Solving product-mix markets and learning agents' preferences



Edwin Lock

Merton College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2021

To my father

Acknowledgements

Firstly, I thank Paul Goldberg, my supervisor, for providing invaluable advice and support throughout my doctoral journey. It has been a privilege to work under your guidance. I am also profoundly grateful to Paul Klemperer and Elizabeth Baldwin for introducing me to the product-mix auction (through Paul Goldberg), inspiring me through our collaborations and, more generally, encouraging me to work in econCS.

Going back in time, I would like to highlight the crucial role that my good friends Austin Kinross and Ventsislav Chonev played in inspiring me early on, during my music days, to study computer science and mathematics. In a sense, this thesis is a consequence of your doing.

My 'second stint' at Oxford would not have been the same without my friends at Merton and beyond. Valerian Hall-Chen, Jakob Jonnerby, and all my other fellow Mertonians have all made their own, unforgettable mark on the last four years. León Brandt, for being an excellent friend, and for introducing me to the world of half-price sushi, I cannot be more grateful.

I would also like to thank the theoretical computer science gang, and all my Oxford collaborators, for their inspiration and for sharing glimpses of their brilliant minds. Special mentions go out to Simon Finster and Francisco Marmolejo Cossío, who have enriched my life with friendship and research.

I am incredibly grateful to my mother, father and sister for their unquestioning support throughout. Without the encouragement of my parents, I would not have made it this far.

To Divya Sridhar, my wonderful wife, I owe a special debt of gratitude. This thesis would not have been possible without you. Thank you for supporting me through this journey with all its ups and downs.

Abstract

This thesis addresses computational questions arising in auctions with multiple goods available in multiple quantities, with a focus on establishing the tractability of auction mechanisms used in practice. It presents algorithms and hardness results for solving these auctions with the goal of maximising social welfare or revenue, and develops procedures to facilitate participating agents in expressing their preferences in the auction's bidding language.

The 'strong-substitutes product-mix auction' was originally designed for the Bank of England by Paul Klemperer and continues to be run at least monthly. It introduces a novel bidding language, which allows agents to submit sealed-bid strong-substitutes preferences over an arbitrary number of goods available in multiple discrete units. We study this language geometrically and computationally, and establish connections to related bidding languages in the literature. In order to solve the strong-substitutes product-mix auction for social welfare, we develop the first efficient algorithms to find market-clearing prices and envy-free allocations of supply to participating bidders. The latter, in particular, requires solving a novel constrained matching problem. By contrast, we show that solving the auction for maximum revenue is APX-hard even in special cases, and present initial algorithms for this.

Agents participating in this auction may have trouble expressing their preferences in the auction's bidding language when faced with a large number of goods or non-trivial demand. Instead, they may be able to answer queries about their value of a given bundle or their demand at given prices. We propose algorithms that learn a bidder's preferences, assuming access to a demand or valuation oracle. In a special case currently implemented by the Bank of England, we present a linear-time algorithm. We also show super-polynomial lower bounds on the query complexity in the general case. Our algorithms constitute the first approach for bidders to express non-trivial preferences in these languages, lowering the barrier for participation in the strong-substitutes product-mix auction.

In the related 'arctic product-mix market', originally designed for the Icelandic government by Klemperer, buyers use a conceptually similar bidding language to express preferences across multiple *divisible* goods in conjunction with monetary budgets. In this setting we present a new coincidence of the solution concepts of competitive equilibrium and optimal revenue: market-clearing prices are unique, and these prices maximise not only social welfare but also revenue. We also provide an algorithm to identify these prices.

Contents

1 Introduction **1**

1.1 Background 1
1.2 Thesis overview 5
1.3 The market model 6
1.4 Demand types 11

2 Bidding languages **18**

2.1 Chapter outline 18
2.2 Introduction 19
2.3 Languages in the literature 20
2.4 The logical languages 22
2.5 The tropical language 24
2.6 The arctic language 40
2.7 Conclusion and outlook 42

3 Solving the strong-substitutes product-mix auction **44**

3.1 Chapter outline 44
3.2 Introduction 45
3.3 The computational challenges 48
3.4 Finding market-clearing prices 51
3.5 Allocations to the individual bidders 58
3.6 Experiments 79
3.7 Conclusion and outlook 81

4 Maximising revenue in the strong-substitutes product-mix auction **83**

4.1 Introduction 83
4.2 Problem statement 84
4.3 The hardness of solving for revenue 89
4.4 Algorithms for maximising revenue 92
4.5 Conclusion and outlook 96

5 Learning strong-substitutes demand **97**
5.1 Chapter outline . . . 97
5.2 Introduction . . . 98
5.3 The geometry of tropical bids . . . 101
5.4 The computational challenges . . . 103
5.5 Simulating $\mathcal{Q}_{\mathcal{B}}$ with a valuation oracle . . . 104
5.6 Learning positively-weighted bids . . . 104
5.7 Learning positive and negative bids . . . 108
5.8 Lower bounds . . . 114
5.9 Conclusion and outlook . . . 120

6 Solving the arctic product-mix auction **122**
6.1 Chapter outline . . . 122
6.2 Introduction . . . 123
6.3 Preliminaries . . . 126
6.4 Efficiency and competitive equilibrium . . . 128
6.5 Element-wise minimal prices . . . 129
6.6 Maximising revenue . . . 131
6.7 Market-clearing prices are unique . . . 132
6.8 Minimal feasible prices clear the market . . . 133
6.9 Finding market-clearing prices . . . 139
6.10 Conclusion and outlook . . . 140

7 Concluding remarks **142**

Bibliography **145**

1 Introduction

Contents

1.1 Background 1
1.1.1 The product-mix auctions 4
1.2 Thesis overview 5
1.3 The market model 6
1.3.1 Bidder preferences 7
1.3.2 Solution concepts 8
1.3.3 Demand and valuation oracles 9
1.4 Demand types 11
1.4.1 Strong-substitutes demand 12
1.4.2 Discrete convex analysis 14
1.4.3 Tropical geometry 15

1.1 Background

Auction markets have long become indispensable for driving revenue and effecting social welfare. They lie at the heart of stock exchanges (Friederich and Payne, 2007; Hasbrouck et al., 1993), the allocation of broadcasting spectrum for telecommunication (Bichler, 2017; Bichler and Goeree, 2017; Cramton et al., 2002), the efficient (or socially optimal) use of natural resources (Cramton, 2009; Teytelboym, 2019), ad auctions (Conitzer et al., 2017, 2019; Varian, 2009), and improving liquidity of banks (Klemperer, 2008, 2010). In these markets, agents with various buying and selling preferences are typically coordinated by a central actor who sets market prices with the goal of maximising revenue or social welfare.[1]

[1]Other objectives or solution concepts, such as Nash equilibrium, with and without a central auctioneer are also often considered in combinatorial auction formats, but are beyond the scope of this thesis.

For centuries, auction markets have been prized for their ability to discover prices at which supply meets demand (Cassady, 1967; Milgrom, 1987). This is perfectly encapsulated by the classic English auction, in which bidders openly announce increasingly higher bids for a single item. The highest bidder then pays their bid to receive the item. The Vickrey auction (or second-price auction), another auction on a single item, asks bidders to submit sealed bids before the auction is run, and then allocates the item to the bidder with the highest bid, but at the price of the second-highest bid. In contrast with the English auction, the Vickrey auction incentivises bidders to participate truthfully, that is to report their values accurately. Yet auction theory has long progressed beyond the classic English and Vickrey auctions. Auction formats typically incorporate multiple indistinguishable items of a good (multi-unit auctions), or multiple distinct goods available simultaneously in one or more items. This introduces additional complexities on the side of buyers and sellers, as an agent's value for an item may depend on the quantities of other items that it receives or sells. For example, a box of cereal may only have value to an agent if they also receive milk. Hence, these goods are complementary. Some goods also exhibit substitutes relations: an agent may buy a car, or a yearly train subscription, to commute, but not both. In order to capture such combinatorial relations in preferences, bidders (and sellers) are typically permitted to place bids on bundles (or packages) of goods to capture complements and combine multiple bids with 'either-or' relationships to reflect substitutes, allowing agents to submit preferences that more accurately mirror real-world demand. For this reason, the combinatorial auction format, in which multiple goods are available in single indivisible units, has been much studied over the last decades (Colini-Baldeschi et al., 2017; Cramton et al., 2007; De Vries and Vohra, 2003; Lehmann et al., 2002).

When goods are divisible, general equilibrium theory has established many fundamental results that illuminate the deep links between competitive equilibrium and social welfare. The Arrow-Debreu model tells us that, under certain reasonable assumptions (convex preferences, perfect competition and demand independence between bidders), competitive equilibrium (also known as Walrasian equilibrium) prices are guaranteed to exist; that is, prices at which supply can be divided between buyers so that each buyer receives quantities of goods they demand at these prices (Arrow and Debreu, 1954). This work has led to the seminal fundamental theorems of welfare economics: applied to auctions, the first theorem establishes that any competitive equilibrium prices support a Pareto-efficient allocation of supply to agents, while the second theorem conversely states that any Pareto-efficient allocation is supported by some market-clearing prices. However, these results, which were established within the Arrow-Debreu model, do not generally carry over to settings with indivisible goods and non-convex preferences. As auctions with indivisible goods have been proposed, and run, in settings including financial, electricity and environmental markets, the questions of existence and computation of competitive equilibria in such markets has received considerable attention over the last few decades (Ausubel, 2004, 2006; Baldwin and Klemperer, 2019; Bikhchandani and Mamer, 1997; Bikhchandani and Ostroy, 2002; Finster, 2020b; Paes Leme and Wong, 2017). This body of work has established deep connections with optimisation

theory, and utilises a diverse set of techniques from operations research, discrete convex analysis (Murota, 2003) and, more recently, tropical geometry (Baldwin and Klemperer, 2019). Moreover, auctions are only applicable in practice if they can be solved computationally in a 'reasonable' amount of time. The development of efficient algorithms to approximately or exactly solve auctions, or computational lower bounds that establish the impossibility of such algorithms, has long drawn the attention of economists and computer scientists alike (see, for instance, (Ausubel, 2004, 2006; Baldwin et al., 2021a; Bichler et al., 2021; Bikhchandani et al., 2001; Fujishima et al., 1999; Garg and Kapoor, 2006; Guruswami et al., 2005; Murota et al., 2016; Nisan, 2000; Orlin, 2010; Paes Leme, 2017; Sandholm, 1999)). The question of solving the product-mix auctions for social welfare and revenue is the subject of Chapters 3, 4 and 6.

In the setting where multiple goods are available in single indivisible units, Kelso and Crawford (1982) show that competitive equilibrium is guaranteed to exist if agents have gross substitutes valuations. When we allow multiple (indivisible) units per good, gross substitutes can be extended to either general substitutes, or the more restrictive strong-substitutes preferences. The latter is appealing because it guarantees existence of market-clearing prices, while the former does not. Section 1.4.1 introduces both notions of substitutes and provides more detail. For recent surveys on Walrasian equilibria and substitutes preferences, we also refer to (Paes Leme, 2017) and (Bichler et al., 2021). Moreover, it is known that competitive equilibrium may not exist when agents have complements preferences (Gul and Stacchetti, 1999; Kelso and Crawford, 1982). However, some work on auctions which allow limited complementarity for agents and support competitive equilibrium exists (Finster, 2020b; Sun and Yang, 2006, 2014; Teytelboym, 2014).

The type of market with which we concern ourselves in this thesis are the product-mix markets, or auctions, developed by Klemperer (2008, 2010), which can be seen as combinatorial auctions in which each good may be available in multiple indistinguishable units. In the arctic product-mix auction, goods are divisible, while the strong-substitutes product-mix auction is designed to allocate indivisible goods. Both product-mix auctions implement a single-round sealed-bid mechanism, which requires agents to submit substitutes bidding preferences before the auction is run. (The central 'auctioneer' runs the auction by setting appropriate uniform market prices and allocating supply.) This is a convenient mechanism which avoids repeated preference elicitation and accompanying price adjustments over time. In addition, sealed-bid mechanisms, which eliminate iterative bidding, may reduce the risk of strategic behaviour by agents (along with inefficiencies this may incur), which has been observed in many sequential auctions such as simultaneous multi-round auctions (SMRAs), and spectrum auctions specifically (Bichler, 2017; Kroemer et al., 2016).[2] Consequently, considerations such as incentive compatibility or the ability of agents to influence prices ('price-making') are desiderata we do not address in this work. We assume that all agents are price-taking and must thus accept prevailing prices, with no opportunity (or motivation) to influence market prices.

[2]In practice, bidders often fail to exhibit perfectly rational behaviour, and sealed bid mechanisms may also lead to overbidding and other 'strategic' actions.

Moreover, our assumption is that bidders will reveal their preferences truthfully. This assumption is reasonable when the number of bidders is large, as is the case in current applications of the product-mix auction. However, in certain settings, e.g. when bidders have asymmetric endowments or the number of agents is small, agents may be able to influence the outcome of the auction by communicating strategically, and so the product-mix auctions, like many multi-good auctions, are not incentive compatible. We refer to (Holmberg et al., 2019) and (Finster, 2020a) for economic studies on strategic behaviour in multi-good auctions, and product-mix auctions in particular. We also emphasise that auctions have been studied extensively from the perspective of mechanism design and algorithmic game theory. Under the assumption of private valuations, for instance, a primary goal is to design auction mechanisms that achieve a subset of the following objectives: incentive-compatibility (ensuring truthful reporting of preferences), efficiency (maximising welfare), maximising revenue, Nash equilibrium and envy-freeness (ensuring that no agent prefers another agent's allocation) (Branzei et al., 2014; Menezes and Monteiro, 2004; Nisan et al., 2007). In these settings, auctions may be interactive, in that agents might act and react in response to observations about other agents' behaviour. The auctions we study in this thesis preclude this kind of behaviour, as bidders submit preferences as sealed bids and are not permitted to revise their submissions.

Sealed-bid mechanisms assume that agents communicate their buying or selling preferences in a systematic fashion. When multiple items of distinct goods are available, the space of preferences grows exponentially, which raises questions about efficient ways to encode, and verify sealed bids. Various bidding languages have been proposed in the literature to allow bidders to express preference classes of varying complexity (Klemperer, 2008; Lehmann et al., 2006, 2002; Milgrom, 2009; Nisan, 2000). Chapter 2 introduces these languages in some detail, and establishes initial connections between different languages, and some limitations of each. More generally, when agents do not submit sealed bids but provide oracle access to a demand or valuation oracle, a growing body of literature has studied upper and lower bounds on the communication complexity of solving a market (e.g. combinatorial auctions and two-sided matching markets) with various demand types (Ashlagi et al., 2020; Dobzinski and Vondrák, 2013; Nisan and Segal, 2006) with regard to maximising revenue or social welfare. As the product-mix auctions ask bidders to submit sealed bids, this body of work is not directly applicable to these auctions. However, in Chapter 5 we make initial steps towards bridging this gap by studying the communication complexity of eliciting strong-substitutes sealed bids (expressed in the 'tropical' bidding language) from agents purely through valuation or demand queries.

1.1.1 The product-mix auctions

The family of product-mix auctions was designed by Klemperer (2008) following the financial crisis of 2007-8. The strong-substitutes product-mix auction (Klemperer, 2008, 2010, 2018) was devised as a means of providing liquidity to commercial banks and has been used regularly by the Bank of England since 2011. In it, there are a

number of distinct goods available in multiple indivisible units, and a set of buyers who express strong-substitutes demands amongst these goods. In the banking context, the goods correspond to liquidity secured against alternative kinds of collateral. Commercial banks pay for liquidity 'products' by committing to interest rates. The values of the bids submitted by the commercial banks correspond to the interest rates they are willing to pay. In this auction, the objective pursued by the auctioneer is to maximise social welfare by solving for competitive equilibrium.

In contrast, the arctic product-mix auction (also known as the Icelandic auction) was devised by Klemperer for the government of Iceland to exchange blocked accounts for other financial assets, e.g. cash or bonds with different properties, which were available only in limited quantities. Unlike in the strong-substitutes product-mix auction, bidders submit preferences for *divisible* goods and can express budgets as well as trade-offs between different assets through bid prices. The objective of this auction, as originally proposed, was to maximise revenue. A variant of the arctic auction has also been suggested as a novel sovereign debt restructuring mechanism to the International Monetary Fund in response to a surge in public debt due to the COVID-19 pandemic[3].

The product-mix auctions are novel in that they allow bidders to express buying preferences for multiple goods in natural bidding languages. The product-mix languages encode agent preferences in a concise fashion and are described in detail in Chapter 2. This provides our algorithms for solving the auctions (cf. Chapters 3, 4 and 6) with complete, and computationally efficient, access to the information required for computing the uniform market prices that solve the auctions. While the original product-mix auctions were designed with a specific objective in mind, both can be studied fruitfully from the perspective of social welfare *and* revenue. Moreover, while the bidding languages are primarily intended for bidders to express substitutes preferences, the auction also allows for the auctioneer to set reserve prices by submitting 'dummy' bids in the same language.

1.2 Thesis overview

This thesis presents computational results that allow general product-mix auctions to be implemented in practice. We propose algorithms for solving both product-mix auctions with the goal of maximising social welfare or revenue, and develop procedures to facilitate agents in participating by systematically learning their preferences. We now give a brief overview of the contributions in each chapter of this thesis. In every chapter, we provide a literature review and sketch out future work. We conclude this thesis in Chapter 7. The remainder of this chapter introduces the underlying market model and defines demand types, with a focus on strong-substitutes preferences.

Chapter 2 introduces the arctic and tropical product-mix bidding languages, respectively used in the strong-substitutes product-mix auction and arctic auction,

[3]https://voxeu.org/article/proposal-auction-based-sovereign-debt-restructuring-mechanism

and establishes novel connections between these and other bidding languages. We also set out computational challenges that our bidding languages induce, which frames the research questions motivating the remainder of this thesis, and show that verifying the validity of bidder preferences expressed in the tropical product-mix bidding language is computationally hard; this hardness result can be found in (Baldwin et al., 2021b).

Chapter 3 presents efficient algorithms for solving the general strong-substitutes product-mix auction for competitive equilibrium, thereby maximising social welfare. Current implementations of this auction only permit bidders to submit restrictive preferences, on the basis of which the auction can be solved using linear programs. Allowing bidders to express arbitrary strong-substitutes demand introduces additional complexities which required the development of a novel algorithm. This work appears in (Baldwin et al., 2021b) and is joint work with Elizabeth Baldwin, Paul Goldberg and Paul Klemperer.

Chapter 4 studies the strong-substitutes product-mix auction from the perspective of maximising revenue. We prove that maximising revenue in this setting is computationally hard (APX-hard) even when restricting bidders to significantly constrained preferences. Moreover, we present simple algorithms that run in super-polynomial time, which may constitute building blocks for future, more sophisticated algorithms.

Chapter 5. While the tropical and arctic product-mix bidding languages are intuitive geometrically, bidders may find it difficult to express non-trivial preferences in these, especially when the number of goods available is high. In Chapter 5, we concern ourselves with expressing demand in the tropical product-mix bidding language. We present algorithms to do so, assuming access to a demand or valuation oracle, and prove a super-polynomial query complexity for this problem using an adversarial game setup. This work has been published in (Goldberg et al., 2020) and is joint work with Paul Goldberg and Francisco Marmolejo-Cossío.

Chapter 6. The final chapter examines the related arctic auction on divisible goods under the objectives of revenue, welfare and competitive equilibrium. Our main result is to establish a novel coincidence of these solutions concepts: the unique market-clearing prices also maximise revenue. We also provide an algorithm to find the market-clearing prices. These results are taken from (Finster et al., 2021), co-authored with Simon Finster and Paul Goldberg.

1.3 The market model

In this thesis, we work in an auction market model with n distinct goods denoted $[n] := \{1, \ldots, n\}$ that are to be allocated to bidders $[m]$. These goods may be available, and demanded, in arbitrary quantities (which we call the *multi-unit*

setting). This generalises the *single-unit* regime commonly assumed in the literature, in which multiple goods are available in a single indivisible quantity. Moreover, we study the cases that goods are divisible and indivisible. In the indivisible case, a single copy of a good is called an *item* or *unit*.

A *bundle* of goods, typically denoted $\boldsymbol{x}$ or $\boldsymbol{y}$ in this work, is a vector in $\mathbb{R}^n_+$ whose i-th entry denotes the quantity of good i in the bundle. In our setting, the auctioneer wishes to allocate (a subset of) his *supply bundle* $\boldsymbol{s} \in \mathbb{N}^n$ to multiple bidders participating in the auction. He does this by setting uniform market prices, which consist of price vectors with a price entry for each of the n goods and are typically denoted $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n$. For any vectors $\boldsymbol{v}, \boldsymbol{w}$ (bundles, prices, or otherwise), we write $\boldsymbol{v} \leq \boldsymbol{w}$ when the inequality holds element-wise.

Occasionally, we will see that it is convenient to work with a notional *reject good* 0. The set of goods is then $[n]_0 := \{0, \dots, n\}$ and we identify bundles and prices with the $(n+1)$-dimensional vectors obtained by adding a 0-th entry of value 0 (in the indivisible case) or 1 (in the divisible case).

Additional notation. For any subset $X \subseteq [n]$, we let $\boldsymbol{e}^X$ denote the characteristic vector of X, i.e. an n-dimensional vector whose i-th entry is 1 if $i \in X$, and 0 otherwise. Furthermore, $\boldsymbol{e}^i$ denotes the vector whose i-th entry is 1 while all other entries are 0. We define $\boldsymbol{e}^0 = \boldsymbol{0}$ for notational convenience. The dot product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is written as $\boldsymbol{a} \cdot \boldsymbol{b}$. We make use of implicit summation to simplify notation. Hence, for any function $f : A \to \mathbb{R}$, we let $f(A') = \sum_{a \in A'} f(a)$ for any $A' \subseteq A$. This extends to functions in two arguments as follows. For any $f : A \times B \to \mathbb{R}$ we have $f(A', b) := \sum_{a \in A'} f(a, b)$, $f(a, B') := \sum_{b \in B'} f(a, b)$, and $f(A', B') = \sum_{a \in A', b \in B'} f(a, b)$, for any $A' \subseteq A$ and $B' \subseteq B$.

1.3.1 Bidder preferences

Throughout, we assume that each bidder has a (finite or infinite) set $A \subseteq \mathbb{Z}^n_+$ (for indivisible goods) or $A \subseteq \mathbb{R}^n_+$ (for divisible goods) of bundles they are interested in. Moreover, in the indivisible goods setting, we let the bidder possess a *valuation* $v : A \to \mathbb{R}$ that denotes, for each bundle $\boldsymbol{x} \in A$, the value that the bidder assigns bundle $\boldsymbol{x}$. This valuation may be provided directly, or given implicitly through some other revelation mechanism. We also equivalently define the valuation as $v : \mathbb{Z}^n \to \underline{\mathbb{R}}$, where $\underline{\mathbb{R}} := \mathbb{R} \cup \{-\infty\}$ denotes the partially extended reals, and we assume that the *effective domain* $\operatorname{dom} v = \{\boldsymbol{x} \in \mathbb{Z}^n \mid v(x) > -\infty\}$ of v is non-negative in the sense that $\boldsymbol{x} \geq \boldsymbol{0}$ for all $\boldsymbol{x} \in \operatorname{dom} v$. In Section 1.4, we will (explicitly and implicitly) impose structures on the valuation function in order to capture different 'natural' valuations.

Moreover, we assume that each bidder has *quasi-linear utilities*, i.e. the utility they derive from bundle $\boldsymbol{x}$ at prices $\boldsymbol{p}$ is given by

$$u(\boldsymbol{x}; \boldsymbol{p}) := v(\boldsymbol{x}) - \boldsymbol{p} \cdot \boldsymbol{x}.$$

At given market prices set by the auctioneer, a bidder demands the bundles of goods that maximise her utility; in the divisible goods setting, the set of 'feasible'

bundles over which to maximise is additionally constrained by budgets. Formally, this is expressed by the *demand correspondence* D, which maps prices $\boldsymbol{p}$ to the set of bundles $\boldsymbol{x}$ that maximise $u(\boldsymbol{x}; \boldsymbol{p})$ for this $\boldsymbol{p}$ (subject to budget constraints in the divisible goods setting). Note that D is not a classical function from prices to bundles, as it is possible for a bidder to be indifferent between multiple bundles at given prices. When D is defined by a valuation function v, we write D_v. Finally, we define the *indirect utility function* as follows in order to capture the utility that a bidder derives at given prices $\boldsymbol{p}$.

Definition 1.1. The *indirect utility function* f_v of valuation $v : A \to \mathbb{R}$ maps a price vector $\boldsymbol{p}$ to the utility that a bidder with valuation v has for receiving her preferred bundle at a given price vector $\boldsymbol{p}$ in the following way.

$$f_v(\boldsymbol{p}) \coloneqq \max_{\boldsymbol{x} \in A} u(\boldsymbol{x}, \boldsymbol{p}) = v(\boldsymbol{y}) - \boldsymbol{p} \cdot \boldsymbol{y}, \text{ where } \boldsymbol{y} \in D_v(\boldsymbol{p}). \tag{1.1}$$

1.3.2 Solution concepts

When solving an auction, the auctioneer may pursue one of multiple objectives. In this work, we study the two objectives of maximising social welfare and revenue. Throughout, we assume that the auctioneer has no costs associated with (partially) allocating supply to buyers. It is a standard result in many auction settings that maximising social welfare is equivalent to finding competitive equilibrium; this also holds in the strong-substitutes and arctic product-mix auctions that we study. We begin by defining a *solution* to an auction market as a tuple $(\boldsymbol{p}, \pi)$, where $\boldsymbol{p} \in \mathbb{R}^n$ denotes market prices and $\pi : [m] \to \mathbb{R}^n$ is an *allocation* that maps each bidder $j \in [m]$ to the bundle $\pi(j) \in \mathbb{R}^n$ they receive. An allocation is *feasible* if $\pi([m]) \leq \boldsymbol{s}$ (here we used implicit summation). We now introduce the three objectives of revenue, social welfare, and competitive equilibrium. In Chapters 3 and 4, we address the problems of solving the strong-substitutes product-mix auction under social welfare and revenue, respectively. Chapter 6 does the same for the arctic product-mix auction, noting the unusual fact that the two solutions concepts coincide. All three objectives guarantee that the optimal solution is *envy-free* for bidders, as defined below.

Definition 1.2. An allocation π to bidders is considered *envy-free*[4] at prices $\boldsymbol{p}$ if every bidder receives a bundle they demand at $\boldsymbol{p}$. In other words, we have $\pi(j) \in D_{v^j}(\boldsymbol{p})$ for each bidder j with valuation v^j. We extend this definition to solutions $(\boldsymbol{p}, \pi)$.

Maximising revenue. Recall that the auctioneer has supply $\boldsymbol{s}$ of goods, and is able to allocate any subset $\boldsymbol{t} \leq \boldsymbol{s}$ among the bidders. Due to our assumption

[4]We have retained the moniker 'envy-free' from the literature (cf. (Chen and Deng, 2014; Chen et al., 2016; Guruswami et al., 2005)) even though it is, strictly speaking, stronger than the notion of envy-freeness typically defined in mechanism design, where every bidder weakly prefers their own bundle to the bundles that other bidders receive. In the latter definition, it is possible, for instance, for a bidder not to receive a bundle they demand but still not envy any other bidders for their allocated bundle.

that the auctioneer sets uniform prices, selling $\boldsymbol{t}$ results in a total revenue of $\boldsymbol{t} \cdot \boldsymbol{p}$. Under the revenue-maximising objective, the auctioneer seeks to find a feasible and envy-free solution $(\boldsymbol{p}, \pi)$ to the auction that maximises revenue $\pi([m]) \cdot \boldsymbol{p}$. In particular, we emphasise that it is possible for revenue to be maximised when the auctioneer sells a strict subset of supply at higher prices than they would have been able to set if they had insisted on selling their entire supply. This solution concept has been studied in the profit-maximising envy-free pricing literature (cf. (Chen and Deng, 2014; Chen et al., 2016; Guruswami et al., 2005)).

Maximising social welfare. Suppose, instead, that the auctioneer wishes to maximise social welfare. For each solution $(\boldsymbol{p}, \pi)$, we can express the social welfare as the sum of utilities that the auctioneer and each bidder enjoy under $(\boldsymbol{p}, \pi)$. The auctioneer's utility is given by the revenue $\pi([m]) \cdot \boldsymbol{p}$ they achieve. Meanwhile, each bidder j incurs a utility of $u(\pi(j), \boldsymbol{p})$. In the indivisible goods setting, we have $u(\boldsymbol{p}(j), \boldsymbol{p}) = v(\pi(j)) - \pi(j) \cdot \boldsymbol{p}$. Hence, the social welfare obtained by solution $(\boldsymbol{p}, \pi)$ is $\sum_{j \in [m]} v(\pi(j))$. Under the social welfare objective, the auctioneer wishes to find a feasible solution that maximises this term. Note that a social welfare optimal allocation is Pareto-efficient, as it is not possible to improve any one bidder's utility without decreasing another bidder's utility.

Competitive equilibrium. A solution $(\boldsymbol{p}, \pi)$ is considered a competitive equilibrium if π is envy-free at $\boldsymbol{p}$ and supply is exhausted ($\pi([m]) = \boldsymbol{s}$). We emphasise that competitive equilibrium may not exist (e.g. if bidders have ordinary substitutes preferences in the multi-unit setting). We discuss this further in Section 1.4. Instead, we may instead seek to achieve quasi-equilibrium, in which the auctioneer sells as many items of supply as possible subject to envy-freeness. However, when bidders have strong-substitutes preferences, then competitive equilibrium exists and all competitive equilibria coincide with social welfare-optimal solutions, mirroring by the fundamental theorems of welfare economics. We state this result for the two markets we study in subsequent chapters. A formal proof in the arctic product-mix auction setting is given in Chapter 6, and an analogous proof holds for the strong-substitutes product-mix auction.

Theorem 1.3. *In the strong-substitutes and arctic product-mix auctions, $(\boldsymbol{p}, \pi)$ is a competitive equilibrium if, and only if, it is a social welfare optimal solution.*

1.3.3 Demand and valuation oracles

In the classical action literature, it is typically assumed that bidder preferences are either fully known a priori, or the auctioneer has direct access to the valuation function or demand correspondence. We briefly describe and formalise these two notions of accessing bidder preferences using the concept of oracles.

Note that in sealed-bid auctions, bidders are required to submit their preferences in 'one shot'. This may avoid strategic behaviour such as that found in sequential auctions (Bichler, 2017), but also precludes algorithms for solving these auctions from making sequential queries to the valuation or demand oracle. Instead, bidding

languages are used to capture preferences concisely before the auction is run. This topic is explored in detail in Chapter 2. Nevertheless, while preference oracles will not be directly queried by our algorithms for solving auctions, they play an important role when we turn to expressing bidder demand in the product-mix bidding languages in Chapter 5.

Valuation oracle. In the classic sealed-bid second-price auction and in many combinatorial auctions, a common assumption is that bidders submit valuations for each bundle in which they are interested. When multiple goods are available in multiple quantities, this can become combinatorially complex. Instead of requesting bidders to submit valuations in their entirety, the auctioneer elicits the valuation of a given bundle as and when required by their algorithm (or procedure) for solving the auction. This is formalised by the *valuation oracle*, which is defined for each bidder to map bundles $\boldsymbol{x}$ to the value $v(\boldsymbol{x})$ that the bidder has for them.

Demand oracle. Alternatively, some authors assume that the auctioneer is able to elicit one or more bundles of goods that a bidder demands at given prices. One such example is Ausubel's well-known ascending-bid auction (Ausubel, 2004), which was later extended to the multi-unit setting in (Ausubel, 2006). We denote such an oracle a *demand oracle*. The demand oracle is also noted in Paes Leme (2017) as an algorithmic primitive needed to implement the Walrasian tâtonnement procedure in auctions where bidders have substitutes preferences. While Ausubel's procedure in (Ausubel, 2006) requires the demand oracle to return *all* bundles demanded at given prices, we will work with a weaker *adversarial demand oracle* $\mathcal{Q}_{\mathcal{B}}$ in Chapter 5; given any price vector $\boldsymbol{p}$, $\mathcal{Q}_{\mathcal{B}}(\boldsymbol{p})$ returns a single bundle from $D_{\mathcal{B}}(\boldsymbol{p})$. A bidder may demand multiple bundles at some price (i.e. when $|D_{\mathcal{B}}(\boldsymbol{p})| > 1$), in which case the adversarial oracle simply returns a single demanded bundle at that price, and we have no control over which bundle is returned.

Converting between oracles. Bidders may find it natural to answer queries to one oracle but not the other. In this case, we may wish to simulate one oracle with a bounded number of queries to the other. This is particularly pertinent when algorithms are designed for use of a specific oracle, as is the case in (Murota, 2003) (to solve multi-unit auctions with strong-substitutes preferences). (Nisan, 2005, Lemma 1.5.1) describes a procedure to simulate a valuation query with a polynomial number of demand queries. This result assumes that valuations have effective domain $\{0,1\}^n$ and satisfy free disposal on this domain. We say that a valuation satisfies *free disposal* if, for any bundles $\boldsymbol{x} \leq \boldsymbol{y}$, we have $v(\boldsymbol{x}) \leq v(\boldsymbol{y})$. Intuitively, this means that adding items to a bundle does not decrease its value (as they can be disposed of freely after the fact). It is straightforward to see that the procedure can be generalised to multi-unit domains to simulate a valuation oracle with a weakly polynomial number of demand queries. In Chapter 5, we will also present a procedure for simulating a demand query with a strongly polynomial number of queries to a valuation oracle. This result requires that the underlying valuation is strong-substitutes, a demand type introduced in detail below.

1.4 Demand types

In Section 1.3, we introduced the notion of bidder valuations v and the demand correspondences D_v they induce. In practice, auctions are only tractable w.r.t. revenue or welfare if we place certain restrictions on the structure of v or, equivalently, D_v. We begin by listing some elementary demand types on indivisible goods. We emphasise that this list by no means complete; for a survey of demand types, we refer to (Baldwin and Klemperer, 2019; Nisan and Segal, 2006). Rather, our main focus is on motivating and defining strong-substitutes demand, which underlies the strong-substitutes product-mix auction and is characterised in different ways in Sections 1.4.1 to 1.4.3.

Additive valuations. A valuation v is *additive* (or *linear*) if there exists a vector $\boldsymbol{c}$ such that $v(\boldsymbol{x}) = \boldsymbol{c} \cdot \boldsymbol{x}$ for all bundles $\boldsymbol{x} \in \mathbb{Z}^n$.

Unit demand. Unit demand valuations capture bidders who, at any prices, want to buy at most one item. Formally, v is *unit demand* if there exists $\boldsymbol{c} \in \mathbb{R}^n$ such that $v(\boldsymbol{0}) = 0$, $v(\boldsymbol{e}^i) = c_i$ and $v(\boldsymbol{x}) = -\infty$ for all $\boldsymbol{x} \notin \{\boldsymbol{0}, \boldsymbol{e}^1, \dots, \boldsymbol{e}^n\}$.

Single-minded demand. Single-minded bidders wish to acquire a specific bundle $\boldsymbol{x}$, and attach no value to any other bundles. Hence they have valuations $v(\boldsymbol{0}) = 0, v(\boldsymbol{x}) > 0$ and $v(\boldsymbol{y}) = -\infty$ for all $\boldsymbol{y} \neq \boldsymbol{x}$.

It is common to assume $v(\boldsymbol{0}) = 0$ (Boutilier and Hoos, 2001; Lehmann et al., 2006; Nisan, 2000; Nisan and Segal, 2006). It is straightforward to see that unit demand agents have substitutes preferences, while single-minded agents exhibit complementarities between the goods in the bundle $\boldsymbol{x}$ they wish to acquire. Under the assumption of free disposal, Nisan (2000) and others define unit demand as follows.[5]

Unit demand with free disposal. In any bundle, unit demand bidders only value a single item (even if the bundle contains more than one item of various goods). The value of bundle $\boldsymbol{x}$ is given as the maximum over all valuations c_i for single units of goods that are present in $\boldsymbol{x}$. Formally, we write $v(\boldsymbol{x}) = \max\{c_i \mid i \in [n], x_i > 0\}$.

So far, we have seen demand types defined by restrictions on the valuation function directly. We now present the *gross substitutes* (GS) demand type, which is typically defined via the demand correspondence. GS captures the notion that if we increase prices of some goods, then the demand of the goods whose prices have remained unchanged does not decrease. This is captured formally by Definition 1.4.

Definition 1.4. A valuation function $v : \{0,1\}^n \to \mathbb{R}^n$ is *gross (or weak) substitutes (GS)* if, for all $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n$ with $\boldsymbol{p} \leq \boldsymbol{q}$, there exist $\boldsymbol{x} \in D_v(\boldsymbol{p})$ and $\boldsymbol{y} \in D_v(\boldsymbol{q})$ so that

$$y_i \geq x_i, \quad \forall i \in [n] \text{ with } q_i = p_i.$$

[5]Strictly speaking, they define valuations only in the single-unit setting. However, the generalisation to the multi-unit setting presented here is natural and we refer to it in Chapter 2.

GS valuations are interesting precisely because they guarantee the existence of competitive equilibrium in the single-unit setting. This restriction to single units is important: as soon as we move to the multi-unit setting, GS no longer guarantees equilibrium; for examples of this, as well as more details, we refer to (Shioura and Tamura, 2015).

Theorem 1.5 ((Kelso and Crawford, 1982; Paes Leme, 2017)). *If goods are available in single units and all bidders have gross substitutes valuations (and quasi-linear utilities) $v : \{0,1\}^n \to \mathbb{R}^n$, then competitive equilibrium exists.*

1.4.1 Strong-substitutes demand

In the strong-substitutes product-mix auction, bidders have strong-substitutes (SS) demand preferences expressed through the tropical bidding language. The SS property is appealing because it is an extension of *gross substitutes* (GS) from the single-unit setting that guarantees the existence of a competitive equilibrium in multi-unit auction markets. An alternative, more general, extension of GS to multiple units, called *ordinary substitutes* (OS), does not provide the same guarantees (Paes Leme, 2017).

Definition 1.6 defines OS and SS analogously to the definition of GS in Definition 1.4, by capturing how OS and SS demand changes when we (weakly) increase prices. Intuitively, the OS property states that the bidder's demand for those goods with unchanged prices does not decrease, while the law of aggregate demand (LAD) guarantees that the overall number of items that are demanded does not increase. Note that OS is the immediate extension of GS to the multi-unit setting, while the SS property combines the OS property with LAD.

Definition 1.6 (cf. (Shioura and Tamura, 2015)). A demand correspondence D_v is *ordinary substitutes* (OS) if, for any prices $\boldsymbol{p}' \geq \boldsymbol{p}$ with $D_v(\boldsymbol{p}) = \{\boldsymbol{x}\}$ and $D_v(\boldsymbol{p}') = \{\boldsymbol{x}'\}$, we have $x'_k \geq x_k$ for all k such that $p_k = p'_k$. D_v is *strong-substitutes* (SS) if $\boldsymbol{x}$ and $\boldsymbol{x}'$ additionally satisfy $\|\boldsymbol{x}'\|_1 \leq \|\boldsymbol{x}\|_1$ (the law of aggregate demand).

It is also instructive to view strong-substitutes preferences as those that would be ordinary substitutes (OS) 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.

Definition 1.6 is only one of multiple characterisations provided in (Shioura and Tamura, 2015), which elucidates the relationship between GS, OS, and SS. Below, we introduce an equivalent characterisation of SS that has emerged in the discrete convex analysis literature and plays an important role in later chapters. For a detailed survey on the relationship between OS and SS, we refer to (Shioura and Tamura, 2015). Finally, Section 1.4.3 introduces a third characterisation of SS from the perspective of tropical geometry, which we make use of in Chapter 3 and Chapter 5.

Motivation. We motivate SS preferences in the following example, which is illustrated in Fig. 1.1a. We will refer to this example again in Chapters 2 and 3, when we present the tropical product-mix bidding language and discuss algorithms to solve the strong-substitutes product-mix auction. It is straightforward to verify that the demand in our example is strong-substitutes. In Section 1.4.3, we will see that this also follows directly from the simple observation that all facets dividing the demand regions in Fig. 1.1a are horizontal, vertical or have slope 1.

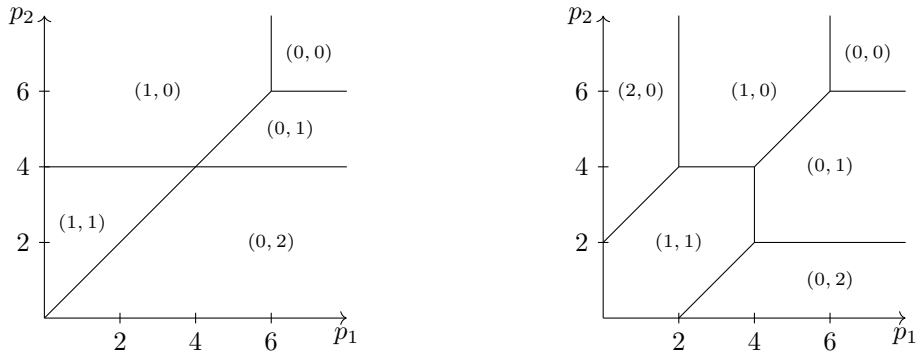
Example 1.7. *Alice participates in an auction held with two goods, say apples (good 1) and bananas (good 2). For her breakfast, she requires a single item of fruit and is willing to pay up to £6 for either an apple or a banana. This is expressed as a unit demand valuation v with $v(\boldsymbol{e}^1) = v(\boldsymbol{e}^2) = 6$, $v(\mathbf{0}) = 0$ and value $-\infty$ for all other bundles. Suppose, for instance, that 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. On its own, this second demand is expressed as valuation v' with $v'(\boldsymbol{e}^2) = 4$, $v'(\mathbf{0}) = v'(\boldsymbol{e}^1) = 0$ and $-\infty$ for all other bundles. We can combine v and v' to get Alice's aggregate valuation v''. Indeed, it is straightforward to check that $v''(\mathbf{0}) = 0$, $v''(0,1) = v''(1,0) = 6$, $v(1,1) = 10$, $v''(0,2) = 10$ and all other bundles have valuation $-\infty$. (In Chapter 2, we formalise how valuations can be combined to achieve, for instance, v''.)*

To illustrate Alice's aggregate demand correspondence, 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: by valuation v, she wants an apple, as her utility for an apple or a banana is 5 and 3, respectively, and by v' she demands in an additional banana, as its price is less than £4. On other hand, if auction prices are $\boldsymbol{p} = (6, 5)$, Alice demands only one banana in total: v results in the demand for a banana and v' induces no demand, as the price of a banana is too high. Alice's aggregate demand correspondence is shown in Fig. 1.1a.

Example 1.8. *Bob participates in the same auction as Alice with two goods, an apple and a banana. His valuation is given by $v(\mathbf{0}) = 0$, $v(0,1) = v(1,0) = 6$, $v(0,2) = v(2,0) = 8$ and $v(1,1) = 10$. Bob's demand correspondence is shown in Fig. 1.1b.*

Note that Alice derives the same marginal value from adding a banana to bundles $(0,1)$ and $(1,0)$ (as $v''(0,2) - v''(0,1) = v''(1,1) - v''(1,0)$). This is not the case for Bob, who derives more utility from adding a banana to $(1,0)$, thus preferring 'more variety'. (This follows from the simple observation $v(0,2) - v(0,1) < v(1,1) - v(1,0)$.) In Section 2.5, we argue that Bob's demand is an example of a 'generic' strong-substitutes demand, which must always exhibit this property of preferring more variety.



(a) Alice's demand **(b)** Bob's demand

Figure 1.1: Two examples of strong-substitutes demand correspondences on two goods {1,2} belonging to Alice and Bob. Price space is divided by facets into polyhedral regions in which a unique bundle is demanded. Here (x, y) denotes the bundle with x units of good 1 and y units of good 2.

