b_n)$ consisting of the first n components lies in it. Moreover, we say that H_i^p or F_{ij}^p is *negative* for a list of bids if it contains more negative than positive bids, otherwise it is *non-negative.*

Observation 31. *For any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n_+$, we have $\boldsymbol{x} \in F_{ij}^y$ if and only if $(x-y)_i = (x-y)_j$ and $\boldsymbol{x} - (x-y)_i \mathbf{1} \leq \boldsymbol{y}$.*

Observation 32. *Fix $i, j \in [n], i \neq j$. Then containment in the regions given in* (10) *and* (11) *is transitive in the sense that, for any $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^n_+$,*

- *$\boldsymbol{x} \in H_i^y$ and $\boldsymbol{y} \in H_i^z$, implies $\boldsymbol{x} \in H_i^z$.*
- *$\boldsymbol{x} \in F_{ij}^y$ and $\boldsymbol{y} \in F_{ij}^z$, implies $\boldsymbol{x} \in F_{ij}^z$.*

The following definition of valid bids restates the definition given in Theorem 1, 3., in terms of regions (10) and (11).

Definition 9 (Valid bids). A list of positive and negative bids is *valid* if, for any point $\boldsymbol{p} \in \mathbb{R}^n_+$ and two coordinates $i, j \in [n]$, H_i^p and F_{ij}^p are non-negative.

On the basis of Definition 9, the proof of Theorem 2 works by reducing the well-known NP-complete problem 3-CNF Satisfiability to Valid Bids by means of an intermediate NP-complete decision problem More Negatives Dominated, which we define in Definition 10. The NP-completeness of More Negatives Dominated is established in Theorem 33 and the reduction from More Negatives Dominated to Valid Bids is given below as the proof of Theorem 2. For any two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, we say that $\boldsymbol{x}$ dominates $\boldsymbol{y}$ if $x_i \geq y_i$ for all $i \in \{1, \ldots, n\}$.

Definition 10. Given two lists P and N of vectors in $\mathbb{R}^n_+$, More Negatives Dominated is the problem of deciding whether there exists a vector $\boldsymbol{z} \in \mathbb{R}^n_+$ that dominates more vectors in N than in P.

Theorem 33. *The problem* More Negatives Dominated *is NP-complete.*

Proof. Note that verifying whether a given point $\boldsymbol{z} \in \mathbb{R}^n_+$ dominates more vectors in N than in P can be done in polynomial time. This establishes membership of More Negatives Dominated in the class NP. We show completeness by reducing from 3-CNF Satisfiability. Recall that a boolean formula ϕ defined on n variables $x_1, \ldots, x_n$ is 3-CNF if

$$\phi = \bigwedge_{i=1}^{m} C_i,$$

where $C_i = L_1^i \vee L_2^i \vee L_3^i$ are its clauses and $L_j^i \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$ are its literals. Here $\overline{x_i}$ denotes the negation of x_i. Without loss of generality, we assume that the three variables appearing in each clause are distinct. Given such a 3-CNF formula ϕ, first construct a pair of lists (P_i, N_i) of vectors in $\mathbb{R}^n_+$ for each clause C_i as follows. For notational convenience, $\boldsymbol{e}^{a\ldots z}$ denotes the characteristic vector $\boldsymbol{e}^{\{a,\ldots,z\}}$ of $\{a, \ldots, z\}$.

- If $C_i = x_a \vee x_b \vee x_c$, let $N_i = \{\boldsymbol{e}^{\emptyset}\}$ and $P_i = \{\boldsymbol{e}^{abc}\}$.
- If $C_i = x_a \vee x_b \vee \overline{x_c}$, let $N_i = \{\boldsymbol{e}^{\emptyset}, \boldsymbol{e}^{abc}\}$ and $P_i = \{\boldsymbol{e}^{ab}\}$.
- If $C_i = x_a \vee \overline{x_b} \vee \overline{x_c}$, let $N_i = \{\boldsymbol{e}^{\emptyset}, \boldsymbol{e}^{ab}, \boldsymbol{e}^{ac}\}$ and $P_i = \{\boldsymbol{e}^{a}, \boldsymbol{e}^{abc}\}$.
- If $C_i = \overline{x_a} \vee \overline{x_b} \vee \overline{x_c}$, let $N_i = \{\boldsymbol{e}^{a}, \boldsymbol{e}^{b}, \boldsymbol{e}^{c}, \boldsymbol{e}^{abc}\}$ and $P_i = \{\boldsymbol{e}^{ab}, \boldsymbol{e}^{ac}, \boldsymbol{e}^{bc}\}$.

Finally, let N be the sum of the N_i and let P be the sum of the P_i together with $m - 1$ copies of $\boldsymbol{e}^{\emptyset}$. This completes our reduction from 3-CNF Satisfiability to More Negatives Dominated. Clearly, P and N can be constructed from ϕ in polynomial time. To show correctness of the reduction, associate with every vector $\boldsymbol{z} \in \mathbb{R}^n_+$ a truth assignment β_z (on boolean variables $x_1, \ldots, x_n$) that sets x_i to TRUE if $z_i < 1$ and FALSE otherwise.

Observation 34. *Let $\boldsymbol{z} \in \mathbb{R}^n_+$ be a point and β_z be its associated truth assignment. For any $i \in \{1, \ldots, m\}$, $\boldsymbol{z}$ dominates one more point in N_i than in P_i if β_z satisfies C_i and an equal number of points in N_i and P_i if β_z does not satisfy C_i.*

Suppose ϕ is a satisfiable 3-CNF formula with satisfying truth assignment β. Then define $\boldsymbol{z} \in \mathbb{R}^n_+$ by

$$z_i = \begin{cases} 0 & \text{if } \beta[x_i] = \text{TRUE}, \\ 1 & \text{else.} \end{cases}$$

Hence β is the truth assignment associated with z. By Observation 34, $\boldsymbol{z}$ dominates one more point in N_i than in P_i for each i, so due to the additional $m - 1$ origin points $\boldsymbol{e}^{\emptyset}$ added to P, $\boldsymbol{z}$ dominates exactly one more point in N than in P. Conversely, suppose there exists a point $\boldsymbol{z} \in \mathbb{R}^n_+$ that dominates more points in N than in P, and let β_z be its corresponding truth assignment. Then $\boldsymbol{z}$ dominates at least m more points in N than in $\sum_{i=1}^m P_i$. By Observation 34, we know that $\boldsymbol{z}$ dominates at most one more point in N_i than in P_i, which implies that this is the case for each i. Hence by the same observation, β_z satisfies all clauses of ϕ and thus ϕ itself. □

Proof of Theorem 2. In order to show that a list of bids $\mathcal{B}$ is not valid, it suffices to provide a certificate in the form of a point $\boldsymbol{p}$ and coordinates $i, j \in [n]$. We can verify in polynomial time whether H_i^p or F_{ij}^p is negative. This establishes membership of Valid Bids in coNP.

Let (N, P) be an instance of More Negatives Dominated and n be the dimension of its vectors. We construct an $(n+1)$-dimensional instance $\mathcal{B}$ of Valid Bids such that $(N, P) \in$ More Negatives Dominated if and only if $\mathcal{B} \notin$ Valid Bids as follows. Let the negative and positive bids of $\mathcal{B}$ be given by prepending a component to the vectors as follows.

$$\mathcal{B}^- \coloneqq \{(1, \boldsymbol{v}) \mid \boldsymbol{v} \in N\}$$
$$\text{and } \mathcal{B}^+ \coloneqq \{(1, \boldsymbol{w}) \mid \boldsymbol{w} \in P\} + \{(0, \boldsymbol{v}) \mid \boldsymbol{v} \in N\} + \{(1, \boldsymbol{v}) + \boldsymbol{e}^{[n+1]} \mid \boldsymbol{v} \in N\}.$$

Clearly, $\mathcal{B} = \mathcal{B}^+ + \mathcal{B}^-$ can be constructed efficiently. Note that all F_{ij}^x and all H_i^x with $i \neq 0$ are non-negative by construction. Indeed, fix F_{ij}^x and suppose it contains r negative bids. Then for each such bid $(1, \boldsymbol{v})$, where $\boldsymbol{v} \in N$, there exists a positive bid $(1, \boldsymbol{v}) + \boldsymbol{e}^{[n+1]}$ that is contained in F_{ij}^x, so that the region is non-negative. Analogously, for any H_i^x with $i \neq 0$, each negative bid $(1, \boldsymbol{v})$ has a corresponding positive bid $(0, \boldsymbol{v})$. Further, as H_0^x contain no negative points unless $x_0 = 1$, the correctness of our reduction thus follows from the fact that $\boldsymbol{z}$ dominates r points from N and s points from P if and only if H_0^x with $\boldsymbol{x} = (1, \boldsymbol{z})$ contains r negative and s positive bids. □

C.2 A simple algorithm for testing validity.

In Section C.1, we show that checking the validity of a bid list in the general case in coNP-complete. Hence we cannot expect an efficient algorithm for the task of checking validity of a list of bids in the general case. Here we present a simple algorithm that tests validity of a given list of bids in polynomial time if the number of goods or the number of negative bids is bounded by a constant. Such constraints may be reasonable in certain economic settings.

C.2.1 The algorithm

Our procedure rests on Theorem 35, which reduces the validity condition from Definition 9 to a finite number of checks. Lemma 36 reduces the number of checks further. Together, the two results immediately yield Algorithm 10, whose running time is given in Theorem 37. For any set of bids U, define the minimal dominating vector $md(U)$ of U as the component-wise maximum over the valuation vectors of the bids in U, so that $md(U)_i = \max_{\boldsymbol{b} \in U} b_i$ for all $i \in [n]$. We note that if all bids of U agree on some coordinate i (that is, if $b_i = b'_i$ for all $\boldsymbol{b}, \boldsymbol{b}' \in U$), they lie in H_i^x for large enough $\boldsymbol{x}$, and $md(U)$ is the minimal such $\boldsymbol{x}$ so that H_i^x contains all points in U. Similarly, for any set of bids U and $i \in [n]$, define $mdF(i, U)$ as $mdF(i, U) \coloneqq \min_{\boldsymbol{b} \in U} b_i \mathbf{1} + md(\{\boldsymbol{b} - b_i \mathbf{1} \mid \boldsymbol{b} \in U\})$. We note that if, for some $i \neq j$, the set U satisfies $b_i - b_j = b'_i - b'_j$ for all $\boldsymbol{b}, \boldsymbol{b}' \in U$, then the bids in U lie in F_{ij}^x for small enough $\boldsymbol{x}$, and it follows from Observation 31 that $mdF(i, U) = mdF(j, U)$ is the maximal such $\boldsymbol{x}$.

Theorem 35. *A list of bids $\mathcal{B}$ is valid if and only if the following two conditions hold.*

1. *For every set $U \subseteq \mathcal{B}^-$ of negative bids that agree on the i-th coordinate, that is $b_i = b'_i \ \forall b, b' \in U$, the region $H_i^{md(U)}$ is non-negative.*
2. *For every set $U \subseteq \mathcal{B}^-$ of negative bids that satisfies $b_i - b'_i = b_j - b'_j$ for all $b, b' \in U$ and some $i \neq j$, the region $F_{ij}^{mdF(i,U)}$ is non-negative.*

Proof. Proof. The implication is immediate by Definition 9. Conversely, suppose $\mathcal{B}$ satisfies the second condition. First we show that condition 1 of Definition 9 is satisfied. Fix H_i^x, let $U = \mathcal{B}^- \cap H_i^x$ be the set of negative bids in H_i^x and $\boldsymbol{y} = md(U)$. Then by construction, we have $\boldsymbol{b} \in H_i^y$ for all $\boldsymbol{b} \in U$ and $\boldsymbol{y} \in H_i^x$. As H_i^y has $|U|$ negative bids, it also contains at least $|U|$ positive bids by assumption and by Observation 32, these positive bids are also in H_i^x. Condition 2 is shown analogously. Fix F_{ij}^x, let U be the negative bids in this region and $\boldsymbol{z} = mdF(U)$. Suppose $\boldsymbol{b} \in F_{ij}^z$ for every $\boldsymbol{b} \in U$ and $\boldsymbol{z} \in F_{ij}^x$. Then Observation 32 implies that F_{ij}^x is non-negative. To see that $\boldsymbol{b} \in F_{ij}^z$ for every $\boldsymbol{b} \in U$, we verify the conditions of Observation 31. Firstly,

$$z_i - z_j = \min_{\boldsymbol{b} \in U} b_i - (\min_{\boldsymbol{b} \in U} b_i + \max_{\boldsymbol{b} \in U} (b_j - b_i)) = \max_{\boldsymbol{b} \in U} (b_j - b_i) = b_j - b_i$$

for all $\boldsymbol{b} \in U$, as the difference $b_j - b_i$ is the same for all $\boldsymbol{b} \in F_{ij}^z$. Secondly, for any $k \in [n]$ and $\boldsymbol{b} \in U$, we have

$$(\boldsymbol{b} - (b_i - z_i)\mathbf{1})_k = z_i + b_k - b_i \leq z_i + \max_{\boldsymbol{b} \in U} (b_k - b_i) = z_k.$$

We can verify $\boldsymbol{z} \in F_{ij}^x$ similarly. □

Lemma 36. *For any list of negative bids $U \subseteq \mathcal{B}^-$ there exists a list $U' \subseteq U$ with $|U'| \leq n$ so that $md(U') = md(U)$. Moreover, for each coordinate $i \in [n]$, there exists a list $U'' \subseteq U$ with $|U''| \leq n+1$ so that $mdF(i, U'') = mdF(i, U)$.*

Proof. Proof. Let U' be a list of bids $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^n$, where $\boldsymbol{u}^k$ is a bid from U that maximises the k-th component, i.e. $\boldsymbol{u}^k \in \arg\max_{\boldsymbol{v} \in U} v_k$. Then $md(U') = md(U)$ by construction. Secondly, fix $i \in [n]$ and let U'' be a list of bids $\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{n+1}$, where $\boldsymbol{u}^1 \in \arg\min_{\boldsymbol{b} \in U} b_i$ and $\boldsymbol{u}^k \in \arg\max_{\boldsymbol{b} \in U}(b_k - b_i)$ for $k \in \{2, \ldots, n+1\}$. Then $mdF(U'') = mdF(U)$ by construction and we are done. □

Algorithm 10 Checking the validity of a bid list $\mathcal{B}$

1: **for all** subsets $U \subseteq \mathcal{B}^-$ of negative bids with $|U| \leq n+1$ **do**
2: **for all** $i, j \in [n]$ **do**
3: Verify the two conditions given in Theorem 35.

Theorem 37. *Algorithm 10 runs in time $O\left(\binom{|\mathcal{B}^-|}{n} n^3 |\mathcal{B}|\right)$. Moreover, if $|\mathcal{B}^-| \leq k$ or $n \leq k$, the algorithm is polynomial in the input size $n|\mathcal{B}|$.*

Proof. Proof. There are $\binom{|\mathcal{B}^-|}{n+1}$ subsets U of $\mathcal{B}^-$ for the algorithm to iterate over, while there are n^2 possibilities for the index pair i, j. For each U and i, j, checking the conditions in Theorem 35 takes $n|\mathcal{B}|$ time. This implies the running time. Suppose the number of negative bids $|\mathcal{B}^-| \leq k$ is bounded by some $k \in \mathbb{N}$. Then $\binom{|\mathcal{B}^-|}{n+1}$ is constant and the running time of our algorithm is $O(n^3(|V| + |W|))$. Secondly, if the number of goods $n \leq k$ is bounded, we have $\binom{|\mathcal{B}^-|}{n+1} \leq |\mathcal{B}^-|^{k+1}$ and hence a running time of $O(|\mathcal{B}^-|^{k+1}|\mathcal{B}|)$. □

D Additional proofs.