1.4.2 Discrete convex analysis

A 'price-free' characterisation of the strong-substitutes property using the language of discrete convex optimisation is given by Shioura and Tamura (2015). A function $f : \mathbb{Z}^n \to \underline{\mathbb{R}}$ is called $M^\natural$-*concave* if it satisfies the following *exchange property*. For any $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$ and $i \in \operatorname{supp}^+(\boldsymbol{x} - \boldsymbol{y})$, there exists $j \in \operatorname{supp}^-(\boldsymbol{x} - \boldsymbol{y}) \cup \{0\}$ such that

$$f(\boldsymbol{x}) + f(\boldsymbol{y}) \leq f(\boldsymbol{x} - \boldsymbol{e}^i + \boldsymbol{e}^j) + f(\boldsymbol{y} + \boldsymbol{e}^i - \boldsymbol{e}^j). \tag{1.2}$$

Here we recall that $\boldsymbol{e}^0 = 0$, and define the positive and negative support of a vector $\boldsymbol{z} \in \mathbb{Z}^n$ as $\operatorname{supp}^+(\boldsymbol{z}) = \{i \in [n] \mid z_i > 0\}$ and $\operatorname{supp}^-(\boldsymbol{z}) = \{i \in [n] \mid z_i < 0\}$.

Theorem 1.9 now allows us to characterise SS as a structural property of the valuation function.

Theorem 1.9 ((Shioura and Tamura, 2015)). *A quasi-linear demand correspondence D_v is strong-substitutes if, and only if, its valuation v is $M^\natural$-concave.*

$M^\natural$-concave functions are closely related to M-concave functions, which satisfy the exchange property (1.2) for some non-zero $j \in \operatorname{supp}^-(\boldsymbol{x} - \boldsymbol{y})$. Every n-dimensional $M^\natural$-concave function can be obtained as the projection of an $(n+1)$-dimensional M-concave function onto an n-dimensional hyperplane. Conversely, we can obtain the corresponding M-concave function $\hat{f}$ of an $M^\natural$-concave function f as

$$\hat{f}(x_0, \boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if } x_0 = -\sum_{i \in [n]} x_i, \\ -\infty & \text{otherwise,} \end{cases} \tag{1.3}$$

where $(x_0, \boldsymbol{x}) \in \mathbb{Z}^{n+1}$ is an $(n+1)$-dimensional vector. For details on $M^\natural$- and M-concave functions, we refer to (Murota et al., 2013).

1.4.2.1 The indirect utility function

$M^\natural$-concavity is closely related to $L^\natural$-convexity, an alternative notion of convexity from the discrete convex optimisation literature. A function $f : \mathbb{Z}^n \to \overline{\mathbb{R}}$ is $L^\natural$-convex if it satisfies the *translation submodularity* property,

$$f(\boldsymbol{p}) + f(\boldsymbol{q}) \geq f((\boldsymbol{p} - \alpha\mathbf{1}) \vee \boldsymbol{q}) + f(\boldsymbol{p} \wedge (\boldsymbol{q} + \alpha\mathbf{1})) \quad (\forall \boldsymbol{p}, \boldsymbol{q} \in \mathbb{Z}^n_+, \forall \alpha \in \mathbb{Z}_+). \tag{1.4}$$

Here, $\vee$ and $\wedge$ denote the element-wise maximum and minimum, respectively. Using this definition, it is possible to show directly that the indirect utility function f_v of a strong-substitutes valuation, as defined in (1.1), is $L^\natural$-convex. A more elegant proof, however, is achieved using a discrete version of the Legendre-Fenchel duality stated, for example, in (Murota, 2003). Indeed, let $f : \mathbb{Z}^n \to \underline{\mathbb{R}}$ with $\operatorname{dom} f \neq \emptyset$. Then the *discrete concave Legendre-Fenchel transformation* (or *discrete concave conjugate*) f° is defined as

$$f^\circ(\boldsymbol{p}) = \inf_{\boldsymbol{x} \in \mathbb{Z}^n} (\boldsymbol{p} \cdot \boldsymbol{x} - f(\boldsymbol{x})). \tag{1.5}$$

Theorem 1.10 ((Murota, 2003)). *If f is an $M^\natural$-concave function, then its discrete concave conjugate is $L^\natural$-convex.*

We note that the indirect utility function f_v, by definition, corresponds to the discrete concave conjugate of the valuation function v. As strong-substitutes valuations are $M^\natural$-concave by Theorem 1.9, the $L^\natural$-convexity of f_v follows by Theorem 1.10.

1.4.3 Tropical geometry

We give some geometric intuition for strong-substitutes demand correspondences. It is well-known that any quasi-linear demand divides price space into piecewise-linear convex regions corresponding to bundles. When demand is SS, each such region is a convex lattice (Murota et al., 2013). Figure 1.2 depicts a SS demand on two goods that illustrates this.

Recently, Baldwin and Klemperer (2019) proposed a new way of characterising demand types. Borrowing from the tropical geometry literature, they introduce the *Locus of Indifference Prices* (LIP), a piecewise-linear geometric object $\mathcal{L}_v = \{\boldsymbol{p} \in \mathbb{R}^n_+ \mid |D_v(\boldsymbol{p})| > 1\}$ associated with every valuation v that consists of the set of all prices at which the bidder is indifferent between two or more bundles. They show that the LIP decomposes in a natural way into an $(n-1)$-dimensional rational polyhedral complex Π_v. Moreover, each facet of Π_v has a positive weight attached to it, expressed by the weight function w. Figure 1.2 depicts an example strong-substitutes demand whose polyhedral complex is drawn using solid lines. Noting that the orientation of the separating facet between two adjacent demand regions, together with the facet weights, characterises how demand changes when moving from one region to the other, Baldwin and Klemperer (2019) propose a new way of defining demand types by the set of facet-normal vectors of the LIP. In this new paradigm, the strong-substitutes demand type is defined as the family of demand correspondences whose LIP facets are normal to $\boldsymbol{e}^i$ or $\boldsymbol{e}^i - \boldsymbol{e}^j$ for some

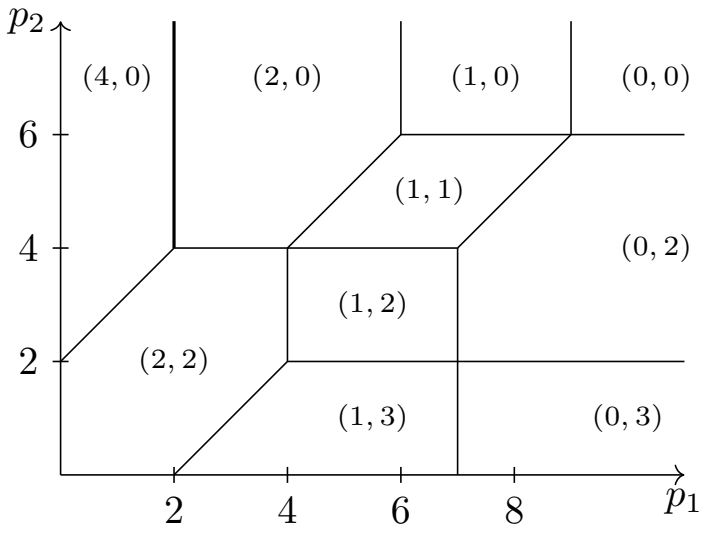


Figure 1.2: An illustration of a strong-substitutes demand correspondence with two goods, partitioning price space into piecewise-linear convex regions. Each region is labelled with the bundle demanded at prices in the region. The lines comprise the polyhedral complex induced by the Locus of Indifference Prices (LIP). The facet from $(2, 4)$ to $(2, 8)$ (drawn thicker) has weight 2, and all other facets have weight 1.

$i, j \in [n]$. We call such normal vectors *strong substitutes*. In two dimensions, facets of SS LIPs are either horizontal, vertical or normal to $(1, -1)$. Hence it follows directly from this definition that the demand correspondences in Figs. 1.1 and 1.2 enjoy the strong-substitutes property. The ordinary substitutes demand type, on the other hand, is defined as the family of demand correspondences whose LIP facets are normal to $a\boldsymbol{e}^i - b\boldsymbol{e}^j$ for some $i, j \in [n]_0$ (where $\boldsymbol{e}^0 := 0$). We illustrate such demand correspondences in Fig. 2.3 of Chapter 2.

It turns out that the facet-normal vectors express the intuitive notion of change in demand as follows. Suppose $\boldsymbol{p}$ and $\boldsymbol{p}'$ are separated by facet F normal to $\boldsymbol{n}$ with weight $w(F)$, where $\boldsymbol{n}$ has been normalised to the vector with co-prime integer elements. Then the bundles demanded at $\boldsymbol{p}$ and $\boldsymbol{p}'$ can be obtained from each other by adding, or subtracting $w(F)\boldsymbol{n}$. It follows immediately that strong-substitutes demand is characterised by 'one-to-one' exchanges between pairs of goods, whereas general substitutes corresponds to 'many-to-many' changes in demand between two goods across facets. This is reflected in Fig. 1.2.

Building on the topological work of Mikhalkin (2004), one of the central results of (Baldwin and Klemperer, 2019) is to characterise the LIP of any concave valuations in terms of a simple but powerful geometric property. Simply stated, this property guarantees the following. Suppose we start at a given price vector $\boldsymbol{p}$ in the interior of a polyhedron of a weighted polyhedral complex, and move along a closed path that ends at $\boldsymbol{p}$. Then the sum of demand changes that we have undergone while crossing facets (as specified by the weight and normal vector of each facet) must be $\mathbf{0}$, as the demand is the same at the start and end.

Definition 1.11 ((Mikhalkin, 2004)). An $(n-1)$-dimensional weighted polyhe-

dral complex (Π, w) is *balanced* if, for every $(n-2)$-cell $G \in \Pi$, the weights $w(F_j)$ on the facets $F_1, \ldots, F_l$ that contain G, and primitive integer normal vectors $\boldsymbol{v}_{F_j}$ for these facets that are defined by a fixed rotational direction about G, satisfy $\sum_{j=1}^{l} w(F_j)\boldsymbol{v}_{F_j} = 0$. (The fixed rotational direction can be obtained, for instance, by taking a circle around any point in G of sufficiently small circumference, and embedding it into a 2-dimensional plane that is normal to G. We require all vectors v_{F_j} to point in a consistent direction around this circle.)

Now that we have defined the notion of balanced polyhedral complexes, we are able to state one of Baldwin and Klemperer (2019)'s central results, the valuation-complex equivalence theorem. We will use this result in Chapter 2 to prove that valid tropical bid lists encode strong-substitutes demand.

Theorem 1.12 ((Baldwin and Klemperer, 2019)). *Suppose* (Π, w) *is an* $(n-1)$*-dimensional weighted rational polyhedral complex in* $\mathbb{R}^n$*, that* $\mathcal{L}$ *is the union of the cells in* Π*, and* $\boldsymbol{p}$ *is any price not contained in* $\mathcal{L}$*. Then the following holds.*

1. *There exists a finite set* $A \subsetneq \mathbb{Z}^n$ *and a function* $u : A \to \mathbb{R}$ *such that* $\mathcal{L}_u = \mathcal{L}$ *and* $w_u = w$ *if, and only if,* (Π, w) *is balanced.*
2. *If* (Π, w) *is balanced, then there exists a finite set* $A \subsetneq \mathbb{Z}^n$ *and a unique concave valuation* $u : A \to \mathbb{R}$ *such that* $D_u(\boldsymbol{p}) = \{\mathbf{0}\}$*,* $u(\mathbf{0}) = 0$*,* $\mathcal{L}_u = \mathcal{L}$ *and* $w_u = w$*.*

2
Bidding languages

Contents

2.1 Chapter outline 18
2.2 Introduction 19
2.3 Languages in the literature 20
2.4 The logical languages 22
2.4.1 Grammar 22
2.4.2 A family of logical languages 23
2.5 The tropical language 24
2.5.1 Representing strong-substitutes demand 27
2.5.2 Understanding tropical bids geometrically 29
2.5.3 Expressivity and validity 30
2.5.4 Checking validity is coNP-hard 32
2.5.5 A simple algorithm for testing validity 35
2.5.6 The relation between OXS and positive tropical bids . . 37
2.5.7 Expressing general substitutes 39
2.6 The arctic language 40
2.7 Conclusion and outlook 42

2.1 Chapter outline

This chapter introduces bidding languages that allow agents to express preferences for distinct goods in varying quantities in the form of sealed bids. Bidding languages are useful only if they are simple to use, expressive enough for bidders to be able to capture their desired preferences, and encode preferences in a way that is conducive to algorithmic treatment. These considerations are formulated in terms of various computational challenges in Section 2.2 and motivate all subsequent

results in this thesis.

After briefly discussing existing bidding languages in the literature in Section 2.3, we focus in particular on the 'tropical' and 'arctic' product-mix languages, which appear in all subsequent chapters of this thesis and allow bidders to represent substitutes preferences for indivisible and divisible goods, respectively. These languages are presented in Sections 2.5 and 2.6. A first result is to highlight the connection between the tropical language and OXS, a member of the 'logical languages' proposed by Lehmann et al. (2006): In Section 2.5.6, we show that these languages are essentially equivalent if we restrict ourselves to strictly positive prices.

A peculiarity of the tropical product-mix bidding language is the possibility of a bidder formulating a syntactically correct 'tropical bid list' that does not express any quasi-linear demand correspondence.[1] We show in Section 2.5.3 that checking whether a bid list does so, or not, is coNP-complete and present an algorithm for this.[2] However, other computational challenges, such as computing a bundle demanded by a tropical bid list at given prices, can be solved efficiently. Finally, Section 2.7 concludes.

2.2 Introduction

In sealed-bid auctions, each bidder communicates their bidding preferences ('bids') to the auctioneer prior to the start of the auction. For auctions on a single indivisible good, which may be available in one or multiple units, a bidder can submit their preferences as a single value or vector denoting the highest price that the bidder is willing to pay for each possible number of items. This mechanism of communicating demand is commonly assumed, for instance, in the uniform price auction and the Vickrey-Clarke-Groves auction. In contrast, combinatorial auctions are typically run over multiple distinct and indivisible goods that may be available in one or more units, and bidders must express preferences over exponentially many possible bundles of goods. When the auction is run on n single-unit goods, for instance, there are 2^n possible bundles (including the empty bundle). Hence a naïve representation of a given valuation requires 2^n values. This is impractical for bidders even for small n, and quickly becomes impossible as n grows. Instead, this chapter discusses bidding languages that allow the bidder to express their preferences or, equivalently, valuations in a succinct and intuitive fashion.[3]

Language considerations. Bidding languages can be thought of as a collection of strings, say over $\{0, 1\}$ or the ASCII alphabet, with each string mapping to a valuation. In the design and application of a bidding language, one must balance the competing objectives of simplicity, expressivity and computational

[1] In practice, this can easily be mitigated by placing certain syntactic restrictions on the bidding language at the cost of expressivity. This is currently done by the Bank of England, for instance.

[2] This result appears in (Baldwin et al., 2021b).

[3] In this sense, our bidding languages might more accurately be called 'valuation languages' and used in more general market settings. We retain the more established moniker 'bidding language', as we utilise these languages for bidders to participate in auctions.

complexity. Simplicity denotes the informal but intuitive notion that bidders must understand how to represent their demand in the given language. The 'complexity' of understanding the language can be mitigated by the provision of algorithms that assist the bidder in expressing their demand in the language (as done in Chapter 5). Secondly, the expressivity of a language relates directly to the types of demand that it can express (e.g. additive or strong-substitutes). Unlike the notion of simplicity, we can study the expressivity of bidding languages rigorously. The tropical product-mix language, for instance, captures exactly the strong-substitutes demand type. Finally, this work concerns itself primarily with notions of computational complexity of the product-mix languages, which can be studied in the context of the following computational challenges.

Expression. Given preferences that can be queried through a valuation or demand oracle, compute a representation ('string') of these preferences in the desired bidding language.

Evaluation. Given preferences expressed in a bidding language, and a bundle of goods $\boldsymbol{x}$, compute the valuation $v(\boldsymbol{x})$ of $\boldsymbol{x}$ implied by the underlying valuation function v.

Demand evaluation. Given preferences expressed in a bidding language, and market prices $\boldsymbol{p}$, compute a bundle $\boldsymbol{x} \in D(\boldsymbol{p})$ of goods that is demanded by these preferences, where $D(\cdot)$ denotes the underlying demand correspondence.

Solving auctions. Given a number of bidders whose preferences are expressed in a bidding language, solve the auction with a given supply with respect to a solution concept such as maximising social welfare or revenue, or achieving competitive equilibrium.

Note that by a simple information-theoretic argument, there cannot exist a bidding language that encodes all valuations succinctly. Assume, without loss of generality, that a 'bid' consists of a string of bits. Even with n goods available in single units, the space of all valuations is exponential in 2^n. It follows that the encoding of most valuations is exponential in n. In face of this, it is reasonable to focus on bidding languages that can concisely capture 'natural' demand types such as general or strong substitutes. Intuitively, a language achieves this if it is able to exploit structural or geometric properties of the demand type is intends to express. We will see below how the tropical product-mix language achieves this.

2.3 Languages in the literature

Logical languages. Bidding languages for combinatorial auctions (with multiple indivisible goods) were first introduced explicitly by Sandholm (1999) and Fujishima et al. (1999). These languages were subsequently studied systematically by Boutilier and Hoos (2001); Nisan (2000) and Lehmann et al. (2006), who set out a framework for a family of 'logical languages' to express arbitrary complements and substitutes demand for indivisible goods available in a single unit. An overview of this family is

given in Section 2.4; for a more detailed survey, we refer to (Nisan, 2005). Moreover, we note that the framework of logical languages was more recently extended to divisible goods by Kaleta (2013).

Assignment messages. Milgrom (2009) proposed 'integer assignment messages', a bidding language designed to represent strong-substitutes preferences in a multi-good, multi-unit indivisible good setting. In this language, preferences are expressed as a linear program; in particular, evaluating the value for a given bundle of goods is achieved by solving the program in question. This language is related to 'endowed assignment valuations' introduced by Hatfield and Milgrom (2005) to encode gross substitutes preferences in the single-unit setting. However, it was recently shown by Fichtl (2021) that assignment messages cannot express arbitrary strong-substitutes preferences, limiting the expressivity of the language. The proof proceeds by characterising the demand type expressed by integer assignment messages, and providing an example of strong-substitutes demand preferences that do not satisfy this characterisation. This result is analogous to that of Ostrovsky and Paes Leme (2015), who showed that endowed assignment valuations cannot express arbitrary gross substitutes preferences (in the setting with one unit per good).

The product-mix languages. In all subsequent chapters, we assume the use of the 'arctic' or 'tropical' product-mix bidding languages, which respectively allow bidders to express substitutes preferences for divisible or indivisible goods. The *tropical product-mix bidding language* (also referred to as the *tropical language*) was introduced to allow bidders taking part in the strong-substitutes product-mix auction to express preferences for indivisible goods in an intuitive and natural fashion, while the *arctic product-mix bidding language* (or simply *arctic language*) allows bidders participating in the arctic auction to express substitutes preferences for divisible goods. Both the arctic and the tropical language require the bidders to submit their bidding preferences as a set of 'bids', each of which consisting of a vector of prices (one for each good) as well as a weight. In both languages, the price vector denotes an upper threshold on the prices that the bid is willing to accept: if market prices strictly dominate the bid's prices in all goods, the bid demands no good. In contrast, the role of the bid weight differs between the two languages. In the tropical language, which expresses demand for indivisible goods, the (integral) bid weight denotes the number of items that the bid demands if market prices do not dominate bid prices. The arctic language, meanwhile, expresses demand for divisible goods, and interprets the (possibly non-integral) bid weight as the bid's budget. At given market prices, each bid is allocated quantities of goods that exhaust its budget.

Developed independently by Klemperer (2008, 2010), the tropical language can be seen to generalise the logical language OXS of Lehmann et al. (2006) to the multi-unit setting and arbitrary strong-substitutes valuations. Unlike the logical languages, which act in 'bundle space' by recursively combining 'atomic' valuations using logical $\vee$ and $\oplus$ operators, the tropical language conceptually constructs preferences by combining 'atomic' bids in price space. Moreover, it has been shown by Baldwin and Klemperer (2021) that the tropical product-mix language is able

to express arbitrary strong-substitutes preferences (unlike, for instance, Milgrom's assignment messages). Importantly, this language also admits polynomial-time algorithms for three of the four computational challenges introduced in Section 2.2: evaluating the valuation function for a given bundle, computing a bundle demanded at given prices, and solving auctions for social welfare. We argue the tractability of computing a demanded bundle below, and refer to Chapter 3 for the other two. However, some challenges remain computationally hard: In Chapter 4, we show that solving the auction for revenue maximisation is APX-hard; this was also previously proved by Guruswami et al. (2005). Moreover, the challenge of computing the representation of given preferences that are accessible through a valuation or demand oracle is shown to be hard in Chapter 5.

2.4 The logical languages

The logical languages set out in Nisan (2000, 2005) and Lehmann et al. (2006) work in a combinatorial auction setting in which n goods are available in single units. Hence bundles of goods are elements in $\{0,1\}^n$ and can alternatively be identified by subsets $S \subseteq [n]$. As usual, a bidder's valuation v maps each bundle $\boldsymbol{x}$ to a real value. By assumption, $v(\mathbf{0}) = 0$ and the valuations satisfy free disposal, so $v(\boldsymbol{x}) \leq v(\boldsymbol{y})$ for all $\boldsymbol{x} \leq \boldsymbol{y}$.

2.4.1 Grammar

Bids in logical languages are constructed recursively using logical $\vee$ (OR) and $\oplus$ (XOR) operators and directly specify valuations according to the following rules.

Atomic bids. For any bundle $\boldsymbol{x} \in \{0,1\}^n$ and value $p \geq 0$, we let $(\boldsymbol{x}, p)$ denote an *atomic bid*. This bid specifies the valuation that assigns value p to any bundle $\boldsymbol{y}$ that contains $\boldsymbol{x}$, and 0 otherwise (so $v(\boldsymbol{y}) = p$ for $\boldsymbol{y} \geq \boldsymbol{x}$ and $v(\boldsymbol{y}) = 0$ otherwise). These valuations are called *single-minded* in (Lehmann et al., 2002). An important special case arises when the bundle $\boldsymbol{x}$ of bid $(\boldsymbol{x}, p)$ is a singleton, so $\boldsymbol{x} = \boldsymbol{e}^i$ for some $i \in [n]$; in this case, we call $(\boldsymbol{e}^i, p)$ a *singleton atomic bid*, or simply *singleton atom*.

$\vee$ combinations. For any two logical bids b and b' with valuations v and v', the bid $(b \vee b')$ specifies valuation v defined by $v(\boldsymbol{x}) := \max_{\boldsymbol{y} \leq \boldsymbol{x}}(v(\boldsymbol{y}) + v'(\boldsymbol{x} - \boldsymbol{y}))$. Intuitively, the $\vee$ operator 'accepts' both bids b and b', and finds a partition $\boldsymbol{y}$ and $\boldsymbol{x} - \boldsymbol{y}$ of $\boldsymbol{x}$ to maximise the joint valuation $v(\boldsymbol{y}) + v'(\boldsymbol{x} - \boldsymbol{y})$. Note that, this valuation may be maximised for $\boldsymbol{y} = \mathbf{0}$ or $\boldsymbol{y} = \boldsymbol{x}$, so it is possible for only one bid to contribute to the joint valuation. This motivates the moniker $\vee$.

$\oplus$ combinations. For any two logical bids b and b' with valuations v and v', the bid $(b \oplus b')$ denotes the valuation defined by $v(\boldsymbol{x}) := \max\{v(\boldsymbol{x}), v'(\boldsymbol{x})\}$. Intuitively, the $\oplus$ operator 'accepts' only one bid, namely the one with the higher valuation for $\boldsymbol{x}$.

We note that $\vee$ and $\oplus$ are associative and commutative, and $\vee$ distributes over $\oplus$. Moreover, the grammar does not rely on goods being available in single units and can be extended immediately to the multi-unit setting.

2.4.2 A family of logical languages

While the grammar defined above allows us to construct bid formulas of arbitrary complexity from atomic bids, it is instructive to study first- and second-order languages, and languages built from singleton atomic bids are of particular interest to us. We now briefly list some logical languages studied in the literature.

The OR bidding language. This language consists of all first-order formulas constructed from atomic bids with the $\vee$ operator. Due to associativity and commutativity, we can write such formulas as $(\boldsymbol{x}^1, p^1) \vee \cdots \vee (\boldsymbol{x}^k, p^k)$. We call such a formula an OR bid. In the single-unit setting, this language captures the valuations that have no substitutabilities (so $v(\boldsymbol{x} + \boldsymbol{y}) \geq v(\boldsymbol{x}) + v(\boldsymbol{y})$ for all disjoint $\boldsymbol{x}$ and $\boldsymbol{y}$), but cannot express unit-demand valuations (Nisan, 2005).

The XOR bidding language. This language consists of the first-order formulas $(\boldsymbol{x}^1, p^1) \oplus \cdots \oplus (\boldsymbol{x}^k, p^k)$ constructed from atomic bids using only the $\oplus$ operator. We call such formulas XOR bids. In the single-unit setting, the XOR language can represent all valuations (Nisan, 2005). However, this representation may not be succinct: additive valuations on n goods can be represented in the OR language with n atomic bids, while even simple additive valuations require 2^n atomic bids in the XOR language (Nisan, 2005).

The OR-of-XOR and XOR-of-OR bidding languages. In an effort to construct expressive languages that can succinctly express a variety of valuations, Sandholm (1999) and Nisan (2000) considered second-order languages. The OR-of-XOR language considers formulas constructed by combining XOR bids with the $\vee$ operator, while the XOR-of-OR language combines OR bids with the $\oplus$ operator. Nisan (2005) gives examples of valuations whose representation in the OR-of-XOR language requires exponentially many more atomic bids than that in the XOR-of-OR language, and vice versa. Hence, neither language is strictly more succinct.

Formulas from singleton atomic bids. Of particular interest to us are the bidding languages that can be constructed recursively from singleton atomic bids. Singleton atoms are appealing because of their simplicity. Moreover, it is straightforward to see that OR bids of singletons correspond to additive valuations, while XOR bids of singletons express the class of unit demand valuations (with free disposal). In second-order 'singleton' languages, we again combine these bids to gain more expressivity: OXS and XOS respectively denote OR-of-XOR and XOR-of-OR of singletons. These languages were introduced by Lehmann et al. (2006) in the single-unit setting, who established the characterisation of complement-free preferences given in Theorem 2.1. A valuation v on n goods (in the single-unit

setting) is called *complement-free* if $v(A) + v(B) \geq v(A \cup B)$ for all $A, B \subseteq [n]$, and *submodular* if the stronger inequality $v(A) + v(B) \geq v(A \cup B) + v(A \cap B)$ holds.

Theorem 2.1 ((Lehmann et al., 2006)). *Let GS, SM and CF respectively denote gross-substitutes, submodular and complement-free valuations (in the single-unit setting). Then the following strict inclusions hold for the languages OXS and XOS:*

$$OXS \subset GS \subset SM \subset XOS \subset CF.$$

We emphasise that OXS, as well as all other logical languages, can be extended to bundles with multiple units per good without changing the grammar defined above, and the definitions do not require free disposal per se.[4] We make use of this observation in Section 2.5.6, where we show that the OXS language generalised to such multi-unit bundles is essentially equivalent to the language of positive tropical bids.

2.5 The tropical language

We now turn to the tropical product-mix bidding language. The original implementation of the strong-substitutes product-mix auction allowed each bidder to submit a list of *tropical bids*, where each bid has $n+1$ integral elements: an integral price for each of the n goods available, and a total (integral) quantity of goods sought by that bid.[5] 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 quasi-linear preferences. 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. (In particular, if a bid only wants one specific good, it simply sets prices of zero for all the other $(n-1)$ goods.) A bid that creates a tie (i.e. when 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 quantities of either of these goods. Fig. 2.1 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 tropical bidding language by expressing Alice's demand of Example 1.7 with two bids. Fig. 2.2a illustrates this example further.

Example 2.2 (Continuation of Example 1.7). *Recall that 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*

[4] To the best of our knowledge, this has not been done in the literature.

[5] 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 £).

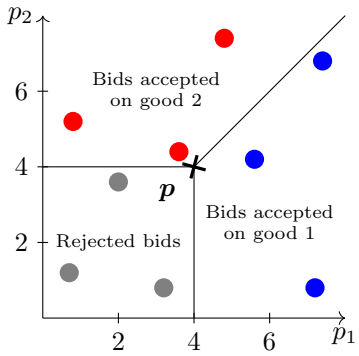
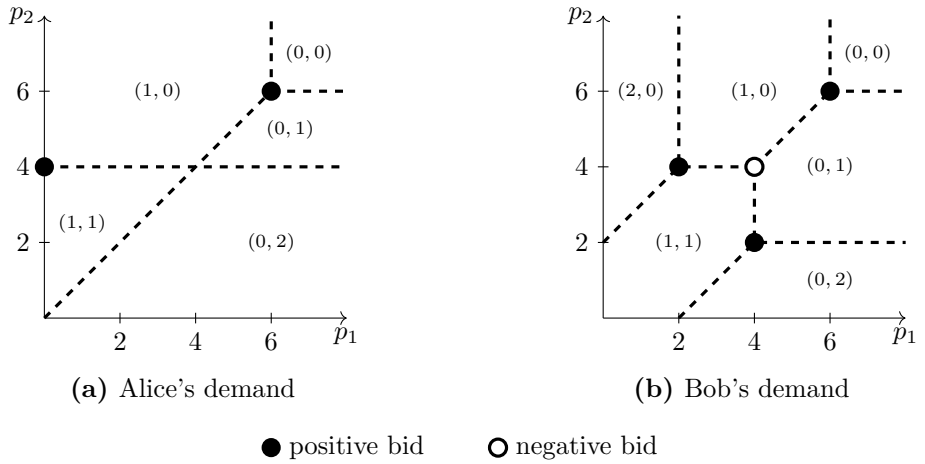


Figure 2.1: Price vectors divide $\mathbb{R}^n$ into regions that specify which good is allocated to bids lying in these region. This example works in the setting with two goods. The price vector $\boldsymbol{p} = (4, 4)$ (marked by a cross) divides $\mathbb{R}^2$ into three regions, each of which labelled according to the good that bids in this region are allocated.

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 to $\boldsymbol{p} = (1, 3)$*, 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 Fig. 2.2a.*

Positive bids only. In versions of the product-mix auction thus far implemented, all bids are for positive quantities of goods. This syntactically defines a demand type of particular practical interest, as the Bank of England currently runs the strong-substitutes product-mix auction with positive bids only. Alice's demand in Example 1.7, for instance, belongs to this demand type. As we will see in subsequent chapters, restricting ourselves to positive bids simplifies many computational challenges, including expressing demand as tropical bids, implementing a demand oracle and solving the strong-substitutes product-mix auction for social welfare. However, we see in Chapter 4 that solving for maximum revenue remains hard. Section 2.5.6 shows that the class of valuations expressible by positive tropical bids is essentially identical (up to free disposal) to the valuations expressed by the generalisation of OXS introduced in (Lehmann et al., 2006) to multi-unit bundles, and we can translate between the two languages easily.



(a) Alice's demand **(b)** Bob's demand

Figure 2.2: Examples of strong-substitutes demand correspondences on two goods {1,2} belonging to two bidders, Alice and Bob, expressed using tropical bids. 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 $\boldsymbol{p}$.

Positive and negative bids. Some strong-substitutes preferences (over bundles of non-negative quantities of goods) can only be expressed by using lists of bids for both positive *and negative* quantities,[6] and it has been shown that *any* strong-substitutes preferences can be represented using lists of positive and negative bids.[7] 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. In Chapter 1, we noted 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.

1. $v((1, 1)) \leq v((0, 1)) + v((1, 0))$,

[6]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)).

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

2. $v((2,0)) - v((1,0)) \leq v((1,1)) - v((0,1))$.

3. $v((0,2)) - v((0,1)) \leq v((1,1)) - v((1,0))$.

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 value of adding an item of good i to a singleton bundle $\boldsymbol{e}^j$ is greater if the items are distinct ($i \neq j$); in other words, 'more variety' is valuable. This reflects "within-good" substitutability (decreasing marginal utility of the good) jointly with an assumption about the "relative size/importance" of the goods. 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 given in Example 1.8 and illustrated in Fig. 1.1b, 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 2.3 (Continuation of Example 1.8). *A combination of positive and negative bids, depicted in Fig. 2.2b, is required to express Bob's demand correspondence in Example 1.8, which divides price space into six demand regions for bundles with at most two units. Recall that the Bob's valuation is* $v(0,1) = v(1,0) = 6$, $v(0,2) = v(2,0) = 8$ *and* $v(1,1) = 10$. *The corresponding bid list is* $\mathcal{B}' = \{(2,4;+1),(4,2;+1),(4,4;-1),(6,6;+1)\}$.

2.5.1 Representing strong-substitutes demand

We now formally describe how every strong-substitutes demand can be represented by a finite list $\mathcal{B}$ of positive and negative tropical bids. A *tropical bid* $\boldsymbol{b}$ consists of an $(n+1)$-dimensional integral 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, in Chapter 3 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. We note that it is also possible to consider bids with arbitrary rational entries; this is done internally by our allocation algorithm in Chapter 3.

It is often convenient to regard a rejected bid as being accepted on a notional *reject good* for which bid entries and prices are always zero. Letting the reject good be 0, the set of goods is then $[n]_0 := \{0,\ldots,n\}$. In this setting, we identify bundles and prices with the $(n+1)$-dimensional vectors obtained by prepending a 0-th entry of value 0 (and thus indexing from 0 instead of 1). Similarly, 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.

The demand correspondence. Suppose $\mathcal{B}$ is a list of tropical bids. The setup described above 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 2.2 (Fig. 2.2a). 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 Fig. 2.2a 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 2.2, 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. Fig. 2.2a 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 Fig. 2.2a, 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 $D_{\mathcal{B}}(\boldsymbol{p}) = \{(1, 0), (0, 1), (1, 1), (0, 2)\}$.

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

$$f_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \max_{i \in [n]_0}(b_i - p_i). \tag{2.1}$$

It is clear that we can compute $f_{\mathcal{B}}(\boldsymbol{p})$ efficiently. In Section 2.5.3, we will see how bid lists, and the demand correspondence they induce, correspond to strong-substitutes demand. In particular, we establish that for any *valid* bid list, $f_{\mathcal{B}} = f_v$ for some strong-substitutes valuation v. Validity of bid lists is introduced in Section 2.5.3.

Computing demand. Suppose we are given a list $\mathcal{B}$ of bids, and wish to implement a demand oracle procedure, which returns some bundle $\boldsymbol{x} \in D_{\mathcal{B}}(\boldsymbol{p})$ demanded at given prices $\boldsymbol{p}$. This solves the 'demand evaluation' challenge described in Section 2.2. If $\boldsymbol{p}$ is non-marginal, simply allocate each bid $\boldsymbol{b}$ a positive or negative quantity $w(\boldsymbol{b})$ of the good $i(\boldsymbol{b})$ it uniquely demands and add up these items to obtain the unique demanded bundle $\boldsymbol{x} = \sum_{\boldsymbol{b}\in\mathcal{B}} w(\boldsymbol{b})\boldsymbol{e}^{i(\boldsymbol{b})}$. Similarly, if $\boldsymbol{p}$ is marginal but all bids are positive, allocate each bid $\boldsymbol{b}$ a positive quantity $w(\boldsymbol{b})$ of any good $i(\boldsymbol{b})$ it demands.

However, if $\boldsymbol{p}$ is marginal and the bid list contains both positive and negative bids, independently allocating to each bid one of the goods it demands may result in a bundle that does not lie in $D_{\mathcal{B}}(\boldsymbol{p})$. In Example 2.3 (depicted in Fig. 2.2b), 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 (and thus obtain a non-marginal price vector), and then note that the resulting bundle is demanded at the unperturbed prices. A suitable such perturbation is to sum $\boldsymbol{p}$ with the vector $\sum_{i\in[n]} \frac{1}{2n}\boldsymbol{e}^i \in \mathbb{R}^n$.

2.5.2 Understanding tropical bids geometrically

In Chapter 1, one characterisation of strong-substitutes demand was given as the family of demand correspondences whose weighted polyhedral complex (Π, w) has facets normal to strong-substitutes vectors (vectors $\boldsymbol{e}^i - \boldsymbol{e}^j$ for $i, j \in [n]_0$). We now formulate analogous geometric objects corresponding to the demand of tropical bid lists. This will be used in the next section to prove that tropical bids express strong-substitutes preferences. We use terminology and material developed in (Baldwin and Klemperer, 2019) and refer to the same for further details. Just as we defined the Locus of Indifference Prices (LIP) for valuations in Section 1.4.3 of Chapter 1, we now introduce the LIP for a bid $\boldsymbol{b}$ as

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

This corresponds to the prices that are marginal w.r.t. $\boldsymbol{b}$. Similarly, the LIP of a list of bids $\mathcal{B}$ is given by

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

and corresponds to the prices that are marginal w.r.t. the entire bid list $\mathcal{B}$. 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 2.4 (cf. (Baldwin and Klemperer, 2019, Definition 3.9)). The *strong-substitutes vectors* are the 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.

2.5.3 Expressivity and validity

A result by Baldwin and Klemperer (2021) shows that any integral SS demand correspondence can be represented as a finite list of bids (which may contain positive and negative bids), and this representation is essentially unique (up to redundancies). This establishes the tropical language as an intuitive language to fully capture strong-substitutes demand. Conversely, however, not all lists of positive and negative bids induce a strong-substitutes valuation function. Indeed, there exist syntactically correct lists of positive and negative bids that do not correspond to any quasi-linear demand; a simple example of this is the list consisting of a single bid $(1, 1; -1)$. We call a bid list *valid* if it induces quasi-linear demand.

Recall that the indirect utility function for a bid list is defined by $f_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\boldsymbol{b}\in\mathcal{B}} w(\boldsymbol{b}) \max_{i\in[n]_0}(b_i - p_i)$. In a first result, Theorem 2.5 gives three equivalent characterisations of validity. This characterisation was first given in (Baldwin et al., 2021b). In Chapter 3, we will also introduce a weaker local notion of validity by introducing validity in the ε-neighbourhood of a price $\boldsymbol{p}$. The proof of Theorem 2.5 makes use of the valuation-complex equivalence theorem from Baldwin and Klemperer (2019), which establishes deep connections between demand types and the geometry of the demand correspondence. In Section 2.5.4, we apply the second characterisation to establish the complexity of checking the validity of a given bid list.

Theorem 2.5. *Let $\mathcal{B}$ be a list of positive and negative bids and $f_{\mathcal{B}}$ be the associated indirect utility function as defined in* (2.1)*. $\mathcal{B}$ is valid if, and only if, the following equivalent conditions hold.*

1. *The indirect utility function $f_{\mathcal{B}}$ as defined in* (2.1) *is convex.*
2. *There is no price vector $\boldsymbol{p}$ and pair of distinct goods $i, i' \in [n]_0$ at which the weights $w(\boldsymbol{b})$ of the bids $\boldsymbol{b} \in \mathcal{B}$ marginal on i and i' sum to a negative number.*
3. *$f_{\mathcal{B}}$ is the indirect utility function f_u of a strong-substitutes valuation u with quasi-linear utilities so that $D_{\mathcal{B}}(\boldsymbol{p}) = D_u(\boldsymbol{p})$ for all $\boldsymbol{p} \in \mathbb{R}^n$.*

Proof. The equivalence of (1) and (2) follows from Proposition 3.12 (in Chapter 3) if we set $P = \mathbb{R}^n$. The implication (3) $\Rightarrow$ (1) is well-known; see for instance (Shioura and Tamura, 2015, Theorem 7.1) and (Murota et al., 2016, Theorem 4.3) for a stronger statement. We now show (2) $\Rightarrow$ (3), using the valuation-complex equivalence theorem from (Baldwin and Klemperer, 2019). Let $(\Pi_{\mathcal{B}}, w_{\mathcal{B}})$ 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. 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 also the proof of Lemma 3.13 in Baldwin and Klemperer (2019) for details.)

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, every bid is either non-marginal at $\boldsymbol{p}$ or marginal between the same two distinct goods, $i, i' \in [n]_0$ (see also (Baldwin and Klemperer, 2019, Proposition 2.4)). 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, $i(\boldsymbol{b}) \in D_{\boldsymbol{b}}(\boldsymbol{p})$ for every $\boldsymbol{b} \in \mathcal{B}$ and $\boldsymbol{x} = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \boldsymbol{e}^{i(\boldsymbol{b})}$. Then we claim that

$$u(\boldsymbol{x}) = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) b_{i(\boldsymbol{b})}. \tag{2.2}$$

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 (2.2) 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

$$u(\boldsymbol{x}') - \boldsymbol{p} \cdot \boldsymbol{x}' = u(\boldsymbol{x}'') - \boldsymbol{p} \cdot \boldsymbol{x}'' \tag{2.3}$$

and

$$f_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b})(b_{i'(\boldsymbol{b})} - p_{i'(\boldsymbol{b})}) = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b})(b_{i''(\boldsymbol{b})} - p_{i''(\boldsymbol{b})}). \tag{2.4}$$

Suppose (2.2) holds for bundle $\boldsymbol{x}'$ and $(i'(\boldsymbol{b}))_{\boldsymbol{b} \in \mathcal{B}}$. Then (2.3) and (2.4) imply that (2.2) 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 (2.2) 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). □

2.5.4 Checking validity is coNP-hard

Suppose bidders submit their lists of positive and negative bids to the auctioneer prior to the auction. We prove in Theorem 2.7 that checking the validity of a given list of bids is computationally hard. The proof rests on the second characterisation of validity in Theorem 2.5.

Definition 2.6. Given a list of positive and negative bids $\mathcal{B}$, the problem ValidBids is to decide whether $\mathcal{B}$ is valid.

Theorem 2.7. ValidBids *is* coNP*-complete.*

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

Theorem 2.8. *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 $\mathcal{B}$.*

The proof of Theorem 2.7 uses an equivalent definition of Theorem 2.5, (2) stated geometrically. For any bid list $\mathcal{B}$, let $\mathcal{B}^+$ and $\mathcal{B}^-$ denote its positive and negative bids. Define regions in $\mathbb{R}^n_+$ that are generated by a point $\boldsymbol{p}$ and coordinates $i, j \in [n]$ as follows.

$$H_i^p \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n_+ \mid \boldsymbol{x} \leq \boldsymbol{p}, x_i = p_i\} \tag{2.5}$$

$$F_{ij}^p \coloneqq \{\boldsymbol{x} + \beta \boldsymbol{e}^{[n]} \mid \beta \in \mathbb{R}_+, \boldsymbol{x} \leq \boldsymbol{p}, x_i = p_i \text{ and } x_j = p_j.\} \tag{2.6}$$

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 its 'valuation' component $(b_1, \ldots, b_n)$ 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, and *non-negative* otherwise.

Observation 2.9. *For any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n_+$, we have $\boldsymbol{x} \in F^{\boldsymbol{y}}_{ij}$ if, and only if, $x_i - y_i = x_j - y_j$ and $\boldsymbol{x} - (x_i - y_i)\mathbf{1} \leq \boldsymbol{y}$.*

Observation 2.10. *Fix distinct $i, j \in [n]$. Then containment in the regions given in (2.5) and (2.6) is transitive in the sense that, for any $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^n_+$,*

- *$\boldsymbol{x} \in H^{\boldsymbol{y}}_i$ and $\boldsymbol{y} \in H^{\boldsymbol{z}}_i$, implies $\boldsymbol{x} \in H^{\boldsymbol{z}}_i$.*
- *$\boldsymbol{x} \in F^{\boldsymbol{y}}_{ij}$ and $\boldsymbol{y} \in F^{\boldsymbol{z}}_{ij}$, implies $\boldsymbol{x} \in F^{\boldsymbol{z}}_{ij}$.*

The following definition of valid bids restates the definition given in Theorem 2.5, (2), in terms of regions (2.5) and (2.6).