We give some definitions that are used in the proofs below. Here and in the following, we use terminology and material developed in Baldwin and Klemperer [2019] and refer to the same for details. Define the Locations of Indifference Prices (LIP) for a bid $\boldsymbol{b}$ as

$$\mathcal{L}_{\boldsymbol{b}} = \{\boldsymbol{p} \in \mathbb{R}^n \mid |\arg\max_{i \in [n]_0}(b_i - p_i)| > 1\}.$$

Similarly, the LIP of a list of bids $\mathcal{B}$ is given by

$$\mathcal{L}_{\mathcal{B}} = \{\boldsymbol{p} \in \mathbb{R}^n \mid |D_{\mathcal{B}}(\boldsymbol{p})| > 1\}.$$

Any $\mathcal{L}_{\mathcal{B}}$ can be decomposed into an $(n-1)$-dimensional rational polyhedral complex $\Pi_{\mathcal{B}}$. We call the $(n-1)$-dimensional polyhedra in this complex its *facets*. Endow every facet F of $\Pi_{\mathcal{B}}$ with a weight, as follows:

$$w_{\mathcal{B}}(F) \coloneqq \sum\{w(\boldsymbol{b}) \mid \boldsymbol{b} \in \mathcal{B} \text{ and } \mathcal{L}_{\boldsymbol{b}} \supseteq F\}.$$

Finally, we say that $(\Pi_{\mathcal{B}}, w_{\mathcal{B}})$ is *balanced* if for every $(n-2)$-cell G of Π, the weights $w(F_j)$ on the facets $F_1, \ldots, F_l$ that contain G, and primitive integer normal vectors $\mathbf{v}_{F_j}$ for these facets that are defined by a fixed rotational direction about G, satisfy $\sum_{j=1}^{l} w(F_j)\mathbf{v}_{F_j} = 0$.

Definition 11 (cf. Baldwin and Klemperer [2019, Definition 3.9]). The *strong substitutes vectors* are those non-zero vectors in $\mathbb{Z}^n$ which have at most one $+1$ entry, at most one -1 entry, and no other non-zero entries.

D.1 Proof of Theorem 1.

Proof of Theorem 1. The equivalence of (1) and (2) follows from Proposition 6 if we set $P = \mathbb{R}^n$. The implication (3) $\Rightarrow$ (1) is well-known; see for instance Murota and Shioura [2014, Theorem 7.3] for a stronger statement. We now show $eq2 \Rightarrow$ (3), using the Valuation-Complex Equivalence Theorem from Baldwin and Klemperer [2019]. Let $(\Pi, w_{\mathcal{B}})$ be the weighted polyhedral complex associated with $\mathcal{B}$. Note that as well as showing that balancing holds, we must also demonstrate that all weights of the polyhedral complex are positive, which is the setting for Baldwin and Klemperer [2019]. Observe first that every facet of $\Pi_{\mathcal{B}}$ has a strong substitutes vector as its normal vector, as this property holds for any individual $\mathcal{L}_{\boldsymbol{b}}$.

Note that the weighted polyhedral complex $(\Pi_{\boldsymbol{b}}, w_{\boldsymbol{b}})$ associated with a single bid is balanced. If $w(\boldsymbol{b}) = 1$ then this follows because $(\Pi_{\boldsymbol{b}}, w_{\boldsymbol{b}})$ is the weighted polyhedral complex associated with a simple valuation for at most one unit of any good (and see Baldwin and Klemperer [2019, Section 2.3]). This then extends to the case $w(\boldsymbol{b}) = -1$ because changing the sign of the weights of all facets does not affect balancing. Then $(\Pi_{\mathcal{B}}, w_{\mathcal{B}})$ is also balanced, as it is the complex corresponding to the union of balanced LIPs (see the proof of Lemma 3.13 in Baldwin and Klemperer [2019]).

Next we see that all weights of $(\Pi_{\mathcal{B}}, w_{\mathcal{B}})$ are non-negative. Let F be a facet of $\Pi_{\mathcal{B}}$ and fix a point $\boldsymbol{p}$ in the relative interior of F. Then, since the normal vector to F is a strong substitutes vector (and by Baldwin and Klemperer [2019, Proposition 2.4]), every bid is either non-marginal at $\boldsymbol{p}$ or marginal between the same two distinct goods, $i, i' \in [n]_0$. Let $\mathcal{B}_{ii'}$ denote the bids that are marginal between i, i' at $\boldsymbol{p}$. Note that we have $F \subseteq \mathcal{L}_{\boldsymbol{b}}$ if and only if $\boldsymbol{b}$ is marginal on i, i', so $w_{\mathcal{B}}(F) = \sum_{\boldsymbol{b} \in \mathcal{B}_{ii'}} w(\boldsymbol{b}) \geq 0$ (by assumption 2).

$(\Pi_{\mathcal{B}}, w_{\mathcal{B}})$ may have zero-weighted facets: let the set of these be $\Pi^0_{\mathcal{B}}$ and write $\Pi^+_{\mathcal{B}} \coloneqq \Pi_{\mathcal{B}} \setminus \Pi^0_{\mathcal{B}}$. Then $\Pi^+_{\mathcal{B}}$ inherits from $\Pi_{\mathcal{B}}$ the structure of a polyhedral complex. Moreover, if we write $w^+_{\mathcal{B}}$ for the restriction of $w_{\mathcal{B}}$ to the facets of $\Pi^+_{\mathcal{B}}$, then $(\Pi^+_{\mathcal{B}}, w^+_{\mathcal{B}})$ is balanced: removing 0-weighted facets does not affect this criterion.

So, we write $\mathcal{L}^+_{\mathcal{B}}$ for the union of the cells in $\Pi^+_{\mathcal{B}}$. Let $\overline{\boldsymbol{p}}$ be a price vector whose components are strictly larger than the components of any bid, so that $D_{\mathcal{B}}(\overline{\boldsymbol{p}}) = \mathbf{0}$. By the valuation-complex equivalence theorem, there exists a unique concave valuation u such that $\mathcal{L}_u = \mathcal{L}^+_{\mathcal{B}}$, $w_u = w^+_{\mathcal{B}}$, $u(\mathbf{0}) = 0$ and $D_u(\overline{\boldsymbol{p}}) = \mathbf{0}$.

If $w(\boldsymbol{b})$ is positive then $\mathcal{L}_{\boldsymbol{b}}$ has the property that facets and weights define all changes in demand between UDRs, [Baldwin and Klemperer, 2019, Section 2.2], and it is easy to see that the same follows for negative-weighted bids, and thus extends by aggregation to our positive-weighted polyhedral complex $(\Pi^+_{\mathcal{B}}, w^+_{\mathcal{B}})$. The same holds for $\mathcal{L}_u$ for standard reasons. So $D_u(\overline{\boldsymbol{p}}) = D_{\mathcal{B}}(\overline{\boldsymbol{p}})$ implies $D_u(\boldsymbol{p}) = D_{\mathcal{B}}(\boldsymbol{p})$ for all UDR prices $\boldsymbol{p} \in \mathbb{R}^n$, and hence for all prices, since u is concave and $D_{\mathcal{B}}(\boldsymbol{p})$ is defined to be discrete convex.

We now show that $f_{\mathcal{B}} = f_u$. For any bundle $\boldsymbol{x}$, let $\boldsymbol{p}$ be a price at which $\boldsymbol{x}$ is demanded and let $(i(\boldsymbol{b}))_{\boldsymbol{b} \in \mathcal{B}}$ be a list of goods allocated to the bids $\boldsymbol{b} \in \mathcal{B}$ in order to obtain $\boldsymbol{x}$. That is, $i(\boldsymbol{b}) \in D_{\mathcal{B}}(\boldsymbol{p})$ for every $\boldsymbol{b} \in \mathcal{B}$ and $x = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \boldsymbol{e}^{i(\boldsymbol{b})}$. Then we claim that

$$u(\boldsymbol{x}) = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) b_{i(\boldsymbol{b})}. \tag{12}$$

This then immediately implies $f_{\mathcal{B}}(\boldsymbol{p}) = f_u(\boldsymbol{p})$ for any $\boldsymbol{p} \in \mathbb{R}^n$, as

$$f_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \max_{i \in [n]_0} (b_i - p_i) = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b})(b_{i(\boldsymbol{b})} - p_{i(\boldsymbol{b})}) = u(\boldsymbol{x}) - \boldsymbol{x} \cdot \boldsymbol{p} = f_u(\boldsymbol{p}).$$

To see that (12) holds, let $\boldsymbol{x}'$ and $\boldsymbol{x}''$ be bundles demanded in neighbouring UDRs and let $\boldsymbol{p}$ be a price on the relative interior of the facet separating these UDRs. Moreover, let $(i'(\boldsymbol{b}))_{\boldsymbol{b} \in \mathcal{B}}$ and $(i''(\boldsymbol{b}))_{\boldsymbol{b} \in \mathcal{B}}$ be the allocation to bids at $\boldsymbol{p}$ that lead to bundles $\boldsymbol{x}'$ and $\boldsymbol{x}''$. Then we have

$$u(\boldsymbol{x}') - \boldsymbol{p} \cdot \boldsymbol{x}' = u(\boldsymbol{x}'') - \boldsymbol{p} \cdot \boldsymbol{x}'' \tag{13}$$

and

$$f_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\boldsymbol{b}\in\mathcal{B}} w(\boldsymbol{b})(b_{i'(\boldsymbol{b})} - p_{i'(\boldsymbol{b})}) = \sum_{\boldsymbol{b}\in\mathcal{B}} w(\boldsymbol{b})(b_{i''(\boldsymbol{b})} - p_{i''(\boldsymbol{b})}). \tag{14}$$

Suppose (12) holds for bundle $\boldsymbol{x'}$ and $(i'(\boldsymbol{b}))_{\boldsymbol{b}\in\mathcal{B}}$. Then (13) and (14) imply that (12) holds for $\boldsymbol{x''}$ and $(i''(\boldsymbol{b}))_{\boldsymbol{b}\in\mathcal{B}}$. Finally, as $u(\boldsymbol{0}) = 0 = f_{\mathcal{B}}(\boldsymbol{0})$, this implies (12) for all possible bundles $\boldsymbol{x}$, by induction.

Finally, the strong substitutes property for u follows immediately from concavity of u and our observation above that all facets of $\Pi_{\mathcal{B}}$, and hence of $\mathcal{L}_u$, are normal to a strong substitutes vector [Baldwin and Klemperer, 2019, Proposition 3.8]. □

D.2 Proof of Proposition 6.

We make use of the following technical observation to prove Proposition 6.

Observation 38. *Let $f : [0,1] \to \mathbb{R}$ be a continuous, piecewise-linear function. If its slope is non-decreasing as we move from 0 to 1, f is convex. This implies that if f is not convex, there exists a point λ at which two linear segments meet and the slope decreases.*

Proof of Proposition 6. Recall that $f_{\mathcal{B}}$ restricted to P is convex if and only if

$$f_{\mathcal{B}}(\lambda\boldsymbol{p} + (1-\lambda)\boldsymbol{q}) \leq \lambda f_{\mathcal{B}}(\boldsymbol{p}) + (1-\lambda) f_{\mathcal{B}}(\boldsymbol{q}), \quad \forall \boldsymbol{p}, \boldsymbol{q} \in P \text{ and } \lambda \in [0,1]. \tag{15}$$

Suppose that $\boldsymbol{p}$ and $\boldsymbol{q}$ are two non-marginal prices in neighbouring UDRs such that for some $\lambda \in (0,1)$, the point $\boldsymbol{r} = \lambda\boldsymbol{p} + (1-\lambda)\boldsymbol{q}$ lies on the interior of a facet separating the UDRs. This implies two possibilities for each bid $\boldsymbol{b} \in \mathcal{B}$. Either $\boldsymbol{b}$ non-marginally demands the same good at $\boldsymbol{p}$ and $\boldsymbol{q}$ (and thus also at $\boldsymbol{r}$). Or $\boldsymbol{b}$ non-marginally demands i at $\boldsymbol{p}$ and i' at $\boldsymbol{q}$, and is marginal (only) on i and i' at $\boldsymbol{r}$. Moreover, all bids of the latter kind are marginal on the same two goods.

Let $\mathcal{B}_{ii'}$ denote the bids that demand the set $\{i, i'\}$ at $\boldsymbol{r}$. We show that $f_{\mathcal{B}}$ satisfies (15) for $\boldsymbol{p}, \boldsymbol{q}$ and any $\lambda \in [0,1]$ if and only if the weights of the bids marginal on i and i' sum to a non-negative number, i.e. if $\sum_{\boldsymbol{b}\in\mathcal{B}} w(\boldsymbol{b}) \geq 0$. By construction, we have

$$\begin{aligned}
f_{\mathcal{B}}(\boldsymbol{p}) &= \sum_{\boldsymbol{b}\in\mathcal{B}_{ii'}} w(\boldsymbol{b})(b_i - p_i) + \sum_{\boldsymbol{b}\notin\mathcal{B}_{ii'}} w(\boldsymbol{b}) \max_{j\in[n]_0}(b_j - p_j),\\
f_{\mathcal{B}}(\boldsymbol{q}) &= \sum_{\boldsymbol{b}\in\mathcal{B}_{ii'}} w(\boldsymbol{b})(b_{i'} - q_{i'}) + \sum_{\boldsymbol{b}\notin\mathcal{B}_{ii'}} w(\boldsymbol{b}) \max_{j\in[n]_0}(b_j - q_j),\\
f_{\mathcal{B}}(\boldsymbol{r}) &= \sum_{\boldsymbol{b}\in\mathcal{B}_{ii'}} w(\boldsymbol{b})(b_i - r_i) + \sum_{\boldsymbol{b}\notin\mathcal{B}_{ii'}} w(\boldsymbol{b}) \max_{j\in[n]_0}(b_j - r_j).
\end{aligned}$$

Hence the inequality

$$f_{\mathcal{B}}(\boldsymbol{r}) \leq \lambda f_{\mathcal{B}}(\boldsymbol{p}) + (1-\lambda) f_{\mathcal{B}}(\boldsymbol{q}) \tag{16}$$

holds if and only if

$$\sum_{\boldsymbol{b}\in\mathcal{B}_{ii'}} w(\boldsymbol{b})[(b_i - q_i) - (b_{i'} - q_{i'})] \leq 0. \tag{17}$$

Note that the term $(b_i - q_i) - (b_{i'} - q_{i'})$ is strictly negative, as $\boldsymbol{b}$ demands i' and not i at $\boldsymbol{q}$. Moreover, this term is the same for all bids $\boldsymbol{b} \in \mathcal{B}_{ii'}$, as i and i' are both demanded at $\boldsymbol{r}$. This implies by (17) that $\sum_{\boldsymbol{b}\in\mathcal{B}_{ij}} w(\boldsymbol{b}) \geq 0$ if and only if (16) holds.

Now we prove our main statement. Suppose $f_{\mathcal{B}}$ restricted to P is not convex. Then there exist $\boldsymbol{p}, \boldsymbol{q} \in P$ and $\lambda \in [0,1]$ that violate (15). Due to continuity of $f_{\mathcal{B}}$, we can perturb $\boldsymbol{p}$ and $\boldsymbol{q}$ slightly so that (15) is still violated, $\boldsymbol{p}, \boldsymbol{q}$ are non-marginal and the line segment $[\boldsymbol{p}, \boldsymbol{q}]$ crosses only interiors of facets of the LIP $\mathcal{L}_{\mathcal{B}}$. We can assume wlog that $\boldsymbol{p}$ and $\boldsymbol{q}$ are in neighbouring UDRs; otherwise, we can apply Observation 38 to find two non-marginal prices on the interior

of the line segment for which (15) fails. Thus we can apply the above to see the implication $2 \Rightarrow 1$.