Definition 2.11 (Valid bids). A list of positive and negative bids is *valid* if, for any point $\boldsymbol{p} \in \mathbb{R}^n_+$ and two coordinates $i, j \in [n]$, $H^{\boldsymbol{p}}_i$ and $F^{\boldsymbol{p}}_{ij}$ are non-negative.

On the basis of Definition 2.11, the proof of Theorem 2.7 works by reducing the well-known NP-complete problem 3-CNF-Sat to ValidBids by means of an intermediate NP-complete decision problem MoreNegativesDominated, which we define in Definition 2.12. The NP-completeness of MoreNegativesDominated is established in Theorem 2.13 and the reduction from MoreNegativesDominated to ValidBids is given below as the proof of Theorem 2.7. 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 2.12. Given two lists P and N of vectors in $\mathbb{R}^n_+$, MoreNegatives-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 2.13. *The problem* MoreNegativesDominated *is NP-complete.*

Proof. Note that verifying whether a given vector $\boldsymbol{z} \in \mathbb{R}^n_+$ dominates more vectors in N than in P can be done in polynomial time. This establishes membership of MoreNegativesDominated in the class NP. We show completeness by reducing from 3-CNF-Sat. 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 $C_i = L^i_1 \vee L^i_2 \vee L^i_3$ are its clauses and $L^i_j \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$ 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-SAT to MORENEGATIVESDOMINATED. 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 2.14. *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 2.14, $\boldsymbol{z}$ 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 2.14, 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. □

We are now ready to prove the coNP-hardness of checking the validity of a given bid list.

Proof of Theorem 2.7. In order to show that a bid list $\mathcal{B}$ 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 VALIDBIDS in coNP.

Let (N, P) be an instance of MORENEGATIVESDOMINATED and n be the dimension of its vectors. We construct an $(n+1)$-dimensional instance $\mathcal{B}$ of VALIDBIDS such that $(N, P) \in$ MORENEGATIVESDOMINATED if and only if $\mathcal{B} \notin$ VALIDBIDS 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. □

2.5.5 A simple algorithm for testing validity

In Section 2.5.4, we showed that checking the validity of a bid list in the general case is 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 some applications of the product-mix auction.

2.5.5.1 The algorithm

Our procedure rests on Theorem 2.15, which reduces the validity condition from Definition 2.11 to a finite number of checks. Lemma 2.16 reduces the number of checks further. Together, the two results immediately yield Algorithm 2.1, whose running time is given in Theorem 2.17. For any set of bids U, define the minimal dominating vector $\mathrm{md}(U)$ of U as the component-wise maximum over the valuation vectors of the bids in U, so that $\mathrm{md}(U)_i = \max_{\boldsymbol{b} \in U} b_i$ 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 $\mathrm{md}(U)$ 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 $\mathrm{mdF}(i, U)$ as $\mathrm{mdF}(i, U) := \min_{\boldsymbol{b} \in U} b_i \mathbf{1} + \mathrm{md}(\{\boldsymbol{b} - b_i \mathbf{1} \mid \boldsymbol{b} \in U\})$. 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 2.9 that $\mathrm{mdF}(i, U) = \mathrm{mdF}(j, U)$ is the maximal such $\boldsymbol{x}$.

Theorem 2.15. *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 $H_i^{\mathrm{md}(U)}$ 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 $F_{ij}^{\mathrm{mdF}(i,U)}$ is non-negative.*

Proof. The implication is immediate by Definition 2.11. Conversely, suppose $\mathcal{B}$ satisfies the second condition. First we show that condition 1 of Definition 2.11 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

$\boldsymbol{y} = \mathrm{md}(U)$. 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 2.10, 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 $\boldsymbol{z} = \mathrm{mdF}(U)$. Suppose $\boldsymbol{b} \in F_{ij}^z$ for every $\boldsymbol{b} \in U$ and $\boldsymbol{z} \in F_{ij}^x$. Then Observation 2.10 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 2.9. 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 2.16. *For any list of negative bids $U \subseteq \mathcal{B}^-$ there exists a list $U' \subseteq U$ with $|U'| \leq n$ so that $\mathrm{md}(U') = \mathrm{md}(U)$. Moreover, for each coordinate $i \in [n]$, there exists a list $U'' \subseteq U$ with $|U''| \leq n + 1$ so that $\mathrm{mdF}(i, U'') = \mathrm{mdF}(i, U)$.*

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 $\mathrm{md}(U') = \mathrm{md}(U)$ 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 $\mathrm{mdF}(U'') = \mathrm{mdF}(U)$ by construction and we are done. □

Algorithm 2.1 Checking the validity of a bid list $\mathcal{B}$

```
for all subsets U ⊆ B⁻ of negative bids with |U| ≤ n + 1 do
    for all i, j ∈ [n] do
        Verify the two conditions given in Theorem 2.15.
    end for
end for
```

Theorem 2.17. *Algorithm 2.1 runs in time $O\left(\binom{|\mathcal{B}^-|}{n} n^3 |\mathcal{B}|\right)$. Moreover, if $|\mathcal{B}^-| \leq k$ or $n \leq k$ for some constant k, the algorithm is polynomial in the input size $n|\mathcal{B}|$.*

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 2.15 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 $O(|\mathcal{B}^-|^{k+1}|\mathcal{B}|)$. □

2.5.6 The relation between OXS and positive tropical bids

OXS and the language of positive tropical bids bear strong similarities. In this section, we show that both languages induce the same demand correspondences in positive price space, when we generalise OXS to the multi-unit demand setting. Recall that singleton atoms were defined in Section 2.4 as $(\boldsymbol{e}^i, p)$ with the corresponding valuation $v : \{0,1\}^n \to \mathbb{R}$ with $v(\boldsymbol{x}) = p$ if $x_i = 1$ and $v(\boldsymbol{x}) = 0$ otherwise. This corresponds to unit demand valuations. There are multiple natural ways in which these atoms can be generalised to the multi-unit setting. We focus on one method, which preserves the free disposal property that Lehmann et al. (2006) and Nisan and Segal (2006) assume, and define the corresponding valuation $v : \mathbb{N}_0^n \to \mathbb{R}$ as $v(\boldsymbol{x}) = p$ if $x_i \geq 1$ and $v(\boldsymbol{x}) = 0$ otherwise. In this setting, OXS is not entirely equivalent to the language of positive tropical bids. This is intuitive, as tropical bids do not induce valuations that satisfy free disposal. However, we will see that this equivalence does hold if we restrict ourselves to positive prices, or to non-negative prices if we allow ourselves work with a 'slack' bid of weight ∞ at the origin.

We begin by defining a one-to-one mapping between lists of positive tropical bids, and 'standardised' OXS bids. This provides us with a straightforward way to translate between the two languages. We then show how these languages relate in terms of valuations and demand correspondences. In the following, we will assume that all prices are non-negative. In order to define 'standardised' OXS bids, we first note the following about XOR bids on n goods.

Observation 2.18. *Every XOR bid on n goods with $k \neq n$ singleton atoms can be written as an equivalent* standardised *XOR bid $(e^1, p_1) \oplus \ldots \oplus (e^n, p_n)$ consisting of exactly n singleton atoms.*

Proof. Suppose our XOR bid contains singleton atoms (e^i, p) and (e^i, p'). Then we can replace these atoms with the single atom $(e^i, \max\{p, p'\})$. Secondly, suppose the bid contains no atom for bundle e^i. Then we can add the atom $(e^i, 0)$ without changing the underlying valuation. □

We say that a *standardised OXS bid* consists only of standardised XOR bids. This suggests an intuitive way to convert between single tropical bids and standardised XOR bids. The following mapping between bid lists and OXS bids formalises this. Finally, Lemma 2.20 proves the relationship between OXS and positive tropical bids.

Definition 2.19. The positive tropical bid list $\mathcal{B} = \{\boldsymbol{b}^1, \ldots, \boldsymbol{b}^m\}$ maps to L if, and only if, L contains m standardised XOR bids and, for each tropical bid $\boldsymbol{b}^j$, L contains the XOR bid $(\boldsymbol{e}^i, b_1) \oplus \cdots \oplus (\boldsymbol{e}^n, b_n)$.

Lemma 2.20. *Let $\mathcal{B}$ and L be a corresponding tropical bid list and logical bid as defined in Definition 2.19. Then their demand correspondences $D_{\mathcal{B}}$ and D_L agree on all positive prices.*

Proof. Fix some prices $\boldsymbol{p} > 0$ with strictly positive entries for all goods. We first show that $D_{\mathcal{B}}(\boldsymbol{p}) \subseteq D_L(\boldsymbol{p})$. Suppose $\boldsymbol{x} \in D_{\mathcal{B}}(\boldsymbol{p})$. Then there exists an allocation $(\boldsymbol{x}^j)_{j\in[m]}$ of $\boldsymbol{x}$ to the tropical bids $\mathcal{B} = (\boldsymbol{b}^j)_{j\in[m]}$. By construction of XOR bids, $\boldsymbol{x}^j$ is also

demanded by the j-th XOR bid of L. We argue that $v_L(\boldsymbol{x}) = \sum_{j\in[m]} v^k(\boldsymbol{x}^j)$, which implies $\boldsymbol{x} \in D_L(\boldsymbol{p})$. Indeed, let $(\boldsymbol{y}^j)_{j\in[m]}$ be any partition of $\boldsymbol{x}$. As $\boldsymbol{x}^j \in D_{v^j}(\boldsymbol{p})$, we have $v^j(x_j) - x^j \cdot \boldsymbol{p} \geq v^j(\boldsymbol{y}^j) - \boldsymbol{y}^j \cdot p$. Summing over all $j \in [n]$, we get $\sum_{j\in[m]} v^j(\boldsymbol{x}^j) \geq \sum_{j\in[m]} v^j(\boldsymbol{y}^j)$. This implies the claim.

Now we show that $D_L(\boldsymbol{p}) \subseteq D_{\mathcal{B}}(\boldsymbol{p})$. Suppose $\boldsymbol{x} \in D_L(\boldsymbol{p})$ is demanded by logical bid L consisting of m $\oplus$-ed XOR bids, and let v^j denote the valuation associated with the j-th XOR bid in L. By definition of the $\vee$ operator, there exists some partition $(\boldsymbol{x}^j)_{j\in[m]}$ such that $\boldsymbol{x} = \sum_{j\in[m]} \boldsymbol{x}^j$ and $v_L(\boldsymbol{x}) = \sum_{j\in[m]} v^j(\boldsymbol{x}^j)$. We claim that for each $j \in [m]$, $\boldsymbol{x}^j$ maximises utility at $\boldsymbol{p}$ for the j-th XOR bid, so $\boldsymbol{x}^j \in D_{v^j}(\boldsymbol{p})$. Indeed, suppose to the contrary that there exists some $k \in [m]$ and $\hat{\boldsymbol{x}}^k$ with $v^k(\hat{\boldsymbol{x}}^k) - \hat{\boldsymbol{x}}^k \cdot \boldsymbol{p} > v^k(\boldsymbol{x}^k) - \boldsymbol{x}^k \cdot \boldsymbol{p}$. Then we can define $\hat{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}^k + \hat{\boldsymbol{x}}^k$. Again, by definition of the $\vee$ operator, we note that $v_L(\hat{\boldsymbol{x}}) \geq \sum_{j\in[m]-k} v^j(\boldsymbol{x}^j) + v^k(\hat{\boldsymbol{x}}^k)$. Hence,

$$v_l(\hat{\boldsymbol{x}}) - \hat{\boldsymbol{x}} \cdot \boldsymbol{p} \geq \sum_{j\in[m]-k} \Big[v^j(\boldsymbol{x}^j) - \boldsymbol{x}^j \cdot \boldsymbol{p})\Big] + v^k(\hat{\boldsymbol{x}}^k) - \hat{\boldsymbol{x}}^k \cdot \boldsymbol{p} \tag{2.7}$$

$$> \sum_{j\in[m]} \Big[v^j(\boldsymbol{x}^j) - \boldsymbol{x}^j \cdot \boldsymbol{p})\Big] \tag{2.8}$$

$$= v_L(\boldsymbol{x}) - \boldsymbol{x} \cdot \boldsymbol{p}. \tag{2.9}$$

But this contradicts our assumption that $\boldsymbol{x}$ is demanded by L at $\boldsymbol{p}$. We have thus shown that $\boldsymbol{x}^k \in D_{v^k}(\boldsymbol{p})$. Next, we show that $\boldsymbol{x}^k = \boldsymbol{e}^i$ for some $i \in [n]_0$, for each $k \in [m]$. Note first that a XOR bid consisting of singleton atoms demands at most one good. Let this good be i, and suppose $x^k_i > 1$. We know that $v^{\boldsymbol{x}^k} = v(\boldsymbol{e}^i)$ by free disposal and the definition of XOR bids. But then the utility of $\boldsymbol{x}^k$ at $\boldsymbol{p}$ is $v(\boldsymbol{x}^k) - x^k_i p_i < v(\boldsymbol{e}^i) - p_i$, as $p_i > 0$. Again, this contradicts $\boldsymbol{x}^k \in D_{v^k}(\boldsymbol{p})$.

Finally, it is immediate from the construction of $\mathcal{B}$ that $\boldsymbol{x}^k \in D_{\boldsymbol{b}^k}(\boldsymbol{p})$ for all $k \in [m]$, and thus $\boldsymbol{x} = \sum_{j\in[m]} \boldsymbol{x}^j$ is demanded by $\mathcal{B}$ at $\boldsymbol{p}$. □

This result is intuitive, as the valuation induced by tropical bids follows the grammar of logical languages. We formalise this observation in Lemma 2.21. Alternatively, note that we can achieve the demand of a multi-unit OXS bid on all *non-negative* (and not just positive) prices by permitting the addition of a single positive tropical bid of weight ∞ at the origin $\mathbf{0}$ to the bid list $\mathcal{B}$ corresponding to the OXS bid. Intuitively, the addition of the ∞-weighted bid enables free disposal at prices that lie on the axes, the only prices at which the demand differs between any corresponding logical bid L and tropical bid list $\mathcal{B}$. Using this idea, we see that positive tropical bids are, in fact, strictly more expressive than multi-good OXS (as long as we allow ∞ weights) as positive bids can also express valuations without free disposal.

Lemma 2.21. *Let $\mathcal{B}^1$ and $\mathcal{B}^2$ be two lists of positive tropical bids corresponding to strong-substitutes valuations v^1 and v^2. Then the bid list $\mathcal{B} = \mathcal{B}^1 \cup \mathcal{B}^2$ induces a strong-substitutes valuation v that satisfies $v(\boldsymbol{x}) = \max_{\boldsymbol{y}\leq\boldsymbol{x}}(v^1(\boldsymbol{y}) + v^2(\boldsymbol{x} - \boldsymbol{y}))$.*

Proof. Without loss of generality, assume that each bid has weight 1. (Otherwise, if some bid has weight $w(\boldsymbol{b}) \geq 2$, we replace it with $w(\boldsymbol{b})$ bids of weight 1.) We first

recall from Chapter 1 that the indirect utility function f_v of a valuation v is given by $f_v(\boldsymbol{p}) := v(\boldsymbol{x}) - \boldsymbol{p} \cdot \boldsymbol{x}$, where $\boldsymbol{x} \in D_v(\boldsymbol{p})$ is a bundle demanded at $\boldsymbol{p}$. Moreover, the indirect utility function $f_{\mathcal{B}}$ for $\mathcal{B}$ is defined as $f_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\boldsymbol{b} \in \mathcal{B}} \max_{i \in [n]_0}(b_i - p_i)$ (here we make use of $w(\boldsymbol{b}) = 1$). By Theorem 2.5, we have $f_v = f_{\mathcal{B}}$ for the strong-substitutes valuation v corresponding to $\mathcal{B}$.

Fix a bundle $\boldsymbol{x}$. If $\boldsymbol{x}$ is not demanded by bids $\mathcal{B}$ at any prices, then we define $v_{\mathcal{B}}(\boldsymbol{x}) = -\infty$. It also follows for all bundles $\boldsymbol{x}^1, \boldsymbol{x}^2$ with $\boldsymbol{x} = \boldsymbol{x}^1 + \boldsymbol{x}^2$ that $\boldsymbol{x}^k \notin D_{\mathcal{B}^k}(\boldsymbol{p})$ for $k = 1$ or $k = 2$. Hence $v^k(\boldsymbol{x}^k) = -\infty$ for $k = 1$ or $k = 2$, and it follows that $v_{\mathcal{B}^1}(\boldsymbol{x}^1) + v_{\mathcal{B}^2}(\boldsymbol{x}^2) = -\infty$.

Now suppose $\boldsymbol{x}$ is demanded at prices $\boldsymbol{p}$ by $\mathcal{B}$. Let i denote an allocation function that maps each bid to the good $i(\boldsymbol{b})$ it receives in order to achieve bundle $\boldsymbol{x}$. (Note that there may be multiple such functions – we can pick any.) Then we can rewrite $f_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\boldsymbol{b} \in \mathcal{B}} b_{i(\boldsymbol{b})} - \boldsymbol{p} \cdot \boldsymbol{x}$. It follows that $v(\boldsymbol{x}) = \sum_{\boldsymbol{b} \in \mathcal{B}} b_{i(\boldsymbol{b})}$. Let $\boldsymbol{x}^k$ be the bundle that the bids in $\mathcal{B}^k$ receive under allocation function i, for $k \in \{1, 2\}$, so $\boldsymbol{x}^k = \sum_{\boldsymbol{b} \in \mathcal{B}^k} \boldsymbol{e}^{i(\boldsymbol{b})}$. We note that $\boldsymbol{x} = \boldsymbol{x}^1 + \boldsymbol{x}^2$ and $\boldsymbol{x}^k \in D_{\mathcal{B}^k}(\boldsymbol{p})$. This immediately implies $v(\boldsymbol{x}) = v(\boldsymbol{x}^1) + v(\boldsymbol{x}^2)$.

Finally, suppose that $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ are two disjoint bundles summing to $\boldsymbol{x}$. We want to show that $v^1(\boldsymbol{y}^1) + v^2(\boldsymbol{y}^2) \leq v^1(\boldsymbol{x}^1) + v^2(\boldsymbol{x}^2)$, which proves our claim. Note that $\boldsymbol{x}^k$ maximises utility at $\boldsymbol{p}$ as $\boldsymbol{x}^k \in D_{\mathcal{B}^k}(\boldsymbol{p})$. (Recall that the quasi-linear utility of bundle $\boldsymbol{x}$ at prices $\boldsymbol{p}$ is given by $u(\boldsymbol{x}) - \boldsymbol{p} \cdot \boldsymbol{x}$). Hence,

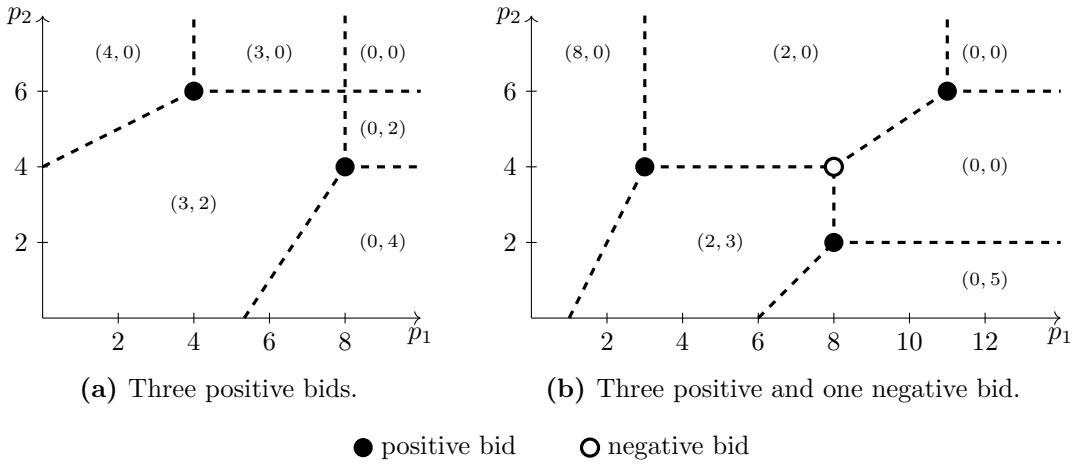
$$v^k(\boldsymbol{y}^k) - \boldsymbol{p} \cdot \boldsymbol{y}^k \leq v^k(\boldsymbol{x}^k) - \boldsymbol{p} \cdot \boldsymbol{x}^k. \tag{2.10}$$

Summing (2.10) for $k = 1$ and $k = 2$ yields $v^1(\boldsymbol{y}^1) + v^2(\boldsymbol{y}^2) \leq v^1(\boldsymbol{x}^1) + v^2(\boldsymbol{x}^2)$ and we are done. □

Finally, Lemma 2.21 also indicates that the language of positive tropical bids can be characterised precisely as the language OR-of-XOR of more restrictive atomic bids $(\boldsymbol{e}^i, b_i)$ corresponding to valuations $v(\boldsymbol{0}) = 0$, $v(\boldsymbol{e}^i) = b_i$ and $v(\boldsymbol{x}) = -\infty$ for all $\boldsymbol{x} \notin \{\boldsymbol{0}, \boldsymbol{e}^i\}$. Moreover, as we have seen, this redefinition of singleton atoms only impacts demand at prices that are not strictly positive.

2.5.7 Expressing general substitutes

The tropical bidding language, as introduced above, allows bidders to express arbitrary strong-substitutes preferences. These are characterised by polyhedral complexes with facets normal to $\boldsymbol{e}^i - \boldsymbol{e}^j$ for some $i, j \in [n]_0$, where we assume $\boldsymbol{e}^0 = 0$. We now view strong-substitutes demand from the perspective of tropical geometry as introduced in Chapter 1. Intuitively, we see that strong-substitutes represents one-to-one demand changes of two goods when we move from one unique demand region (UDR) to another by crossing the relative interior of a facet of the polyhedral complex $\Pi_{\mathcal{B}}$. In contrast, the general substitutes demand type is characterised by many-to-one demand changes, which corresponds to facets normal to $a\boldsymbol{e}^i - b\boldsymbol{e}^j$, where $a, b \in \mathbb{N}$. We can generalise the tropical language as follows to capture these preferences. A *generalised tropical bid* consists of a vector $\boldsymbol{b}$ of length $2n + 1$. We can write $\boldsymbol{b} = (\boldsymbol{v}, \boldsymbol{w}, \sigma)$, where $\boldsymbol{v} \in \mathbb{N}^n$ denotes the bid's valuation (or



(a) Three positive bids. **(b)** Three positive and one negative bid.

Figure 2.3: Two demand correspondences on two goods generated by generalised tropical bids. Note the different slopes of the facets separating demand regions, which express many-to-one demand changes. Left: demand generated by the two positive bids $(4, 6; 1, 2; 1)$ and $(8, 4; 3, 2; 1)$. Right: demand generated by positive bids $(3, 4; 6, 3; 1), (8, 2; 2, 2; 1), (11, 6; 2, 3; 1)$ and negative bid $(8, 4; 2, 3; -1)$.

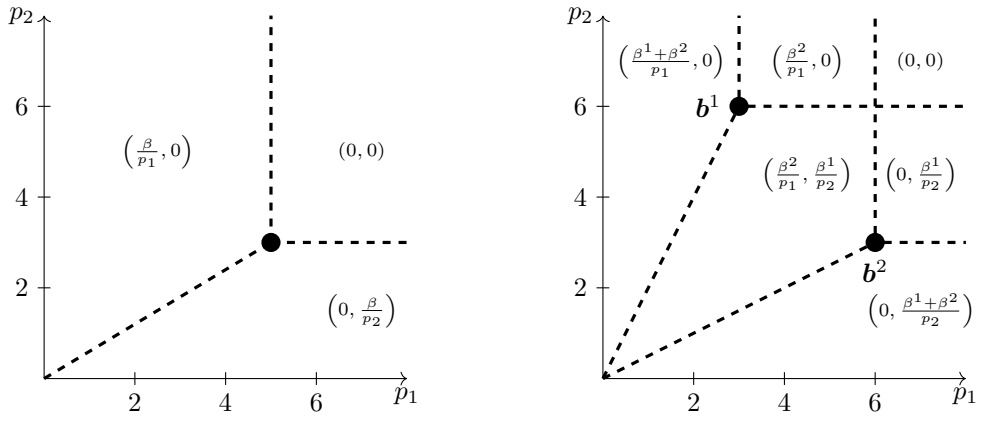
price thresholds, analogous to the first n entries in a standard tropical bid), $\boldsymbol{w} \in \mathbb{N}^n$ denotes the weight vector and $\sigma \in \{-1, 1\}$ denotes the sign of the bid.

At any price vector $\boldsymbol{p} \in \mathbb{R}^n$, we say that $\boldsymbol{b} = (\boldsymbol{v}, \boldsymbol{w}, \sigma)$ demands good i if $i \in \arg\max_{j \in [n]_0} w_j(v_j - p_j)$. Moreover, $\boldsymbol{p}$ is *non-marginal* if every bid uniquely demands a good at $\boldsymbol{p}$. At any given non-marginal price vector $\boldsymbol{p} \in \mathbb{R}^n$, we define the demand correspondence $D_{\boldsymbol{b}}$ of bid $\boldsymbol{b}$ as $D_{\boldsymbol{b}}(\boldsymbol{p}) := \sigma w_{i(\boldsymbol{b})} \boldsymbol{e}^{i(\boldsymbol{b})}$, where $i(\boldsymbol{b})$ is some good demanded by $\boldsymbol{b}$ at $\boldsymbol{p}$. The demand correspondence for marginal prices and for a list of generalised bids can then be defined as for standard tropical bids. In particular, negative bids require careful allocation in order to define a valid demand correspondence as before. We note that a standard tropical bid $\boldsymbol{b}$ can be expressed as a generalised bid $(\boldsymbol{b}, |w(\boldsymbol{b})|, \ldots, |w(\boldsymbol{b})|, \sigma)$, where σ denotes the sign of $w(\boldsymbol{b})$. Generalised bids capture many-to-one demand changes and thus greatly increase expressivity. This is illustrated in Fig. 2.3. Moreover, just as standard tropical bid can express arbitrary strong-substitutes preferences, it has recently been shown that generalised tropical bids are able to express arbitrary ordinary-substitutes preferences, making them an appealing language for agents in many settings.

Theorem 2.22 ((Baldwin and Klemperer, 2021)). *Generalised tropical bids can express arbitrary ordinary-substitutes preferences.*

2.6 The arctic language

In the arctic product-mix auction, each bidder submit a finite list $\mathcal{B}$ of *arctic* bids to express demand over n *divisible* goods. An *arctic bid* consists of an $(n + 1)$-dimensional integral vector $(b_1, \ldots, b_n; b_{n+1}) \in \mathbb{N}^{n+1}$, whose first n entries b_i can be understood as the amount that $\boldsymbol{b}$ is willing to spend per unit of good i, and b_{n+1}



(a) A single arctic bid.

(b) Two arctic bids with budgets β^1 and β^2.

Figure 2.4: The demand correspondences of arctic bids. Left: An arctic bid with budget β divides price-space into three regions (left). At prices that are element-wise larger than $\boldsymbol{b}$, the bid rejects. At prices $\boldsymbol{p}$ that lie above the line connecting $\boldsymbol{b}$ to the origin, the bid demands a quantity β/p_1 of good 1. Conversely, at prices below the line, $\boldsymbol{b}$ demands $\beta(\boldsymbol{b})/p_2$ of good 2. Right: The demand correspondence of two arctic bids $\boldsymbol{b}^1 = (3, 6; \beta^1)$ and $\boldsymbol{b}^2 = (6, 3; \beta^2)$.

denotes the bid's monetary budget. For convenience, we also refer to this budget as $\beta(\boldsymbol{b})$. At a market price *above the stated bid price*, the bid *rejects the good*. At market prices below the stated bid prices, the bid demands the goods that yield the highest 'bang-per-buck', i.e. the highest ratio of value to price, a feature in common with the Fisher market. Through the ability to reject a good, however, arctic bids generalise the demand that can be expressed in the standard Fisher market. If a bid demands a single good, it demands its entire budget's worth of this good. If a bid demands multiple goods, it demands quantities that, multiplied by prices, make up its entire budget. If a bid's values are strictly lower than prices, it demands nothing. If a bid is indifferent between one or multiple goods and receiving nothing, it demands quantities that, multiplied by prices, make up weakly less than its budget.

Defining the demand correspondence. Similar to tropical bids, we introduce the notional reject good 0 and identify bid vectors and prices with the $(n+2)$- and $(n+1)$-dimensional vectors obtained by adding a 0-th entry of value 1. (Note that this differs from the tropical language setting, where the value was 0.) This allows us to formally define the demand correspondence $D_{\mathcal{B}}$ induced by a list of arctic bids as follows. Fix market prices $\boldsymbol{p}$. A bid $\boldsymbol{b}$ is *rejected* at $\boldsymbol{p}$ if $b_i < p_i$ for all goods $i \in [n]$. Otherwise, the bid *demands a good* $i \in [n]$ that maximises $\frac{b_i}{p_i}$ at prices $\boldsymbol{p}$. Recalling that $b_0 = 1 = p_0$, we say that $\boldsymbol{b}$ demands good $i \in [n]_0$ if $i \in \arg\max_{i \in [n]_0} \frac{b_i}{p_i}$, and receiving the 'reject' good is equivalent to the bid being rejected. The demand of a single 'arctic' bid is illustrated in Fig. 2.4a. We let $I_{\boldsymbol{b}}(\boldsymbol{p})$ denote the goods bid $\boldsymbol{b}$ demands at $\boldsymbol{p}$, i.e. $I_{\boldsymbol{b}}(\boldsymbol{p}) = \arg\max_{i \in [n]_0} \frac{b_i}{p_i}$. If $I_{\boldsymbol{b}}(\boldsymbol{p})$ contains more than one good, we say that $\boldsymbol{b}$ is *indifferent* between these goods at $\boldsymbol{p}$. A bid may be indifferent between demanding goods and being rejected when $\max_{i \in [n]_0} \frac{b_i}{p_i} = 1$.

At given market prices $\boldsymbol{p}$, each bid spends its budget $\beta(\boldsymbol{b})$ on a combination of goods $I_{\boldsymbol{b}}(\boldsymbol{p})$. Hence, we define its *demand correspondence* $D_{\boldsymbol{b}}$ as the set of bundles $D_{\boldsymbol{b}}(\boldsymbol{p}) := \left\{\boldsymbol{x} \in \mathbb{R}_+^n \mid \boldsymbol{x} \cdot \boldsymbol{p} = \beta(\boldsymbol{b}) \text{ and } x_i > 0 \implies i \in I_{\boldsymbol{b}}(\boldsymbol{p})\right\}$. The demand correspondence $D_{\mathcal{B}}$ of a bid list $\mathcal{B}$ is then given by the Minkowski sum $D_{\mathcal{B}}(\boldsymbol{p}) = \left\{\sum_{\boldsymbol{b}\in\mathcal{B}} \boldsymbol{x}^{\boldsymbol{b}} \mid \boldsymbol{x}^{\boldsymbol{b}} \in D_{\boldsymbol{b}}(\boldsymbol{p})\right\}$. This is illustrated for non-marginal prices in Fig. 2.4b.

Computing demand. Suppose we wish to compute, for a given list of arctic bids and price vector $\boldsymbol{p}$, a bundle $\boldsymbol{x} \in D_{\boldsymbol{b}}(\boldsymbol{p})$ demanded at $\boldsymbol{x}$. In the tropical language, we allowed bid weights to be positive or negative, which introduced complexities when $\boldsymbol{p}$ is marginal. In the arctic language, each bid's budget is positive, so we can independently compute a demanded bundle for each bid, and compute the sum over all bids.

2.7 Conclusion and outlook

This chapter has provided an overview of bidding languages in the literature and introduced the arctic and tropical product-mix languages in detail, as well as establishing the connection between OXS and positive tropical bids. The remaining chapters of this work answer fundamental questions about the computational complexity arising from the product-mix languages. Chapter 3 establishes that the strong-substitutes product-mix auction, in which bidders express preferences in the tropical language, can be solved for social welfare in polynomial time. In Chapter 4, we see that the same no longer holds for revenue maximisation, as this problem is APX-hard.

Beyond strong-substitutes. While the current tropical bidding language is only able to express strong-substitutes demand, we have seen a principled way of extending the language to general substitutes. It is an interesting question whether it is possible to further generalise this language to incorporate (limited) complements, perhaps using elements from logical languages. Moreover, it is an open question whether more expressible logical or product-mix languages that capture complements as well as substitutes retain tractability for the computational challenges outlined above, such as solving auctions with the social welfare objective. Note that competitive equilibrium need not exist in the multi-good, multi-unit setting when bidders have preferences that are not strong-substitutes. Instead, social welfare is maximised under quasi-equilibrium.

Generalising arctic bids. While bidding languages for indivisible goods have been well-studied, the same does not hold for divisible goods. The arctic bidding language is of particular interest, as its corresponding arctic auction has been studied algorithmically (cf. Fichtl (2020) and Chapter 6) and admits a novel coincidence of welfare and revenue described in Chapter 6. In order to broaden the class of preferences expressible with arctic bids, it is of interest to generalise these bids to positive *and* negative budgets. It is an open question whether this affects

the coincidence of welfare and revenue, or the complexity of solving the arctic auction for either solution concept.

3

Solving the strong-substitutes product-mix auction

Contents

3.1 Chapter outline 44
3.2 Introduction 45
3.3 The computational challenges 48
3.4 Finding market-clearing prices 51
3.4.1 Longer step sizes 54
3.4.2 Some practical improvements 57
3.5 Allocations to the individual bidders 58
3.5.1 The generalised allocation problem 60
3.5.2 The marginal bids graph and the derived graph 63
3.5.3 Allocating unambiguous bids 66
3.5.4 The shift-project-unshift reduction 68
3.5.5 The main algorithm 77
3.6 Experiments 79
3.6.1 Generating test data 79
3.6.2 Testing the allocation algorithm 80
3.6.3 Results 81
3.7 Conclusion and outlook 81

3.1 Chapter outline

This chapter develops polynomial-time algorithms to solve strong-substitutes product-mix auctions by finding a competitive equilibrium, which consists of market-clearing prices and an envy-free allocation of supply to bidders who use the tropical bidding

language (cf. Chapter 2) to truthfully express their strong-substitutes preferences over an arbitrary number of goods, each of which being available in multiple indivisible units.[1] Our use of the tropical bidding language, and the information it provides, contrasts with existing algorithms that rely on access to a valuation or demand oracle, and establishes the tropical language as a tractable language for expressing strong-substitutes preferences. This work appears in (Baldwin et al., 2021b) and is joint work with Elizabeth Baldwin, Paul Goldberg and Paul Klemperer.

Section 3.2 introduces the strong-substitutes product-mix auction in detail. In Section 3.3, we formally set out the computational tasks involved in computing competitive equilibrium, and show that the solution concept is well-defined. The first computational challenge, determining prices at which the market is cleared (so the entire supply is demanded), is solved in Section 3.4 using steepest-descent algorithms that apply submodular minimisation. Allocating the supply among the bidders at these prices then requires solving a novel constrained matching problem. Our allocation algorithm in Section 3.5 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 strongly polynomial running time bounds on both price-finding and allocation, and illustrate experimentally that our allocation mechanism is practical in Section 3.6. Finally, Section 3.7 concludes with further research directions.

3.2 Introduction

The *strong-substitutes 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.) In the product-mix auction, participants submit their preferences in the tropical bidding language presented in Chapter 2. In our setting, we allow bidders to submit lists consisting of positive *and* negative tropical bids. As shown in (Baldwin and Klemperer, 2021), this allows bidders to express any strong-substitutes preferences (introduced in Chapter 1) for an arbitrary number of distinct goods. In contrast, the original implementation of the auction by Bank of England restricted bidders to positive bids only.

The auction sets uniform prices $\boldsymbol{p} \in \mathbb{R}^n$ for the n goods (i.e. every recipient of any particular good pays the same price per unit for that good), and gives each bidder with bid list $\mathcal{B}$ an allocation that lies in its demand set $D_{\mathcal{B}}(\boldsymbol{p})$ as defined in Chapter 2. The auction's prices are chosen to maximise social welfare, which consists of the sum of the bids' and auctioneer's utilities – that is, it finds competitive equilibrium prices and quantities.[2] Moreover, the auctioneer expresses

[1] As discussed in Chapter 1, truthfulness is a reasonable assumption when we have many bidders that act as price-takers; that is, they cannot gain from acting strategically.

[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

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] It is a standard result that the first and second welfare theorems hold in our auction setting (Bichler et al., 2021; Nisan et al., 2007). The first theorem states that social welfare is maximised at a competitive equilibrium, while the second theorem conversely establishes that any social welfare maximising solution is a competitive equilibrium. Hence, the strong-substitutes product-mix auctions is solved to maximise social welfare.

Our contributions. This chapter 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-substitutes 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.

The algorithms for finding component-wise minimal market-clearing prices that we consider in Section 3.4 use a discrete steepest descent method from the discrete optimisation literature (Murota, 2003; Shioura, 2017) that employs submodular minimisation to find the steepest descent directions. We discuss a technique of (Shioura, 2017) that reduces the number of iterations required by taking 'longer steps' in the steepest descent direction. We also present a second method for computing step lengths, and show that both methods yield algorithms for price finding that are fully polynomial in the input size of the bid lists. Since submodular minimisation is rapid in practice (Chakrabarty et al., 2017), we expect our approach to be efficient. Indeed, the results of preliminary experiments in Section 3.6 suggest our method performs well.

Our main contribution, given in Section 3.5, 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

equilibrium price vectors, and also a unique one among them at which every good's price is lowest.

[3]Appendix 1E of (Klemperer, 2018) describes 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 a full discussion.

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 a practical implementation in Python that runs on all major software platforms; this can be found at `https://github.com/edwinlock/product-mix`. We also point towards an existing Haskell implementation developed by Well-Typed[4] that is available for Windows at `http://pma.nuff.ox.ac.uk`. Some effort has been 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 Section 3.6, 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 combinatorial auction setting 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 (Murota et al., 2016; Shioura and Tamura, 2015) 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

[4] `https://well-typed.com`

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 3.3. In particular, the valuation oracle setting provides no straightforward way to compute a demanded bundle or the indirect utility function at given prices.

When preferences are expressed in the tropical bidding language, it is not straightforward to determine a bidder's valuation of a given bundle, so the allocation algorithms of Murota (2003), and (Paes Leme and Wong, 2017) cannot be directly applied. Indeed, 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 3.4, computing the indirect utility $f_{\mathcal{B}}(\boldsymbol{p})$ at this price and then solving (3.1) for $v(\boldsymbol{x})$; this is not effective in practice. However, our algorithms for computing competitive equilibrium do not require access to a valuation function but instead directly use the information encoded in the tropical bids to efficiently compute demanded bundles as well as the indirect utility derived at any given prices. We note that our use of the tropical language is somewhat analogous to but more informative than the demand oracle setting of (Ausubel, 2006) and (Paes Leme and Wong, 2017), who presuppose access to an oracle returning a demanded bundle at given prices to compute market-clearing prices.

This chapter 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. In Chapter 5, we 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.

3.3 The computational challenges

We now state the computational problems to solve the strong-substitutes product-mix auction on n goods. Let J denote the set of bidders and let $\mathcal{B}^j$ denote the bids that bidder j submits, for all bidders $j \in J$. We assume each $\mathcal{B}^j$ is *valid*, as defined in Section 2.5.3, 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 3.1a, for instance, depicts the aggregate bid list of Alice and Bob from Fig. 2.2 in Chapter 2. We will express the running times of our algorithms in

terms of n, $|J|$, $|\mathcal{B}|$ and $M \coloneqq \max_{\boldsymbol{b}\in\mathcal{B}} \|\boldsymbol{b}\|_\infty$, the maximum bid vector entry.

Suppose the auctioneer intends to sell its entire supply $\boldsymbol{s}$. Our aim is to compute a competitive equilibrium: a market-clearing price vector $\boldsymbol{p}$ at which $\boldsymbol{s}$ is demanded and an allocation of $\boldsymbol{s}$ 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 supply 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 $\|\boldsymbol{s}\|_1+1$ positive bids at $\mathbf{0}$.[5] The computation of a competitive equilibrium separates into two problems.

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

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

Theorem 3.1, a standard result which can be found in (Danilov et al., 2001; Milgrom and Strulovici, 2009; Murota, 2003), ensures that the allocation problem is well-defined: if the supply bundle $\boldsymbol{s}$ is aggregately demanded at market-clearing price $\boldsymbol{p}$, then there exists an allocation of supply $\boldsymbol{s}$ among the bidders so that every bidder receives a bundle she demands.

Theorem 3.1 ((Danilov et al., 2001; Milgrom and Strulovici, 2009)). *Suppose that each bidder $j \in J$ has strong-substitutes demand correspondence D^j. Given a supply bundle $\boldsymbol{s}$ and a market-clearing price $\boldsymbol{p}$ for $\boldsymbol{s}$, there exists a partition $\boldsymbol{s} = \sum_{j\in J} \boldsymbol{s}^j$ such that $\boldsymbol{s}^j \in D^j(\boldsymbol{p})$.*

In order to solve these two problems, we recall the following from Chapter 1. Let $\mathcal{B}$ be any valid bid list. By Theorem 2.5 in Chapter 2, the indirect utility function $f_{\mathcal{B}} = \sum_{\boldsymbol{b}\in\mathcal{B}} w(\boldsymbol{b}) \max_{i\in[n]_0}(b_i - p_i)$ of $\mathcal{B}$ as defined in (2.1) is the indirect utility function f_v of some strong-substitutes valuation v. Hence, using (1.1) we have

$$f_{\mathcal{B}}(\boldsymbol{p}) = f_v(\boldsymbol{p}) = v(\boldsymbol{x}) - \boldsymbol{p}\cdot\boldsymbol{x}, \text{ for any } \boldsymbol{x} \in D_{\mathcal{B}}(\boldsymbol{p}) = D_v(\boldsymbol{p}). \tag{3.1}$$

Positive bids only. Suppose bidders only submit positive bids, which corresponds to the current implementation by the Bank of England. Hence they essentially express OXS preferences, as shown in Chapter 2. In this case, finding an envy-free allocation of $\boldsymbol{s}$ to the bidders is a straightforward application of solving a max-flow problem. Indeed, suppose $\boldsymbol{p}$ are market-clearing prices. Then we can construct a network who vertices consists of a source, a sink, the bids $\mathcal{B}$ and the goods $[n]_0$. We add the following arcs: from the source to each bid, an arc with capacity 1; from each good i to the sink, an arc with capacity s_i (where $s_0 \coloneqq \infty$); and for

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

each bid $\boldsymbol{b}$ that demands good $i \in [n]_0$ at $\boldsymbol{p}$, an arc $(\boldsymbol{b}, i)$ with capacity ∞. Recall that every flow network with integral capacities admits an integral max flow that we can find with the Edmonds-Karp algorithm (Ahuja et al., 1993). Moreover, there is a one-to-one correspondence between integral max flows of this network and envy-free allocations at prices $\boldsymbol{p}$.

Positive and negative bids. However, in our setting bidders are permitted to submit both positive *and* negative bids in order to express arbitrary strong-substitutes preferences. In this case, care must be taken in finding an allocation that is envy-free for all bidders, as bidders can no longer simply be allocated an arbitrary bundle they demand at the market-clearing price. This is discussed in more details in our discussion about demand correspondences of tropical bid lists in Chapter 2. Indeed, consider the lowest prices $\boldsymbol{p} = (4, 4)$ at which Alice and Bob aggregately demand $(1, 1)$ in Fig. 2.2. The aggregate demand correspondence of Alice and bob is shown in Fig. 3.1a. If we allocate Alice the whole bundle $(1, 1)$, Bob does not demand the empty remainder $(0, 0)$ at $\boldsymbol{p}$. These inter-dependencies between bids gives rise to a novel matching problem that we solve with our allocation algorithm.