For the converse ($(1) \Rightarrow (2)$), suppose there exist $\boldsymbol{r}$ and $i, i' \in [n]_0$ such that the weights of bids marginal on i, i' at $\boldsymbol{r}$ sum to a negative number. Wlog we can assume that all marginal bids at $\boldsymbol{r}$ are marginal only on goods i, i', by subtracting $\delta \boldsymbol{e}^{\{i,i'\}}$ from $\boldsymbol{r}$ for some infinitesimal positive value of δ if necessary. Hence $\boldsymbol{r}$ lies on the interior of a facet F and for small enough $\varepsilon > 0$, the points $\boldsymbol{p} = \boldsymbol{r} + \varepsilon \boldsymbol{e}^i - \varepsilon \boldsymbol{e}^{i'}$ and $\boldsymbol{q} = \boldsymbol{r} - \varepsilon \boldsymbol{e}^i + \varepsilon \boldsymbol{e}^{i'}$ lie in P and in neighbouring UDRs separated by F. By the above, this implies that $\boldsymbol{p}$ and $\boldsymbol{q}$ with $\lambda = 1/2$ violate (15) and we are done. □

D.3 Proof of Proposition 14.

In order to prove Proposition 14, we first present and prove three technical lemmas and one proposition about local prices and demand for valuations.

Lemma 39. *Suppose G is an invertible matrix whose rows are strong substitute vectors (Definition 11). Then there exists a diagonal matrix U multiplying some (possibly empty) set of rows of G by -1, such that every column of UG contains exactly one $+1$ entry and such that the rows of $(UG)^{-1}$ contain at most two non-zero entries, and all non-zero entries of $(UG)^{-1}$ are $+1$.*

Proof. Note that any vector of the form $\boldsymbol{e}^i - \boldsymbol{e}^j$ for $i \neq j$ has zero inner product with $\sum_k \boldsymbol{e}^k$ and conclude that, since G is invertible, at least one row of G is $\pm$ a coordinate vector. So there exists an elementary row swapping operation T_1, an elementary row multiplying operation U_1 which multiplies one row by either $+1$ or -1, and an elementary column swapping operation V_1, such that $T_1 U_1 G V_1 = G'$ whose first row is $\boldsymbol{e}^1$. Note that all of these elementary matrix operations are themselves unimodular.

Because G' is invertible we can apply the same logic to the matrix that remains when we strike the first row and column from G', obtaining a $(n-1) \times (n-1)$ submatrix whose first row is the first coordinate vector. These elementary operations may be extended to the original first row and column without changing the desired properties. Continuing in this fashion, we obtain an expression $T_n U_n \cdots T_1 U_1 G V_1 \cdots V_n = G^{(n)}$ such that $G^{(n)}$ is an lower triangular matrix with 1s along its main diagonal, where for all i, U_i is an elementary row multiplying operation which multiplies a row by either $+1$ or -1 and T_i, V_i are elementary row and column swapping operations respectively. The rows of $G^{(n)}$ are still strong substitute vectors and so any non-zero entries below the main diagonal are -1s.

It is clear that $(G^{(n)})^{-1}$ is given by the lower triangular matrix whose entries are the absolute value of the entries of $G^{(n)}$. In particular, then, each row of $(G^{(n)})^{-1}$ contains at most two non-zero entries, and all non-zero entries of $(G^{(n)})^{-1}$ are $+1$.

Now note that, by properties of elementary matrices, we may re-write $T_n U_n \cdots T_1 U_1 = TU$ in which T is a permutation matrix and U multiplies some (possibly empty) set of rows by -1. We may similarly write $V_1 \cdots V_n = V$ where V is a permutation matrix. So $G^{(n)} = TUGV$. Since T and V both simply permute rows and columns, we conclude that UG inherits from $G^{(n)}$ the property that each column contains exactly one $+1$ entry.

Thus $(G^{(n)})^{-1} = V^{-1}(UG)^{-1}T^{-1}$ i.e. $(UG)^{-1} = V(G^{(n)})^{-1}T$. As the structure of permutation matrices are identical, whether they act on rows or columns, this implies that $(UG)^{-1}$ inherits from $(G^{(n)})^{-1}$ the properties of each row containing at most two non-zero entries, and all non-zero entries being equal to $+1$. □

Lemma 40. *Suppose that G is an invertible $n' \times n'$ matrix whose rows are strong substitute vectors (Definition 11), and that H_i is the ith column of $H = (h_{ij})_{i,j=1}^n := \begin{pmatrix} 0 \\ G_2 \end{pmatrix}$ where we have*

written G in block form $G = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$. Then either $G^{-1}H_i = \boldsymbol{e}^S$ for some $S \subseteq [n']$ with $i \in S$, or $G^{-1}H_i = -\boldsymbol{e}^S$ for some $S \subseteq [n']$ with $i \notin S$.

Proof. By Lemma 39 there exists a diagonal matrix U with $+1$ and -1 on its diagonal such that the rows of $(UG)^{-1}$ contain at most two non-zero entries, and all non-zero entries of $(UG)^{-1}$ are $+1$, and each column of UG contains precisely one $+1$ entry. Observe that $G^{-1}H_i = (UG)^{-1}UH_i$. We may therefore replace G by UG for brevity in the proof.

By definition $G^{-1}G_i = \boldsymbol{e}^i$. But each row of G^{-1} contains at most two non-zero entries, all $+1$, so this is a stacked set of equations of the form $g_{li} = 0$ or $g_{li} + g_{mi} = 0$ (with any indices $l \neq m$ with exceptionally one which takes the form of either $g_{ji} = 1$ or $g_{ji} + g_{ki} = 1$ (with some indices $j \neq k$). The latter case correponds to the ith row of G^{-1}. Since each $g_{ji} \in \{-1, 0, 1\}$ we conclude for the ith row that there exactly one non-zero entry in G_i which is aligned to a non-zero entry in the ith row of G^{-1}, and this entry is equal to $+1$. Write this as $g_{ji} = 1$, i.e. j is the row in G_i of this entry.

Recall it follows that $g_{ki}\{-1, 0\}$ for $k \neq j$. Translating this across to $G^{-1}H_i$, we find there are two cases:

If $h_{ji} = g_{ji} = 1$ then every incidence of '$g_{li} = 0$' translates to '$h_{li} = 0$' and so gives us a zero entry in H_i; and every incidence of $g_{li} + g_{mi} = 0$ either translates to $h_{li} + h_{mi} = 0$ or to $h_{li} + h_{mi} = 1$ (which holds if, w.l.o.g., $g_{li} = -1$ but $h_{li} = 0$; note that in this case it must follow that $g_{mi} = 1$ and hence that $m = j$ and hence by assumption that $h_{mi} = 1$). That is, every entry in H_i is either 0 or $+1$, and in particular the ith entry is 1.

But suppose $h_{ji} = 0 \neq g_{ji}$. It still holds that every incidence of '$g_{li} = 0$' translates to '$h_{li} = 0$'. But now every incidence of $g_{li} + g_{mi} = 0$ either translates to $h_{li} + h_{mi} = 0$ or to $h_{li} + h_{mi} = -1$. For if w.l.o.g. $g_{li} = -1$ then $g_{mi} = 1$ and so $m = j$ and $h_{mi} = 0$ and hence $h_{li} + h_{mi} \in \{-1, 0\}$. That is, every entry in H_i is either 0 or -1, and in particular the ith entry is 0. □

Lemma 41. *Suppose that G is an $n' \times n$ matrix whose rows are strong subtitute vectors (Definition 11) and that $S \subseteq [n]$. Then $G\boldsymbol{e}^S \in \{-1, 0, 1\}^n$ and in particular $G\boldsymbol{e}^S \geq -\mathbf{1}$ (the vector of 1s).*

Proof. Each row of $G\boldsymbol{e}^S$ is the sum of a subset of the entries in the corresponding row of G, and so by definition of strong substitue vectors it is only possible that this is ± 1 or 0. □

Proposition 42. *Suppose that we have two strong substitute integer-valued valuations, u^1 and u^2. Fix $i \in [n]$ and write $u^{2[\varepsilon,i]}$ for the valuation given by $u^{2[\varepsilon,i]}(\boldsymbol{x}) = u^2(\boldsymbol{x}) + \varepsilon\boldsymbol{e}^i \cdot \boldsymbol{x}$. For any $\boldsymbol{r}^1, \boldsymbol{r}^2 \in \mathbb{Z}^n$ suppose that there exists a price $\boldsymbol{p}^*$ such that $\boldsymbol{r}^1 \in D_{u^1}(\boldsymbol{p}^*)$, and $\boldsymbol{r}^2 \in D_{u^{2[\varepsilon,i]}}(\boldsymbol{p}^*)$. Then there exists a price $\boldsymbol{p}^0$ such that $\boldsymbol{r}^1 \in D_{u^1}(\boldsymbol{p}^0)$, and $\boldsymbol{r}^2 \in D_{u^2}(\boldsymbol{p}^0)$. Moreover then, for any $\boldsymbol{p}$ such that $\boldsymbol{r} \coloneqq \boldsymbol{r}^1 + \boldsymbol{r}^2 \in D_{u\{1,2\}}(\boldsymbol{p})$, it follows that $\boldsymbol{r}^1 \in D_{u^1}(\boldsymbol{p})$, and $\boldsymbol{r}^2 \in D_{u^2}(\boldsymbol{p})$.*

Proof. For $j = 1, 2$ write P^j for the set of prices $\boldsymbol{p}$ at which $\boldsymbol{r}^j \in D_{u^j}(\boldsymbol{p})$. It follows from the definition of $u^{2[\varepsilon,i]}$ that the set of prices $\boldsymbol{p}$ at which $\boldsymbol{r}^2 \in D_{u^{2[\varepsilon,i]}}(\boldsymbol{p})$ is equal to $P^2 + \{\varepsilon\boldsymbol{e}^i\}$. And it follows by assumption that $\boldsymbol{r}^1, \boldsymbol{r}^2$ that $P^1 \cap (P^2 + \{\varepsilon\boldsymbol{e}^i\}) \neq \emptyset$.

By Baldwin and Klemperer [2019, Proposition 2.7], P^1, P^2 are polyhedra in $\mathbb{R}^n$, so each may be written as those $\boldsymbol{p}$ such that:

$$H^j\boldsymbol{p} \geq A^j \tag{18}$$

where H^j is an $a_j \times n$ matrix for some $a_j \in \mathbb{Z}_+$ and $A^j \in \mathbb{R}^{a_j}$. Moreover by Baldwin and Klemperer [2019, Proposition 3.10], we can choose H^j such that each row of H^j is a strong substitutes vector (Definition 11) and then, as we assume each valuation u^j is integer-valued, also $A^j \in \mathbb{Z}^{a_j}$. Then $P^2 + \{\varepsilon\boldsymbol{e}^i\}$ is defined by

$$H^2(\boldsymbol{p} - \varepsilon\boldsymbol{e}^i) \geq A^2 \quad \Leftrightarrow \quad H^2\boldsymbol{p} \geq A^2 + \varepsilon H^2_i \tag{19}$$

where H_i^2 is the ith column of H^2.

Write $H = \begin{pmatrix} H^1 \\ H^2 \end{pmatrix}$ and $A = \begin{pmatrix} A^1 \\ A^2 \end{pmatrix}$, and $\hat{H} = \begin{pmatrix} 0 \\ H^2 \end{pmatrix}$, inequalities defining $P^1 \cap (P^2 + \{\varepsilon \boldsymbol{e}^i\})$ are:

$$H\boldsymbol{p} \geq A + \varepsilon \hat{H}_i \tag{20}$$

Now argue similarly to Theorem 19.1 in Schrijver [1998]. We know that $P^1 \cap (P^2 + \{\varepsilon \boldsymbol{e}^i\}) \neq \emptyset$. Let $F = \{\boldsymbol{p} : H'\boldsymbol{p} = A' + \varepsilon \hat{H}'_i\}$ be a minimal face of this intersection, where $H'\boldsymbol{p} \geq A' + \varepsilon \hat{H}'_i$ is a subsystem of (20) with linearly independent rows. After possibly permuting the columns of H', that is, the indices of the goods, we can write $H' = \begin{pmatrix} G & \tilde{G} \end{pmatrix}$ where G is invertible. Moreover we can chose this permutation such that the ith coordinate remains within the first square matrix – that is, if G is $n' \times n'$ then assume when we re-ordering that $i \leq n'$. Moreover, G inherits from H^1, H^2 the property that its rows are strong substitute vectors. Now define:

$$\boldsymbol{p}^\varepsilon \coloneqq \begin{pmatrix} G^{-1}\left(A' + \varepsilon \hat{H}'_i\right) \\ 0 \end{pmatrix} \text{ and } \boldsymbol{p}^0 \coloneqq \begin{pmatrix} G^{-1}A' \\ 0 \end{pmatrix}$$

Note $\boldsymbol{p}^0$ is an integer vector, since A is integer and G is unimodular. Moreover, we can apply Lemma 40: either $G^{-1}\hat{H}'_i = \boldsymbol{e}^S$ for some $S \subseteq [n']$, with $i \in S$ or $G^{-1}\hat{H}'_i = -\boldsymbol{e}^S$ for some $S \subseteq [n']$ with $i \notin S$ (here n' is the dimension of G). So, either $\boldsymbol{p}^\varepsilon = \boldsymbol{p}^0 + \varepsilon \boldsymbol{e}^S$ for some $S \subseteq [n]$ with $i \in S$ or $\boldsymbol{p}^\varepsilon = \boldsymbol{p}^0 - \varepsilon \boldsymbol{e}^S$ for some $S \subseteq [n]$ with $i \notin S$. Moreover, in either case, $\boldsymbol{p}^\varepsilon - \varepsilon \boldsymbol{e}^i \in \{\boldsymbol{p}^0 \pm \varepsilon \boldsymbol{e}^{S'}\}$ for some $S' \subseteq [n]$.

We also see immediately that $\boldsymbol{p}^\varepsilon$ is a vector in F, and hence in $P^1 \cap (P^2 + \{\varepsilon \boldsymbol{e}^i\})$. So (20) holds at $\boldsymbol{p}^\varepsilon$: $H\boldsymbol{p}^\varepsilon \geq A + \varepsilon \hat{H}_i$. We will show that also $\boldsymbol{p}^0 \in P^1 \cap P^2$.

First consider (18) for $j = 1$, which holds at $\boldsymbol{p}^\varepsilon$. Note also that H^1 has strong substitute rows and $(\boldsymbol{p}^0 - \boldsymbol{p}^\varepsilon) \in \pm\varepsilon \boldsymbol{e}^S$ for $S \subseteq [n]$, so applying Lemma 41 together with (18):

$$H^1\boldsymbol{p}^0 = H^1\boldsymbol{p}^\varepsilon + H^1(\boldsymbol{p}^0 - \boldsymbol{p}^\varepsilon) \geq A^1 - \varepsilon \mathbf{1}.$$

The rows of $H^1\boldsymbol{p}^0$ and A^1 are integer, and $\varepsilon \in (0, 1)$, so $H^1\boldsymbol{p}^0 \geq A^1$.

Next consider that (19) holds at $\boldsymbol{p}^\varepsilon$. Again H^2 has strong substitute rows and this time use that $(\boldsymbol{p}^0 - \boldsymbol{p}^\varepsilon + \varepsilon \boldsymbol{e}^i) \in \pm\varepsilon \boldsymbol{e}^{S'}$ for $S' \subseteq [n]$, so applying Lemma 41 together with (19):

$$H^2\boldsymbol{p}^0 = H^2(\boldsymbol{p}^\varepsilon - \varepsilon \boldsymbol{e}^i) + H^2(\boldsymbol{p}^0 - \boldsymbol{p}^\varepsilon + \varepsilon \boldsymbol{e}^i) \geq A^2 - \varepsilon \mathbf{1}.$$

Again, the rows of $H^2\boldsymbol{p}^0$ and A^2 are integer, and $\varepsilon \in (0, 1)$, so $H^2\boldsymbol{p}^0 \geq A^2$. Thus we have established that $\boldsymbol{p}^0 \in P^1 \cap P^2$, as required to show that $\boldsymbol{r}^j \in D_{u^j}(\boldsymbol{p}^0)$ for $j = 1, 2$.