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 supply 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 Fig. 3.1a.)

If we had (oracle) access to the valuations of the bidders, we could perturb the valuations such that the supply bundle is uniquely demanded at some price by subtracting carefully constructed functions that are discrete-convex in one variable.[6] However, recall that our bidding language does not give straightforward access to bidder valuations. 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 to all the bids in bid set $\mathcal{B}^j$ (a different vector 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 3.25, that small perturbations of bids make only small changes to the market clearing price(s).) However, the perturbations need 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 allocation algorithm avoids this by systematically perturbing and un-perturbing bid sets, making decisions on marginal bids as it proceeds.

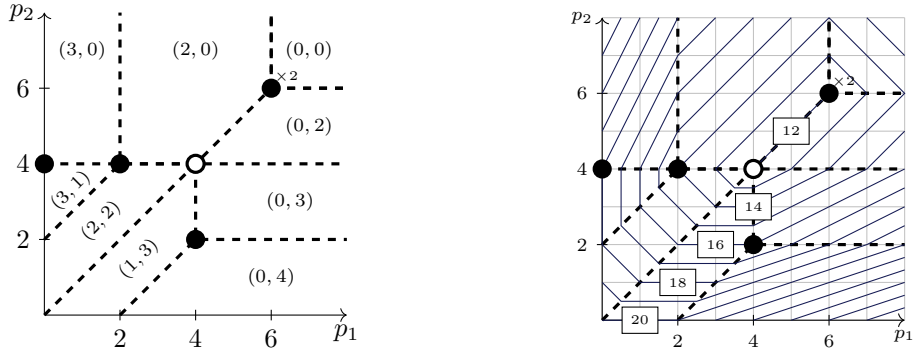
[6] See (Murota, 2003, Theorem 6.13 (4)).

Submodular minimisation. Our algorithms for finding competitive equilibrium rely on a submodular minimisation subroutine. Recall that 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 time polynomial in n and γ, where γ denotes the time it takes to query f, using the improved cutting plane method of Lee et al. (2015). For SFM algorithms that are simple to implement and appear to be efficient in practice, at least for certain problem instances, 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. 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 3.4, 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 $S_0 \coloneqq \{v \in S \mid f(S_v) > f(S)\}$.

3.4 Finding market-clearing prices

In order to determine component-wise minimal prices at which the market is cleared, we apply methods from the discrete convex optimisation literature. Specifically, 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 use the SFM subroutine described in Chapter 1 to find the component-wise minimal discrete steepest descent direction. While the first algorithm takes unit steps, the second algorithm (LongStepMinUp) improves on this by taking longer steps in the descent direction. Shioura (2017) suggests a binary search technique to compute the length of the longer step.

Our contribution in this section is to present a second method for computing step lengths and to show that both methods yield a fully polynomial running time in the encoding size of the aggregate bid list. We note that an alternative approach to finding market-clearing prices that also exploits the direct knowledge provided by the tropical language was recently presented by Baldwin et al. (2021a). Their algorithm minimises the difference in the objective functions of two linear programs; the first program corresponds to one that we would solve if we restricted ourselves to the positive bids only and the second linear program corresponds to the negative bids. This algorithm is competitive in practice but, unlike our approach, it is unknown whether it runs in polynomial time.



(a) Aggregate demand of Alice and Bob

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

Figure 3.1: Left: The demand of the aggregate bid list obtained by aggregating Alice's and Bob's bids from Fig. 2.2 in Chapter 2. Right: A contour plot of the Lyapunov function g for the aggregate bid list and supply bundle $\boldsymbol{s} = (1, 1)$. Note that $\boldsymbol{p} = (4, 4)$ is the minimal market-clearing price for $\boldsymbol{s}$. 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 $\boldsymbol{p}$.

We now summarise the standard discrete steepest descent algorithms. In order to apply the steepest descent method to our price-finding problem, we define a Lyapunov function g and note in Proposition 3.2 that its restriction to $\mathbb{Z}^n_+$ is $L^\natural$-convex. Moreover, Lemma 3.3 states that the lowest market-clearing price is integral and finding it reduces to determining the minimal minimiser of g. We note that this approach generalises an algorithm used by Ausubel's ascending auction design (Ausubel, 2006) and Gul and Stacchetti (1999) for the task of finding equilibrium prices in single-unit markets. The Lyapunov function for the bid list illustrated in Fig. 3.1a is shown in Fig. 3.1b.

Recall that our bid list $\mathcal{B}$ correspond to a strong-substitutes valuation v. The *Lyapunov function* with regard to indirect utility function f_v of this valuation and supply bundle $\boldsymbol{s}$ is defined as $g_{\boldsymbol{s}}(\boldsymbol{p}) \coloneqq f_v(\boldsymbol{p}) + \boldsymbol{s} \cdot \boldsymbol{p}$. We suppress the subscript $\boldsymbol{s}$ if it is clear from context. Using (3.1), we can express $g_{\boldsymbol{s}}$ for list $\mathcal{B}$ as

$$g_{\boldsymbol{s}}(\boldsymbol{p}) \coloneqq f_{\mathcal{B}}(\boldsymbol{p}) + \boldsymbol{s} \cdot \boldsymbol{p} = \sum_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b}) \max_{i \in [n]_0} (b_i - p_i) + \boldsymbol{s} \cdot \boldsymbol{p}. \tag{3.2}$$

From (3.2) it is clear that we can evaluate g at any price $\boldsymbol{p}$ in time $O(n|\mathcal{B}|)$.

Recall from Chapter 1 that a function $f : \mathbb{Z}^n \to \mathbb{R}$ is $L^\natural$-convex if it satisfies the *translation submodularity* property,

$$f(\boldsymbol{p}) + f(\boldsymbol{q}) \geq f((\boldsymbol{p} - \alpha\mathbf{1}) \vee \boldsymbol{q}) + f(\boldsymbol{p} \wedge (\boldsymbol{q} + \alpha\mathbf{1})) \quad (\forall \boldsymbol{p}, \boldsymbol{q} \in \mathbb{Z}^n_+, \forall \alpha \in \mathbb{Z}_+). \tag{3.3}$$

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

Proposition 3.2 and Lemma 3.3 are well-known (cf. (Murota et al., 2013, 2016)) and demonstrate that we can use algorithms for minimising $L^\natural$-convex functions to find the minimal price $\boldsymbol{p}^*$ at which $\boldsymbol{s}$ is demanded. We provide short proofs, as they nicely illustrate the structure of the Lyapunov function.

Proposition 3.2 ((Murota et al., 2013, 2016)). *The Lyapunov function $g_{\boldsymbol{s}}$ restricted to $\mathbb{Z}^n_+$ is $L^\natural$-convex.*

Proof. As mentioned in Chapter 1, the indirect utility function f_v restricted to $\mathbb{Z}^n$ is $L^\natural$-convex if, and only if, the valuation function v is strong-substitutes (see also (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 $f_{\mathcal{B}} = f_v$ for some strong-substitutes valuation v and it follows that the function $f_{\mathcal{B}}(\boldsymbol{p}) + \boldsymbol{s} \cdot \boldsymbol{p}$ is $L^\natural$-convex. □

Lemma 3.3 (cf. (Murota et al., 2016)). *The Lyapunov function $g_{\boldsymbol{s}}$ with regard to a valid (integral) bid list and any supply bundle $\boldsymbol{s}$ is convex, and $\boldsymbol{p}$ is a minimiser of g if, and only if, it is a market-clearing price for supply bundle $\boldsymbol{s}$. Moreover, the minimal minimiser of g is integral.*

Proof. The first statement is immediate from the fact that $f_{\mathcal{B}}$ is convex and $\boldsymbol{s} \cdot \boldsymbol{p}$ is a linear term. To see the second statement, note that for any market-clearing price $\boldsymbol{p}$ of supply bundle $\boldsymbol{s}$, we have $g(\boldsymbol{p}) = v(\boldsymbol{s})$, whereas for any price $\boldsymbol{p}$ at which $\boldsymbol{s}$ is not demanded, we have $g(\boldsymbol{p}) = \max_{\boldsymbol{x} \in D(\boldsymbol{p})}(v(\boldsymbol{p}) - \boldsymbol{x} \cdot \boldsymbol{p}) + \boldsymbol{s} \cdot \boldsymbol{p} > v(\boldsymbol{s})$. Here v is the valuation function corresponding to $\mathcal{B}$ as stated in Theorem 2.5 and we use (3.1).

Integrality of the minimal minimiser of g was shown in (Murota et al., 2016, Corollary 4.4). Alternatively, we can see that this 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* $\boldsymbol{e}^{S_0}$. Moreover, as we can evaluate g in time $O(n|\mathcal{B}|)$, it follows that finding this minimal descent direction can be achieved in time polynomial in n and $|\mathcal{B}|$. We will refer to this time as $T(n, |\mathcal{B}|)$.

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 3.1 (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 3.1 MinUp

1: Pick a point $\boldsymbol{p} \leq \boldsymbol{p}^*$ (e.g. $\boldsymbol{p} = \boldsymbol{0}$).
2: Find the inclusion-wise minimal set $S_0 \subseteq [n]$ 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.
7: **end if**

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 Algorithm 3.1 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 Chapter 1 that finds a minimal minimiser of a given function in time $T(n, |\mathcal{B}|)$. This leads to the following running time for MinUp.

Theorem 3.4 (cf. (Murota and Shioura, 2014)). *The algorithm MinUp finds the component-wise minimal market-clearing price in time $O(MnT(n, |\mathcal{B}|))$, where $T(n, |\mathcal{B}|)$ 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 Chapter 1 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 $\boldsymbol{s}$ and $\boldsymbol{x}$, this implies $g'(\boldsymbol{p}; S) = (\boldsymbol{s} - \boldsymbol{x}) \cdot \boldsymbol{e}^S \leq |\mathcal{B}|$.

3.4.1 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

$$\lambda(\boldsymbol{p}, S_0) = \max\{\lambda \in \mathbb{Z}_+ \mid g'(\boldsymbol{p}; S_0) = g'(\boldsymbol{p} + (\lambda - 1)\boldsymbol{e}^{S_0}; S_0)\}, \tag{3.4}$$

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

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

Algorithm 3.2 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 (3.4), using a method from Section 3.4.1.1.
4: **if** $S_0 \neq \emptyset$ **then**
5: Set $\boldsymbol{p} = \boldsymbol{p} + \lambda(\boldsymbol{p}, S_0)\boldsymbol{e}^{S_0}$ and go to line 2.
6: **end if**
7: **return** $\boldsymbol{p}$.

in the direction of $\boldsymbol{e}^{S_0}$ changes. We give two methods to compute (3.4) in Section 3.4.1.1. This leads to the following algorithm (referred to as GreedyUp-LS in (Shioura, 2017)).

We note that the values of $\boldsymbol{p}$ follow the same trajectory for MinUp and LongStepMinUp. This is an immediate consequence of Lemma 3.6, which rests on the monotonicity properties of $g'(\boldsymbol{p}, S_0)$ stated in Proposition 3.5.

Proposition 3.5 ((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 3.6. *Let S_0 denote the minimal steepest descent at $\boldsymbol{p}$ and let $\lambda(\boldsymbol{p}, S_0)$ be defined by* (3.4). *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 3.7 ((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

$$g'(\boldsymbol{0}; S) = g(\boldsymbol{e}^S) - g(\boldsymbol{0}) = f_v(\boldsymbol{e}^S) + \boldsymbol{s} \cdot \boldsymbol{e}^S - f_v(\boldsymbol{0}) = (\boldsymbol{s} - \boldsymbol{x}) \cdot \boldsymbol{e}^S \geq -\sum_{i \in [n]} x_i \geq -|\mathcal{B}|,$$

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.

3.4.1.1 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 (3.4). The first method uses binary search and is also suggested in (Shioura, 2017). The second method is novel and 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 3.8. *The* LongStepMinUp *algorithm in combination with the* binary search *and* demand change *methods has a respective running time of* $O(n^2|\mathcal{B}|^2 \log M + n|\mathcal{B}|T(n,|\mathcal{B}|))$ *and* $O(n^2|\mathcal{B}|^3 + n|\mathcal{B}|T(n,|\mathcal{B}|))$.

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 (proposed by Shioura (2017)). 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 3.5, 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 3.9 proves that the demand-change procedure correctly computes the value of $\lambda(\boldsymbol{p}; S_0)$ in time $O(n|\mathcal{B}|^2)$.

Lemma 3.9. *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 3.5 implies that it suffices to show

$$g'(\boldsymbol{p}; S_0) = g'(\boldsymbol{p}^K - \boldsymbol{e}^S; S_0) \text{ and } g'(\boldsymbol{p}; S_0) < g'(\boldsymbol{p}^K; S_0) \tag{3.5}$$

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, by applying (3.1), we get

$$g'(\boldsymbol{p}^{k-1}; S_0) = (\boldsymbol{s} - \boldsymbol{x}) \cdot \boldsymbol{e}^{S_0} = g'(\boldsymbol{p}^k - \boldsymbol{e}^{S_0}; S_0).$$

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 (3.5).

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 $\mu \geq \mu_{\boldsymbol{b}} + 1$. □

3.4.2 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

$$g'(\boldsymbol{p}; S_0) = g(\boldsymbol{p} + \boldsymbol{e}^{S_0}) - g(\boldsymbol{p}) = (\boldsymbol{s} - \boldsymbol{x}) \cdot \boldsymbol{e}^{S_0},$$

and S_0 minimises this term if, and only if, S_0 contains all indices $i \in [n]$ with $s_i < x_i$ and no indices $j \in [n]$ with $s_j > x_j$. In particular, we have $S_0 \coloneqq \{i \in [n] \mid s_i < x_i\}$ 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

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

we have $I_{\boldsymbol{p}} \subseteq S_0$ and $J_{\boldsymbol{p}} \cap S_0 = \emptyset$. (Note that the inequalities in the definitions of $I_{\boldsymbol{p}}$ and $J_{\boldsymbol{p}}$ are strict and not strict, respectively.) 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 3.10. *$I_{\boldsymbol{p}}$ and $J_{\boldsymbol{p}}$ can be computed in time $O(n|\mathcal{B}|)$.*

Proof. Note that $I_{\boldsymbol{p}} = \{i \in [n] \mid \min_{\boldsymbol{x} \in D(\boldsymbol{p})} x_i > s_i\}$ and $J_{\boldsymbol{p}} = \{i \in [n] \mid \max_{\boldsymbol{x} \in D(\boldsymbol{p})} x_i \leq s_i\}$, 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

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

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. □

3.5 Allocations to the individual bidders

Suppose we are given a market-clearing price $\boldsymbol{p}$ for supply bundle $\boldsymbol{s}$. We now present an algorithm that solves the allocation problem, i.e. finds a partition $(\boldsymbol{s}^j)_{j \in J}$ of the supply bundle $\boldsymbol{s}$ such that $\boldsymbol{s}^j$ is demanded by bidder j at price $\boldsymbol{p}$. We note that while the market-clearing prices $\boldsymbol{p}$ returned by our steepest descent approach in Section 3.4 is component-wise minimal, our allocation algorithm works for any integral market-clearing prices.[7]

Our algorithm Allocate, illustrated in Fig. 3.2 and stated formally in Algorithm 3.3, repeatedly simplifies the problem until it becomes vacuous. It generates a sequence of *allocation problems* by iteratively allocating parts of the supply bundle to bidders and removing satisfied bids from the bid lists until the residual supply bundle (initially the supply bundle) is empty. In order to capture this formally, we define a generalised allocation problem in Section 3.5.1; 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 supply 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 3.5.2. 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.

[7] Recall that the component-wise minimal market-clearing prices are always integral by Lemma 3.3. This is a consequence of the fact that tropical bid vectors are integral by definition. If we allowed non-integral bids, we could apply the same allocation algorithm by scaling up bids, or modify some constants for shifting and perturbing bids.

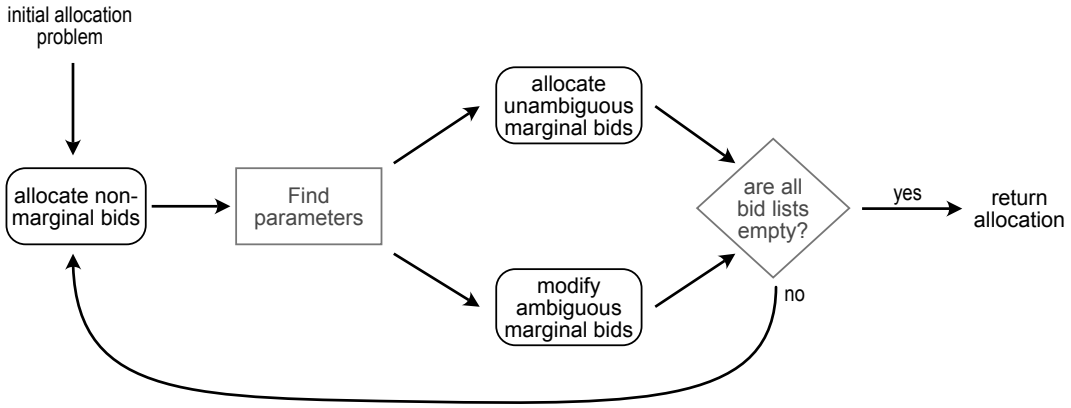


Figure 3.2: An overview of the allocation algorithm, which repeatedly applies two out of three subroutines (depicted as rounded rectangles) to simplify the problem until it becomes vacuous (which is asserted at the diamond step). A special subroutine FindParams, depicted as a greyed-out rectangle, is used to decide which of the two 'marginal bid' subroutines to call in each iteration, and computes parameters required for this subroutine.

Algorithm 3.3 Allocate

1: **Input:** Initial allocation problem $\mathcal{A}$.
2: **Output:** Supply bundle allocation.
3: Allocate all non-marginal bids by applying NonMarginals (Algorithm 3.5).
4: **while** $\mathcal{A}$ has a non-empty bid list **do**
5: Run the FindParams subroutine (Algorithm 3.4).
6: **if** FindParams returns a demand cluster (I, j^*) with at most one link good i^* **then**
7: Process unambiguous marginal bids with UnambiguousMarginals (Algorithm 3.6).
8: **else**
9: Simplify the allocation problem with ShiftProjectUnshift (Algorithm 3.7).
10: **end if**
11: Allocate all (newly) non-marginal bids with NonMarginals (Algorithm 3.5).
12: **end while**
13: **return** the partial allocation bundle $\boldsymbol{m}^j$ for each bidder $j \in J$.

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 3.5.3. Next, the algorithm calls the FindParams subroutine (defined in Section 3.5.2), 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 3.5.3) to process these bids. Otherwise FindParams identifies parameters that Allocate uses to call

the ShiftProjectUnshift procedure. ShiftProjectUnshift is defined in Section 3.5.4; 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 supply bundle $\boldsymbol{s}$ at prices $\boldsymbol{p}$. Section 3.5.5 gives the running time and a proof of correctness for Allocate.

3.5.1 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 2.5. (Figure 3.4 in Section 3.5.3 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 3.11. 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 $w(\boldsymbol{b})$ of the bids $\boldsymbol{b} \in \mathcal{B}$ 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 3.12. *The two criteria in Definition 3.11 are equivalent.*

We make use of the following technical observation to prove Proposition 3.12.

Observation 3.13. *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 3.12. Recall that $f_{\mathcal{B}}$ restricted to P is convex if, and only if,

$$f_{\mathcal{B}}(\lambda \boldsymbol{p} + (1 - \lambda)\boldsymbol{q}) \leq \lambda f_{\mathcal{B}}(\boldsymbol{p}) + (1 - \lambda) f_{\mathcal{B}}(\boldsymbol{q}), \quad \forall \boldsymbol{p}, \boldsymbol{q} \in P \text{ and } \lambda \in [0, 1]. \tag{3.6}$$

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 in 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 (3.6) 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 $\sum_{\boldsymbol{b}\in\mathcal{B}_{ii'}} w(\boldsymbol{b}) \geq 0$. 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

$$f_{\mathcal{B}}(\boldsymbol{r}) \leq \lambda f_{\mathcal{B}}(\boldsymbol{p}) + (1-\lambda) f_{\mathcal{B}}(\boldsymbol{q}) \tag{3.7}$$

holds if, and only if,

$$\sum_{\boldsymbol{b}\in\mathcal{B}_{ii'}} w(\boldsymbol{b})[(b_i - q_i) - (b_{i'} - q_{i'})] \leq 0. \tag{3.8}$$

(To see this, note that $f_{\mathcal{B}}(\boldsymbol{r}) = \lambda \sum_{\boldsymbol{b}\in\mathcal{B}_{ii'}} w(\boldsymbol{b})(b_i - p_i) + (1\lambda) \sum_{\boldsymbol{b}\in\mathcal{B}_{ii'}} w(\boldsymbol{b})(b_i - q_i)$.) 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 (3.8) that $\sum_{\boldsymbol{b}\in\mathcal{B}_{ij}} w(\boldsymbol{b}) \geq 0$ if, and only if, (3.7) 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 (3.6). Due to continuity of $f_{\mathcal{B}}$, we can perturb $\boldsymbol{p}$ and $\boldsymbol{q}$ slightly so that (3.6) 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 without loss of generality that $\boldsymbol{p}$ and $\boldsymbol{q}$ are in neighbouring UDRs; otherwise, we can apply Observation 3.13 to find two non-marginal prices on the interior of the line segment for which (3.6) 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. Without loss of generality, 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 (3.6) and we are done. □

Definition 3.14. 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 $\boldsymbol{s}^j \coloneqq \boldsymbol{r}^j + \boldsymbol{m}^j$ 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 supply bundle $\boldsymbol{s}$ 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 $\boldsymbol{s}$ already allocated to j, while $\boldsymbol{r}$ denotes the remaining part of $\boldsymbol{s}$ 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 3.15. 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 supply bundle and the partial allocation vectors are $\mathbf{0}$. Note that s_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 $\boldsymbol{s}$. 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 3.5.4 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 3.18 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 3.16. *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 3.17. *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 3.18. *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 $\boldsymbol{p}$.*

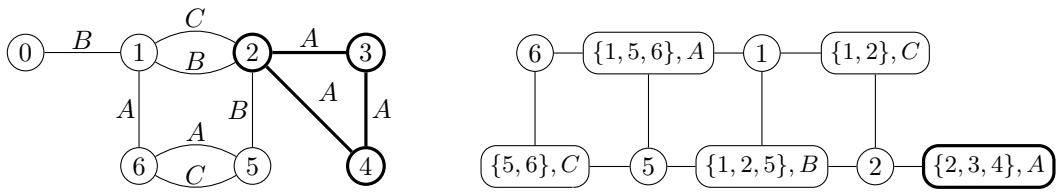
Proof. We show that $f_{\mathcal{B}}$ is convex on $B(\boldsymbol{p}, \varepsilon/2)$ by verifying that it satisfies midpoint convexity. Note that $f_{\mathcal{B}}$ is linear on the line segment connecting $\boldsymbol{p}$ to $\boldsymbol{q}$, for any $\boldsymbol{q} \in B(\boldsymbol{p}, \varepsilon/2)$. 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 3.16, 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 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, $f\left(\frac{1}{2}(\boldsymbol{q} + \boldsymbol{q}')\right) \leq \frac{1}{2}f(\boldsymbol{q}) + \frac{1}{2}f(\boldsymbol{q}')$. □

3.5.2 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 3.19. 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



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

Figure 3.3: Example of a marginal bids multigraph (a) and derived graph (b) with three bidders A, B, C and six goods $1, \ldots, 6$ together with reject good 0. 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).

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 3.20. 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 3.3 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 an $\alpha(n)$-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.

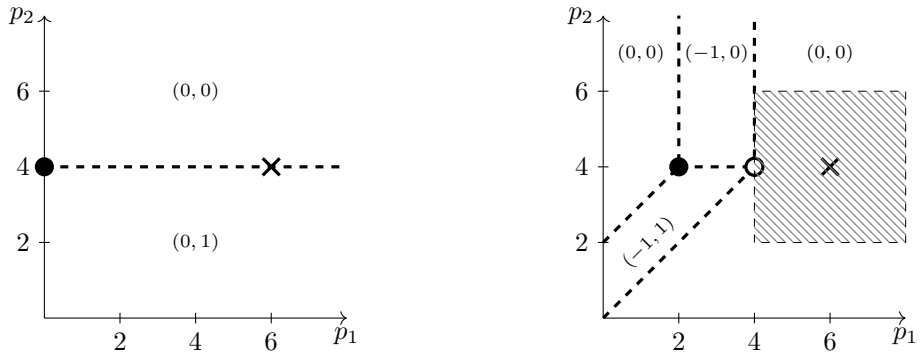
The FindParams subroutine. In every iteration, the ALLOCATE algorithm employs a subroutine to decide, using the derived graph, whether to invoke UNAMBIGUOUSMARGINALS or SHIFTPROJECTUNSHIFT, and to compute the input parameters for the respective procedure. This subroutine FINDPARAMS, given in Algorithm 3.4, 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 3.4 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).
end if
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^*.
end if

Lemma 3.21. *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.*



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

Figure 3.4: The resulting bid lists after applying NonMarginals (Algorithm 3.5) to Alice's and Bob's bid lists from Fig. 2.2 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).

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^*. □

3.5.3 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 3.5, 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 3.4 gives an example. Allocate calls NonMarginals in every iteration in order to process non-marginal bids that can be allocated unambiguously.

Lemma 3.22. *Given a valid allocation problem $\mathcal{A}$, NonMarginals (Algorithm 3.5) outputs a reduction $\mathcal{A}'$ of $\mathcal{A}$ in linear time. Moreover, we have $G_{\mathcal{A}} = G_{\mathcal{A}'}$.*

Algorithm 3.5 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 $w(\boldsymbol{b})$.
    end for
end for

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 utility function is also convex in some open neighbourhood of $\boldsymbol{p}$. It is straightforward to see that $(\boldsymbol{s}^j)_{j\in J}$ 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 3.6.

Lemma 3.23. *UnambiguousMarginals, given in Algorithm 3.6, 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 $G_{\mathcal{A}}$.*

Algorithm 3.6 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.
    end for
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.
    end for
end if
Remove all bids marginal on I from $\mathcal{B}^j$.

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 3.11. 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 demand clusters. 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. □

3.5.4 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 Shift-ProjectUnshift 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. Moreover,

as the new marginal bids graph has strictly fewer edges, this procedure has broken ties and we have made progress. 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 3.24 and Proposition 3.25 together show that we can use SFM to find a price $\boldsymbol{p}^\varepsilon \in \left\{\boldsymbol{p} + \varepsilon \boldsymbol{e}^S, S \subseteq [n]\right\}$, at which the new allocation problem $\mathcal{A}'$ with shifted bids and price vector $\boldsymbol{p}^\varepsilon$ is a valid reduction.

Lemma 3.24. *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 3.18, 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 3.16 and 3.17, 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 that 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 3.25. *Let $\mathcal{A}$ be an allocation problem with price vector $\boldsymbol{p}$ and fix ε with $|\varepsilon| < 1/4$. Suppose we shift all bids of bidder j's in $\mathcal{B}$ by ε to obtain (aggregate) bid list $\mathcal{B}'$. Then there exists a price $\boldsymbol{p}^\varepsilon \in \left\{\boldsymbol{p} \pm \varepsilon \boldsymbol{e}^S, S \subseteq [n]\right\}$, at which the residual supply $\boldsymbol{r}$ is demanded by $\mathcal{B}'$. We can determine $\boldsymbol{p}^\varepsilon$ using submodular minimisation of the Lyapunov function g with regard to list $\mathcal{B}'$ and the residual supply $\boldsymbol{r}$.*

In order to prove Proposition 3.25, we first establish a technical lemma and a proposition that illustrates the robustness of demand when we perturb valuations by some small value ε.

Lemma 3.26. *Suppose that $G = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$ is an invertible $n \times n$ matrix with strong-substitutes rows (cf. Definition 2.4). Moreover, let H_i denote the i-th column of the matrix $H =\coloneqq \begin{pmatrix} 0 \\ G_2 \end{pmatrix}$. Then, for any $i \in [n]$ we have either (i) $G^{-1}H_i = \boldsymbol{e}^S$ for some $S \subseteq [n]$ with $i \in S$, or (ii) $G^{-1}H_i = -\boldsymbol{e}^S$ for some $S \subseteq [n]$ with $i \notin S$.*

Proof. Suppose G is a lower triangular matrix with diagonal values $g_{ii} = 1$ and $g_{ij} \in \{0, -1\}$ for all $i > j$. This is without loss of generality, as we can permute columns and multiply by a diagonal matrix U with entries ± 1 on its diagonals otherwise; in this case, replace G by PUG (where P denotes the permutation

matrix), below and adjust the proof slightly. Note that the rows of PUG remain strong-substitutes.

In order to prove our lemma, we associate G with a directed graph $\mathcal{D}$ on vertices $[n]_0$ as follows. As G is invertible and the rows are strong-substitutes, each row takes the form $\boldsymbol{e}^j - \boldsymbol{e}^i$ for some distinct $i, j \in [n]_0$, where we define $\boldsymbol{e}^0 = \mathbf{0}$. Thus, for each row $\boldsymbol{e}^j - \boldsymbol{e}^i$, we add an arc from i to j.

We establish some properties of $\mathcal{D}$. Firstly, note that $\mathcal{D}$ is an arborescence rooted at 0; that is, it is a directed graph in which there is exactly one path from 0 to any other vertex. (Such a graph is also known as an out-tree.) This can be seen by induction on $k \in [n]$, if we define G_{kk} as the sub-matrix of G consisting of the first k rows and columns, and let $\mathcal{D}_{kk}$ denotes its graph. It is immediate that $\mathcal{D}_{11}$ is an arborescence. Secondly, consider $\mathcal{D}_{kk}$ and assume the claim holds for $k-1$. Then we see that the last row (and column) of G_{kk} adds a single arc from a good $j < k$ to k, so $\mathcal{D}_{kk}$ remains an arborescence.

Secondly, we can express G as $G = I_n - L$, where I_n is the identity matrix and L is a strictly lower triangular matrix. Moreover, L admits the intuitive interpretation of being the transpose of the adjacency matrix of $\mathcal{D}$. (To see this, note that $\boldsymbol{e}^j - \boldsymbol{e}^i$ is the j-th row in G by assumption, and so we have $L_{ji} = 1$ if, and only if, there exists an arc from i to j.) In the following, let L^k be the k-th power of L (with $L^0 = I_n$). We recall the well-known result that L^k_{ji} denotes the number of directed walks of length k from vertex i to j. In particular, as $\mathcal{D}$ is a tree, all walks are paths. This implies that, for all $i, j \in [n]$, we have $L^k_{ij} \in \{0, 1\}$ and $L^k_{ij} = 1$ for at most one value of k.Moreover, $L^n = \mathbf{0}$, as no path can have length n. It follows that the inverse of G is

$$G^{-1} = \sum_{k=0}^{n-1} L^k, \tag{3.9}$$

and G^{-1} has only entries $\{0, 1\}$.

We now prove the following claim, which immediately implies our main claim.

Claim 3.27. *Fix $i \in [n]$ and let J_i be any subset of the in-neighbours of i (in-neighbours are the vertices j for which there exists an edge from i to j). Then the following holds.*

1. *For $\boldsymbol{h} = -\sum_{j \in J_i} \boldsymbol{e}^j$, there exists a subset S such that $i \in S$ and $G^{-1}h = -\boldsymbol{e}^S$.*
2. *For $\boldsymbol{h} = \boldsymbol{e}^i - \sum_{j \in J_i} \boldsymbol{e}^j$, there exists a subset S such that $i \in S$ and $G^1 h = \boldsymbol{e}^S$.*

Proof. We note that $J_i \subseteq \{i+1, \ldots, n\}$. For any $j \in [n]_0$, let $S(j)$ denote the vertices reachable from vertex j in $\mathcal{D}$. As $L^k_{jl} = 1$ if there exists a path of length k from vertex l to j, and $L^k_{jl} = 0$ otherwise, (3.9) implies that $G^{-1}\boldsymbol{e}^i = \boldsymbol{e}^{S(j)}$ is the indicator vector denoting, for each $l \in [n]$, whether l is reachable from j (along a path of arbitrary length).

Next, observe that J_i contains endpoints of arcs from i, so the arborescence property of $\mathcal{D}$ implies that the sets $S(j)$ for $j \in J_i$ are pairwise disjoint. Moreover, we see that $S(j) \subseteq S(i)$, as there is an arc from i to every $j \in J_i$. This allows us to say the following.

1. If $\boldsymbol{h} = \sum_{j \in J_i} \boldsymbol{e}^j$, then we have $G^{-1}\boldsymbol{h} = -\sum_{j \in J_i} \boldsymbol{e}^{S(j)} = \boldsymbol{e}^S$, where we define $S \coloneqq \bigcup_{j \in J_i} S(j)$, and we note that $i \notin S$.
2. If $\boldsymbol{h} = \boldsymbol{e}^i - \sum_{j \in J_i} \boldsymbol{e}^j$, then we have $G^{-1}\boldsymbol{h} = \boldsymbol{e}^{S(i)} - \sum_{j \in J_i} \boldsymbol{e}^{S(j)} = \boldsymbol{e}^S$, where we define $S \coloneqq S(i) \setminus \bigcup_{j \in J_i} S(j)$, and we note that $i \in S$. The last equality holds because $S(j) \subseteq S(i)$ for all $j \in J_i$. □

To see why Claim 3.27 proves our main claim, note that H_i takes the form $H_i = -\sum_{j \in J_i} \boldsymbol{e}^j$ or $H_i = \boldsymbol{e}^i - \sum_{j \in J_i} \boldsymbol{e}^j$ for some J_i. This concludes the proof. □

Observation 3.28. *Suppose that G is an $n' \times n$ matrix with strong-substitutes rows (cf. Definition 2.4) and that $S \subseteq [n]$. Then $G\boldsymbol{e}^S \in \{-1, 0, 1\}^n$ and, in particular, $G\boldsymbol{e}^S \geq -\mathbf{1}$.*

Proposition 3.29. *Suppose we have two strong-substitutes integer-valued valuations, v^1 and v^2, and let $v^{\{1,2\}}$ denote the aggregate valuation. Fix $i \in [n]$ and $\varepsilon \in (0, 1)$, and write $v^{2[\varepsilon,i]}$ for the valuation given by $v^{2[\varepsilon,i]}(\boldsymbol{x}) = v^2(\boldsymbol{x}) + \varepsilon x_i$. For any $\boldsymbol{r}^1, \boldsymbol{r}^2 \in \mathbb{Z}^n$, suppose there exists a price $\boldsymbol{p}^*$ such that $\boldsymbol{r}^1 \in D_{v^1}(\boldsymbol{p}^*)$, and $\boldsymbol{r}^2 \in D_{v^{2[\varepsilon,i]}}(\boldsymbol{p}^*)$. Then there exists a price $\boldsymbol{p}^0$ such that $\boldsymbol{r}^1 \in D_{v^1}(\boldsymbol{p}^0)$, and $\boldsymbol{r}^2 \in D_{v^2}(\boldsymbol{p}^0)$. Moreover, for any price $\boldsymbol{p}$ such that $\boldsymbol{r} \coloneqq \boldsymbol{r}^1 + \boldsymbol{r}^2 \in D_{v^{\{1,2\}}}(\boldsymbol{p})$, we have $\boldsymbol{r}^1 \in D_{v^1}(\boldsymbol{p})$, and $\boldsymbol{r}^2 \in D_{v^2}(\boldsymbol{p})$.*

Proof. Fix bundles $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ that are, by assumption, demanded at $\boldsymbol{p}^*$ under v^1 and $v^{2[\varepsilon,i]}$. For $j \in \{1, 2\}$, we define $P^j \coloneqq \{\boldsymbol{p} \geq 0 \mid \boldsymbol{r}^j \in D_{v^j}(\boldsymbol{p})\}$, the set of prices at which $\boldsymbol{r}^j$ is demanded under v^j. It follows from the definition of $v^{2[\varepsilon,i]}$ that the set $P^{2[\varepsilon,i]}$ of prices at which $\boldsymbol{r}^2$ is demanded by $\boldsymbol{p}$ under $v^{2[\varepsilon,i]}$ is equal to $P^2 + \varepsilon\boldsymbol{e}^i$. Moreover, by assumption we have $\boldsymbol{p}^* \in P^1 \cap P^{2[\varepsilon,i]}$, so this intersection is non-empty.

As P^1 and P^2 are polyhedra in $\mathbb{R}^n$ (cf. (Baldwin and Klemperer, 2019, Proposition 2.7)), we can describe each region P^j by the system of linear inequalities

$$H^j\boldsymbol{p} \geq A^j, \tag{3.10}$$

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, Proposition 3.10 in (Baldwin and Klemperer, 2019) tells us that we can choose H^j such that each row of H^j is a strong-substitutes vector (cf. Definition 2.4). Additionally, as valuation v^j is integer-valued, we have $A^j \in \mathbb{Z}^{a_j}$. It follows that $P^{2[\varepsilon,i]} = P^2 + \varepsilon\boldsymbol{e}^i$ is defined by

$$H^2(\boldsymbol{p} - \varepsilon\boldsymbol{e}^i) \geq A^2 \quad \Leftrightarrow \quad H^2\boldsymbol{p} \geq A^2 + \varepsilon H^2_i, \tag{3.11}$$

where H^2_i is the i-th column of H^2. This allows us to express the intersection $P^1 \cap P^{2[\varepsilon,i]}$ as

$$H\boldsymbol{p} \geq A + \varepsilon\hat{H}_i, \tag{3.12}$$

where we define $H = \begin{pmatrix} H^1 \\ H^2 \end{pmatrix}$, $A = \begin{pmatrix} A^1 \\ A^2 \end{pmatrix}$, and $\hat{H} = \begin{pmatrix} 0 \\ H^2 \end{pmatrix}$.

Now we argue similarly to Theorem 19.1 in (Schrijver, 1998). We know that $P^1 \cap P^{2[\varepsilon,i]} \neq \emptyset$. Let $F = \{\boldsymbol{p} \geq 0 \mid H'\boldsymbol{p} = A' + \varepsilon\hat{H}'_i\}$ be a minimal face of

this intersection, where $H'\boldsymbol{p} \geq A' + \varepsilon\hat{H}'_i$ is a subsystem of (3.12) with linearly independent rows. Without loss of generality, we can write $H' = \begin{pmatrix} G & \tilde{G} \end{pmatrix}$, where G is invertible (permuting the columns of H' if necessary, which is equivalent to relabelling goods). Moreover, we can choose this permutation such that the i-th coordinate remains within the first square matrix – that is, if G is $n' \times n'$ then $i \leq n'$. In addition, G inherits from H^1, H^2 the property that its rows are strong-substitutes vectors. Now define the two price vectors

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

Note that $\boldsymbol{p}^0$ is an integral vector, since A is integral and G is unimodular. Moreover, by Lemma 3.26, we know that either (i) $G^{-1}\hat{H}'_i = \boldsymbol{e}^S$ for some $S \subseteq [n']$ with $i \in S$, or (ii) $G^{-1}\hat{H}'_i = -\boldsymbol{e}^S$ for some $S \subseteq [n']$ with $i \notin S$ (here n' is the dimension of G). 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 lies in $P^1 \cap P^{2[\varepsilon,i]}$, so $\boldsymbol{p}^\varepsilon$ satisfies (3.12). We now use this knowledge to show that $\boldsymbol{p}^0 \in P^1 \cap P^2$.

As $\boldsymbol{p}^\varepsilon \in P^1$, it satisfies (3.10) for $j = 1$. Note also that the rows of H^1 are strong-substitutes and $(\boldsymbol{p}^0 - \boldsymbol{p}^\varepsilon) \in \{\pm\varepsilon\boldsymbol{e}^S\}$ for some $S \subseteq [n]$, so (3.10) and Observation 3.28 imply

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

The integrality of H^1, $\boldsymbol{p}^0$ and A^1, together with $\varepsilon \in (0, 1)$, imply $H^1\boldsymbol{p}^0 \geq A^1$. Similarly, we argue that $H^2\boldsymbol{p}^0 \geq A^2$. Indeed, $\boldsymbol{p}^\varepsilon$ satisfies (3.11) and we have $(\boldsymbol{p}^0 - \boldsymbol{p}^\varepsilon + \varepsilon\boldsymbol{e}^i) \in \pm\varepsilon\boldsymbol{e}^S$ for some $S \subseteq [n]$, so (3.11) and Observation 3.28 imply

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

The last inequality again follows due to integrality of $H^2, \boldsymbol{p}^0$ and A^2. Thus we have established that $\boldsymbol{p}^0 \in P^1 \cap P^2$, which is equivalent to $\boldsymbol{r}^j \in D_{v^j}(\boldsymbol{p}^0)$ for $j = 1, 2$, our first claim.

Now let $\boldsymbol{r} := \boldsymbol{r}^1 + \boldsymbol{r}^2$ and let $\boldsymbol{p}$ be a price at which $\boldsymbol{r}$ is aggregately demanded, so $\boldsymbol{r} \in D_{v^{\{1,2\}}}(\boldsymbol{p})$. Suppose $\boldsymbol{r} = \boldsymbol{s}^1 + \boldsymbol{s}^2$ is a potentially different equilibrium allocation of bundles across the two agents, so that $\boldsymbol{s}^j \in D_{v^j}$ holds. By Proposition 2.F.1 in (Mas-Colell et al., 1995), we know that $(\boldsymbol{s}^j - \boldsymbol{r}^j) \cdot (\boldsymbol{p} - \boldsymbol{p}^0) \leq 0$ for $j = 1, 2$, with equality holding only if the agent is indifferent between both bundles at both prices. Hence, 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 both agents are indifferent between their two bundles at both prices. But as $\boldsymbol{s}^1 + \boldsymbol{s}^2 = \boldsymbol{r}$, equality holds, so we arrive at $\boldsymbol{r}^j \in D_{v^j}(\boldsymbol{p})$ for $i = 1, 2$. □

Proof of Proposition 3.25. Let v^2 denote the strong-substitutes integral-valued valuation of bidder j corresponding to $\mathcal{B}^j$, and let v^1 denote the (strong-substitutes integral-valued) aggregate valuation of all other bidders corresponding to $\mathcal{B} \setminus \mathcal{B}^j$. As above, $v^{\{1,2\}}$ denotes the aggregate valuation corresponding to $\mathcal{B}$. By assumption, we have $\boldsymbol{r} \in D_{v^{\{1,2\}}}(\boldsymbol{p})$.

As in Proposition 3.29, we write $v^{2[\varepsilon,i]}$ for the valuation given by $v^{2[\varepsilon,i]}(\boldsymbol{x}) = v^2(\boldsymbol{x}) + \varepsilon x_i$. Since both v^1 and $v^{2[\varepsilon,i]}$ are strong-substitutes valuations, there exists a price $\boldsymbol{p}^* \in \mathbb{R}^n$ such that $\boldsymbol{r} \in D_{v^{1,2[\varepsilon,i]}}(\boldsymbol{p}^*)$, and thus there exist $\boldsymbol{r}^1, \boldsymbol{r}^2 \in \mathbb{Z}^n$ with $\boldsymbol{r}^1 + \boldsymbol{r}^2 = \boldsymbol{r}$ such that $\boldsymbol{r}^2 \in D_{v^1}(\boldsymbol{p}^*)$ and $\boldsymbol{r}^2 \in D_{v^{2[\varepsilon,i]}}(\boldsymbol{p}^*)$. By Proposition 3.29, it follows that $\boldsymbol{r}^j \in D_{v^j}(\boldsymbol{p})$ for $j = 1, 2$.