Now take any $\boldsymbol{p}$ such that $\boldsymbol{r} \coloneqq \boldsymbol{r}^1 + \boldsymbol{r}^2 \in D_{u^{\{1,2\}}}(\boldsymbol{p})$, and suppose that $\boldsymbol{s}^j \in D_{u^j}(\boldsymbol{p})$ for $j = 1, 2$ with $\boldsymbol{r} = \boldsymbol{s}^1 + \boldsymbol{s}^2$: we have an equilibrium with potentially a different allocation of bundles across the two agents. By Mas-Colell et al. [1995, Proposition 2.F.1] we know that for $j = 1, 2$ we have $(\boldsymbol{s}^j - \boldsymbol{r}^j) \cdot (\boldsymbol{p} - \boldsymbol{p}^0) \leq 0$, with equality holding only if the agent is indifferent between both bundles at both prices. So, adding across $j = 1, 2$, we obtain $((\boldsymbol{s}^1 + \boldsymbol{s}^2) - \boldsymbol{r}) \cdot (\boldsymbol{p} - \boldsymbol{p}^0) \leq 0$, with equality holding only if the two agents are both indifferent between both bundles at both prices. But as $\boldsymbol{s}^1 + \boldsymbol{s}^2 = \boldsymbol{r}$ it must indeed follow that $((\boldsymbol{s}^1 + \boldsymbol{s}^2) - \boldsymbol{r}) \cdot (\boldsymbol{p} - \boldsymbol{p}^0) = 0$ and so in particular that also $rb^j \in D_{u^j}(\boldsymbol{p})$ for $i = 1, 2$, as required. □

Proof of Proposition 14. Re-label so that bidder j has valuation u^2 and the aggregate valuation from all remaining agents is u^1. We know that $\boldsymbol{r} \in D_{u^{\{1,2\}}}(\boldsymbol{p})$ by definition of an allocation problem.

As in Proposition 42 we write $u^{2[\varepsilon,i]}$ for the valuation given by $u^{2[\varepsilon,i]}(\boldsymbol{x}) = u^2(\boldsymbol{x}) + \varepsilon \boldsymbol{e}^i \cdot \boldsymbol{x}$. Since both u^1 and $u^{2[\varepsilon,i]}$ are strong substitute valuations, there exists a price $\boldsymbol{p}^* \in \mathbb{R}^n$ such that $\boldsymbol{r} \in D_{u^{1,2[\varepsilon,i]}}(\boldsymbol{p}^*)$, and thus there exist $\boldsymbol{r}^1, \boldsymbol{r}^2 \in \mathbb{Z}^n$ such that $\boldsymbol{r}^1 + \boldsymbol{r}^2 = \boldsymbol{r}$ and such that $\boldsymbol{r}^2 \in D_{u^1}(\boldsymbol{p}^*)$, $\boldsymbol{r}^2 \in D_{u^{2[\varepsilon,i]}}(\boldsymbol{p}^*)$. By Proposition 42 it follows that $\boldsymbol{r}^j \in D_{u^j}(\boldsymbol{p})$ for $j = 1, 2$.

As in Proposition 42 write P^j for the set of prices $\boldsymbol{p}'$ at which $\boldsymbol{r}^j \in D_{u^j}(\boldsymbol{p}')$, and observe that the set of prices $\boldsymbol{p}'$ at which $\boldsymbol{r}^2 \in D_{u^{2[\varepsilon,i]}}(\boldsymbol{p}')$ is equal to $P^2 + \{\varepsilon \boldsymbol{e}^i\}$. We know that $\boldsymbol{p} \in P^1 \cap P^2$ (so this is non-empty) and $\boldsymbol{p}^* \in P^1 \cap (P^2 + \{\varepsilon \boldsymbol{e}^i\})$ (so this is non-empty). We seek $\boldsymbol{p}^\varepsilon \in P^1 \cap (P^2 + \{\varepsilon \boldsymbol{e}^i\})$ with $\boldsymbol{p}^\varepsilon \in \boldsymbol{p} \pm \varepsilon \boldsymbol{e}^S$ for some $S \subseteq [n]$; this is sufficient for $\boldsymbol{r} \in D_{u^{\{1,2[\varepsilon,i]\}}}(\boldsymbol{p}^\varepsilon)$, as we require.

As at line (18) above we express both polyhedra as $H^i \boldsymbol{p}' \geq A^j$, where rows of H^j are strong substitutes vectors. Therefore $P^1 \cap (P^2 + \{\varepsilon \boldsymbol{e}^i\})$ is defined by the set of inequalties:

$$H^1 \boldsymbol{p}' \geq A^1 \text{ and } H^2(\boldsymbol{p}' - \varepsilon \boldsymbol{e}^i) \geq A^2 \tag{21}$$

Now, for $j = 1, 2$, let

$$(H^j)' \boldsymbol{p}' \geq (A^j)' \tag{22}$$

be the subsystem of $H^j \boldsymbol{p}' \geq A^j$ consisting of all rows which hold with equality at $\boldsymbol{p}$. (If both of these subsystems are empty then subsequent arguments will simply set $\boldsymbol{p}^\varepsilon = \boldsymbol{p}$). Write $H = \begin{pmatrix} (H^1)' \\ (H^2)' \end{pmatrix}$, $A = \begin{pmatrix} (A^1)' \\ (A^2)' \end{pmatrix}$ and $\hat{H} = \begin{pmatrix} 0 \\ (H^2)' \end{pmatrix}$. Thus when we stack the corresponding rows of (21) we obtain the new system:

$$H \boldsymbol{p}' \geq A + \varepsilon \hat{H}_i. \tag{23}$$

This defines a superset of $P^1 \cap (P^2 + \varepsilon)$, since we have already removed some rows from $H^j \boldsymbol{p}' \geq A^j$ for $j = 1, 2$. But $\boldsymbol{p}^* \in P^1 \cap (P^2 + \varepsilon)$ so (23) defines a non-empty polytope. So let $F = \{\boldsymbol{p}' : H' \boldsymbol{p}' = A' + \varepsilon \hat{H}'_i\}$ be a minimal face of this polytope, where

$$H' \boldsymbol{p}' \geq A' + \varepsilon \hat{H}'_i \tag{24}$$

is a subsystem of (23) with linearly independent rows.

After possibly permuting the columns of H', that is, the indices of all goods, we can write $H' = \begin{pmatrix} G & \tilde{G} \end{pmatrix}$, where G is invertible. Moreover we can choose this perumtation such that the ith coordinate remains within the first square matrix – that is, if G is $n' \times n'$ then assume when we re-order that $i \leq n'$. Correspondingly write $\boldsymbol{p}' = \begin{pmatrix} \boldsymbol{p}'_1 \\ \boldsymbol{p}'_2 \end{pmatrix}$ So our system (23) is now written:

$$\begin{pmatrix} G & \tilde{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}'_1 \\ \boldsymbol{p}'_2 \end{pmatrix} \geq A' + \varepsilon \hat{H}'_i. \tag{25}$$

Observe that, by definition of the rows we identified at (22), we have $\begin{pmatrix} G & \tilde{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \end{pmatrix} = A'$ and so $\boldsymbol{p}_1 = G^{-1} A' - G^{-1} \tilde{G} \boldsymbol{p}_2$. Write now:

$$\boldsymbol{p}^\varepsilon := \begin{pmatrix} G^{-1}(A' + \varepsilon \hat{H}'_i) - G^{-1} \tilde{G} \boldsymbol{p}_2 \\ \boldsymbol{p}_2. \end{pmatrix} \tag{26}$$

so that

$$H' \boldsymbol{p}^\varepsilon = \begin{pmatrix} G & \tilde{G} \end{pmatrix} \boldsymbol{p}^\varepsilon = A' + \varepsilon \hat{H}'_i. \tag{27}$$

We wish to show that (21) holds for $\boldsymbol{p}' = \boldsymbol{p}^\varepsilon$. We can immediately observe that $\boldsymbol{p}^\varepsilon - \boldsymbol{p} = \begin{pmatrix} \varepsilon G^{-1} \hat{H}'_i \\ 0 \end{pmatrix}$. So, by Lemma 40, either $\boldsymbol{p}^\varepsilon = \boldsymbol{p} + \varepsilon \boldsymbol{e}^S$ for some $S \subseteq [n]$ with $i \in S$, or $\boldsymbol{p}^\varepsilon = \boldsymbol{p} - \varepsilon \boldsymbol{e}^S$ where $S \subseteq [n]$ with $i \notin S$. Moreover, in either case, $\boldsymbol{p}^\varepsilon - \varepsilon \boldsymbol{e}^i \in \{\boldsymbol{p} \pm \varepsilon \boldsymbol{e}^{S'}\}$ for some $S' \subseteq [n]$.

We also immediately observe that $\boldsymbol{p}^\varepsilon$ satisfies (24) with equality, and so $\boldsymbol{p}^\varepsilon \in F$. Thus by definition of F, also $\boldsymbol{p}^\varepsilon$ satisfies (23). It follows by definition of (22) that, for every row in the original systems $H^j \boldsymbol{p}' \geq A^j$ that holds with equality at $\boldsymbol{p}$, the corresponding shifted equation in (21) holds at $\boldsymbol{p}^\varepsilon$. It remains to show that this is also true for rows that are slack at $\boldsymbol{p}$. Let

$(H^j)''\boldsymbol{p}' \geq (A^j)''$ be the subsystems of such rows, for $j = 1, 2$, and observe that by integrality it follows that

$$(H^j)''\boldsymbol{p} \geq (A^j)'' + \mathbf{1}. \tag{28}$$

Now, applying Lemma 41 and (28) for $j = 1$, we obtain:

$$(H^1)''\boldsymbol{p}^\varepsilon = (H^1)''\boldsymbol{p} + (H^1)''(\boldsymbol{p}^\varepsilon - \boldsymbol{p}) \geq (A^1)'' + (1-\varepsilon)\mathbf{1} \geq (A^1)''.$$

And applying Lemma 41 to $\boldsymbol{p}^\varepsilon - \boldsymbol{p} - \varepsilon\boldsymbol{e}^i$, and (28) for $j = 2$

$$(H^2)''(\boldsymbol{p}^\varepsilon - \varepsilon\boldsymbol{e}^i) = (H^2)''\boldsymbol{p} + (H^2)''(\boldsymbol{p}^\varepsilon - \boldsymbol{p} - \varepsilon\boldsymbol{e}^i) \geq (A^2)'' + (1-\varepsilon)\mathbf{1} \geq (A^2)''.$$

So, for every row in the original systems $H^j\boldsymbol{p}' \geq A^j$ that is slack at $\boldsymbol{p}$, the corresponding shifted equation in (21) holds for $\boldsymbol{p}^\varepsilon$. This completes the proof that $\boldsymbol{p}^\varepsilon \in P^1 \cap (P^2 + \{\varepsilon\boldsymbol{e}^i\})$.

It follows that $\boldsymbol{r} \in D_{u^{1,2[\varepsilon,i]}}(\boldsymbol{p}^\varepsilon)$, which was constructed to be positioned relative to $\boldsymbol{p}$ as described in the proposition.

Finally, recall that the prices at which $\boldsymbol{r}$ is demanded are equivalent to the prices that minimise the Lyapunov function $g(\boldsymbol{p}) = f_{\mathcal{B}}(\boldsymbol{p}) + \boldsymbol{r}\cdot\boldsymbol{p}$. It is known that $f_{\mathcal{B}}$ (and thus g) are submodular [Murota and Shioura, 2014, Theorem 7.2], so

$$h^+(S) \coloneqq g(\boldsymbol{p} + \varepsilon\boldsymbol{e}^S) - g(\boldsymbol{p}), h^-(S) \coloneqq g(\boldsymbol{p} + \varepsilon\boldsymbol{e}^S) - g(\boldsymbol{p}),$$

are submodular set functions. Hence in order to determine a price $\boldsymbol{p}^\varepsilon \in \left\{\boldsymbol{p} + \varepsilon\boldsymbol{e}^S \mid S \subseteq [n]\right\}$ at which $\boldsymbol{r}$ is demanded, we find minimisers S^+ and S^- of h^+ and h^- using SFM, then let

$$\boldsymbol{p}^\varepsilon = \arg\min\left\{g\left(\boldsymbol{p} + \varepsilon\boldsymbol{e}^{S^+}\right), g\left(\boldsymbol{p} - \varepsilon\boldsymbol{e}^{S^-}\right)\right\}.$$

Note that we do not require *minimal* submodular minimisers to find $\boldsymbol{p}^\varepsilon$, and so this step takes $2T(n)$ time, where $T(n)$ is the time is takes to perform submodular minimisation on $h^\pm$. We also note that finding $\boldsymbol{p}^\varepsilon$ is analogous to finding a steepest descent direction in Appendix B. □

D.4 Proof of Lemma 17.

Here we prove that applying ShiftProjectUnshift strictly reduces the number of edges in the marginal bids graph $G_{\mathcal{A}}$.

Proof of Lemma 17. First we show that every edge in $G_{\mathcal{A}'}$ is present in $G_{\mathcal{A}}$. To see this, fix some bid $\boldsymbol{b}$ and note by Observation 7 and Lemma 16 that Steps 1, 2 and 3 do not make $\boldsymbol{b}$ marginal on any new goods.

Secondly we prove that for any multi-bidder cycle C with cycle-link good i^* and incident label j^*, ShiftProjectUnshift removes at least one of the edges of C from the marginal bids graph. Re-label the goods going around cycle C as $1, \ldots, k$, so that $1 = i^*$ and so that bidder j^* placed the bid marginal between goods 1 and 2. Also, for convenience, index the marginal bid that is marginal between goods i and $i+1$ as 'i', and index the marginal bid between goods k and 1 as 'k'. Thus $\boldsymbol{b}^1 \in \mathcal{B}^{j^*}$ and $\boldsymbol{b}^k \notin \mathcal{B}^{j^*}$. (In general the differently labelled bids need not be from different bidders). The existence of a marginal bid between goods i and $i+1$ means that we have an equality

$$b^i_i - p_i = b^i_{i+1} - p_{i+1} \tag{29}$$

for $i = 1, \ldots, k-1$, and also

$$b^k_k - p_k = b^k_1 - p_1 \tag{30}$$

But if we take the sum of the first $k-1$ equations, and cancel, we find

$$\sum_{i=1}^{k-1} b^i_i - \sum_{i=1}^{k-1} b^i_{i+1} = \sum_{i=1}^{k-1}(p_i - p_{p+1}) = p_1 - p_k$$

So it must hold that

$$\sum_{i=1}^{k-1}(b_i^i - b_{i+1}^i) = b_1^k - b_k^k. \tag{31}$$

For any bid $\boldsymbol{b}$ in the bid lists $(\mathcal{B}^j)_{j\in J}$, fixed $j^* \in J$ and $i^* \in [n]$, let

$$\tilde{\boldsymbol{b}} \coloneqq \begin{cases} \boldsymbol{b} + \frac{1}{10}\boldsymbol{e}^{i^*} & \boldsymbol{b} \in \mathcal{B}^{j^*} \\ \boldsymbol{b} & \text{otherwise} \end{cases}$$

Suppose we replace each bid $\boldsymbol{b}$ by $\tilde{\boldsymbol{b}}$ as above and recompute the prices. Since prices shift by at most $\frac{1}{10}$, a bid that is *not* marginal on a pair of goods, cannot *become* marginal on them. If we assume for a contradiction that all edges in C are all still present in G' then we can write down similar expressions to (29) and (30) (w.r.t. new prices), then eliminate those prices obtaining a version of (31), namely $\sum_{i=1}^{k-1}\left(\tilde{b}_i^i - \tilde{b}_{i+1}^i\right) = \tilde{b}_1^k - \tilde{b}_k^k$. But by definition of $\tilde{\boldsymbol{b}}$, we know that

$$\sum_{i=1}^{k-1}(\tilde{b}_i^i - \tilde{b}_{i+1}^i) = \sum_{i=1}^{k-1}(b_i^i - b_{i+1}^i) + \frac{1}{10},$$
$$\text{and } \tilde{b}_1^k - \tilde{b}_k^i = b_1^k - b_k^k.$$

This is inconsistent with (31), so we have the required contradiction. □