Like in the proof of Proposition 3.29, we let $P^j := \{\boldsymbol{p}' \geq 0 \mid \boldsymbol{r}^j \in D_{v^j}(\boldsymbol{p}')\}$ for $j \in \{1, 2\}$ and observe that $P^{2[\varepsilon,i]} := \{\boldsymbol{p}' \geq 0 \mid \boldsymbol{r}^2 \in D_{v^{2[\varepsilon,i]}}(\boldsymbol{p}')\} = P^2 + \varepsilon \boldsymbol{e}^i$. We know that $\boldsymbol{p} \in P^1 \cap P^2$ (so this intersection is non-empty) and $\boldsymbol{p}^* \in P^1 \cap P^{2[\varepsilon,i]}$ (so this intersection is also non-empty). We seek $\boldsymbol{p}^\varepsilon \in P^1 \cap P^{2[\varepsilon,i]}$ with $\boldsymbol{p}^\varepsilon \in \boldsymbol{p} \pm \varepsilon \boldsymbol{e}^S$ for some $S \subseteq [n]$; this is sufficient to our first claim $\boldsymbol{r} \in D_{v^{\{1,2[\varepsilon,i]\}}}(\boldsymbol{p}^\varepsilon)$.

Analogous to line (3.10) above, we express the polyhedron P^j as $H^j \boldsymbol{p}' \geq A^j$, for $j \in \{1, 2\}$, where the rows of H^j are strong-substitutes vectors. Therefore $P^1 \cap P^{2[\varepsilon,i]} = P^1 \cap (P^2 + \varepsilon \boldsymbol{e}^i)$ is defined by the set of inequalities

$$H^1 \boldsymbol{p}' \geq A^1 \text{ and } H^2(\boldsymbol{p}' - \varepsilon \boldsymbol{e}^i) \geq A^2. \tag{3.13}$$

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

$$(H^j)' \boldsymbol{p}' \geq (A^j)' \tag{3.14}$$

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 (3.13), we obtain the new system

$$H \boldsymbol{p}' \geq A + \varepsilon \hat{H}_i. \tag{3.15}$$

This defines a superset of $P^1 \cap (P^2 + \varepsilon)$, since we have removed some rows from $H^j \boldsymbol{p}' \geq A^j$ for $j = 1, 2$. But we have seen that $P^1 \cap (P^2 + \varepsilon)$ contains $\boldsymbol{p}^*$, so (3.15) defines a non-empty polytope. So let $F = \{\boldsymbol{p}' \geq 0 \mid H' \boldsymbol{p}' = A' + \varepsilon \hat{H}_i'\}$ be a minimal face of this polytope, where

$$H' \boldsymbol{p}' \geq A' + \varepsilon \hat{H}_i' \tag{3.16}$$

is a subsystem of (3.15) with linearly independent rows.

Without loss of generality, we can write $H' = \begin{pmatrix} G & \tilde{G} \end{pmatrix}$, where G is invertible (permuting the columns of H' if necessary, which is equivalent to relabelling goods). Moreover we can choose this permutation such that the i-th 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 $\boldsymbol{p}' = \begin{pmatrix} \boldsymbol{p}_1' \\ \boldsymbol{p}_2' \end{pmatrix}$. In summary, our system (3.15) is now written

$$\begin{pmatrix} G & \tilde{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_1' \\ \boldsymbol{p}_2' \end{pmatrix} \geq A' + \varepsilon \hat{H}_i'. \tag{3.17}$$

By definition of the rows identified at (3.14), we have $\begin{pmatrix} G & \tilde{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \end{pmatrix} = A'$ and so

$$\boldsymbol{p}_1 = G^{-1} A' - G^{-1} \tilde{G} \boldsymbol{p}_2. \tag{3.18}$$

We now define $\boldsymbol{p}^\varepsilon$ as follows

$$\boldsymbol{p}^\varepsilon \coloneqq \begin{pmatrix} G^{-1}(A' + \varepsilon \hat{H}'_i) - G^{-1}\tilde{G}\boldsymbol{p}_2 \\ \boldsymbol{p}_2. \end{pmatrix} \tag{3.19}$$

in order to obtain

$$H'\boldsymbol{p}^\varepsilon = \begin{pmatrix} G & \tilde{G} \end{pmatrix} \boldsymbol{p}^\varepsilon = A' + \varepsilon \hat{H}'_i. \tag{3.20}$$

We wish to show that (3.13) holds for $\boldsymbol{p}' = \boldsymbol{p}^\varepsilon$. From (3.18) we observe that $\boldsymbol{p}^\varepsilon - \boldsymbol{p} = \begin{pmatrix} \varepsilon G^{-1}\hat{H}'_i \\ 0 \end{pmatrix}$. So, by Lemma 3.26, 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 $i \notin S$. 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 (3.16) with equality, and so $\boldsymbol{p}^\varepsilon \in F$. Thus, by definition of F, $\boldsymbol{p}^\varepsilon$ satisfies (3.15). It follows by definition of (3.14) 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 (3.13) 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

$$(H^j)''\boldsymbol{p} \geq (A^j)'' + \mathbf{1}. \tag{3.21}$$

Now, applying Observation 3.28 to $\boldsymbol{p}^\varepsilon - \boldsymbol{p}$ and (3.21) 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 Observation 3.28 to $\boldsymbol{p}^\varepsilon - \boldsymbol{p} - \varepsilon \boldsymbol{e}^i$, and (3.21) 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 (3.13) 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 $\boldsymbol{r} \in D_{v^{1,2[\varepsilon,i]}}(\boldsymbol{p}^\varepsilon)$, which concludes the proof of our first claim.

To see what we can compute $\boldsymbol{p}^\varepsilon$ in polynomial time, 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 (Shioura and Tamura, 2015, Theorem 7.3), 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 vector $\boldsymbol{p}^\varepsilon \in \left\{\boldsymbol{p} + \varepsilon \boldsymbol{e}^S \mid S \subseteq [n]\right\}$ 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 $2T(n, |\mathcal{B}|)$ time, where $T(n, |\mathcal{B}|)$ 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 Section 3.4. □

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.

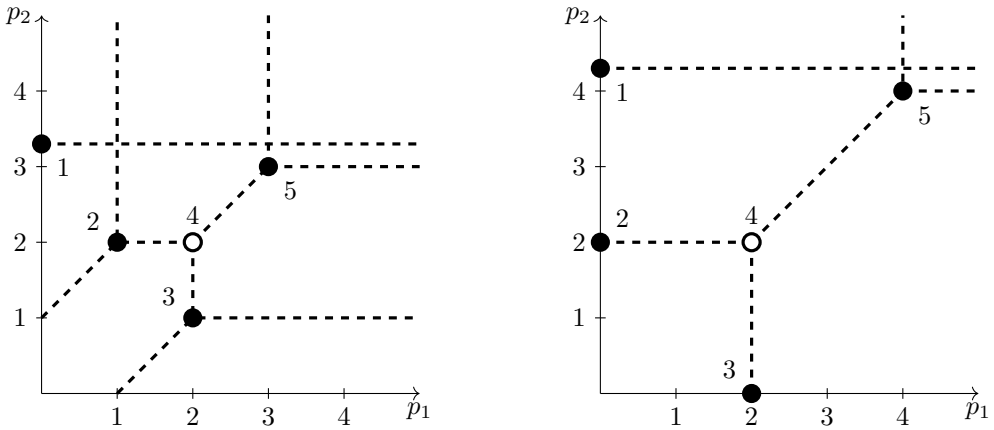
$$\boldsymbol{b}' \coloneqq \begin{cases} \boldsymbol{b} - \boldsymbol{e}^{[n]_0 \setminus I} & \text{if } 0 \in I, \\ \boldsymbol{b} + \boldsymbol{e}^{I} & \text{otherwise.} \end{cases}$$

Note that we allow for bid vector entries to be negative. Figure 3.5 illustrates the projection operation.

Observation 3.30. *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 3.31. *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 Lemma 3.18 and Observation 3.30. □



(a) Five numbered bids before projecting. **(b)** After projecting w.r.t. prices $\boldsymbol{p} = (4, 4)$.

Figure 3.5: 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 3.7. It shifts and projects to strictly 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 3.32 and 3.33 establish that the reduction obtained by ShiftProjectUnshift is valid and makes progress.

Algorithm 3.7 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 $\boldsymbol{p}$.

Lemma 3.32. *The marginal bids graph $G_{\mathcal{A}'}$ of $\mathcal{A}'$ has strictly fewer edges than $G_{\mathcal{A}}$.*

Proof. 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 3.16 and Lemma 3.31 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

$$b^i_i - p_i = b^i_{i+1} - p_{i+1} \tag{3.22}$$

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

$$b^k_k - p_k = b^k_1 - p_1 \tag{3.23}$$

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

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

So it must hold that

$$\sum_{i=1}^{k-1}(b^i_i - b^i_{i+1}) = b^k_1 - b^k_k. \tag{3.24}$$

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

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

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 (3.22) and (3.23) (w.r.t. new prices), then eliminate those prices obtaining a version of (3.24), 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 (3.24), so we have the required contradiction. □

Lemma 3.33. *ShiftProjectUnshift, given in Algorithm 3.7, returns a reduction $\mathcal{A}'$ of $\mathcal{A}$ in time $O(T(n,|\mathcal{B}|) + n|\mathcal{B}|)$, where $T(n,|\mathcal{B}|)$ is the time it takes to minimise an n-dimensional submodular function.*

Proof. The procedure ShiftProjectUnshift takes as input an allocation problem and then modifies it three times in lines 3 to 5. Our goal is to show that each successive problem remains valid, and is a reduction of the previous problem. To do this, we apply lemmas and propositions from above. Let $\mathcal{A}^3$ and $\mathcal{A}^4$ denote the allocation problems after executing lines 3 and 4, respectively. First we show that $\mathcal{A}^3$ is a reduction of $\mathcal{A}$. Indeed, by Proposition 3.25 and Step 1, the residual bundle $\boldsymbol{r}$ is aggregately demanded at $\boldsymbol{p}^\varepsilon$ in $\mathcal{A}^3$. Furthermore, for any allocation $(\boldsymbol{r}^j)_{j\in J} = (\boldsymbol{s}^j - \boldsymbol{m}^j)_{j\in J}$ of the residual bundle $\boldsymbol{r}$ at price $\boldsymbol{p}^\varepsilon$ in $\mathcal{A}^3$, Lemma 3.24 implies that $(\boldsymbol{m}^j + \boldsymbol{r}^j)_{j\in J}$ is a solution of $\mathcal{A}$. Secondly, Lemma 3.31 implies that $\mathcal{A}^4$ is a reduction of $\mathcal{A}^3$. Finally, after line 5, the bid lists are locally valid at $\boldsymbol{p}$ by Lemma 3.24. Proposition 3.25 implies that the residual bundle $\boldsymbol{r}$ is demanded at some price $\boldsymbol{p}' \in \left\{\boldsymbol{p}^\varepsilon \pm \varepsilon \boldsymbol{e}^S \mid S \subseteq [n]\right\}$. 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. □

3.5.5 The main algorithm

The algorithm Allocate, stated in Algorithm 3.3 above, combines the procedures NonMarginals, UnambiguousMarginals and ShiftProjectUnshift in order to solve the allocation problem described in Section 3.3. Recall that it uses FindParams as a subroutine to decide whether to call UnambiguousMarginals or ShiftProjectUnshift in each iteration of the loop. Theorem 3.34 proves correctness and gives a running time bound for Allocate. We note that this bound has not been shown to be tight, and a more refined analysis may lead to a better running time guarantee.

Theorem 3.34. *Allocate solves the allocation problem in time $O\left(n^2|J|(\alpha(n)n|\mathcal{B}| + T(n,|\mathcal{B}|))\right)$, where $T(n,|\mathcal{B}|)$ 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 3.23 and 3.32). 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 3.15, 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 3.5.2 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 supply 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 favour 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.

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.

Algorithm 3.8 PRIORITYPARAMS

Compute the derived graph $D_{\mathcal{A}}$.
for all pairs (i, j) in the priority list **do**
    if i is a link good and $i \in I$ for some demand cluster (I, j) **then**
        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.
        return (i, j) if the edge lies in a cycle.
    end if
end for
return some leaf demand cluster (I, j) and the adjacent link good i.

3.6 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.

3.6.1 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 $\boldsymbol{p}_\pi = \boldsymbol{p} + \sum_{i\in[n]} \frac{\varepsilon}{2i}\boldsymbol{e}^{\pi(i)}$ for some $\varepsilon < 0.1$. Note that a unique bundle is demanded at $\boldsymbol{p}_\pi$ for any permutation π.

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 3.35. *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 2.5 from Chapter 2. □

Algorithm 3.9 GenerateList(n,M,q)

Initialise: Empty bid list $\boldsymbol{b}$ and bundle $\boldsymbol{x} = \mathbf{0}$.
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 $\boldsymbol{b} + \lambda \mathbf{1}$ 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.
end if
if $\boldsymbol{p}$ is the component-wise minimal market-clearing price vector of $\boldsymbol{x}$ **then**
    return $\mathcal{B}$.
else
    Repeat the algorithm.
end if

3.6.2 Testing the allocation 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. 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.

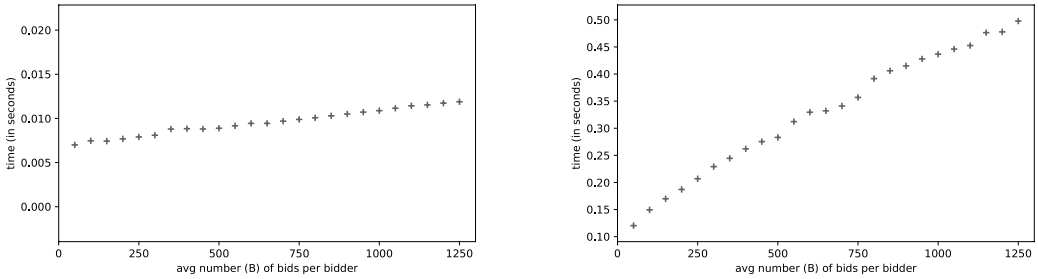
2. For the second pair of experiments, we vary the number of goods from $n = 10$ to 50 in steps of 5 and fix q to 50 and 100, respectively.

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.

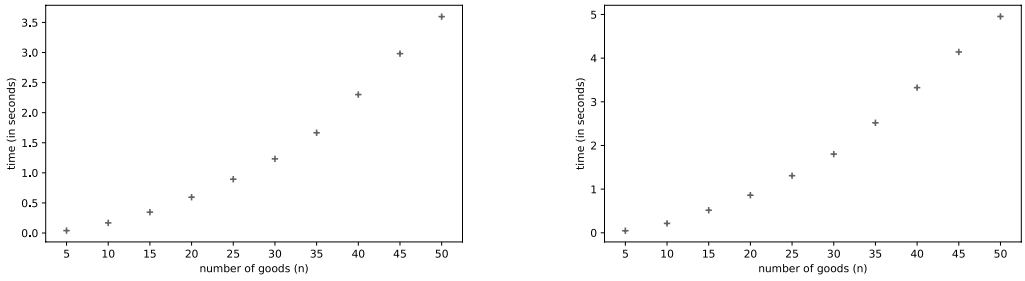
3.6.3 Results

The outcomes of the three pairs of experiments are shown in Fig. 3.6 to 3.8. 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 3.8 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 3.6: 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.

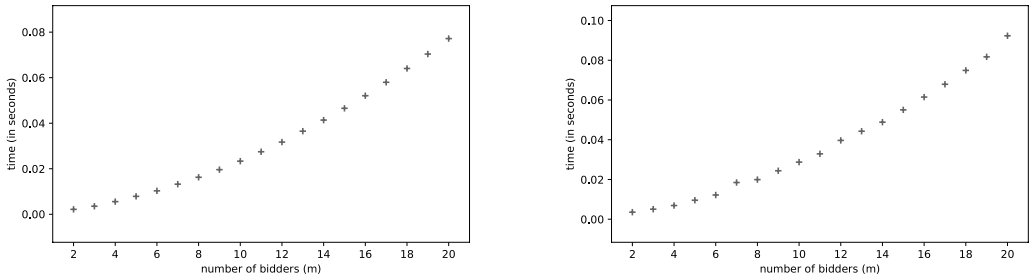


(a) Parameters: $m = 5, M = 100, q = 50$. **(b)** Parameters: $m = 5, M = 100, q = 100$.

Figure 3.7: 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.

3.7 Conclusion and outlook

This chapter has developed fully polynomial algorithms for solving the strong-substitutes product-mix auction by finding a competitive equilibrium. In this auction, bidders express arbitrary strong-substitutes preferences via the tropical bidding language by submitting a valid list of positive and negative bids. This



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

Figure 3.8: 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.

greatly expands the expressivity for bidders, compared to previous implementations of the auctions that only permitted positive bids.

Future work. 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. Recall that we used submodular minimisation in each iteration of the price-finding algorithm (cf. Section 3.4) and in the SHIFTPROJECTUNSHIFT procedure of our allocation algorithm (cf. Section 3.5.4). One immediate question is whether we can also exploit the information provided by the bidding language to improve the computational efficiency of the submodular function minimisation subroutine. Another obvious question is the extent to which our methods can be extended to broader classes of valuations. In Chapter 2, for instance, we presented a way to generalise the tropical language to general substitutes. Recall that a multi-unit auction with ordinary-substitutes bidders may not admit a competitive equilibrium. In light of this limitation, is an open question whether the techniques presented in this chapter can be applied to finding a quasi-equilibrium as described in Chapter 1. Other generalisations of the tropical language that incorporate complements may also be tractable from the point of computing competitive equilibrium.

4

Maximising revenue in the strong-substitutes product-mix auction

Contents

4.1 Introduction 83
4.2 Problem statement 84
4.2.1 Preliminaries 86
4.3 The hardness of solving for revenue 89
4.4 Algorithms for maximising revenue 92
4.4.1 Additive demand 92
4.4.2 Iterating over bundles 93
4.5 Conclusion and outlook 96

4.1 Introduction

In Chapter 3, the strong-substitutes product-mix auction is solved for competitive equilibrium, thus maximising social welfare. In other words, we find prices at which demand meets supply (market-clearing prices) and every bidder is allocated a bundle they demand (envy-free allocation). The social welfare objective is natural in many settings and has been studied extensively, both for divisible and indivisible goods. In many practical applications, however, auctions are solved with the objective of maximising revenue instead of social welfare. Important examples of revenue-focused auctions can be found in the advertising industry, for instance (Conitzer et al., 2019) and (Colini-Baldeschi et al., 2016). In this chapter, we investigate the computational complexity of solving the strong-substitutes product-mix auction with the objective of maximising revenue. Specifically, the auctioneer is willing

to sell only a subset of its supply, if doing so increases revenue, as long as the resulting allocation remains envy-free.

Chapter outline. The remainder of this section reviews related literature. In Section 4.2 we formally state the problem(s) that we solve and make some preliminary observations about revenue-maximising prices and allocations. We also recall the complexity-theoretic notion of APX-hardness. Section 4.3 shows that, unlike the social welfare setting, optimising for revenue is computationally hard even when we restrict our bidding language to positive tropical bids with two positive entries only. Finally, Section 4.4 presents some initial algorithmic results that may be extendable in future.

Literature. The problem of maximising revenue has received some attention in the literature under the moniker of profit-maximising envy-free pricing. In (Guruswami et al., 2005), the authors study this problem for two classes of demand: unit demand consumers (which correspond to single positive unit bids), and single-minded consumers (who are only interested in a particular subset of goods). In particular, both problems are APX-hard, and admit a logarithmic approximation algorithm. Working in the unit supply setting (so setting supply $\boldsymbol{s} = \mathbf{1}$), Chen and Deng (2014) study the setting where each bidder has unit demand with a positive valuation for at most k goods. In the case of $k = 2$, the authors demonstrate a polynomial-time algorithm for maximising revenue, whereas the case where $k \geq 3$ is shown to be NP-hard. In contrast, we show below that relaxing supply to multiple units per good makes the case of $k = 2$ APX-hard. Other variations of the problem have also been explored, including bundle pricing schemes where the seller stipulates prices according to bundle sizes (Monaco et al., 2015), settings that place per-bidder constraints on desired bundles (Mizuno and Takahashi, 2019), versions in which bidders have budgets (Colini-Baldeschi et al., 2016; Feldman et al., 2012), and restrictions to 'sharp' multi-unit demands in which each bidder wants a specific number of goods (Chen et al., 2016).

4.2 Problem statement

We now present the problem formally before making some preliminary observations. In this chapter, we restrict bidders to submitting positive tropical bids. Without loss of generality, we can assume that each bidder submits a single such bid (treating bids as separate bidders, if necessary). Hence, each bidder has unit demand on indivisible goods. We emphasise again that we assume truthful bidding, as described in Chapter 1. Hence we let the list of buyer bids be $\mathcal{B} = \{\boldsymbol{b}^1, \ldots, \boldsymbol{b}^m\}$, where $\boldsymbol{b}^j$ denotes the bid of bidder j. Moreover, the auctioneer submits a list of k positive tropical bids denoted $\mathcal{A} = \{\boldsymbol{b}^{m+1}, \ldots, \boldsymbol{b}^{m+k}\}$. The purpose of the auctioneer's bids is to set any reserve prices that she might have. Our goal is to find an envy-free allocation to bidders $[m]$ that maximises revenue while not exceeding supply $\boldsymbol{s} \in \overline{\mathbb{R}}^n$. Note that we allow for supply in some (or all) goods to be infinite. This makes sense only in the revenue maximisation objective, not for competitive equilibrium.

Moreover, if we let $M := \max_{\boldsymbol{b}\in\mathcal{B}} \|\boldsymbol{b}\|_\infty$, then we can restrict price space to the cube $[0, M]^n$. This is because any price vector outside this cube can be projected onto the cube without changing the prices of goods that are demanded, so the cube contains a revenue-maximising price vector.

The problem. Recall that a solution $(\boldsymbol{p}, \pi)$ for the strong-substitutes product-mix auction consists of prices $\boldsymbol{p}$ and an allocation function $\pi \geq 0$, which maps each bidder $j \in [m]$ to the bundle $\pi(j) \geq \mathbf{0}$ they receive. In order for the allocation to be feasible, that is not exceed supply, we require $\pi([m+k]) \leq \boldsymbol{s}$. Secondly, the solution is envy-free if $\pi(j) \in D_{\boldsymbol{b}^j}(\boldsymbol{p})$ for all $j \in [m+k]$. Hence we can express the problem of finding a revenue-maximising solution in terms of the following quadratic program.

$$\begin{aligned}
\max_{\boldsymbol{p},\pi} \quad & \boldsymbol{p} \cdot \pi([m]) && (4.1)\\
\text{s.t.} \quad & \pi(j) \in D_{\boldsymbol{b}^j}(\boldsymbol{p}),\ \forall j \in [m+k], && (4.2)\\
& \mathbf{0} \leq \boldsymbol{p} \leq \boldsymbol{M}, && (4.3)\\
& \pi([m+k]) \leq \boldsymbol{s}, && (4.4)\\
& \pi \geq 0. && (4.5)
\end{aligned}$$

We are interested both in finding algorithms to solve this problem, and in understanding its complexity. To this end, we introduce the optimisation problems RevenueMax and k-RevenueMax in Definition 4.1. The latter captures the intuitive notion that each bidder has unit demand for a small subset of goods only. Without loss of generality, we can assume that a bidder who is indifferent between the reject good and other goods is allocated the empty bundle.

Definition 4.1. The optimisation problem RevenueMax consists of instances $(\mathcal{A}, \mathcal{B}, \boldsymbol{s})$, where $\mathcal{A}$ and $\mathcal{B}$ are lists of positive tropical bids and $\boldsymbol{s}$ is the supply vector. Furthermore, k-RevenueMax is the subset of RevenueMax in which all bid lists contain only bids with at most k non-zero entries. When the auctioneer submits no bids, we simply write $(\mathcal{B}, \boldsymbol{s})$.

We now introduce the notion of PTAS reductions and APX-hardness. This is analogous to polynomial-time reductions and NP-hardness, except that we allow algorithms to find an approximate, instead of exact, solution in polynomial time. For details on the APX complexity class, and PTAS reductions, we refer to (Papadimitriou, 1994). In Section 4.3, we construct a PTAS reduction from Max-2Sat to show that 2-RevenueMax is APX-hard.

Definition 4.2. A *reduction* (f, g) from problem $\mathcal{X}$ to $\mathcal{Y}$ consists of two polynomial-time computable functions f and g, with f mapping each problem instance I of $\mathcal{X}$ to a problem instance $f(I)$ of $\mathcal{Y}$, and g mapping each solution to instance $f(I) \in \mathcal{Y}$ to a solution to instance $I \in \mathcal{X}$.

We say that (f, g) is a *PTAS-reduction* if, for all $\varepsilon > 0$, there exists $\delta = \delta(\varepsilon) > 0$ such that if $g(x)$ is a $(1 + \delta(\varepsilon))$-approximate solution to $f(I) \in \mathcal{Y}$, then x is a $(1 + \varepsilon)$-approximate solution to $I \in \mathcal{X}$.

Definition 4.3. The class APX is the set of optimisation problems in NP that admit polynomial-time algorithms with constant-factor approximation ratios. A problem $\mathcal{P}$ is APX-hard if there exists a PTAS reduction from every problem in APX to $\mathcal{P}$.

4.2.1 Preliminaries

We make some preliminary observations that help lead us to initial algorithms for solving RevenueMax and its parametrised variant. First we note the intuitive geometric observation that for every demanded bundle there exists a unique and integral element-wise maximal price vector at which it is demanded. Secondly, given a revenue-maximising price vector $\boldsymbol{p}$ of a revenue-maximising solution $(\boldsymbol{p}, \pi)$, we can compute a supporting allocation π in polynomial time, and vice versa. Hence, in order to find a solution to the revenue maximisation program (4.1), it suffices to find either its prices, or its allocation. We will see below that solving (4.1) is APX-hard, even if the auctioneer submits no bids and supply is unconstrained. Hence, another way to interpret our preliminary lemmas is that the problems of finding prices and allocations are equally hard.

A unique element-wise maximal price vector. Recall from Chapters 1 and 2 that every list $\mathcal{B}$ of tropical bids partitions price space into convex *demand regions*: for any bundle that is demanded by $\mathcal{B}$ at some prices, the set of all prices at which some bundle $\boldsymbol{x}$ is demanded is its demand region and forms a polyhedron. This partitioning of price space is captured by the strong-substitutes polyhedral complex.[1]

Lemma 4.4. *Let R be the demand region in which $\boldsymbol{x}$ is demanded. Then R contains a unique element-wise maximal price vector $\boldsymbol{p}$, that is $\boldsymbol{p} \geq \boldsymbol{q}$ for all $\boldsymbol{q} \in R$. Moreover, $\boldsymbol{p}$ is integral.*

Proof. The fact that R contains a unique element-wise maximal price vector was shown by Murota (2003, Theorem 11.16). It remains to prove that $\boldsymbol{p}$ is integral. We do this by showing that $\boldsymbol{x}$ is also demanded at $\lceil \boldsymbol{p} \rceil$. Specifically, for every bid $\boldsymbol{b} \in \mathcal{B}$, if $\boldsymbol{b}$ demands good i at $\boldsymbol{p}$, then it also demands i at $\lceil \boldsymbol{p} \rceil$. (This implies that $\boldsymbol{p}$ and $\lceil \boldsymbol{p} \rceil$ lie in the demand region of $\boldsymbol{x}$.) Fix a bid $\boldsymbol{b} \in \mathcal{B}$ and a good $i \in [n]$ that is demanded by $\boldsymbol{b}$ at $\boldsymbol{p}$. Moreover, let $j \in [n]$ be any other good. Then

$$b_j - \lceil p_j \rceil \leq b_j - p_j \leq b_i - p_i < b_i - \lceil p_i \rceil + 1.$$

Here the second inequality holds because good i is demanded at $\boldsymbol{p}$. The integrality of $\boldsymbol{b}$ and $\lceil \boldsymbol{p} \rceil$ then implies $b_j - \lceil p_j \rceil \leq b_i - \lceil p_i \rceil$. Secondly, if $b_i > p_i$, then by the integrality of $\boldsymbol{b}$ we also have $b_i \geq \lceil p_i \rceil$. Hence we have shown that good i is still demanded at $\lceil \boldsymbol{p} \rceil$. □

[1] We note that demand regions may or may not be full-dimensional.

Computing revenue-maximal prices. Fix some allocation π, so $\pi(j) \in \mathbb{Z}_+^n$, with $\pi([m+k]) = \boldsymbol{s}$. We now discuss the problem of computing revenue-maximal prices at which π is feasible, or reporting that no such vector exist. Let $\boldsymbol{x} = \pi([m+k)$ be the bundle aggregately allocated by π. This is analogous to the problem of finding market-clearing prices in Chapter 3. Indeed, we know that if $\boldsymbol{x}$ is demanded anywhere, then its demand region has unique element-wise minimal and element-wise maximal price vectors. In Chapter 3, our price-finding algorithm finds the element-wise minimal prices. Now we wish to find the element-wise maximal prices in order to maximise revenue.

Lemma 4.5. *Let $\boldsymbol{x} \in \mathbb{Z}^n$ be a bundle demanded at some prices. We can compute the unique revenue-maximal price vector at which $\boldsymbol{x}$ is demanded in polynomial time.*

Proof. The algorithm we use is a simple adaptation of the LongStepMinUp algorithm presented in Chapter 3 to find the element-wise minimal price at which $\boldsymbol{x}$ is demanded. Instead of starting at $\mathbf{0}$ and moving upwards, we start at $\boldsymbol{M}$ and move downwards, repeatedly finding a steepest descent direction among the set $\{-1, 0\}^n$. At each step, finding the descent direction is equivalent to submodular minimisation, which is polynomial-time. This algorithm, with unit steps, has been described in the discrete optimisation literature (Murota and Shioura, 2014; Shioura, 2017). The same improvements described in Chapter 3 to take 'long steps' can be applied to yield the algorithm LongStepMaxDown. In particular, its running time can be bounded by a polynomial, analogously to the LongStepMinUp algorithm. □

Computing a revenue-maximal bundle. Suppose $\boldsymbol{p}$ is a fixed price vector. We now describe how to compute an envy-free allocation π at $\boldsymbol{p}$ in polynomial time that does not exceed supply and maximises $\pi([m]) \cdot \boldsymbol{p}$. This is achieved by reducing the problem to a min-cost network flow problem.[2] In particular, if $\boldsymbol{p}$ is a revenue-maximising price vector, then our procedure finds a revenue-maximising solution $(\boldsymbol{p}, \pi)$. On the other hand, if all possible envy-free allocations exceed supply, the procedure returns nothing.

Construct the min-cost network flow problem as follows. The directed network is defined on the set of vertices consisting of the source s, the sink t, the bids $\boldsymbol{b} \in \mathcal{B}$ and the goods $i \in [n]_0$ (including the reject good). Recall that a min-cost network flow problem specifies a cost $c(v, w)$, a capacity $u(v, w)$ and a lower bound $l(v, w)$ for each arc (v, w) in the network (Ahuja et al., 1993). In our network, all arcs have cost 0 unless otherwise specified. For each bid $\boldsymbol{b} \in \mathcal{A} \cup \mathcal{B}$, we add an arc $(s, \boldsymbol{b})$ with $u(s, \boldsymbol{b}) = l(s, \boldsymbol{b}) = w(\boldsymbol{b})$. For each good $i \in [n]_0$, we add an arc (i, t) with $l(i, t) = 0$ and $u(i, t) = s_i$. Finally, we add an arc $(\boldsymbol{b}, i)$ with $l(\boldsymbol{b}, i) = 0$ and $u(\boldsymbol{b}, i) = \infty$ if, and only if, $\boldsymbol{b}$ demands good i at $\boldsymbol{p}$. We set $c(\boldsymbol{b}, i) = -p_i$ if $\boldsymbol{b} \in \mathcal{B}$. Figure 4.1 illustrates this network.

[2]This is conceptually similar to the allocation problem solved in Chapter 3. However, as we only have positive bids in this setting, the problem admits a simpler solution. Moreover, we now wish to find an allocation that maximises revenue instead of units of goods demanded.

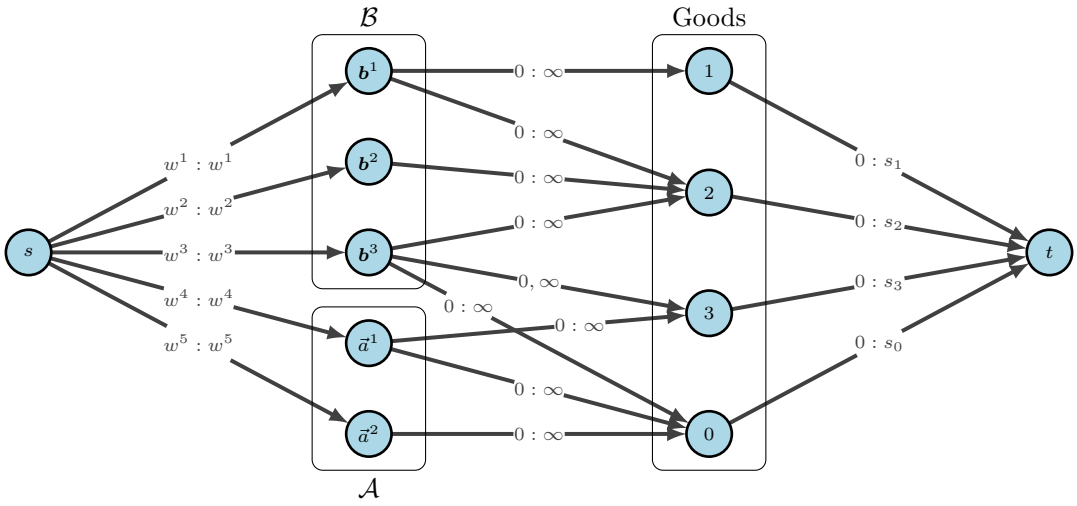


Figure 4.1: The min-cost network flow we solve to find a revenue-maximising allocation for fixed prices $\boldsymbol{p}$. There is an arc from bid $\boldsymbol{b}$ to good $i \in [n]_0$ if, and only if, $\boldsymbol{b}$ demands i at $\boldsymbol{p}$. We label each arc (x, y) with the upper and lower bounds on flow as $(l(x, y) : u(x, y))$. Here we also use the shorthand $w^j := w(\boldsymbol{b}^j)$ for all $j \in [m + k]$. Arcs $(\boldsymbol{b}, i)$ with $\boldsymbol{b} \in \mathcal{B}$ have cost $-p_i$, all other arcs have 0 cost.

The min-cost integrality theorem states that, for every min-cost network flow problem with integral costs, capacities and lower bounds, there exists an integral flow that minimises cost. Moreover, the network simplex method finds such an integral min-cost flow in strongly polynomial time, or reports that no feasible flow exists (Orlin, 1997). Note that there is a one-to-one correspondence between (feasible) integral flows f in this network and envy-free allocations π to bids defined by $f(\boldsymbol{b}^j, i) = \pi(j)_i$. It follows immediately that $\boldsymbol{p}$ is feasible if, and only if, the network admits a feasible integral flow. Moreover, an optimal integral solution to the min-cost network flow minimises $-\sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]_0} p_i f(\boldsymbol{b}, i) = -\pi([m]) \cdot \boldsymbol{p}$, and so corresponds to an envy-free allocation that maximises revenue. This implies Lemma 4.6.

Lemma 4.6. *Given prices $\boldsymbol{p} \in \mathbb{R}^n$, we can compute, in polynomial time, an envy-free allocation π that maximises revenue $\boldsymbol{p} \cdot \pi([m])$ if $\boldsymbol{p}$ is feasible, or report that $\boldsymbol{p}$ is not feasible otherwise.*

A naive algorithm. We now show how these preliminary insights lead to a naive algorithm that runs in exponential time. Recall that each demand region has a unique element-wise maximal price vector, and Lemma 4.4 ensures that revenue is maximised at some maximal vector. Hence, a straightforward approach is to iterate over all such vectors, computing revenue at each price vector by means of the min-cost network flow problem, and return the prices at which revenue is maximised.

In order to enumerate the element-wise maximal price vectors of all demand regions, we briefly recall that each positive tropical bid induces $O(n^2)$ facets, in which it is indifferent between goods (n facets normal to $\boldsymbol{e}^i$, $i \in [n]$, and $\binom{n}{2}$ facets normal to $\boldsymbol{e}^i - \boldsymbol{e}^j$, $i, j \in [n]$). Hence, there are $O((|\mathcal{A}| + |\mathcal{B}|)n^2$ facets in total.

Furthermore, we note that every element-wise maximal price vector coincides with the intersection of n such facets or the boundaries of the price-space cube $[0, M]^n$. To see this, note that we could otherwise increase some prices without changing the demand of any bids, which contradicts our assumption that the prices are revenue-maximal. Hence, we iterate over all possible intersections of n facets or boundaries, which guarantees that we visit the element-wise maximal price vector each demand region. If n is constant, this yields an efficient algorithm. However, for general n this procedure is exponential.

4.3 The hardness of solving for revenue

We now show that solving the strong-substitutes product-mix auction for revenue is APX-hard even when we restrict ourselves to positive bids that have only two non-zero entries. As mentioned above, we can assume that each bidder submits exactly one bid, which expresses unit demand. Moreover, in this section we assume that supply is unconstrained and the auctioneer submits no reserve bids. Hence our setting is equivalent to that of Guruswami et al. (2005), who previously showed that RevenueMax is APX-hard by means of a PTAS reduction from VertexCover-B, the vertex cover problem with vertex degrees bounded by a constant B. However, while VertexCover-B is known to be APX-hard, there exists no explicit bound on its approximation ratio to the best of our knowledge.

In our proof, we show APX-hardness of 2-RevenueMax by providing a PTAS reduction from Max-2Sat. This result establishes that maximising revenue remains hard even when every bidder is indifferent between only two goods. Moreover, if we assume that the unique games conjecture or the weaker assumption $P \neq NP$ are true, then explicit bounds on the approximation ratio of Max-2Sat are known, and our PTAS reduction thus implies an explicit bound on the approximation ratio of 2-RevenueMax that we give in Corollary 4.15.

Recall that a Boolean formula is in 2-CNF form if it consists of a conjunction of clauses, each of which being the disjunction of exactly two distinct literals. We assume without loss of generality that each variable occurs at most once in a clause. Max-2Sat is the problem of finding a truth assignment β of a given formula in 2-CNF form that satisfies the largest number of clauses. It is known that Max-2Sat is APX-hard (Håstad (2001)). We establish the APX-hardness of 2-RevenueMax in Theorem 4.14. To begin with, we restate the following well-known fact about the largest number of clauses satisfiable by any truth assignment, which is used in our proof of Theorem 4.14.

Lemma 4.7 ((Håstad, 2001)). *Every Boolean formula in 2-CNF form has a truth assignment that satisfies at least 3/4 of the clauses.*

The PTAS reduction. Let φ be an instance of Max-2Sat, i.e. a Boolean formula in 2-CNF form with K clauses and N variables. From φ, we construct a 2-RevenueMax instance $\mathcal{P}_\varphi$ on $n = 2N$ goods that has a single bidder with the following bid list $\mathcal{B}$. For each literal L in φ, let $\mathrm{sgn}(L)$ be the function

that evaluates to 1 if the literal is positive and 0 otherwise. Moreover, for each variable X_i, let $\rho(X_i)$ denote the number of occurrences of X_i in φ. Note that $\sum_{i\in[N]} \rho(X_i) = 2K$. $\mathcal{B}$ consists of the following bids. For each variable X_i, $i \in [N]$, add 'variable' bids of weight $3\rho(X_i)$ at $3\boldsymbol{e}^{2i-1}, 3\boldsymbol{e}^{2i}$ and $2\boldsymbol{e}^{\{2i-1,2i\}}$. For each clause (L_1, L_2) with variables X_i and X_j, place a 'clause' bid of weight 1 at $2\boldsymbol{e}^S$, where $S = \{2i - \operatorname{sgn}(L_1), 2j - \operatorname{sgn}(L_2)\}$.

Lemma 4.8. *For any price vector $\boldsymbol{p} \notin \{2,3\}^{2N}$, let $\boldsymbol{p}'$ be the vector obtained from $\boldsymbol{p}$ by rounding as follows.*

$$p'_i := \begin{cases} 2 & \text{if } p_i \leq 2, \\ 3 & \text{else.} \end{cases}$$

Then the revenue at $\boldsymbol{p}'$ is strictly higher than the revenue at $\boldsymbol{p}$.

Proof. Suppose $p_i < 2$. We claim that increasing p_i to 2 increases the total revenue. First we note that increasing p_i does not reduce the revenue contribution from any bids, and the contribution from at least one variable bid strictly increases. A similar argument shows for the case $p_i > 2$ that setting p_i to 3 strictly increases revenue.□

We now observe how much revenue the variable bids and the clause bids contribute to the total revenue if $\boldsymbol{p} \in \{2,3\}^{2N}$.

Observation 4.9. *Suppose $\boldsymbol{p} \in \{2,3\}^{2N}$ and fix a variable X_i. The three variable bids corresponding to this variable contribute a revenue of $18\rho(X_i)$ if $p_{2i-1} = p_{2i}$ and $21\rho(X_i)$ otherwise.*

Proof. If both p_{2i-1} and p_{2i} are set to 2, then the three bids each contribute a revenue of $6\rho(X_i)$. If both prices are set to 3, the two bids at $\boldsymbol{e}^{2i-1}$ and $\boldsymbol{e}^{2i}$ each contribute $9\rho(X_i)$ and the third bid at $\boldsymbol{e}^{\{2i-1,2i\}}$ contributes 0. On the other hand, if $p_{2i-1} = 2$ and $p_{2i} = 3$, then the bid at $3\boldsymbol{e}^{2i}$ contributes $9\rho(X_i)$ and the other two bids each contribute $6\rho(X_i)$. Similarly, the three bids also contribute $21\rho(X_i)$ if $p_{2i-1} = 3$ and $p_{2i} = 2$. □

Observation 4.10. *Suppose $\boldsymbol{p} \in \{2,3\}^{2N}$ and fix a clause (L_1, L_2) with $L_1 \in \{X_i, \overline{X_i}\}$ and $L_2 \in \{X_j, \overline{X_j}\}$. If $p_{2i-\operatorname{sgn}(L_1)}$ and $p_{2j-\operatorname{sgn}(L_2)}$ are both 3, then the clause bid corresponding to (L_1, L_2) contributes no revenue. Otherwise, if at least one of these two prices is 2, the bid contributes revenue 2.*

Lemma 4.11. *Suppose $\boldsymbol{p} \in \{2,3\}^{2N}$ satisfies $p_{2i-1} = p_{2i}$ for some $i \in [N]$. Then changing the value of p_{2i} from 2 to 3 (or vice versa) increases revenue by at least 1.*

Proof. Note that the variable X_i occurs in $\rho(X_i)$ clauses. Hence by Observation 4.10, changing the value of p_{2i} reduces the revenue contribution from clause bids by at most $2\rho(X_i)$. On the other hand, Observation 4.9 implies that the contribution from the variable bid corresponding to X_i increases by $3\rho(X_i)$, while the contributions from all other variable bids are unchanged. Hence the total revenue increases by at least $\rho(X_i) \geq 1$ if we change the value of p_{2i} so that it does not equal p_{2i-1}. □

We now describe how to construct a solution to Max-2Sat from a solution to the corresponding 2-RevenueMax problem. Lemma 4.8 and Lemma 4.11 imply a specific characterisation of revenue-maximising prices. In order to translate between (approximate) solutions of Max-2Sat and 2-RevenueMax, we introduce a natural one-to-one correspondence between truth assignments on N variables and valid price vectors $\boldsymbol{p} \in \{2,3\}^{2N}$.

Definition 4.12. We say that a price vector $\boldsymbol{p} \in \mathbb{R}^n$ is *valid* if $\boldsymbol{p} \in \{2,3\}^{2N}$ and $p_{2i-1} \neq p_{2i}$ for every $i \in [N]$.

A valid price vector $\boldsymbol{p}$ corresponds to truth assignment β if and only if $\boldsymbol{p}$ and β satisfy

$$\beta(X_i) = \begin{cases} \text{True} & \text{if } p_{2i-1} = 2, \\ \text{False} & \text{if } p_{2i} = 2. \end{cases}$$

For convenience, we will write $\beta(\boldsymbol{p})$ for the truth assignment corresponding to $\boldsymbol{p}$ and $\boldsymbol{p}(\beta)$ for the price vector corresponding to a given truth assignment β.

Lemma 4.13. *For any valid price vector $\boldsymbol{p}$, the revenue obtained at $\boldsymbol{p}$ in $\mathcal{P}_\varphi$ is $42K + 2k$, where k denotes the number of clauses of φ satisfied by the truth assignment $\beta(\boldsymbol{p})$. Moreover, revenue for $\mathcal{P}_\varphi$ is maximised at price vectors that correspond to truth assignments satisfying the largest number of clauses.*

Proof. Let $\boldsymbol{p}$ be a valid price vector and $\beta(\boldsymbol{p})$ be its corresponding truth assignment. Then each variable bid contributes a revenue of $21\rho(X_i)$ by Observation 4.9. Thus, recalling that $\sum_{i\in[N]} \rho(X_i) = 2K$, the total contribution from variable bids to overall revenue is $\sum_{i\in[N]} 21\rho(X_i) = 42K$. On the other hand, the definition of $\beta(\boldsymbol{p})$ implies that each clause bid contributes a revenue of 2 if $\beta(\boldsymbol{p})$ satisfies its clause and 0 otherwise. Hence it follows that total revenue is $42K + 2k$, where k denotes the number of clauses that $\beta(\boldsymbol{p})$ satisfies.

By Lemma 4.8 and Lemma 4.11, revenue for $\mathcal{P}_\varphi$ is maximised at some valid price vector $\boldsymbol{p}^*$. Let k^* be the number of clauses that $\beta(\boldsymbol{p}^*)$ satisfies. We claim that $\beta(\boldsymbol{p}^*)$ is an assignment that satisfies that the largest number of clauses. Indeed, suppose there exists a truth assignment β that satisfies $k > k^*$ clauses. Then overall revenue at $\boldsymbol{p}(\beta)$ is $42K + 2k > 42K + 2k^*$, in contradiction to our assumption that $\boldsymbol{p}^*$ is revenue-maximising. □

Theorem 4.14. *If 2-RevenueMax can be approximated to within a factor of $1 - \frac{3}{87}\varepsilon$ in polynomial time, then Max-2Sat can be approximated to within a factor of $1 - \varepsilon$ in polynomial time.*

Proof. Suppose there exists an algorithm $\mathcal{A}$ that finds a $1-\frac{3}{87}\varepsilon$-approximate solution of any 2-RevenueMax instance in polynomial time. We show how this leads to a polynomial-time algorithm for approximating Max-2Sat up to factor $1 - \varepsilon$.

Let φ be an instance of Max-2Sat and $\mathcal{P}_\varphi$ be the corresponding 2-RevenueMax instance. We can run $\mathcal{A}$ to obtain a price vector $\hat{\boldsymbol{p}}$ at which revenue is within a factor $1 - \frac{3}{87}\varepsilon$ of the maximum revenue. Without loss of generality, we assume that $\hat{\boldsymbol{p}}$ is valid; otherwise, we round the price vector as in Lemma 4.8 and flip values

of p_i without decreasing the revenue. Let $\hat{k}$ be number of clauses of φ satisfied by $\beta(\hat{\boldsymbol{p}})$ and let k^* be the largest number of clauses satisfiable by any truth assignment. By Lemma 4.13, revenue at $\hat{p}$ is $42K + 2\hat{k}$ and the maximum achievable revenue for $\mathcal{P}_\varphi$ is $42K + 2k^*$.

We show that $\beta(\hat{\boldsymbol{p}})$ is a $1-\varepsilon$-approximate solution to the Max-2Sat instance φ, i.e. that $\hat{k} \geq (1-\varepsilon)k^*$. Indeed, suppose to the contrary that $\hat{k} < (1-\varepsilon)k^*$. Then we can upper bound the approximation ratio of $\hat{\boldsymbol{p}}$ as

$$\frac{42K + 2\hat{k}}{42K + 2k^*} < \frac{42K + 2(1-\varepsilon)k^*}{42K + 2k^*} = 1 - \frac{2\varepsilon k^*}{42K + 2k^*} \leq 1 - \frac{3}{87}\varepsilon,$$

where the last inequality follows from the fact that $\frac{2\varepsilon k^*}{42K+2k^*}$ is monotonically increasing and $k^* \geq \frac{3}{4}K$ by Lemma 4.7. This contradicts our assumption that $\hat{p}$ is a $1 - \frac{3}{87}\varepsilon$-approximate solution. □

Under the reasonable assumption that the unique games conjecture and the Majority is Stablest conjecture hold, Khot et al. (2007) showed that Max-2Sat is NP-hard to approximate to within factor β, where

$$\beta = \min_{\theta \in (\frac{\pi}{2}, \pi)} \frac{2 + \frac{2}{\pi}\theta}{3 - \cos\theta} \approx 0.943943.$$

We note that the Majority is Stablest conjecture has been proved (De et al., 2013). Moreover, under the weaker assumption that $P \neq NP$, Håstad (2001) proved a bound of 21/22 for polynomial-time approximability of Max-2Sat. This implies the following for 2-RevenueMax.

Corollary 4.15. *If the unique games conjecture holds, then it is NP-hard to approximate* 2-RevenueMax *to within an approximation ratio of* $\frac{84+3\beta}{87} \approx 0.998$. *Under the weaker assumption of* $P \neq NP$, *this approximation ratio becomes* $\frac{637}{638} \approx 0.99643$.

4.4 Algorithms for maximising revenue

In light of the APX-hardness established in Section 4.3, we cannot expect to find an algorithm that (approximately) maximises revenue in polynomial time. However, this does not preclude the development of algorithms that are efficient in special cases, and of approximation algorithms that run in super-polynomial time. The algorithms for finding an exact solution presented below can be seen as first steps towards more sophisticated procedures for solving the product-mix auction for revenue efficiently in practice.

4.4.1 Additive demand

Suppose each bid has only one non-zero entry. In this case, the bids express additive demand. In particular, we can solve the auction by solving n one-dimensional

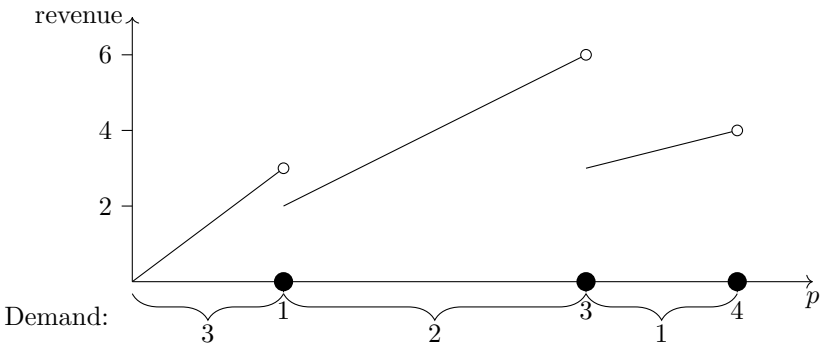


Figure 4.2: An example of a one-dimensional RevenueMax instance with three bids $\{(1;1),(3;1),(4;1)\}$. Note that revenue is maximised at $p = 3$.

RevenueMax instances with supplies s_i and taking the element-wise maximum over all n revenue-maximising price vectors. Let $\mathcal{A}^i := \{\boldsymbol{b} \in \mathcal{A} \mid b_i > 0\}$ and $\mathcal{B}^i := \{\boldsymbol{b} \in \mathcal{B} \mid b_i > 0\}$ denote the auctioneer and buyer bids that have a positive entry for good i. Note that solving the one-dimensional RevenueMax problem for $(\mathcal{B}^i, s_i)$ using our naive method reduces to checking the set of prices coinciding with the bid values $\mathcal{A}^i \cup \mathcal{B}^i$. Moreover, by sorting these bid values in descending order, we can iterate over the bid values and incrementally build up the bundle (of good i) demanded at each price directly (instead of solving the min-cost network flow problem described above). This results in a running time of $O((|\mathcal{A}^i| + |\mathcal{B}^i|)\log(|\mathcal{A}^i| + |\mathcal{B}^i|))$ for each one-dimensional instance, and an overall complexity of $O(n(|\mathcal{A}|+|\mathcal{B}|)\log(|\mathcal{A}|+|\mathcal{B}|))$. This is closed to the information-theoretic lower bound of $(n(|\mathcal{A}| + |\mathcal{B}|)$ to even read the entire bid list. Figure 4.2 illustrates a one-dimensional RevenueMax instance.

4.4.2 Iterating over bundles

Suppose we fix $k \leq n$. We now present an algorithm for solving k-RevenueMax that works in demand space by iterating over a set of candidate bundles and determining a revenue-maximising bundle among them. This can be seen as the 'dual' to our naive approach, which works in price-space by iterating over a set of candidate price vector and determine a revenue-maximising vector among them. Our initial algorithm runs in time $(k+1)^B f(n, B, \log M)$, where f denotes the time it takes the LongStepMinUp algorithm (cf. Lemma 4.5) to compute revenue-maximising prices for a given bundle. In the general case of RevenueMax (so setting $k = n$), we have $O(n^B f(n, B, \log M))$. In Section 4.4.2.2, we show how we can eliminate a factor n from this running time.

4.4.2.1 The general idea

Recall that we assumed (without loss of generality) that the auctioneer rejects a bid if it is indifferent between the reject good 0 and another good. At the heart of our algorithm lies the observation that a bid with at most k non-zero entries can only demand one of $k + 1$ different goods (the reject good 0 and the non-reject goods i for which $b_i > 0$). For each bid $\boldsymbol{b}$, we define the set of demanded

goods $I_{\boldsymbol{b}} := \{0\} \cup \{i \in [n] \mid b_i > 0\}$, which has a cardinality of at most $k+1$. Hence, if π is an envy-free allocation at $\boldsymbol{p}$, then $\pi(\boldsymbol{b}) = \boldsymbol{e}^i(\boldsymbol{b})$, where $i(\boldsymbol{b})$ is one of the goods in $I_{\mathcal{B}}$. It follows that there are at most $(k+1)^B$ distinct bundles that can be demanded by $\mathcal{B}$, where we recall that B denotes the number of bids. Let $\mathcal{X}$ be this set of possible bundles.

Lemma 4.5 discusses how to find, for any given bundle $\boldsymbol{x}$, the element-wise maximal price vector $\boldsymbol{p}$ at which $\boldsymbol{x}$ is demanded. We iterate over all bundles $\boldsymbol{x}$ in $\mathcal{X}$, computing for each bundle that satisfies $\boldsymbol{x} \leq \boldsymbol{s}$ the corresponding element-wise maximal price vector $\boldsymbol{p}^{\boldsymbol{x}}$ of its demand region and checking the revenue attained for this pair $(\boldsymbol{p}^{\boldsymbol{x}}, \boldsymbol{x})$. As $\mathcal{X}$ contains all possible bundles demanded by the bidder, it also contains all bundles that form part of a solution $(\boldsymbol{p}, \boldsymbol{x})$ to k-RevenueMax, so our approach is guaranteed to find such a solution. To see the running time, note that the main loop iterates over $(k+1)^B$ bundles and each iteration is dominated by the task of computing the element-wise maximal price vector, which can be done in time polynomial in n, B and $\log M$.

4.4.2.2 An improvement

We now turn to improving the algorithm above in the general case of RevenueMax (where $k = n$). Recall that iterating over all possible goods for each bid and checking whether the resulting bundle is revenue-maximising takes time $O\left(n^B f(n, B, \log M)\right)$. We describe a modification that brings down the running time by a factor of n at the cost of increasing the dependence in terms of B. The overall running time is $O(B^2 n^{B-1} f(n, B, \log M))$.

The improvement exploits the intuition that for some bids we do not need to iterate over all goods in $[n]_0$ and can instead restrict ourselves to a subset of size at most B. We now develop this idea formally. Let $(\boldsymbol{p}, \pi)$ be a revenue-maximising solution and let $I = \{i \in [n] \mid \pi(\boldsymbol{b}) = \boldsymbol{e}^i \text{ for some } \boldsymbol{b} \in \mathcal{B}\}$ be the set of goods of which non-zero quantities are allocated to bidders. Every revenue-maximising price vector $\boldsymbol{p}$ has a highest-priced good i among all the goods allocated to bids. Without loss of generality, we can assume that good 1 lies in I and is priced highest, i.e. $p_1 \geq p_i$ for all $i \in I$ (relabelling goods if necessary).

Fix a bid $\boldsymbol{b} \in \mathcal{B}$ that demands good 1. Such a bid must exist by construction of I. Lemma 4.16, stated below, establishes that $b_1 \geq b_i$ for all $i \in [n]$. Consider the set $M = \arg\max_{i \in [n]} b_i$ and let M' denote any set obtained by removing elements from M one by one until the resulting set has at most B entries. Note that if $|M| \leq B$, then we have $M = M'$ and thus $1 \in M'$. On the other hand, if $|M| > B$, then M' may, or may not, contain good 1.

Suppose $1 \notin M'$. Then we show that there exists an alternative revenue-maximising solution $(\boldsymbol{p}', \pi')$ in which $\boldsymbol{b}$ demands some other good $i \in M' \setminus I$ instead. We do this by modifying $(\boldsymbol{p}, \pi)$ slightly. First of all, we see that $M' \setminus I$ is not empty. This follows from the fact that $1 \in I \setminus M'$, while $|M'| = B$ and $|I| \leq B$. Fix any good i in $M' \setminus I$. We obtain prices $\boldsymbol{p}'$ from $\boldsymbol{p}$ by setting the price p_i of good i to the value of p_1 and keeping all other prices the same. As $i \in M'$, we have $b_i = b_1$, so bid $\boldsymbol{b}$ is now indifferent between goods 1 and i, and we can obtain a new allocation π' from π by changing $\boldsymbol{b}$'s allocation from good 1 to the good

$\pi'(\boldsymbol{b}) = \boldsymbol{e}^i$ without affecting the revenue contribution from $\boldsymbol{b}$. Similarly, every other bid $\boldsymbol{b}' \in \mathcal{B}$ now uniquely demands good i, so we change its allocation to $\pi'(\boldsymbol{b}') = i$, or it can be allocated the same good as before. As $p_i \geq p_k$ for all $k \in [n]$, the revenue contribution from $\boldsymbol{b}'$ cannot decrease. Moreover, as $\boldsymbol{p}$ is revenue-maximal, it follows that all bids contribute the same revenue at $\boldsymbol{p}'$ and $\boldsymbol{p}$, and thus $(\boldsymbol{p}', \pi')$ is also a revenue-maximising solution.

This implies Algorithm 4.1, where we iterate over all goods in M' to allocate to bid $\boldsymbol{b}$, and over all goods $[n]_0$ for all other bids.

Algorithm 4.1 Computing a revenue-maximising bundle

Input: List of bids $\mathcal{B}$
Output: Revenue-maximising bundle $\boldsymbol{x}$

1: **for** every bid $\boldsymbol{b} \in \mathcal{B}$ **do**
2: Label the remaining bids $\mathcal{B} \setminus \{\boldsymbol{b}\} = \{b^1, \ldots, b^{B-1}\}$.
3: Compute $M = \arg\max_{i \in [n]} \boldsymbol{b}$ and remove elements until it contains at most B goods.
4: **for** every good g in M and every $(g^1, \ldots, g^{B-1}) \in [n]_0^{B-1}$ **do**
5: Define allocation π with $\pi(\boldsymbol{b}) = g$ and $\pi(\boldsymbol{b}^j) = g^j$ for $j \in \mathcal{B} \setminus \{\boldsymbol{b}\}$.
6: Attempt to find revenue-maximal prices $\boldsymbol{p}$ for π (cf. Lemma 4.5). If they exist, determine $\boldsymbol{p} \cdot \pi(\mathcal{B})$.
7: **end for**
8: **end for**
9: **return** $(\boldsymbol{p}, \pi)$ that achieves the greatest revenue.

We argue that the algorithm returns a revenue-maximising allocation. Let $\boldsymbol{b}$ be a bid that demands a highest-priced good for some revenue-maximal solution. By the above, we know that there exists a revenue-maximal solution $(\boldsymbol{p}', \pi')$ at which some good $i \in M'$ is allocated to $\boldsymbol{b}$. The algorithm will consider the allocation π', and thus return a revenue-maximising solution.

Lemma 4.16. *Let $(\boldsymbol{p}, \pi)$ be a revenue-maximising solution and, without loss of generality, let $p_1 \geq p_i$ for all goods $i \in I$ allocated to some bid. If $\boldsymbol{b}$ is a bid that receives an item of good i under π, then $1 \in \arg\max_{i \in [n]} b_i$.*

Proof. Suppose that $b_i > b_1$ for some $i \in [n]$. First we show that good i cannot be demanded by any bid. Indeed, if i were demanded by some bid we would have $p_i \leq p_1$ and thus $b_i - p_i > b_1 - p_1$. In other words, $\boldsymbol{b}$ would demand i instead of 1, in contradiction to the fact that $\boldsymbol{b}$ demands good 1. Hence, as no bid is assigned an item of good i, we have $p_i = \infty$ by assumption. Consider the modified price vector $\boldsymbol{p}'$ obtained from $\boldsymbol{p}$ by changing p_i to $p'_i = b_i - (b_1 - p_1)$ and leaving all other prices unchanged. As $b_i > b_1$, we have $p'_i > p'_1$ and it follows that $p'_i > p'_j$ for all goods $j \in I \setminus \{i\}$ that are allocated to some bid. Moreover, bid $\boldsymbol{b}$ is now indifferent between goods 1 and i, so we modify the existing allocation by allocating an item of good i to $\boldsymbol{b}$, which strictly increases its revenue contribution. All other bids either continue to demand the same good, or switch their demand to good i, so their revenue contribution either remains unchanged or increases. Hence the

overall revenue strictly increases, in contradiction to our assumption that $\boldsymbol{p}$ is revenue-maximal. □

4.5 Conclusion and outlook

This chapter has set out the computational challenge of solving the strong-substitutes product-mix auction with respect to maximising revenue. This objective occurs in many natural applications such as in ad auctions. In contrast to the social welfare objective, which admits polynomial-time algorithms (e.g. those presented in Chapter 3), we see that maximising revenue is APX-hard even if we restrict bidders to submitting positive bids with at most two non-zero bid entries only (this is the 2-RevenueMax problem). Given these complexity-theoretic insights, it is of interest to explore algorithms that either exactly, or approximately, determine optimum revenue in special cases of practical interest. In future work, we hope to build on the algorithmic insights presented in this chapter to improve on existing results. Of special theoretical interest would be to identify whether RevenueMax or k-RevenueMax admits a fixed-parameter tractable algorithm. Here our problem can be parametrised by various parameters including the number of goods, and the number of bids. (We note that the maximum number of non-zero entries in a bid is not suitable parameter for parametrised complexity, as we have shown that k-RevenueMax is APX-hard even for $k = 2$, and trivial for $k = 1$.) Another pertinent question with regard to the tropical bidding language is whether existing approximation algorithms by Guruswami et al. (2005) and others can be extended to bidder demands with positive *and* negative tropical bids, as this would allows bidders to express general strong-substitutes demands.

5

Learning strong-substitutes demand

Contents

5.1 Chapter outline 97
5.2 Introduction 98
5.2.1 Preliminaries 100
5.3 The geometry of tropical bids 101
5.4 The computational challenges 103
5.5 Simulating $\mathcal{Q}_{\mathcal{B}}$ with a valuation oracle 104
5.6 Learning positively-weighted bids 104
5.6.1 Finding a single positive bid 105
5.7 Learning positive and negative bids 108
5.7.1 Bid existence queries 108
5.7.2 Super queries 110
5.7.3 Finding a separating hyperplane 111
5.7.4 The main algorithm 112
5.8 Lower bounds 114
5.8.1 The island gadget 115
5.8.2 A lower bound for positive and negative bids 117
5.9 Conclusion and outlook 120
5.9.1 Future work 120

5.1 Chapter outline

This chapter addresses the computational challenges of expressing strong-substitutes demand in the tropical bidding language when given access to a demand (or valuation) oracle. This work has been published in (Goldberg et al., 2020) and is joint work with Paul Goldberg and Francisco Marmolejo-Cossío.

Recall from Chapter 2 that the language of positive tropical bids is essentially equivalent to the logical language OXS extended to multiple units. We will see that restricting ourselves to learning positive bids (or, equivalently, OXS preferences) reduces the computational complexity of learning demand considerably.

Section 5.2 motivates the problem, outlines our contributions and briefly surveys related work before introducing notation we will use throughout the chapter. Our algorithms for learning bid lists exploit the geometry of strong-substitutes demand. To this end, Section 5.3 outlines a geometric characterisation of tropical bids. A first result of this chapter, given in Section 5.5, is to show that demand oracles are not unreasonably powerful: when given access instead to a valuation oracle, it is possible to simulate a demand oracle with $O(n^3 \log(W/n))$ valuation queries, where n is the number of goods and W is the maximum weight of a bid vector.

In Sections 5.6 and 5.7, we consider algorithms that learn the unique bid list corresponding to a bidder's demand. The algorithm in Section 5.6 learns demands that can be represented by lists of positive bids, and has linear query complexity, while our algorithm in Section 5.7 is applicable in the setting where demand may require positive and negative bids to express.

Finally, in Section 5.8 we consider lower bounds on the query complexity of learning bid lists. We note briefly that $\Omega(B \log M)$ queries are required to learn a list of B positive bids, where M is the magnitude of the bid vectors w.r.t. the L_∞ norm and show super-polynomial lower bounds on the query complexity of computing the list of bids in the general case where bids may be positive and negative.

5.2 Introduction

In the version of the strong-substitutes product-mix auction currently implemented by the Bank of England, bidders are restricted to submitting bids with one non-zero vector entry and positive weights, and it is straightforward for bidders to construct bid lists corresponding to their demand. However, in the past, the Bank also allowed bidders to submit arbitrary positively-weighted bids. Moreover, in its most general form, bidders participating in the strong-substitutes product-mix auction are able to submit arbitrary strong-substitutes preferences as a list of positive and negative tropical bids. Indeed, we saw in Chapter 2 that negative bids are required to express some strong-substitutes preferences. Moreover, we recall that any strong-substitutes demand can be expressed using positive and negative bids (Baldwin and Klemperer, 2021).

But while this gives the buyer a general-purpose means of communicating any strong substitutes demand, the buyer faces the practical problem of expressing her demand in this language. It may be easier for a buyer to answer queries of the form "what bundle would you demand, given the following per-unit prices of goods?". In this chapter, we develop query protocols that assist a buyer in constructing her demand function based on a sequence of queries. Given an unknown demand function, the algorithm is assumed to have access to a demand oracle: for any given prices for goods, the algorithm can learn a bundle of goods demanded at those prices. We are interested in minimising the number of queries

to the demand oracle. Moreover, we also show how to simulate a demand oracle query with a polynomial number of queries to the valuation oracle. This answers the computational challenge of 'expression' outlined in Chapter 2 for the tropical bidding language. To the best of our knowledge, this work constitutes the first study of learning algorithms for bidding languages.

Our contributions. This chapter addresses the computational challenges of learning strong substitutes demand when given access to a demand oracle. Under the mild assumption that bidders are able to answer questions of the form "What bundle do you demand at the following per-unit prices?", our algorithms constitute a systematic approach for bidders to generate a bid list corresponding to their demand, allowing them to participate in product-mix auctions with non-trivial demand preferences. We provide upper and lower bounds on the query complexity of learning demand preferences and expressing these in the tropical bidding language, which is able to encode any strong substitutes demand in a conceptually simple and natural fashion.

A first result of this chapter, given in Section 5.5, is to show that demand oracles are not unreasonably powerful: when given access instead to a valuation oracle, it is possible to simulate a demand oracle with $O(n^3 \log(W/n))$ valuation queries, where n is the number of goods and W is the maximum weight of a bid vector.

Our algorithms for learning the unique bid list corresponding to a bidder's demand are motivated by the tropical geometric insights of Baldwin and Klemperer (2019). If demand is expressible with positive bids only, we provide a linear-time learning algorithm. In the setting where demand may require positive and negative bids to express, we provide an exponential-cost algorithm that proceeds by learning all hyperplanes that contain facets of the Locus of Indifference Prices (LIP), a geometric object introduced by Baldwin and Klemperer (2019) and introduced in Chapter 1 to characterise demand.

Complementing the exponential query complexity of our general learning algorithm, we consider lower bounds on the query complexity of learning bid lists. These lower bounds hold for adaptive deterministic learning algorithms; adaptive algorithms make successive queries to the demand oracle and are permitted to take into account the results from prior queries to decide on which future queries to make. We note briefly that $\Omega(B \log M)$ queries are required to learn a list of B positive bids, where M is the magnitude of the bid vectors w.r.t. the L_∞ norm. In order to identify the dependence on the number of goods n, we construct a game that an adversary can play to force a large number of queries using a novel 'island gadget' consisting of bids with weight ± 1. Crucially, the island gadget only changes demand in a local region. For fixed n, we identify the overall query complexity of learning bid lists corresponding to strong substitutes demand as $\Theta(B \log M + B^n)$.

Overall, our algorithms constitute the first systematic approach for bidders to construct a bid list corresponding to non-trivial demand, allowing them to participate in the strong-substitutes product-mix auction.

Related work. Our work relates to the theory of preference elicitation. In this setting, a centralised agent, such as an auctioneer, wishes to identify an optimal allocation of goods via queries to participants' preferences. Queries typically take the form of value queries, where an agent reports a valuation for a given bundle of goods, or demand queries, where an agent reports a bundle that is demanded at given prices. This chapter focuses on using demand queries to learn the bid list representation of strong substitutes demand preferences. This representation can then be used to compute an optimal allocation of goods to agents via the methods developed in Chapter 3.

Although the goal in preference elicitation is to compute an optimal allocation without necessarily learning entire valuation functions of participants, a strong connection between preference elicitation and active learning with membership and equivalence queries in computational learning theory has been established. Much early work in preference elicitation highlights the deep connections to exact learning via membership and equivalence queries from computational learning theory, and our results can also be viewed through this lens. Some notable examples include (Zinkevich et al., 2003) and (Blum et al., 2004). Conen and Sandholm (2001) explore the use of ranking oracles to exploit the topological structure of bidder preferences to learn optimal allocations. This approach is extended in (Conen and Sandholm, 2002a,b), and verified empirically in (Hudson and Sandholm, 2004). Moreover, Nisan and Segal (2006) explore the communication complexity of preference elicitation in combinatorial auctions, where they show that for general valuations, finding a value-maximising allocation requires an exponential communication cost in the number of items. In (Lahaie and Parkes, 2004), the authors explore connections between preference elicitation and exact learning, but they demonstrate that the representation length of the valuation is an important parameter in the query complexity of computing optimal allocations. This dependence on the representation length provides a way of side-stepping lower bounds from (Nisan and Segal, 2006), and further justifies the need for succinct yet expressive bidding languages, as described in Chapter 2.

Our problem is conceptually similar to a problem studied recently in (Zhang and Conitzer, 2020), in which the authors also consider algorithms with access to demand queries, and attempt to learn the underlying valuation that gives rise to the demand correspondence. The main difference between their work and ours however is that they consider a different class of value functions. In (Zhang and Conitzer, 2020), there are n goods, each in unit supply (whereas we allow multiple copies of goods), and the buyer wants at most k goods, and has additive valuations (whereas our strong substitutes valuations are more general).

5.2.1 Preliminaries

We assume that a bidder has strong-substitutes demand over n goods. As before, 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 element-wise. For any vector $\boldsymbol{v} \in \mathbb{R}^n$, the L_1 and L_∞ norms are defined as $\|\boldsymbol{v}\|_1 = \sum_{i \in [n]} |v_i|$ and

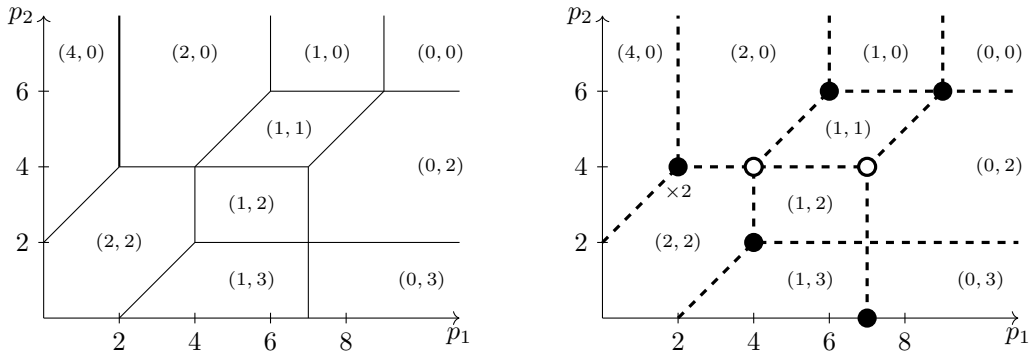


Figure 5.1: Left: An illustration of a strong-substitutes demand correspondence with two goods, partitioning price space into piecewise-linear convex regions. Each region is labelled with the bundle demanded at prices in the region. The dashed lines comprise the Locus of Indifference Prices (LIP). Right: Six positive (solid) and two negative (hollow) bids of unit weight are required to express this demand.

$\|\boldsymbol{v}\|_\infty = \max_{i\in[n]} |v_i|$. The L_∞ ball $B^\varepsilon_{\boldsymbol{p}}$ of radius ε at $\boldsymbol{p}$ consists of all points $\boldsymbol{q} \in \mathbb{R}^n$ that satisfy $\|\boldsymbol{p} - \boldsymbol{q}\|_\infty \leq \varepsilon$; note that this a hypercube with edge length 2ε. Any hypercube centred at $\boldsymbol{p}$ can be partitioned into 2^n *orthants*, where every orthant O_a is described by some vector $\boldsymbol{a} \in \{-1, 1\}^n$ and consists of the set of points $O_a = \boldsymbol{p} + \{\boldsymbol{x} \in \mathbb{R}^n \mid a_i x_i \geq 0, \forall i \in [n]\}$.[1] Every such orthant can be triangulated into $n!$ simplices as follows. For every ordering $[i_1, \ldots, i_n]$ of the indices $[n]$, we define a simplex as the set of points in the orthant that satisfy $a_{i_1} x_{i_1} \leq a_{i_2} x_{i_2} \leq \ldots \leq a_{i_n} x_{i_n}$.

5.3 The geometry of tropical bids

We first recall from Chapter 1 some geometric intuition for strong substitutes demand correspondences that underpins the algorithmic ideas in this chapter. Any quasi-linear demand divides price space into piecewise-linear convex regions corresponding to bundles. When demand is strong-substitutes, each such region is a convex lattice (Murota et al., 2013). Figure 5.1 illustrates this. Recall from Chapter 1 that Baldwin and Klemperer (2019) propose a new way of characterising demand types by introducing the *Locus of Indifference Prices* (LIP), a weighted polyhedral complex with $(n-1)$-dimensional facets. (In Fig. 5.1, the LIP is drawn using dashed lines.) In particular, the strong substitutes demand type is defined as the family of demand correspondences whose LIP facets are normal to $\boldsymbol{e}^i$ or $\boldsymbol{e}^i - \boldsymbol{e}^j$ for some $i, j \in [n]$. (In two dimensions, facets of SS LIPs are either horizontal, vertical or normal to $(1, -1)$. Hence it follows directly from this definition that the demand correspondence in Fig. 5.1 enjoys the strong substitutes property.)

As described in Chapter 2, the tropical bidding language allows bidder to submit tropical bids consisting of an n-dimensional integral vector $\boldsymbol{b} \in \mathbb{Z}^n$ and a weight

[1] Orthants in n-dimensional space generalise the notion of quadrants and octants in two- and three-dimensional space, respectively.

$w(\boldsymbol{b}) \in \mathbb{Z}$. (When working with the notional reject good 0 introduced above, we identify a bid vector $\boldsymbol{b}$ with the $(n+1)$-dimensional vector obtained by adding a 0-th entry of value 0.) We note that any bid with a weight of $w(\boldsymbol{b}) \in \mathbb{Z}$ is equivalent to $w(\boldsymbol{b})$ unit bids with the same vector and sign. This allows us to normalise bid lists to their most succinct form, where no two bids share the same vector. In this chapter, we wish to learn the unique normalised bid list that represents the bidder's demand correspondence.

A special subclass of the strong substitutes demand type is the family of demand correspondences that can be expressed using only positive bids. This family is of particular practical interest, as the Bank of England currently runs the Product-Mix Auction with positive bids only. Note that any list of positive bids is valid, as it trivially satisfies Theorem 2.5.

In Chapter 2, we explained the algebraic relationship between a bid list and its resulting demand correspondence. Here we provide a geometric view on this demand correspondence, which underpins our algorithms. Fix a bid $\boldsymbol{b} \in \mathbb{Z}^n$ and an item $i \in [n]_0$. We let R_i denote the set of prices at which $\boldsymbol{b}$ demands good i. Each R_i is an unbounded convex polytope in $\mathbb{R}^n$, which can be expressed succinctly as $R_i = \boldsymbol{b} + H_i$, where H_i is the conic hull of the vectors $\boldsymbol{e}^j$, $j \in [n]_0 \setminus \{i\}$, if we define $\boldsymbol{e}^0 = -\boldsymbol{e}^{[n]}$ (see Fig. 5.2a). Note that each R_i is of full affine dimension and the R_i together cover the entirety of $\mathbb{R}^n$. In line with the previous section, $\boldsymbol{b}$ is indifferent between two goods $i, j \in [n]_0$ at $\boldsymbol{p}$ if, and only if, $\boldsymbol{p} \in R_i \cap R_j$. Moreover, if a price lies in the interior of any given R_i, good i is the unique good demanded by $\boldsymbol{b}$. For a given list of bids $\mathcal{B}$, we recall that a price $\boldsymbol{p}$ is non-marginal if each bid in $\mathcal{B}$ demands a unique item at $\boldsymbol{p}$. If we define $R_i^j := \boldsymbol{b}^j + H_i$ for each $\boldsymbol{b}^j \in \mathcal{B}$, then it is straightforward to see that $\boldsymbol{p}$ is non-marginal if, and only if, it does not lie on the boundary of any R_i^j. This allows us to describe the geometry of the demand correspondence $D_{\mathcal{B}}$. Suppose that $\boldsymbol{p}$ is non-marginal, and that for each $\boldsymbol{b} \in \mathcal{B}$, $i(\boldsymbol{b})$ is the unique good demanded by $\boldsymbol{b}$ at $\boldsymbol{p}$. We recall that the unique bundle demanded at $\boldsymbol{p}$ is obtained by adding $w(\boldsymbol{b})$ items of good $i(\boldsymbol{b})$ for each $\boldsymbol{b} \in \mathcal{B}$. We can express this as

$$D_{\mathcal{B}}(\boldsymbol{p}) = \sum_{\substack{(i,j):\\ \boldsymbol{p} \in R_i^j}} w(\boldsymbol{b}^j)\boldsymbol{e}^i.$$

This is illustrated in Fig. 5.2b. We extend the definition of $D_{\mathcal{B}}$ to marginal prices as above: if $\boldsymbol{p}$ is marginal, $D_{\mathcal{B}}(\boldsymbol{p})$ is the discrete convex hull of the bundles demanded at non-marginal prices arbitrarily close to $\boldsymbol{p}$.

Valid bid lists. As mentioned in Chapter 2, not all demand correspondences arising from bid lists encode a quasi-linear, strong substitutes demand correspondence D_v. In particular, we established in Chapter 2 that the bid list must satisfy the equivalent validity properties given in Theorem 2.5. We now briefly explore what validity means geometrically. Suppose that $\mathcal{B}$ is a bid list, and for any given price $\boldsymbol{p}$, define the (i, ℓ)-*support* $\mathrm{supp}_{i,\ell}(\boldsymbol{p})$ of $\boldsymbol{p}$ to be the set of all bids $\boldsymbol{b}^j \in \mathcal{B}$ such that $\boldsymbol{p} \in R_i^j \cap R_\ell^j$. In other words, all bids that are indifferent between goods i and ℓ at price $\boldsymbol{p}$. This allows us to give a condition for non-validity of

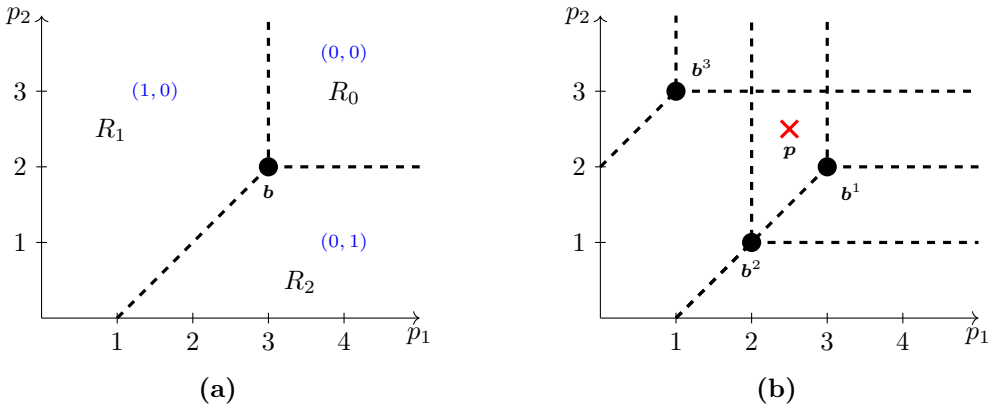


Figure 5.2: Left: The demand regions R_i of a single positive bid $\boldsymbol{b}$ in the case of two goods $1, 2$ and the reject good 0. Right: a three-element bid list $\mathcal{B} = \{\boldsymbol{b}^1, \boldsymbol{b}^2, \boldsymbol{b}^3\}$. The price vector $\boldsymbol{p}$ (marked by a red cross) lies in R_1^1, R_0^2, and R_2^3. Hence, at prices $\boldsymbol{p}$, the bid $\boldsymbol{b}^1$ demands good 1, $\boldsymbol{b}^2$ is rejected, and $\boldsymbol{b}^3$ demands good 2. Putting this together, $D_{\mathcal{B}}(\boldsymbol{p}) = (1, 1)$.

our bid list $\mathcal{B}$. In particular, $\mathcal{B}$ is valid if, and only if, $\sum_{\boldsymbol{b} \in \operatorname{supp}_{i,\ell}(\boldsymbol{p})} w(\boldsymbol{b}) \geq 0$ for all prices $\boldsymbol{p}$ and pairs of goods $i, \ell \in [n]_0$.

5.4 The computational challenges

Consider a bidder who has an (unknown) strong-substitutes demand correspondence D_v on n goods. We study the problem of learning the unique list $\mathcal{B}$ of positive and negative bids of unit weight that represent a bidder's demand correspondence, i.e. such that $D_v = D_{\mathcal{B}}$. We consider algorithms that learn $\mathcal{B}$ by querying the demand correspondence D_v at different price vectors. More specifically, our algorithms have access to an *adversarial demand oracle* $\mathcal{Q}_{\mathcal{B}}$; given any price vector $\boldsymbol{p}$, $\mathcal{Q}_{\mathcal{B}}(\boldsymbol{p})$ returns a bundle from $D_{\mathcal{B}}(\boldsymbol{p})$. A bidder may demand multiple bundles at some price (i.e. when $|D_{\mathcal{B}}(\boldsymbol{p})| > 1$), in which case the adversarial oracle simply returns a single demanded bundle at that price, and we have no control over which such bundle is returned. Another related setting we address in Section 5.5 is the complexity of learning $\mathcal{B}$ given access to a *valuation oracle*, i.e. given a bundle $\boldsymbol{x}$, the bidder reports their valuation $v(\boldsymbol{x})$ for this bundle.

Let $B := |\mathcal{B}|$ be the number of bids we wish to learn. Moreover, let $M := \max_{\boldsymbol{b} \in \mathcal{B}} \|\boldsymbol{b}\|_\infty$ be the *magnitude* of the bids w.r.t to the L_∞ norm and $W := \max_{\boldsymbol{b} \in \mathcal{B}} w(\boldsymbol{b})$ be the maximum bid weight. Note that for any unknown bid list $\mathcal{B}$, we can determine the value of M with $O(\log M)$ demand queries, as M corresponds to the smallest value m such that the bidder demands the empty bundle at price vector $\boldsymbol{p} = m\boldsymbol{e}^{[n]}$, which can be found using binary search. We are interested in the query complexity of learning $\mathcal{B}$, measured in terms of n, B, $\log M$ and $\log W$. Note that $nB \log M + B \log W$ bits are required to store the bid list $\mathcal{B}$, under the natural assumption that bid vectors and weights are encoded in binary.

5.5 Simulating $\mathcal{Q}_{\mathcal{B}}$ with a valuation oracle

Our algorithms assume that the bidder is able to answer demand queries. In this section, we show that our algorithms can be adapted to the setting where we only have access to a valuation oracle, without a super-polynomial increase in the asymptotic query complexity. We do this by showing that demand oracles are not unreasonably powerful, in the sense that we can use a valuation oracle to simulate a demand oracle with polynomial overhead. Consider the setting where we are given query access to a bidder's valuation function v. We show that a single query to $\mathcal{Q}_{\mathcal{B}}$ can be simulated with a polynomial number of queries to a valuation oracle. This result utilises the equivalence of the strong substitutes property and $M^\natural$-convexity from the discrete convex analysis literature introduced in Chapter 1.

Recall that the utility of bundle $\boldsymbol{x}$ at prices $\boldsymbol{p}$ is given by $u(\boldsymbol{x};\boldsymbol{p}) = v(\boldsymbol{x}) - \boldsymbol{p}\cdot\boldsymbol{x}$. We define $u_{\boldsymbol{p}} \coloneqq u(\cdot;\boldsymbol{p})$ for convenience. In order to simulate a demand oracle on input $\boldsymbol{p}$, we wish to compute a bundle $\boldsymbol{x} \in D_v$ that maximises $u_{\boldsymbol{p}}(\boldsymbol{x})$. Note that we can compute $u_{\boldsymbol{p}}(\boldsymbol{x})$ for any bundle $\boldsymbol{x}$ using a single query to the valuation oracle. In order to compute a maximiser of $u_{\boldsymbol{p}}(\cdot)$, we draw from the discrete convex analysis literature. Firstly, we see that $u_{\boldsymbol{p}}$ is $M^\natural$-concave. Indeed, it is well-known that strong substitutes valuations are $M^\natural$-concave (Shioura and Tamura, 2015) and subtracting a linear term preserves this property. Secondly, let $\hat{u}$ be the corresponding M-concave function to $u_{\boldsymbol{p}}$ as defined in (1.3). We see that maximising $u_{\boldsymbol{p}}$ is equivalent to maximising $\hat{u}$. Moreover, we can compute $\hat{u}(x_0, \boldsymbol{x})$ using at most one query to the valuation oracle. Thirdly, note that we have $\|x\|_1 \leq BW$ for any bundle $\boldsymbol{x}$ that the bidder demands, as every bid $\boldsymbol{b} \in \mathcal{B}$ contributes at most W items to $\boldsymbol{x}$.

Murota (2003, Chapter 10) provides multiple algorithms for maximising M-concave functions f with bounded effective domains $\operatorname{dom} f$. The simplest such algorithm, a straightforward steepest descent method, finds a maximiser with $O(n^2 L)$ queries, where $L \coloneqq \max\{\|x-y\|_1 \mid x, y \in \operatorname{dom} f\}$. In our setting, we have $L = BW$, yielding a query complexity of $O(n^2 BW)$. This query complexity can be improved to $O(n^3 \log(BW/n))$ by applying the more involved algorithms for maximising M-concave functions given in (Shioura, 2004) and (Tamura, 2004). We note that this query complexity is polynomial in n, B and $\log W$.

5.6 Learning positively-weighted bids

In this section we assume that the bidder's demand correspondence can be expressed by a list of positive bids.[2] Our algorithm learns a list of B positive bids using $O(nB \log M)$ demand queries. This is close to our lower bound of $\Omega(B \log M)$ given in Theorem 5.15 below.

We proceed by repeatedly finding a bid and 'removing' it, thereby reducing the size of the remaining demand correspondence until all bids have been found. Let $\mathcal{L}$ denote the subset of bids from $\mathcal{B}$ that have already been learnt, and let $\mathcal{B}' \coloneqq \mathcal{B} - \mathcal{L}$ be the list of remaining bids. We can simulate a demand oracle $\mathcal{Q}_{\mathcal{B}'}(\boldsymbol{p})$ for the demand correspondence associated with $\mathcal{B}'$ as follows. At price vector $\boldsymbol{p}$,

[2] As we showed in Chapter 2, this essentially corresponds to the OXS demand type.

first determine a bundle $\boldsymbol{x}$ demanded by all bids in $\mathcal{B}$ with a single query $\mathcal{Q}_{\mathcal{B}}(\boldsymbol{p})$, and then subtract from $\boldsymbol{x}$ a bundle $\boldsymbol{y}$ demanded at $\boldsymbol{p}$ by the bids in $\mathcal{L}$.[3]

In this way, the problem of learning a list of positive bids reduces to repeatedly identifying a single bid. In the next section, we describe a subroutine that learns the location of a single bid in $\mathcal{B}'$ using $O(n \log M)$ queries. As this subroutine is called B times, this yields an overall query complexity of $O(nB \log M)$ for learning all bids in $\mathcal{B}$. Recall that we can compute M with $O(\log M)$ queries.

5.6.1 Finding a single positive bid

We present an algorithm that performs binary searches using *delta queries* to successively learn the coordinates $x_1, x_2, \ldots$ of a bid's location $\boldsymbol{x}$ together with its weight. We begin by defining delta queries and establishing some fundamental facts about the results returned by these queries.

Definition 5.1. A *delta query* $\Delta(\boldsymbol{q})$ at $\boldsymbol{q} \in \mathbb{Z}^n$ consists of two queries $\boldsymbol{q}^+$ and $\boldsymbol{q}^-$ defined by $\boldsymbol{q}^+ = \boldsymbol{q}' + \frac{1}{2n}\boldsymbol{e}^1$ and $\boldsymbol{q}^- = \boldsymbol{q}' - \frac{1}{2n}\boldsymbol{e}^1$, where we let

$$\boldsymbol{q}' \coloneqq \boldsymbol{q} + \sum_{i \in \{2,\ldots,n\}} \frac{1}{2(n-i+1)} \boldsymbol{e}^i.$$

The return value of the delta query is defined as $\Delta(\boldsymbol{q}) = x_1^- - x_1^+$, where $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ are the bundles of goods uniquely demanded at $\boldsymbol{q}^+$ and $\boldsymbol{q}^-$.

Note that $q_1^+ = q_1 + \frac{1}{2n}$ and $q_1^- = q_1 - \frac{1}{2n}$, and the two query points $\boldsymbol{q}^+$ and $\boldsymbol{q}^-$ agree on all other coordinates $i \geq 2$. Secondly, $\boldsymbol{q}^\pm$ is non-marginal by construction, so any bid $\boldsymbol{b} \in \mathbb{Z}^n$ uniquely demands some good i at $\boldsymbol{q}^\pm$. The intuition behind delta queries is as follows. Consider the hyperplane normal to $\boldsymbol{e}^1$ that contains $\boldsymbol{q}$. In a first step, we carefully perturb $\boldsymbol{q}$ such that the resulting point $\boldsymbol{q}'$ remains on the hyperplane and no bid is indifferent between any two goods in $\{2, \ldots, n\}$. The points $\boldsymbol{q}^-$ and $\boldsymbol{q}^+$ are then obtained by perturbing $\boldsymbol{q}'$ in directions $\pm\boldsymbol{e}^1$ such that the prices become non-marginal.

Lemma 5.2 makes the observation that bids $\boldsymbol{b}$ satisfying $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}$ demand good 1 at $\boldsymbol{q}^-$ and are rejected at $\boldsymbol{q}^+$, while all other bids demand the same good at both prices $\boldsymbol{q}^\pm$. Hence demand changes only in terms of good 1, and $\Delta(\boldsymbol{q})$ captures the magnitude of this change. In our current setting where all bids have positive weights, Corollary 5.3 notes that this is equivalent to summing the weights of the bids $\boldsymbol{b}$ that satisfy $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}$. Our algorithm exploits this fact in order to learn the coordinates of a bid location as well as the bid weight.

Lemma 5.2. *Suppose we place a delta query at $\boldsymbol{q} \in \mathbb{Z}^n$. Then any bid $\boldsymbol{b} \in \mathbb{Z}^n$ demands different goods at $\boldsymbol{q}^-$ and $\boldsymbol{q}^+$ if, and only if, $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}'$. Moreover, any such bid demands an item of good 1 at $\boldsymbol{q}^-$ and an item of the reject good 0 at $\boldsymbol{q}^+$.*

[3]Note that $\mathcal{B}'$ is valid, as lists of positive bids are always valid. If $\mathcal{B}$ consisted of positive and negative bids, removing a single positive bid might result in a bid list that is no longer valid. In this case, the algorithm described in this section may fail and return points not corresponding to bid locations.

Algorithm 5.1 Learning positive bids

1: Perform binary search to find largest price $\boldsymbol{p} \in \{(k, M, \ldots, M) \mid k \in [M]_0\}$ at which $\Delta(\boldsymbol{p}) > 0$, and fix $x_1 = p_1$.
2: **for** $i = 2 \ldots n$ **do**
3: Binary search to find the smallest price $\boldsymbol{p}$ at which $\Delta(\boldsymbol{p}) > 0$ in search domain $\{(x_1, \ldots, x_{i-1}, k, M, \ldots, M) \mid k \in [M]_0\}$, and fix $x_i = p_i$.
4: **end for**
5: **return** bid vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ and weight $\Delta(\boldsymbol{x})$.

Proof. Suppose $\boldsymbol{b}$ is a bid with $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}'$. As $\boldsymbol{b}$ is integral, we have $b_i < q_i^+ = q_i^-$ for all goods $i \geq 2$, which implies that $\boldsymbol{b}$ can only demand goods 0 or 1 at $\boldsymbol{q}^-$ and $\boldsymbol{q}^+$. At $\boldsymbol{q}^-$, bid $\boldsymbol{b}$ uniquely demands good 1, as we have $b_1 - q_1^- = \frac{1}{2n} > 0$. Similarly, we see that $\boldsymbol{b}$ uniquely demands the reject good 0 at $\boldsymbol{q}^+$, as $b_1 - q_1^+ = -\frac{1}{2n} < 0$.

Conversely, suppose $\boldsymbol{b}$ demands distinct goods i and j at $\boldsymbol{q}^-$ and $\boldsymbol{q}^+$, respectively. This implies the two fundamental inequalities (a) $b_i - q_i^- > b_j - q_j^-$ and (b) $b_j - q_j^+ > b_i - q_i^+$. We first show that we must have $i = 1$ and $j = 0$ by excluding all other possibilities. (Recall that prices p_0 and bid values b_0 are 0 for the reject good 0, by definition.) Suppose that $i = 0$ or $i \geq 2$. Then $q_i^- = q_i^+$ by construction of $\boldsymbol{q}$, so (a) and (b) imply $b_j - q_j^- < b_j - q_j^+$ in contradiction to $q_j^- \leq q_j^+$, which holds by construction of $\boldsymbol{q}$. Next suppose $i = 1$ and $j \geq 2$. Then $q_j^- = q_j^+ = q_j + \frac{1}{2(n-j+1)}$. The inequalities (a) and (b) imply $b_1 - q_1 - \frac{1}{2n} < b_j - q_j^- < b_1 - q_1 + \frac{1}{2n}$. As $\boldsymbol{b}$ and $\boldsymbol{q}$ are integral, it follows that $b_j - q_j^-$ must lie within $\frac{1}{2n}$ of an integral point, in contradiction to $q_j^- = q_j + \frac{1}{2(n-j+1)}$.

Finally, we show that a bid $\boldsymbol{b}$ that demands good $i = 1$ at q^- and $j = 0$ at q^+ satisfies $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}'$, again using the integrality of $\boldsymbol{b}$ and $\boldsymbol{q}$. Firstly, (a) and (b) imply $q_1 - \frac{1}{2n} < b_1 < q_1 + \frac{1}{2n}$ and it follows that $b_1 = q_1$. Secondly, we have $b_k < q_k^+$ for all $k \geq 1$, as $\boldsymbol{b}$ uniquely demands the reject good 0 at $\boldsymbol{q}^+$, which implies $\boldsymbol{b} \leq \boldsymbol{q}$ by integrality. □

Corollary 5.3. *$\Delta(\boldsymbol{q})$ is the sum of the weights of all bids $\boldsymbol{b} \in \mathcal{B}$ satisfying $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}$.*

5.6.1.1 The algorithm

Algorithm 5.1 learns the vector $\boldsymbol{x}$ and weight of a single positive bid with $O(n \log M)$ queries. It determines the value of x_i, $i \in [n]$, by performing a binary search on line segment $L_i \coloneqq \{(x_1, \ldots, x_{i-1}, z, M, \ldots, M) \mid 0 \leq z \leq M\}$, where the values of $x_1, \ldots, x_{i-1}$ have already been determined and are fixed. As L_i is well-ordered, we can define the 'smallest' and 'largest' points on L_i as $\boldsymbol{s}^i \coloneqq (x_1, \ldots, x_{i-1}, 0, M, \ldots, M)$ and $\boldsymbol{l}^i \coloneqq (x_1, \ldots, x_{i-1}, M \ldots, M)$.

In a first step, Algorithm 5.1 performs binary search on L_1 in order to find the largest point at which demand for good 1 is positive. Note that at any $\boldsymbol{p} \in L_1$, no bid demands items of good $i \geq 2$, i.e. every bid demands an item of good 0 or 1 (or is indifferent between the two). Moreover, the function mapping prices $\boldsymbol{p}$ on

L_1 to the demand of good 1 at $\boldsymbol{p}$ is monotonically decreasing and changes only at integral points, as the bids are integral. As B items of good 1 are demanded at $\boldsymbol{s}^1$, there is a largest price $\boldsymbol{p}^* \in L_1$ at which demand is positive. Hence, we can find $\boldsymbol{p}^*$ using binary search on L_1 by querying demand at $O(\log M)$ prices of the form $(k, M, \ldots, M)$ with $k \in [M]_0$.

The second kind of binary search uses delta queries to find the smallest point $\boldsymbol{p}^*$ on each line segment L_i, $i \geq 2$, at which $\Delta(\boldsymbol{p}^*)$ is positive. Suppose $i \geq 2$. Corollary 5.3 implies that $\Delta(\boldsymbol{q})$ restricted to the line L_i is monotonically increasing. We show in the proof of Theorem 5.5 that the invariant $\Delta(\boldsymbol{l}^i) > 0$ holds when we perform binary search on L_i. Moreover, Δ only changes in value at integral points along L_i, so we can perform binary search to find $\boldsymbol{p}^*$ with $O(\log M)$ delta queries at prices $(x_1, \ldots, x_{i-1}, k, M, \ldots, M)$, where $k \in [M]_0$.

Theorem 5.5 establishes the correctness and running time of Algorithm 5.1. The proof proceeds by induction and makes use of Observation 5.4.

Observation 5.4. *Let $i \geq 2$. Suppose Algorithm 5.1 has successfully determined the first $i-1$ coordinates $x_1, \ldots, x_{i-1}$ and binary search (using delta queries) finds the smallest point $\boldsymbol{p} = (x_1, \ldots, x_i, M, \ldots, M)$ on L_i at which $\Delta(\boldsymbol{p}) > 0$. By Corollary 5.3, (a) none of the bids $\boldsymbol{b} \in \mathcal{B}$ satisfy $\boldsymbol{b} \leq \boldsymbol{p}$ and $b_i < p_i$, and (b) there is at least one bid $\boldsymbol{b}$ with $\boldsymbol{b} \leq \boldsymbol{p}$ and $b_i = p_i$.*

Theorem 5.5. *Algorithm 5.1 returns the vector and weight of a bid using $O(n \log M)$ queries.*

Proof. First we show that the algorithm is well-defined. Clearly, the first coordinate x_1 can be found as outlined above. To show that the binary search on L_i using delta queries to find x_i is well-defined for all $i \geq 2$, it suffices to show that $\Delta(\boldsymbol{l}^i)$ is positive. Fix $i \leq 2$ and suppose the algorithm has found $x_1, \ldots, x_{i-1}$. If $\boldsymbol{p}$ is the point found by binary searching on line L_{i-1}, then Observation 5.4 (b) tells us that there is at least one bid $\boldsymbol{b}$ satisfying $b_1 = p_1$ and $\boldsymbol{b} \leq \boldsymbol{p}$. Moreover, we have $p_1 = \boldsymbol{l}^i_1$ and $\boldsymbol{p} \leq \boldsymbol{l}^i$ by construction. By transitivity and Corollary 5.3, it follows that $\Delta(\boldsymbol{l}^i) > 0$.

Next, we prove that $\boldsymbol{x} = (x_1, \ldots, x_n)$ returned by the algorithm corresponds to the vector of a bid and that $\Delta(\boldsymbol{x})$ is its weight. By Observation 5.4 (b) and construction of x_n, we know that there is a bid $\boldsymbol{b}^*$ that satisfies $\boldsymbol{b}^* \leq \boldsymbol{x}$ and $b^*_n = x_n$. On the other hand, for any $1 \leq i \leq n-1$, the construction of x_i together with Observation 5.4 (a) implies that no bid $\boldsymbol{b}$ satisfies $\boldsymbol{b} \leq \boldsymbol{x}$ and $b_i < x_i$. As a result, we get $b^*_i = x_i$ for all $1 \leq i \leq n$ and it follows that $\boldsymbol{b}^*$ lies at point $\boldsymbol{x}$. Moreover, we see that $\boldsymbol{b}^*$ is the only bid in $\mathcal{B}$ that satisfies $b_1 = x_1$ and $\boldsymbol{b} \leq \boldsymbol{x}$. By Corollary 5.3, $\Delta(\boldsymbol{x})$ is the weight of $\boldsymbol{b}^*$.

Finally, to see that the algorithm has query complexity $O(n \log M)$, note that it performs n binary searches along lines L_i, each of which incurring $O(\log M)$ queries. □

5.7 Learning positive and negative bids

In this section, we provide an algorithm for learning valid bid lists that may contain both positive and negative bids. Introducing negative bids significantly complicates learning the location of bids, as negative bids are able to 'cancel out' facets. Consider, for instance, a bid $\boldsymbol{b}$ of weight 2 at position $(2, 4)$ in the setting with two goods. Without negative bids, its horizontal and vertical facets extend indefinitely from $\boldsymbol{b}$ in direction $\boldsymbol{e}^1$ and $\boldsymbol{e}^2$. In contrast, Fig. 5.1 demonstrates how two negative bids of weight 1 cancel out $\boldsymbol{b}$'s horizontal facet from $(7, 4)$ onwards. Similarly, in the same figure the vertical facet of the bid at $(4, 2)$ is cancelled out by the negative bid at $(4, 4)$. As a result, some bids may only be detectable if we query in their local neighbourhood; for more details see Section 5.8.2, where we exploit this phenomenon to show super-polynomial lower bound on learning positive and negative bids.

Recall from Chapter 1 and Section 5.3 that the demand correspondence of a strong-substitutes demand (and hence of a valid bid list) corresponds to a polyhedral complex over price space, the LIP, where the boundaries between unique demand regions are $n-1$-dimensional facets. Our algorithm learns the collection of all hyperplanes that contain these facets, as well as each vertex arising from the intersection of n such hyperplanes. We note that every bid must lie at a vertex but, conversely, not every such vertex contains a bid. In order to check for existence of a bid at a vertex, we introduce bid existence queries at integral price vectors, which consist of a set of 2^{n-1} demand queries in the local neighbourhood of the vector. In addition, we define super queries which are strictly more powerful and most costly than bid existence queries. Super queries, applied at integral points $\boldsymbol{p}$, provide complete information about the demand correspondence in the local neighbourhood of $\boldsymbol{p}$. This allows us to simulate a bid existence query with a super query and also to perform a principled search for new hyperplanes: at each iteration of the algorithm, either the local information around two vertices points us in the direction of a new hyperplane, or we have succeeded in learning all hyperplanes, and thus all bids.

5.7.1 Bid existence queries

Our algorithm computes a set of integral candidate price vectors at which bids may lie. In order to check for any such candidate vector $\boldsymbol{p} \in \mathbb{Z}^n$ whether there is a bid at this location and, if so, what its weight is, we first introduce a generalisation of the delta queries that were first introduced in Section 5.6.1. This in turn allows us to define the *bid existence query*, which consists of 2^{n-1} generalised delta queries. In total, we see that a total of 2^n queries to the demand oracle is sufficient to determine the existence and weight of a bid at a given integral price vector.

5.7.1.1 Generalising delta queries

Recall that a delta query at $\boldsymbol{q} \in \mathbb{Z}^n$ works by first perturbing the entries $i \geq 2$ of $\boldsymbol{q}$ in a specific way. We now generalise the directions in which these perturbations happen. For any point $\boldsymbol{q} \in \mathbb{Z}^n$ and every $S \subseteq \{2, \ldots, n\}$, we define $\boldsymbol{q}(S)$ by $q(S)_1 := q_1$ and

$q(S)_i := q_i - \frac{1}{2(n-i+1)}$ if $i \in S$ and $q(S)_i := q_i + \frac{1}{2(n-i+1)}$ otherwise for all $i \geq 2$. This allows us to define a *generalised delta query* at $\boldsymbol{q}$ w.r.t. S as follows.

Definition 5.6. Let $\boldsymbol{q} \in \mathbb{Z}^n$ and $S \subseteq \{2, \ldots, n\}$. A (generalised) delta query consists of two queries $\boldsymbol{q}^+(S) := \boldsymbol{q}(S) + \frac{1}{2n}\boldsymbol{e}^1$ and $\boldsymbol{q}^-(S) := \boldsymbol{q}(S) - \frac{1}{2n}\boldsymbol{e}^1$, where we let

$$\boldsymbol{q}(S) := \boldsymbol{q} - \sum_{i \in S} \frac{1}{2(n-i+1)}\boldsymbol{e}^i + \sum_{i \notin S \cup \{1\}} \frac{1}{2(n-i+1)}\boldsymbol{e}^i.$$

The return value of the delta query is defined as $\Delta(\boldsymbol{q}; S) = x_1^- - x_1^+$, where $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ are the bundles of goods uniquely demanded at $\boldsymbol{q}^+$ and $\boldsymbol{q}^-$.

Note that $q^+(S)_1 = q(S)_1 + \frac{1}{2n}$ and $q^-(S)_1 = q(S)_1 - \frac{1}{2n}$, and the two query points $\boldsymbol{q}^+(S)$ and $\boldsymbol{q}^-(S)$ agree on all other coordinates $i \geq 2$. Secondly, $\boldsymbol{q}^\pm$ are non-marginal prices by construction, so any bid $\boldsymbol{b} \in \mathbb{Z}^n$ uniquely demands some good i.

Lemma 5.7 makes the observation that bids satisfying $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}(S)$ demand good 1 at $\boldsymbol{q}^-$ and are rejected at $\boldsymbol{q}^+$, while all other bids demand the same good at both prices $\boldsymbol{q}^\pm$. Hence demand changes only in terms of good 1, and $\Delta(\boldsymbol{q}; S)$ captures this change. Another useful interpretation of $\Delta(\boldsymbol{q}; S)$ is that it returns the sum of the weights of all bids that satisfy $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}(S)$.

Lemma 5.7. *Suppose we make a delta query at $\boldsymbol{q}(S) \in \mathbb{Z}^n$ w.r.t. $S \subseteq \{2, \ldots, n\}$. Then any bid $\boldsymbol{b} \in \mathbb{Z}^n$ demands different goods at $\boldsymbol{q}^-$ and $\boldsymbol{q}^+$ if, and only if, $b_1 = q_1$ and $\boldsymbol{b} \leq \boldsymbol{q}(S)$. Moreover, any such bid demands an item of good 1 at $\boldsymbol{q}^-$ and an item of reject good 0 at $\boldsymbol{q}^+$.*

Proof. Analogous to the proof of Lemma 5.2. □

5.7.1.2 Defining bid existence queries

In order to check for the existence (and weight) of a bid at a given price vector $\boldsymbol{p} \in \mathbb{Z}^n$, we introduce the existence query consisting of 2^{n-1} generalised delta queries. Indeed, an existence query performed at $\boldsymbol{p} \in \mathbb{Z}^n$ consists of the set of delta queries $\{\Delta(\boldsymbol{p}; S) \mid S \subseteq \{2, \ldots, n\}\}$. We now show how the existence and weight of a bid at $\boldsymbol{p}$ can be inferred from the return values of these 2^{n-1} delta queries. For convenience, we introduce the notation $w(\mathcal{C}) = \sum_{\boldsymbol{b} \in \mathcal{C}} w(\boldsymbol{b})$ for any list of bids $\mathcal{C}$. Our approach is based on the following straightforward observation.

Observation 5.8. *For any bid list $\mathcal{B}$, define $\mathcal{B}' = \{\boldsymbol{b} \in \mathcal{B} \mid b_1 = p_1, \boldsymbol{b} \leq \boldsymbol{p}\}$ and $\mathcal{B}'' = \{\boldsymbol{b} \in \mathcal{B} \mid b_1 = p_1, \boldsymbol{b} \leq \boldsymbol{p}, \boldsymbol{b} \neq \boldsymbol{p}\}$. There exists a bid at $\boldsymbol{p}$ in $\mathcal{B}$ if, and only if, the term $w := w(\mathcal{B}') - w(\mathcal{B}'')$ is non-zero. Moreover, if the bid exists, then its weight is w.*

Note that $\mathcal{B}'$ contains all bids dominated by $\boldsymbol{p}$ and the delta query $\Delta(\boldsymbol{p}, \emptyset)$ returns the value $w(\mathcal{B}')$. Moreover, the set $\mathcal{B}''$ contains all bids that are dominated by $\boldsymbol{p}$ and do not lie at $\boldsymbol{p}$. In order to determine $w(\mathcal{B}'')$, we first partition $\mathcal{B}''$ into $2^{n-1} - 1$ sub-lists as follows. For each non-empty set of goods $T \subseteq \{2, \ldots, n\}$, let $\mathcal{B}_T := \{\boldsymbol{b} \in \mathcal{B} \mid b_i < p_i, \forall i \in T \text{ and } b_j = p_j, \forall j \notin T\}$. Lemma 5.9 now provides us with a method of computing $w(\mathcal{B}'')$ using the $2^{n-1} - 1$ delta queries $\Delta(\boldsymbol{p}, S)$ with $\emptyset \subsetneq S \subseteq \{2, \ldots, n\}$.

Lemma 5.9. *We have*

$$w(\mathcal{B}'') = \sum_{\emptyset \subsetneq T \subseteq \{2,\dots,n\}} w(\mathcal{B}_T) = \sum_{\emptyset \subsetneq S \subseteq \{2,\dots,n\}} (-1)^{|S|+1} \Delta(\boldsymbol{p}; S).$$

Proof. Firstly, observe how the result of the delta query $\Delta(\boldsymbol{p}; S)$ for a given $S \subseteq \{2, \dots, n\}$ satisfies $\Delta(\boldsymbol{p}; S) = \sum_{T \supseteq S} w(\mathcal{B}_T)$. Indeed, recall that $\Delta(S)$ sums the weights of the bids $\boldsymbol{b} \in \mathcal{B}$ satisfying $b_1 = p_1$ and $\boldsymbol{b} \leq \boldsymbol{p}(S)$. As we have $p(S)_i < p_i$ for all $i \in S$ and the bids are integral, the set of bids that $\Delta(\boldsymbol{p}; S)$ considers is identical to $\bigcup_{T \supseteq S} \mathcal{B}_T$. Hence

$$\begin{aligned}
\sum_{\emptyset \subsetneq S \subseteq \{2,\dots,n\}} (-1)^{|S|+1} \Delta(\boldsymbol{p}; S) &= \sum_{\emptyset \subsetneq S \subseteq \{2,\dots,n\}} (-1)^{|S|+1} \sum_{T \supseteq S} w(\mathcal{B}_T) \\
&= \sum_{\emptyset \subsetneq T \subseteq \{2,\dots,n\}} w(\mathcal{B}_T) \sum_{S \subseteq T} (-1)^{|S|+1} \\
&= \sum_{\emptyset \subsetneq T \subseteq \{2,\dots,n\}} w(\mathcal{B}_T).
\end{aligned}$$

The last equality follows from the binomial identity in Lemma 5.10 below. □

Lemma 5.10. *For any $n \geq 1$ we have $\sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} = 1$.*

Proof. Recalling Pascal's Rule, $\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}$, we have

$$\begin{aligned}
\sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} &= \sum_{i=1}^{n} (-1)^{i+1} \binom{n-1}{i} + \sum_{i=1}^{n} (-1)^{i+1} \binom{n-1}{i-1} \\
&= \sum_{i=1}^{n-1} (-1)^{i+1} \binom{n-1}{i} + \sum_{i=1}^{n-1} (-1)^{i} \binom{n-1}{i} + (-1)^2 \binom{n-1}{0} = 1. \square
\end{aligned}$$

5.7.2 Super queries

Suppose $\boldsymbol{p} \in \mathbb{Z}^n$ is an integral price vector. We show that it is possible to obtain complete knowledge of $D_{\mathcal{B}}(\boldsymbol{p}')$ for all prices $\boldsymbol{p}'$ with $\|\boldsymbol{p} - \boldsymbol{p}'\|_\infty < 1$ using a *super query*, which consists of a specific set of demand queries at non-marginal query points in the vicinity of $\boldsymbol{p}$. Intuitively, this works because bid vectors are integral and facets of an SS LIP can only have specific orientations. Super queries are used by our algorithm in two ways: firstly to determine the existence and weight of a bid at a given integral point $\boldsymbol{p}$, and secondly to provide information that leads to a new separating hyperplane.

Let $U_1(\boldsymbol{p}), \dots, U_{2^n}(\boldsymbol{p})$ denote the 2^n orthants of the unit L_∞-ball around $\boldsymbol{p}$. Each orthant is a hypercube that can be triangulated into $n!$ simplices (one for each permutation of the coordinates $[n]$), as described in Section 5.2.1. We denote these simplices for the i-th orthant by $U_i^1(\boldsymbol{p}), \dots, U_i^{n!}(\boldsymbol{p})$.

Definition 5.11. A *super query* at $\boldsymbol{p} \in \mathbb{Z}^n$ is a collection $\mathrm{SQ}(\boldsymbol{p})$ of representative prices from the interior of each $U_i^j(\boldsymbol{p})$, where $i \in [2^n]$ and $j \in [n!]$.

With a slight abuse of notation, we say that we 'super query' a price vector $\boldsymbol{p}$ if we query all price vectors in $SQ(\boldsymbol{p})$. Figure 5.3 illustrates super queries in the case of two goods ($n = 2$). The following lemma demonstrates the use of super queries.

Lemma 5.12. *Querying the points in* $\mathrm{SQ}(\boldsymbol{p})$ *once is sufficient to ascertain* $D_{\mathcal{B}}(\boldsymbol{p}')$ *for any* $\boldsymbol{p}'$ *with* $\|\boldsymbol{p} - \boldsymbol{p}'\|_\infty < 1$. *With this information, we can learn all facets of the LIP containing* $\boldsymbol{p}$ *and establish the existence and weight of a bid at* $\boldsymbol{p}$.

Proof. Suppose that $\boldsymbol{p}'$ is such that $\|\boldsymbol{p} - \boldsymbol{p}'\|_\infty < 1$. From construction, it follows that $\boldsymbol{p}' \in U_i^j(\boldsymbol{p})$ for some $U_i^j(\boldsymbol{p})$. However, we also know that facets in the polyhedral complex resulting from $D_{\mathcal{B}}$ can only be normal to a unit vector ($\boldsymbol{e}^i$ for some i), or a vector of the form $\boldsymbol{e}^i - \boldsymbol{e}^j$ for some $i, j \in [n]$. As a consequence, any price in the interior of $U_i^j(\boldsymbol{p})$ demands the same unique bundle. If $\boldsymbol{p}'$ also lies in the interior of $U_i^j(\boldsymbol{p})$, then we know the bundle it demands and we are done. On the other hand, if $\boldsymbol{p}'$ is on the boundary $U_i^j(\boldsymbol{p})$, then it could be marginal, but if this is the case, from construction, we know all bundles demanded at non-marginal prices neighbouring $\boldsymbol{p}'$, as they must lie in the interior of other $U_{i'}^{j'}(\boldsymbol{p})$. This in turn implies that we can infer all bundles demanded at $\boldsymbol{p}'$ as desired. Moreover, it follows immediately that $SQ(\boldsymbol{p})$ holds all information regarding facets containing $\boldsymbol{p}$, as facets can only take specific orientations in a SS demand correspondence. Finally, as have complete information about demand in the unit ball around $\boldsymbol{p}$, we can simulate a bid existence query as defined in Section 5.7.1 at $\boldsymbol{p}$ without making any further queries to the demand oracle. This tells us whether there exists a bid at $\boldsymbol{p}$ and, if so, what its weight is.

5.7.3 Finding a separating hyperplane

Suppose $\mathbf{0} \leq \boldsymbol{q}, \boldsymbol{q}' \leq (M+1)\boldsymbol{e}^{[n]}$ are distinct price vectors that lie in the interiors of different demand regions. Note that $\|\boldsymbol{q} - \boldsymbol{q}'\|_\infty \leq M + 1$. As demand regions are convex and have piecewise-linear boundaries, there exists some facet F of the LIP separating $\boldsymbol{q}$ and $\boldsymbol{q}'$. In order to find the hyperplane containing F, we first perform $O(\log M)$ steps of binary search on $\mathrm{conv}(\boldsymbol{q}, \boldsymbol{q}')$ to obtain a pair of points $\boldsymbol{s}$ and $\boldsymbol{s}'$ on either side of F with $\|\boldsymbol{s} - \boldsymbol{s}'\|_\infty \leq 1/4$. By the geometry of SS demand, we know that F must be normal to a vector $\boldsymbol{e}^i - \boldsymbol{e}^j$, where $i, j \in [n]_0$ and we denote $\boldsymbol{e}^0 = 0$ for convenience.

Suppose we know that F is normal to $\boldsymbol{e}^i - \boldsymbol{e}^j$ for some fixed i and j. Then we can determine the point $\boldsymbol{p} = \lambda\boldsymbol{s} + (1-\lambda)\boldsymbol{s}'$ $(0 \leq \lambda \leq 1)$ at which the line segment $\overline{\boldsymbol{s}\boldsymbol{s}'}$ intersects F, as this is obtained for the unique value $0 \leq \lambda \leq 1$ such that $\boldsymbol{p}$ satisfies $p_i - p_j \in \mathbb{Z}$. Next, we round up $\boldsymbol{p}$ to the nearest integer and claim that $\lceil \boldsymbol{p} \rceil$ also lies in F. Indeed, we see that if a bid $\boldsymbol{b}$ demands good i at $\boldsymbol{p}$, then it also demands i at $\lceil \boldsymbol{p} \rceil$:

$$b_j - \lceil p_j \rceil - 1 \leq b_j - p_j - 1 \leq b_i - p_i - 1 < b_i - \lceil p_i \rceil.$$

Here the second inequality follows from the fact that $\boldsymbol{b}$ demands good i at $\boldsymbol{p}$. As $\boldsymbol{b}$ and $\lceil \boldsymbol{p} \rceil$ are integral, this chain of inequalities implies $b_i - \lceil p_i \rceil \geq b_j - \lceil p_j \rceil$. Hence, as this

Algorithm 5.2 Finding a separating hyperplane

Input: Two hyperplane witnesses $\boldsymbol{q}$ and $\boldsymbol{q}'$ that are separated by a facet.
Output: Hyperplane containing this facet.

1: Binary search between $\boldsymbol{q}$ and $\boldsymbol{q}'$ to find a pair of points $\boldsymbol{s}, \boldsymbol{s}'$ with distance $\|\boldsymbol{s} - \boldsymbol{s}'\|_\infty < 1/4$ and distinct demands.
2: **for all** $(i, j) \in [n]_0^2$ **do**
3: Suppose $\boldsymbol{s}$ and $\boldsymbol{s}'$ are separated by a facet F normal to $\boldsymbol{e}^i - \boldsymbol{e}^j$.
4: Find unique intersection point $\boldsymbol{p}$ of F with the line segment from $\boldsymbol{s}$ to $\boldsymbol{s}'$.
5: Super query at $\lceil \boldsymbol{p} \rceil$.
6: **if** the super query learns a facet containing $\lceil \boldsymbol{p} \rceil$ **then**
7: **return** the hyperplane containing this facet.
8: **end if**
9: **end for**

holds for every bid, we see that $\boldsymbol{p}$ and $\lceil \boldsymbol{p} \rceil$ lie in the same facet. We perform a super query at $\lceil \boldsymbol{p} \rceil$ in order to find F (as well any other adjacent facets that $\boldsymbol{p}$ might lie in).

This suggests the following algorithm for finding a separating hyperplane, specified in Algorithm 5.2. First we perform binary search between $\boldsymbol{q}$ and $\boldsymbol{q}'$ in order to find points $\boldsymbol{s}$ and $\boldsymbol{s}'$ that are $1/4$ apart (w.r.t. the L_∞-norm). Then, for every possible pair of goods $(i, j) \in [n]_0^2$, we suppose that $\boldsymbol{s}$ and $\boldsymbol{s}'$ are separated by a facet that is normal to $\boldsymbol{e}^i - \boldsymbol{e}^j$ and perform a super query at prices $\lceil \boldsymbol{p} \rceil$ obtained as described above. As soon as such a super query finds a facet containing $\lceil \boldsymbol{p} \rceil$, we return the hyperplane containing this facet. Overall, finding a separating hyperplane between $\boldsymbol{q}$ and $\boldsymbol{q}'$ in this way costs $O(\log M + n^2 2^n n!)$ queries, as each super query costs $O(2^n n!)$. Moreover, our procedure is guaranteed to find a separating hyperplane, as it exhaustively checks all possible orientations for the separating facet F between $\boldsymbol{s}$ and $\boldsymbol{s}'$.

5.7.4 The main algorithm

Algorithm 5.3 learns bid lists that may comprise positive and negative bids. The algorithm maintains a set of hyperplanes $\mathcal{H}$ that it has learnt. We initialise $\mathcal{H}$ with the axis-aligned hyperplanes $\boldsymbol{e}^i = 0$ and $\boldsymbol{e}^i = M$ for all $i \in [n]$. The algorithm also keeps track of the corresponding set of vertices $\mathcal{V}$ arising from intersections of hyperplanes in $\mathcal{H}$, the set of query points $\mathcal{Q} = \bigcup_{\boldsymbol{v} \in \mathcal{V}} \mathrm{SQ}(\boldsymbol{v})$, as well as the set of polytopes $\mathcal{P}$ of the subdivision of $\mathbb{R}^n$ by the hyperplanes in $\mathcal{H}$. Finally, $\hat{\mathcal{B}}$ denotes the set of bids that the algorithm has learnt. Two query points $\boldsymbol{q}, \boldsymbol{q}' \in \mathcal{Q}$ are *hyperplane witnesses* if they lie in the same polytope $P \in \mathcal{P}$ but have different demand. Figure 5.3 shows a snapshot of Algorithm 5.3 after learning a single hyperplane.

We now argue that Algorithm 5.3 is well-defined and learns a bid list in $O(Bn^2)$ iterations by identifying all the hyperplanes containing a facet of the LIP. As each bid gives rise to at most $O(n^2)$ facets, the total number of such hyperplanes is $O(Bn^2)$. The algorithm learns a new hyperplane in each iteration, as hyperplane witnesses lie in the same polytope by definition, which implies that Step 6 of the algorithm finds a new hyperplane that is not in $\mathcal{H}$. Moreover, every bid must lie

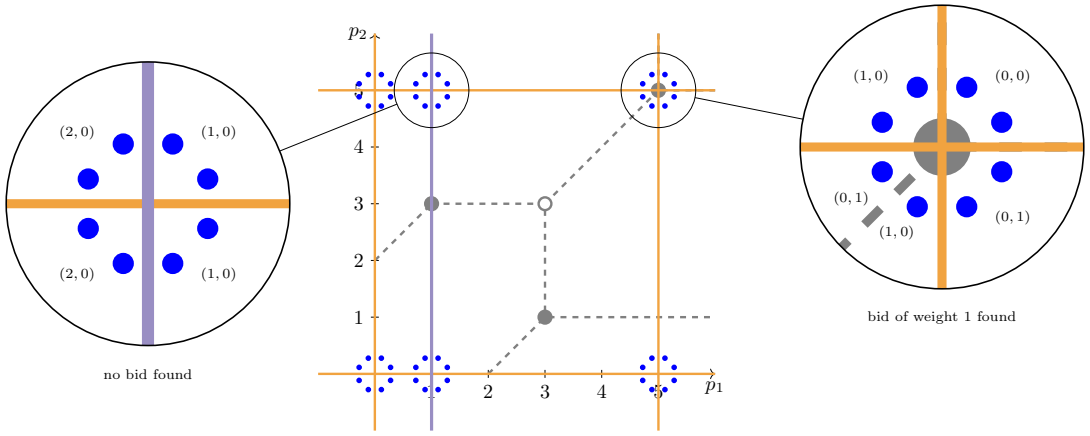


Figure 5.3: A snapshot of Algorithm 5.3 learning the bid list consisting of one negative bid $(3, 3, -1)$ and three positive bids $(1, 3; 1), (3, 1; 1)$ and $(5, 5; 1)$. The initial axis-aligned hyperplanes are drawn in orange. Moreover, the algorithm has learnt one additional vertical hyperplane $(x = 1)$ drawn in purple. The super queries made at the hyperplane intersections are represented by blue query points. Two super queries are highlighted to illustrate how the existence of a bid is determined.

Algorithm 5.3 Learning positive and negative bids

Initialisation:

Let $\mathcal{H}$ be hyperplanes of the form $\boldsymbol{e}^i \cdot \boldsymbol{x} = 0$ and $\boldsymbol{e}^i \cdot \boldsymbol{x} = M$ for all $i \in [n]$.
Update $\mathcal{V}, \mathcal{Q}$ and $\mathcal{P}$ according to $\mathcal{H}$.
Initialise $\hat{\mathcal{B}}$ as an empty list of bids.
Superquery at each $\boldsymbol{v} \in \mathcal{V}$ to check for bid and add each new-found bid to $\hat{\mathcal{B}}$.

Main loop:

while $\exists$ hyperplane witnesses $\boldsymbol{q}, \boldsymbol{q}' \in \mathcal{Q}$ **do**
    Use Algorithm 5.2 to find a hyperplane separating $\boldsymbol{q}$ and $\boldsymbol{q}'$ and add it to $\mathcal{H}$.
    Update $\mathcal{V}, \mathcal{Q}$ and $\mathcal{P}$ accordingly.
    for all new vertices $\boldsymbol{v}$ in $\mathcal{V}$ **do**
        Superquery at $\boldsymbol{v}$ to check for bid at $\boldsymbol{v}$ and add a newly found bid to $\hat{\mathcal{B}}$.
    end for
end while
return $\hat{\mathcal{B}}$

at the intersection of n such hyperplanes, and we perform a super query at each intersection to check for the existence of a bid at that point. All that remains is to show that Algorithm 5.3 does not terminate until all $O(Bn^2)$ hyperplanes containing facets of the LIP have been learnt, as this immediately implies that the algorithm identifies the locations and weights of all bids.

Lemma 5.13. *Algorithm 5.3 learns all $O(Bn^2)$ hyperplanes containing a facet of the LIP.*

Proof. It suffices to show that there always exists a pair of hyperplane witnesses if we have not learnt all hyperplanes. Suppose F is a facet of the LIP that is not

contained in any hyperplane in $\mathcal{H}$. Then F separates two neighbouring demand regions. Moreover, by assumption there is a polytope $P \in \mathcal{P}$ that intersects both these regions, hence there exist two non-marginal points $\boldsymbol{p}, \boldsymbol{p}' \in P$ at which demand differs.

Next, recall that we perform a super query at every vertex of the subdivision of $\mathbb{R}^n$ by the hyperplanes in $\mathcal{H}$. Hence, for every polytope $P \in \mathcal{P}$ we have a query point close to its vertices. Suppose all these query points in P demand the same bundle $\boldsymbol{x}$. This implies that $\boldsymbol{x}$ is also demanded at all vertices of P. By convexity of demand, $\boldsymbol{x}$ is then uniquely demanded at *any* non-marginal point of the polytope. But this contradicts the fact that demand differs at $\boldsymbol{p}$ and $\boldsymbol{p}'$. It follows that at least two of the query points close the vertices of P have distinct demand. □

Theorem 5.14 gives the overall query complexity of learning a bid list with Algorithm 5.3.

Theorem 5.14. *Algorithm 5.3 requires* $O\left(Bn^2 \log M + 2^n n! \binom{Bn^2}{n}\right)$ *queries to learn a bid list that may consist of positive and negative bids. For* n *constant, this is* $O\left(B \log M + B^n\right)$.

Proof. Let H, V and Q be the number of hyperplanes, vertices and query points in the sets $\mathcal{H}, \mathcal{V}$ and $\mathcal{Q}$ at the conclusion of Algorithm 5.3. Note that Algorithm 5.3 only makes demand queries in Steps 4, 6 and 9. We first count the number of queries that arise from checking for existence of a bid at vertex locations (Steps 4 and 9). $\mathcal{H}$ is initialised with $2n$ axis-aligned hyperplanes and learns an additional $O(Bn^2)$ hyperplanes (cf. Lemma 5.13). Hence $H = O(Bn^2)$. As every vertex in $\mathcal{V}$ arises as the intersection of n distinct hyperplanes, we have $V = O\left(\binom{Bn^2}{n}\right)$. The algorithm performs a super query of cost $2^n n!$ at each vertex in $\mathcal{V}$, leading to a total cost of $O\left(2^n n! \binom{Bn^2}{n}\right)$. Next we count the number of queries required to find new hyperplanes (Step 6). Section 5.7.3 tells us that finding a single hyperplane costs $O(\log M + n^2 2^n n!)$ queries. As the algorithm learns $O(Bn^2)$ hyperplanes, the aggregate number of queries performed by Step 6 is $O\left(Bn^2 \left(\log M + n^2 2^n n!\right)\right)$. Summing these two cost terms for super queries and hyperplane finding gives the desired query complexity for Algorithm 5.3. □

5.8 Lower bounds

We describe lower bounds for the complexity of learning bid lists. First we show that learning B positive bids requires $\Omega(B \log M)$ queries. Our other lower bounds apply in the setting where the bid list may comprise positive and negative bids and make use of a carefully constructed 'island gadget' that consists of 2^n positive and 2^n negative bids of unit weight. We construct an adversarial game that an adversary can play to force a learner to identify the location of the island gadget. For bid lists $\mathcal{B}$ that need not be valid, the island gadget construction immediately implies that $\Omega\left(\left(\frac{M}{4}\right)^n\right)$ queries are required to learn $\mathcal{B}$ (roughly, price-space has to be queried exhaustively). For the case with valid bid lists, the adversarial game

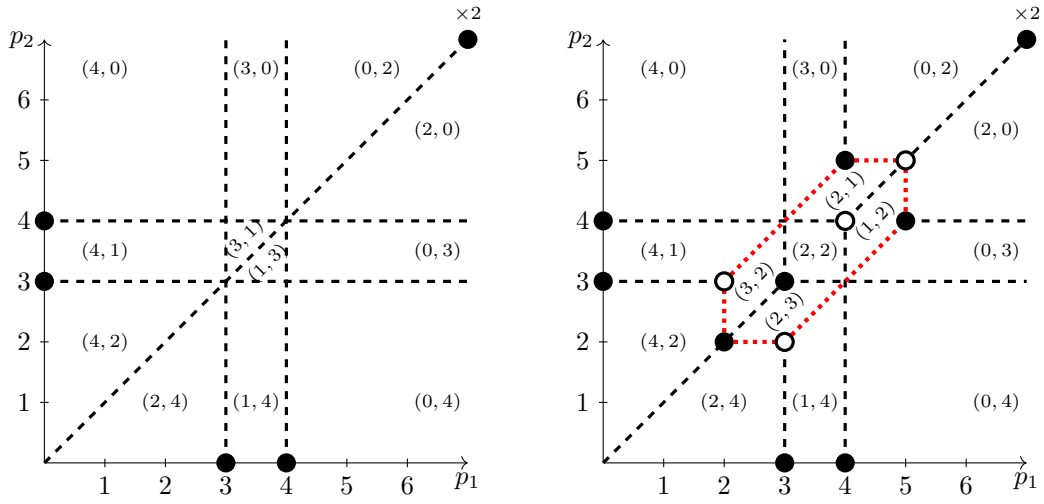


Figure 5.4: A demand correspondence with (right) and without (left) an *island gadget* placed at $(2, 2)$. The gadget consists of the eight bids contained in the convex region denoted by the red dotted lines. Note that demand differs only at prices *inside* inside this region.

implies a lower bound of $\Omega\left(((B - 2^{n+1})/8n^2)^n\right)$ on the query complexity. We see that n must be held constant for the query complexity to be polynomial. In this regime, our lower bounds and the upper bounds from Section 5.7 imply a query complexity of $\Theta\left(B \log M + B^n\right)$ for constant n.

Theorem 5.15. *Any algorithm for learning bid lists requires* $\Omega(B \log M)$ *queries.*

Proof. Suppose an adversary places B positive bids at B integral points in the space $\{(x_1, 0, \ldots, 0) \mid x_1 \in [M]_0\}$. By a standard decision tree argument, any algorithm must make $\Omega(B \log M)$ queries to learn the location of the bids. □

5.8.1 The island gadget

We now introduce the *island gadget*, which allows us to locally change the demand correspondence without affecting demand outside the convex hull of the gadget bids. This is illustrated in Fig. 5.4. Lemma 5.17 establishes that the gadget only influences demand locally. Let $\rho(\boldsymbol{x})$ denote the weight function that assigns positive weight 1 to bids with an even number of odd entries and negative weight -1 otherwise.

Definition 5.16. The gadget G at position $\mathbf{0}$ consists of the following 2^{n+1} positive and negative unit bids that sit on the vertices of two unit hyper-cubes. The bids on the first hyper-cube lie at $\boldsymbol{b} \in \{0, 1\}^n$ and have weight $\rho(\boldsymbol{b})$. The bids on the second hyper-cube lie at $\boldsymbol{b} \in \{2, 3\}^n$ and have weight $-\rho(\boldsymbol{b})$. In order to place G at position $\boldsymbol{x}$, we add $\boldsymbol{x}$ to the position of each bid (without changing the weights).

Lemma 5.17. *The bids of gadget* G *placed at position* $\boldsymbol{x}$ *demand nothing at prices* $\boldsymbol{p} \notin \boldsymbol{x} + [3]_0^n$.

The proof of Lemma 5.17 makes use of the following technical result.

Lemma 5.18. *Let $\boldsymbol{c} \in \mathbb{R}^n$ be a constraint vector. If $c_i \geq 1$ for some $i \in [n]$, then the number of vectors with an even number of 1 entries in $\{\boldsymbol{x} \in \{0,1\}^n \mid \boldsymbol{x} \leq \boldsymbol{c}\}$ is equal to the number of vectors with an odd number of 1 entries.*

Proof. The proof proceeds by induction on n. The case $n = 1$ is immediate. For the inductive step, suppose, without loss of generality, that $c_1 \geq 1$. By the induction hypothesis, the set $X^{n-1} := \{\boldsymbol{x} \in \{0,1\}^{n-1} \mid x_i \leq c_i, \forall i \in [n-1]\}$ has an equal number of vectors with even and odd numbers of 1 entries. If $c_n = 0$, then the set $X^n := \{\boldsymbol{x} \in \{0,1\}^n \mid \boldsymbol{x} \leq \boldsymbol{c}\}$ can be constructed by appending an n-th entry 0 to each vector in X^{n-1}, so the number of entries with even and odd 1 entries remains unchanged. If $c_n \geq 1$, X^n consists of the set of vectors obtained by appending n-th entries 0 or 1 to each vector in X^{n-1}, so the number of odd and even vectors doubles. In both cases, the claim holds. □

Proof of Lemma 5.17. Recall that gadget G at position $\boldsymbol{x}$ consists of the bids $\boldsymbol{b} \in \boldsymbol{x} + \{0,1\}^n$ of weight $\rho(\boldsymbol{b} - \boldsymbol{x})$ and the bids $\boldsymbol{b} \in \boldsymbol{x} + \{2,3\}^n$ of weight $-\rho(\boldsymbol{b} - \boldsymbol{x})$. In order to prove that the bids of G do not aggregately demand items of any goods at prices $\boldsymbol{p} \notin \boldsymbol{x} + [3]_0^n$, we show for every good i and every such price that the number of negative and positive gadget bids demanding i at $\boldsymbol{p}$ is the same.

Fix a good $i \in [n]$ and prices $\boldsymbol{p} \notin \boldsymbol{x} + [0,3]^n$. Without loss of generality, we assume that $\boldsymbol{p}$ is non-marginal and $p_i \neq p_j$ for all $i, j \in [n]$. Recall that an integral bid $\boldsymbol{b}$ uniquely demands good i at $\boldsymbol{p}$ if, and only if, we have $b_i > p_i$ and $b_i - p_i > b_j - p_j$ for all $j \in [n] \setminus \{i\}$. The last condition can be rewritten as $b_i - b_j > p_i - p_j$. Hence we can express the set of bids in the bottom and top cube that (uniquely) demand a positive or negative item of good i as

$$\{\boldsymbol{b} \in \boldsymbol{x} + \{0,1\}^n \mid b_i > p_i \text{ and } b_i - b_j > p_i - p_j,\ \forall j \in [n] \setminus \{i\}\} \tag{5.1}$$

$$\text{and } \{\boldsymbol{b} \in \boldsymbol{x} + \{2,3\}^n \mid b_i > p_i \text{ and } b_i - b_j > p_i - p_j,\ \forall j \in [n] \setminus \{i\}\}, \tag{5.2}$$

respectively. We show by case distinction on the possible value of p_i that the number of positive and negative bids in (5.1) and (5.2) is equal.

Case I: Suppose first that $p_i \notin \boldsymbol{x} + [3]_0^n$. If $p_i > x_i + 3$, the two sets (5.1) and (5.2) are empty and we are done. Hence assume that $p_i < x_i$. Then we can express (5.1) and (5.2) as

$$\boldsymbol{x} + \{\boldsymbol{v} \in \{0,1\}^n \mid v_i - v_j > p_i - p_j,\ \forall j \in [n] \setminus \{i\}\}$$
$$\text{and } \boldsymbol{x} + 2\boldsymbol{e}^{[n]} + \{\boldsymbol{v} \in \{0,1\}^n \mid v_i - v_j > p_i - p_j,\ \forall j \in [n] \setminus \{i\}\}.$$

We see that there is a one-to-one correspondence $\boldsymbol{b} \to \boldsymbol{b} + 2\boldsymbol{e}^{[n]}$ between bids in (5.1) and (5.2) that preserves the number of odd entries in the bid vectors. A bid $\boldsymbol{b}$ in the bottom cube is positive if, and only if, $\boldsymbol{b} - \boldsymbol{x}$ has an even number of odd entries and the reverse is true for the top cube. Hence the total number of positive and negative bids in (5.1) and (5.2) is equal.

Case II: Suppose that $p_i \in [x_i, x_i + 3]$. Then there exists a good $k \in [n] \setminus \{i\}$ with $p_k < x_k$ or $p_k > x_k + 3$. We proceed by further case distinction. Note that $b_i - b_j \in \{-1, 0, 1\}$ for all gadget bids $\boldsymbol{b}$.

Case II.1: Suppose $p_i - p_j \geq 1$ for some $j \in [n] \setminus \{i\}$. Then the sets (5.1) and (5.2) are empty, as the constraint $b_i - b_j > p_i - p_j$ is violated.

Case II.2: Now suppose $p_i - p_j < 1$ for every $j \in [n] \setminus \{i\}$ and we have a non-empty set K of indices k for which $0 < p_i - p_k < 1$. Then for all bids $\boldsymbol{b}$ in (5.1) and (5.2) we have $b_i - b_k = 1$ for all $k \in K$. Hence b_i and b_k can only take one value.

If $K = [n] \setminus \{i\}$, this implies that (5.1) and (5.2) both contain a single bid of odd parity and we are done. If $K \subset [n] \setminus \{i\}$, then for $j \notin K$ we have $p_i - p_j < 0$ and the constraint $b_i - b_j > p_i - p_j$ holds vacuously, so b_j can take any value. We apply Lemma 5.18 to see that (5.1) and (5.2) each contain an equal number of positive and negative bids.

Case II.3: Suppose $p_j > p_i$ for all $j \in [n] \setminus \{i\}$. Then there exists a good $k \in [n] \setminus \{i\}$ such that $p_k > x + 3$. We consider the two subsets $\{b \in (5.1) \mid b_i = x_i\}$ and $\{b \in (5.1) \mid b_i = x_i + 1\}$ of (5.1) separately. If $\boldsymbol{b}$ is a bid in the first subset, then $b_i = x_i$ and $b_i > p_i$ imply that $p_i - p_k < -1$, so there is no constraint on the value of b_k. If $\boldsymbol{b}$ is a bid in the second subset, then there is no constraint on the value of b_k. Hence in both cases each subset is either empty or contains an equal number of positive and negative bids by Lemma 5.18. □

5.8.2 A lower bound for positive and negative bids

From now on, we assume that the unknown bid list may comprise positive and negative bids. We primarily consider the setting where the bid list is assumed to be valid and specify an 'adversarial game' where the adversary can force the player to make $\Omega\left(\left(\frac{B-2^{n+1}}{8n^2}\right)^n\right)$ queries. Note that this yields a query complexity of $\Omega(B^n)$ if n is constant. A similar but simpler adversarial game is then applied to obtain a lower bound for the setting where the bid list may be invalid. The resulting lower bounds for bid lists comprising positive and negative bids are stated in Theorem 5.22.

Fix a parameter $k \in \mathbb{N}$ and let $M = 4k$. The adversary positions the island gadget consisting of 2^{n+1} bids at exactly one of k^n possible points $\boldsymbol{x}$ of the lattice $4[k-1]_0^n$. He also places *boundary bids* of weight 1 on the boundary of the M-cube as follows. For every good $i \in [n]$, there are M positive bids at all points $m\boldsymbol{e}^i$ with $m \in [M-1]_0$, and for every pair of goods $(i,j) \in [n]^2$ with $i \neq j$, there are M positive bids at all points $m\boldsymbol{e}^i + M\boldsymbol{e}^j$ with $m \in [M-1]_0$. Hence, in total, the adversary creates a bid list with $B = 2^{n+1} + 4k\left(n + \binom{n}{2}\right)$ positive and negative bids. The player wishes to identify where the adversary placed the gadget. Lemma 5.17 shows that placing the gadget at $\boldsymbol{x}$ only influences demand at prices inside the cube $\boldsymbol{x} + [3]_0^n$. Hence the player must make queries inside at least $k^n - 1$ cubes to determine where the gadget was placed. Lemma 5.19 shows that the bid list created by the adversary is valid. This leads to the lower bounds stated in Theorem 5.22.

Lemma 5.19. *The bids placed by the adversary are valid.*

Recall that $\mathcal{B}^G$ denotes the bids of the island gadget. In order to prove Lemma 5.19, we define the subset $\mathcal{B}_i^G$ of $\mathcal{B}^G$ that contains all bids indifferent between a good i and the reject good at some fixed price vector $\boldsymbol{p}$. The subset of

bids indifferent between two non-reject goods i and j is defined similarly, albeit separately for the 'lower' and the 'upper' cube of the gadget. Specifically, we define

$$\mathcal{B}_i^G \coloneqq \{\boldsymbol{b} \in \{0,1\}^n \cup \{2,3\}^n \mid b_i = p_i \text{ and } b_k \leq p_k,\ \forall k \in [n]\} \tag{5.3}$$

as well as

$$\mathcal{B}_{ij}^L \coloneqq \{\boldsymbol{b} \in \{0,1\}^n \mid b_i \geq p_i, b_j = b_i - p_i + p_j \text{ and } b_k \leq b_i - p_i + p_k\ \forall k \notin \{i,j\}\} \tag{5.4}$$

$$\mathcal{B}_{ij}^U \coloneqq \{\boldsymbol{b} \in \{2,3\}^n \mid b_i \geq p_i, b_j = b_i - p_i + p_j \text{ and } b_k \leq b_i - p_i + p_k\ \forall k \notin \{i,j\}\}. \tag{5.5}$$

In order to prove Lemma 5.19, we first establish the two technical lemmas Lemmas 5.20 and 5.21.

Lemma 5.20. *Let $n \geq 3$ and $\boldsymbol{p} \in \{0,1\}^n \cup \{2,3\}^n$. The sum of the weights of the gadget bids indifferent between i and the reject good 0 $\boldsymbol{p}$ is -1, 0 or 1.*

Proof. Suppose first that $\boldsymbol{p} \in \{0,1\}^n$. We show that the sum of the weights of bids in $\mathcal{B}_i^G$ is 0 or 1. Recalling the definition of $\mathcal{B}_i^G$ from (5.3), note that if $p_k = 0$ for all $k \neq i$, then $\boldsymbol{b} = \boldsymbol{0}$ is the only bid in this set and its weight is positive. If we have $p_k = 1$ for some $k \neq i$, then the set contains an equal number of positive and negative bids by Lemma 5.18. Similarly, we can show that the sum of the weights of bids in $\mathcal{B}_i^G$ is -1 or 0 when $\boldsymbol{p} \in \{2,3\}^n$. □

Lemma 5.21. *Let $n \geq 3$ and $\boldsymbol{p} \in \{0,1\}^n \cup \{2,3\}^n$. If $\mathcal{B}_{ij}^L$ has more than one bid, the sum of the weights of the bids indifferent between i and j is 0 or 1. Similarly, if $\mathcal{B}_{ij}^U$ has more than one bid, the sum of the respective bid weights is -1 or 0.*

Proof. We prove the claim for $\mathcal{B}_{ij}^L$; the proof for $\mathcal{B}_{ij}^U$ is identical. Note that in order for $\mathcal{B}_{ij}^L$ to contain any bids, we must have $\boldsymbol{p} \in \{0,1\}^n$.

Suppose $p_i = 1$. Then for any bid $\boldsymbol{b} \in \mathcal{B}_{ij}^L$, the values of b_i and b_j are constrained to $b_i = 1$ and $b_j = p_j$. Moreover, the values of b_k are constrained by $b_k \leq p_k$. Hence for $\mathcal{B}_{ij}^L$ to contain at least two bids, we must have $p_k = 1$ for some $k \notin \{i,j\}$, allowing b_k to take values 0 and 1. Lemma 5.18 implies that $\mathcal{B}_{ij}^L$ contains an equal number of positive and negative bids.

Now suppose $p_i = 0$. Then b_i can take value 0 or 1 and we consider the two subsets of $\mathcal{B}_{ij}^L$ where b_i takes each value separately. Let $S \coloneqq \{\boldsymbol{b} \in \mathcal{B}_{ij}^L \mid b_i = 0\}$ and $T \coloneqq \{\boldsymbol{b} \in \mathcal{B}_{ij}^L \mid b_i = 1\}$.

By definition, $\boldsymbol{b}$ is a bid in S if $b_i = 0$, $b_j = p_j$ and $b_k \leq p_k$. By the same argument as above, S either contains a single positive bid at $b_j = p_j$ and $b_k = 0$ for all $k \neq i$, or it contains an equal number of positive and negative bids.

Finally, note that $\boldsymbol{b}$ is a bid in T if $b_i = 1$, $b_j = 1 + p_j$ and $b_k \leq 1 + p_k$ for all $k \in [n]$. The last constraint is vacuous as it holds for all possible values of b_k and p_k. Hence T is empty if $p_j = 1$ and contains an equal number of positive and negative bids otherwise. Together, these properties of S and T imply the claim. □

We are now ready to prove Lemma 5.19.

Proof of Lemma 5.19. Suppose that gadget G is placed at $\boldsymbol{x}$. We make use of Theorem 2.5 from Chapter 2 to prove the claim. Moreover, we can use Theorem 2.15 to restrict ourselves to checking the second validity criterion in Theorem 2.5 only at a finite number of price points. In our case, it suffices to verify the conditions for each vertex $\boldsymbol{p} \in \boldsymbol{x} + \{0,1\}^n$ and $\boldsymbol{p} \in \boldsymbol{x} + \{2,3\}^n$.

Fix $\boldsymbol{p} \in \boldsymbol{x} + \{0,1\}^n \cup \{2,3\}^n$ and distinct goods $i, j \in [n]_0$. We consider first the case where $i = 0$ and $j \in [n]$. By Lemma 5.20, the sum of weights of the gadget bids indifferent between 0 and j is at least -1. Secondly, by construction of the adversarial game, there is a gadget bid at $p_i \boldsymbol{e}^i$ and this bid is indifferent between i and the reject good. Hence the total sum of weights of the bids in the game indifferent between i and rejecting is at least 0.

Now let i, j be distinct (non-reject) goods in $[n]$. Note that $\boldsymbol{b} \in \{0,1\}^n$ is indifferent between i and j at $\boldsymbol{p}$, then $\boldsymbol{b}' = \boldsymbol{b} + \boldsymbol{2} \in \boldsymbol{x} + \{2,3\}^n$ is also indifferent between i and j. Moreover, $\boldsymbol{b}$ and $\boldsymbol{b}'$ have opposite weights. This fact, together with Lemma 5.21, implies that the sum of the weights of the gadget bids is at least -1. Again, note that for any $\boldsymbol{p}$ there is a boundary bid that is indifferent between i and j. Indeed, recall that a bid is indifferent between i and j if, and only if, $b_i - b_j = p_i - p_j$, $b_i \geq p_i$ and $b_k - p_k \leq b_i - p_i$. If $p_i \geq p_j$, then the bid $M\boldsymbol{e}^i + (M - (p_i - p_j))\boldsymbol{e}^j$ satisfies these conditions. Similarly, if $p_i < p_j$, then the bid $(M - (p_i - p_j))\boldsymbol{e}^i - M\boldsymbol{e}^j$ satisfies this condition. As this bid is present by construction of the adversarial game, we see that the total sum of weights of all bids that are indifferent between i and j in the game is at least 0 in all cases. □

Theorem 5.22. *Let $\mathcal{B}$ be a bid list that may comprise positive and negative bids. If $\mathcal{B}$ is valid, any algorithm requires $\Omega\left(\left(\frac{B-2^{n+1}}{8n^2}\right)^n\right)$ queries to learn $\mathcal{B}$. If $\mathcal{B}$ is allowed to be invalid, $\Omega\left((\frac{M}{4})^n\right)$ queries are required.*

Proof. Consider first the case where $\mathcal{B}$ is valid. Let $\mathcal{B}$ be the adversary's bid list, which is valid by Lemma 5.19. By Lemma 5.17, we know that the gadget placed at $\boldsymbol{x}$ does not affect demand outside the cube $\boldsymbol{x} + [3]_0^n$. Hence by a standard decision tree argument, any algorithm requires $k^n - 1$ queries to learn the location of the gadget G. Note that the adversary places a total of $B \leq 2^{n+1} + 8kn^2$ gadget and boundary bids. Solving for k gives $k \geq \frac{B-2^{n+1}}{8n^2}$. Hence expressing the query complexity of $k^n - 1$ in terms of B and n yields the desired expression.

For the case where $\mathcal{B}$ may be invalid, we construct an even simpler adversarial game, where the adversary positions the island gadget as above but places no boundary bids. By the same argument, learning the location of the gadget incurs at least $\Omega\left((\frac{M}{4})^n\right)$ queries. □

Using the fact that $\max\{f, g\} = \Omega(f + g)$ for any positive functions f and g, we can combine the lower bounds from Theorem 5.15 and Theorem 5.22, along with the upper bound from Algorithm 5.3 in Section 5.7, to identify the query complexity of expressing SS demand by means of positive and negative bids.

Corollary 5.23. *For constant n, learning SS demand requires $\Theta(B^n + B \log M)$ queries.*

5.9 Conclusion and outlook

In this chapter, we have presented the first algorithms for bidders to computationally express their strong-substitutes preferences as a list of tropical bids. Our algorithms for learning demand are conceptually simple and provide the first systematic approach for bidders to express their preferences in the tropical bidding language used by the strong-substitutes product-mix auction. This allows bidders with non-technical backgrounds to participate in these auctions under the mild assumption that they are able to answer demand oracle queries. In the setting where demand can be expressed using positive bids only, we present an algorithm that achieves linear query complexity. When demand may only be expressible using positive *and* negative bids, our hyperplane finding algorithm performs well if the number of goods is not too large. Moreover, our lower bound indicates that our algorithm cannot be improved significantly.

5.9.1 Future work

To the best of our knowledge, this work presents the first principled algorithms for expressing demand in a bidding language. Building on our results, there are multiple avenues for future work that we outline below.

Few negative bids. Demand preferences seen in practice tend to require only few negative bids. In light of this observation, it is of considerable practical interest to extend our positive-bid algorithm to allow a small (e.g. constant) number of negative bids. This is analogous to the algorithm for checking the validity of a tropical bid list that we presented in Chapter 2. A central question in this regard is also whether learning bid lists has fixed-parameter tractable query complexity, where the parameter in question is the number of negative bids. An FPT algorithm would be of considerable practical and theoretical interest.

Generalised tropical bids. In Chapter 2, we generalised the tropical bids to capture arbitrary general substitutes demand. As this demand type is significantly more expressive than strong substitutes, it is of great interest to extend our learning algorithms to this setting. If we bound the entries of the generalised tropical bids by some constant, it can be seen that only minor modifications to the definition of delta queries are required to generalise the algorithm for learning positive bids with the same query complexity. In the case of positive and negative bids, we conjecture that our algorithm can be similarly modified to accommodate the more varied orientations of the LIP facets.[4]

Learning arctic bids. Another related question is to adapt our algorithms to learning arctic bids. In this setting, we note that the arctic language admits only bids with positive weight, so that the most promising starting point is the algorithm of Section 5.6.1. Care must be taken to incorporate the fact that the

[4]Work in this direction is ongoing.

quantity demand by an arctic bid is not its weight, but its budget divided by the relevant market price. Hence, price-space can be fruitfully partitioned not according to demanded bundles, but according to how much is spent on each good. This, in turn, suggests the introduction of a *spending oracle*, which, for given prices, returns the amount spent on each good under some arbitrary valid expenditure $e : \mathcal{B} \times \mathbb{R}^n \to \mathbb{R}$ at $\boldsymbol{p}$. It then remains to suitably adapt the delta query procedure to reflect the 'arctic' orientations of facets.[5]

Dealing with inconsistencies. Additionally, one practical limitation of our current algorithm is the fact that bidders may not have strong-substitutes demand in practice, and thus answer the demand queries inconsistently. In this case, we currently cannot make any guarantees on what our algorithm returns. One way of alleviating this issue might be to impose certain restrictions on the bundles that a bidder may report when queries for demand at given prices. These restrictions are computed on the basis of the bidder's previous query replies, to ensure that the demand correspondence reported remains strong-substitutes. (E.g., "At prices $\boldsymbol{p}$, which bundle $\boldsymbol{x}$ of goods do you demand, subject to $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$?", where $\boldsymbol{l}$ and $\boldsymbol{u}$ are constraints on the bundles.) Intuitively, this forces bidders to report strong-substitutes demand. It is an open question whether the demand thus learnt approximates the bidder's 'true' demand well. Indeed, it is of interest to study approximate learning of the demand function, where the goal is to find a strong-substitutes demand that approximates the bidder's (non-substitutes) demand as well as possible, and dealing with errors in answers to queries more generally.

[5]This is work in progress. Note that 'arctic' facets are either normal to a unit vector $\boldsymbol{e}^i$, $i \in [n]$, or are contained in a hyperplane which, in turn, contains the origin $\mathbf{0}$.

6

Solving the arctic product-mix auction

Contents

6.1 Chapter outline 122
6.2 Introduction 123
6.2.1 Our contributions 125
6.2.2 The Fisher market 126
6.3 Preliminaries 126
6.4 Efficiency and competitive equilibrium 128
6.5 Element-wise minimal prices 129
6.6 Maximising revenue 131
6.7 Market-clearing prices are unique 132
6.8 Minimal feasible prices clear the market 133
6.8.1 The demand and expenditure graphs 134
6.8.2 Reducing prices along a tree 135
6.8.3 The procedure for reducing prices 138
6.9 Finding market-clearing prices 139
6.10 Conclusion and outlook 140

6.1 Chapter outline

This chapter studies a version of the arctic product-mix auction introduced by Klemperer (2010) without seller costs. Zero seller costs are natural in many settings, e.g. for digital goods. Unlike the strong-substitutes product-mix auction studied in Chapters 3 and 4, the arctic auction is run to allocate *divisible* goods. Moreover, bidders express their preferences using arctic bids. Due to this, the arctic auction bears similarities to the Fisher market with linear utilities (Nisan et al., 2007); we discuss the connections in Section 6.2.2. Unlike in Fisher markets, buyers

participating in the arctic auction do not necessarily spend their entire budget but instead reject a potential purchase if prices are too high. This allows for the possibility of solving the auction with the aim of maximising social welfare *or* optimal revenue.

In this chapter, we establish a novel coincidence of these two solution concepts of welfare and revenue. To the best of our knowledge, this work is the first to describe a market in which this coincidence holds. A first result, given in Section 6.4, is to show that the first and second welfare theorems hold for our arctic market: in other words, social welfare and competitive equilibrium coincide. Sections 6.5 and 6.6 then prove that there exists an element-wise smallest price vector $\boldsymbol{p}^*$ at which demand does not exceed supply, and this vector maximises revenue. In Section 6.8, we establish that $\boldsymbol{p}^*$ also supports a competitive equilibrium and is thus welfare-maximising. We do this by modifying a price-scaling routine that forms part of a 'simplex-like' algorithm by Adsul et al. (2012) for Fisher markets to our arctic market setting; this is not immediate, as bids can reject if prices are too high. In our approach, we scale prices down instead of up, and apply additional pre-processing for bids that may indifferent between rejecting and spending. Finally, Section 6.9 presents an analogous algorithm to that of Adsul et al. (2012) for finding $\boldsymbol{p}^*$. As for the 'simplex-like' algorithm of Adsul et al. (2012), it is an open question whether this algorithm runs in polynomial time. The results in this chapter appear in (Finster et al., 2021), co-authored with Simon Finster and Paul Goldberg.

6.2 Introduction

Finding equilibrium prices in markets with multiple divisible goods has been a subject of great interest for centuries. The divisible goods market by Walras (1896) and its important special case, the Fisher market (Brainard and Scarf, 2005), have received a great deal of attention from economists and computer scientists alike. This has been motivated, in part, by the celebrated result of Arrow and Debreu (1954) on the existence of a general market equilibrium in the Arrow-Debreu model. Since then, a rich algorithmic literature has emerged to compute market equilibria (e.g. (Adsul et al., 2012; Devanur and Vazirani, 2004; Devanur et al., 2008; Eaves, 1976; Garg et al., 2013; Jain, 2007; Orlin, 2010)), and efficient algorithms have been described for both Walras and Fisher markets for the linear utilities case and beyond. However, in many applications, these markets have one important short-coming: demand is scale-invariant, that is buyers may demand goods at arbitrarily high prices.[1]

In the arctic product-mix auction, buyers submit substitutes preferences for divisible goods by means of the arctic bidding language introduced in Chapter 2. The preferences are, thus, scale-invariant only up to certain price thresholds that are buyer-specific; buyers may choose not to spend on any goods if prices are too high.[2] This property of demand makes it possible to consider revenue, or profit, as

[1] In contrast to Walrasian markets, money has no intrinsic value in Fisher markets (Nisan et al., 2007). Hence bidders always wish to spend their entire budget, which implies the scale-invariance of demand.

[2] We note that by setting these thresholds arbitrarily high, an arctic bid can 'simulate' linear

an objective for the seller: if not all supply is sold by default, it may be profitable to raise prices and withhold some supply from the market instead. In Fisher markets, this is not a viable objective because bidders spend their entire budget at any prices, resulting in identical revenue for the seller. Therefore, our work reveals a surprising and desirable feature of the arctic market: we prove the existence of unique market equilibrium prices, that *simultaneously* maximise the seller's revenue. Because the welfare theorems hold in our setting, these prices have the desirable property of being efficient *and* optimal. To the best of our knowledge, this work is the first to describe a market in which these coincidences holds.

The market we consider is an important variant of the arctic product-mix auction market first proposed by Klemperer (2010) for the government of Iceland. In the general arctic product-mix auction, the seller can additionally choose cost functions to fine-tune its preferences, while in our model, we assume the seller's costs to be zero. We refer to (Fichtl, 2020) for some discussion and algorithmic results of the general case. However, we emphasise that the arctic auction has so far only been studied in terms of revenue, and not social welfare.

In our market, a number of buyers compete for a given supply $\boldsymbol{s}$ of n distinct but divisible goods. Each buyer submits a list of arctic bids, each of which consisting of a budget and per-unit values for each good. We emphasise that unlike the tropical language, all bids in the arctic language have positive weight (which corresponds to their budget). Hence the seller can process each bid independently and it is without loss of generality to assume that each buyer submits a single bid, i.e. has a single budget-value combination. The seller sets a uniform market price for each good, i.e. each buyer who demands a given good pays this price per allocated unit. We briefly recall how arctic bids induce demand. Given a market price vector (subsequently simply called 'price vector'), each buyer demands only those goods that give him the maximal 'bang-per-buck', that is the maximal ratio of value to price among all goods. If a buyer demands a single good, he demands his entire budget's worth of this good. If a buyer demands multiple goods, he demands quantities that, multiplied by prices, make up his entire budget. If a buyer's values are strictly lower than prices, a buyer demands nothing. If a buyer is indifferent between one or multiple goods and receiving nothing, he demands quantities that, multiplied by prices, make up weakly less than his budget.

Applications. The arctic auction offers an intuitive bidding language that, unlike the tropical language, acts on divisible goods. The government planned to use this auction market to exchange blocked accounts for other financial assets, e.g. cash or bonds with different properties, which were available only in limited but divisible quantities. The auction was especially fitting for this use because buyers could express a budget, as well as trade-offs between different assets through bid prices. Other related applications may include debt restructuring and the (re-)division of firms between shareholders (Klemperer, 2018).

Our market may be applicable to advertising auctions (Conitzer et al., 2017, 2019). When advertising companies compete for web space (goods) to display their

'Fisher' preferences. Hence the arctic product-mix auction generalises the Fisher market.

ads, the decision which publisher to choose and bid for is difficult. It may be intuitive, however, to choose an advertising budget and state demand in terms of 'limit market prices' for multiple, distinct goods, below which the seller, or the market platform, allocates those good with the highest bang-per-buck. This is not only conceptually easier for the advertiser (buyer), but also practical and feasible through our results and algorithm for the arctic product-mix setting.

6.2.1 Our contributions

In setting market prices, the seller can pursue a variety of objectives. The focus of our contribution is on the objectives of maximising revenue, achieving competitive equilibrium, and optimising efficiency and social welfare. Our model assumes complete information of all agents in the markets and truthful submission of valuations in the form of arctic bids. As described in Chapter 1, truthful bidding is a natural assumption when we have a large number of bidders with similar budgets. Thus, social welfare is defined with respect to the submitted valuations. If the seller aims to maximise efficiency, it is intuitive that she wants to sell her entire supply; we prove this in Section 6.8. However, if the seller aims to maximise her revenue, she may want to raise prices and keep some of her supply to achieve higher revenue. As our main result, we prove that our market admits a unique price vector $\boldsymbol{p}^*$ that unifies our two objectives: $\boldsymbol{p}^*$ clears the market and supports a competitive equilibrium (Section 6.7), and it maximises revenue (Section 6.6). In Section 6.9, we also present an algorithm to find these prices. While we show that market-clearing prices are unique in our market, the same does not hold for revenue-optimal prices. Our prices $\boldsymbol{p}^*$ are, nevertheless, bidder-optimal revenue-maximising prices in the sense that allocated quantities are maximised. To the best of our knowledge, we are the first to describe a market setting in which revenue and welfare coincide.

We start by establishing a common representation of the buyers' utility, implying that our market is 'well-behaved', that is the first and second welfare theorem hold. We introduce the 'feasible region' in price space, which consists of the set of prices at which demand does not exceed supply. At prices not contained in the region, market demand exceeds the supply of at least one good. We demonstrate that the feasible region has the lower semi-lattice property, guaranteeing the existence of element-wise minimal prices. We proceed to show that at these prices, the seller's revenue is maximised. Intuitively, the seller's revenue only depends on collecting the budgets of those bids that accept to buy at the market prices she sets. Thus, setting prices as low as possible while not violating supply must be revenue-optimal. However, it is not immediately obvious that markets clear at the element-wise minimal feasible price vector. We then modify a tree-based price reduction procedure from (Adsul et al., 2012) that allows us to reduce any given price vector that is not market-clearing. This immediately suggests an algorithm to compute market-clearing prices. It is an open question whether our algorithm admits a polynomial-time running time analysis; we expect our current exponential bound to be loose. Finally, we demonstrate by means of our price reduction procedure that there exist unique market-clearing prices, implying that our algorithm finds

competitive equilibrium and revenue-optimal prices in one.

6.2.2 The Fisher market

We briefly present the Fisher market with linear utilities, which can be seen as a special case of the arctic product-mix auction in which all arctic bid values are set sufficiently high so that the bids never reject. A Fisher market consists of bidders $[m]$ and goods $[n]$. In the linear utilities model, each bidder $j \in [m]$ has a budget β^j as well as a utility $u^j(\boldsymbol{x}) = \boldsymbol{a}^j \cdot \boldsymbol{x}$ for any given bundle $\boldsymbol{x} \in \mathbb{Z}^n$, where $\boldsymbol{a}^j \in \mathbb{Z}^n$ is a vector denoting, for each good $i \in [n]$, the utility that the bidder derives per unit of good i. Note that u^j is scale-invariant. Without loss of generality, we assume unit supply of each good (by re-scaling utilities of all bidders if necessary).

The objective of the Fisher market is to compute a competitive equilibrium: that is, to determine prices $\boldsymbol{p}$ and an allocation $\pi : [m] \to \mathbb{R}^n$ to bidders so that the following holds.

The market is cleared. The entire supply is allocated ($\pi([m]) = \mathbf{1}$), and every bidder spends its entire budget ($\boldsymbol{p} \cdot \pi(k) = \beta^j$).

'Bang per buck' is maximised. For every bidder j, if $\frac{\boldsymbol{a}_i^j}{p_i} < \frac{\boldsymbol{a}_k^j}{p_k}$ for goods i and k, then they are not allocated any quantity of good i (so $\pi_i(j) = 0$).

It can be seen easily that every arctic bid demands the same quantities and maximises 'bang per buck' in the same way as a bidder with linear utilities in the Fisher market, as long as market prices do not lie strictly above the bid prices. In particular, we can simulate a linear-utility Fisher bidder with a single arctic bid by letting the bid prices be sufficiently high.

6.3 Preliminaries

We now formally introduce our market model and define the notions of expenditure, allocation and feasibility of prices. These definitions will be crucial for the subsequent sections.

Market model. Each bidder submits a list of arctic bids. As the seller treats each bid independently, we may assume that each bidder submits a single bid. We let $\mathcal{B}$ denote the list containing each bidder's bid. A seller has a given supply $\boldsymbol{s} \in \mathbb{R}_+^n$ of distinct divisible goods and wishes to allocate some subset of this supply among the buyers. The notional reject good is available with infinite supply $s_0 = \infty$. The seller sets a market price vector $\boldsymbol{p} \in \mathbb{R}_+^n$ that induces an allocation according to bidders' demand. We recall the following from Section 2.6. At given market prices $\boldsymbol{p}$, each bid spends its budget $\beta(\boldsymbol{b})$ on a combination of goods $I_{\boldsymbol{b}}(\boldsymbol{p})$. Hence, we define its *demand correspondence* $D_{\boldsymbol{b}}$ as the set of bundles $D_{\boldsymbol{b}}(\boldsymbol{p}) := \left\{\boldsymbol{x} \in \mathbb{R}_+^n \mid \boldsymbol{x} \cdot \boldsymbol{p} = \beta(\boldsymbol{b}) \text{ and } x_i > 0 \implies i \in I_{\boldsymbol{b}}(\boldsymbol{p})\right\}$. The demand correspondence $D_{\mathcal{B}}$ of a bid list $\mathcal{B}$ is then given by the Minkowski sum $D_{\mathcal{B}}(\boldsymbol{p}) = \left\{\sum_{\boldsymbol{b} \in \mathcal{B}} \boldsymbol{x}^{\boldsymbol{b}} \mid \boldsymbol{x}^{\boldsymbol{b}} \in D_{\boldsymbol{b}}(\boldsymbol{p})\right\}$.

Expenditure and allocation. An expenditure $e : \mathcal{B} \times [n]_0 \to \mathbb{R}_+$ expresses, for all bids $\boldsymbol{b} \in \mathcal{B}$ and all goods $i \in [n]_0$, the amount $e(\boldsymbol{b}, j)$ that bid $\boldsymbol{b}$ spends on good i. For each expenditure, we can define the corresponding *allocation* $\pi : \mathcal{B} \times [n]_0 \to \mathbb{R}_+$ denoting the amount $\pi(\boldsymbol{b}, j) = \frac{e(\boldsymbol{b},j)}{p_j}$ of good j that bid $\boldsymbol{b}$ receives. We also write $\pi(\boldsymbol{b})$ (with a single argument) to refer to the bundle that $\boldsymbol{b}$ receives, so $\pi(\boldsymbol{b})_j = \pi(\boldsymbol{b}, j)$ for each $j \in [n]_0$. For any subset $\mathcal{B}' \subseteq \mathcal{B}$ and $I \subseteq [n]_0$, we recall the implicit summation notation $e(\mathcal{B}', i) := \sum_{\boldsymbol{b}\in\mathcal{B}'} e(\boldsymbol{b}, i)$, $e(\boldsymbol{b}, I) := \sum_{i\in I} e(\boldsymbol{b}, i)$, and $e(\mathcal{B}', I) := \sum_{\boldsymbol{b}\in\mathcal{B}', i\in I} e(\boldsymbol{b}, i)$ (and similarly for allocations π). Given an expenditure e, we say that good $i \in [n]$ is *saturated* if $e(\mathcal{B}, i) = p_i s_i$, and *unsaturated* if $e(\mathcal{B}, i) < p_i s_i$.

Definition 6.1 (Feasibility and validity). Let e be an expenditure. Then e is:

(i) *Supply-feasible* at prices $\boldsymbol{p}$ if supply weakly exceeds the allocation for all goods i, so $e(\mathcal{B}, i)/p_i \leq s_i$ for all $i \in [n]$.

(ii) *Budget-feasible* at prices $\boldsymbol{p}$ if every bid $\boldsymbol{b}$ spends weakly less than its budget on non-reject goods. The remaining (non-negative) amount is always spent on the reject good, so $e(\boldsymbol{b}, [n]_0) = \beta(\boldsymbol{b})$ for all $\boldsymbol{b} \in \mathcal{B}$.

(iii) *Demand-valid* at prices $\boldsymbol{p}$ if each bid only spends on a good if it demands it, so $j \in I_{\boldsymbol{b}}(\boldsymbol{p})$ if $e(\boldsymbol{b}, j) > 0$.

An expenditure is *feasible* if (i) and (ii) hold, and it is *valid* if (iii) also holds. We extend the same terminology to the allocations π corresponding to feasible or valid expenditures defined by $\pi(\boldsymbol{b}, j) = \frac{e(\boldsymbol{b},j)}{p_j}$. Note that for any valid allocation π at prices $\boldsymbol{p}$, the bundle of goods that bidder is allocated under π lies in $D_{\boldsymbol{b}}(\boldsymbol{p})$.

Definition 6.2. Prices $\boldsymbol{p}$ are *feasible* if there exists an expenditure e valid at $\boldsymbol{p}$.

Market objectives. We consider two separate objectives for the seller. She may set prices with the objective of maximising her revenue, or she may set prices to maximise social welfare. Intuitively (and we prove this below), at the social optimum, the seller's entire supply is allocated. To maximise revenue, the seller may prefer to raise prices and keep some of her supply.

The *revenue maximisation problem* consists of a bid list $\mathcal{B}$ and a supply vector $\boldsymbol{s} \in \mathbb{R}^n_+$. A solution to this problem is a feasible price vector $\boldsymbol{p}$ and allocation π that maximise revenue $\sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]} p_i \pi(\boldsymbol{b}, i) = \boldsymbol{p} \cdot \pi(\mathcal{B})$ subject to π being valid at $\boldsymbol{p}$. We define the revenue function R as $R(\boldsymbol{p}) := \max_\pi \boldsymbol{p} \cdot \pi(\mathcal{B})$, where we maximise over valid allocations π.

Recall that the buyers are price takers. Thus, social welfare (and efficiency below) is defined with respect to the submitted values. The *social welfare maximisation problem* consists of a bid list $\mathcal{B}$ and a supply vector $\boldsymbol{s} \in \mathbb{R}^n_+$. A solution $(\boldsymbol{p}, \pi)$ to this problem is a price vector $\boldsymbol{p}$ and allocation π that maximise social welfare $\sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]} b_i \pi(\boldsymbol{b}, i) = \sum_{\boldsymbol{b}\in\mathcal{B}} \boldsymbol{b} \cdot \pi(\boldsymbol{b})$, subject to π being a valid allocation.

6.4 Efficiency and competitive equilibrium

We now formally define efficiency and show that the first and second welfare theorems hold in our market. Hence social welfare is maximised at competitive equilibrium solutions.

Definition 6.3 (Efficiency). We say that a feasible allocation π is *efficient* if $\sum_{\boldsymbol{b}\in\mathcal{B},i\in[n]} b_i\pi(\boldsymbol{b},i) \geq \sum_{\boldsymbol{b}\in\mathcal{B},i\in[n]} b_i\pi'(\boldsymbol{b},i)$ for all feasible allocations $\pi' \neq \pi$.

Definition 6.4 (Competitive equilibrium). Given a bid list $\mathcal{B}$ and a supply vector $\boldsymbol{s}$, prices $\boldsymbol{p}$ and allocation π (at $\boldsymbol{p}$) form a competitive equilibrium if, and only if, π is valid at $\boldsymbol{p}$ and $\pi(\mathcal{B},i) = s_i$ for all goods i with $p_i > 0$.

The following technical lemma allows us to prove the first and second welfare theorem.

Lemma 6.5. *Suppose π is a budget-feasible allocation at prices $\boldsymbol{p}$ and fix bid $\boldsymbol{b}$. Then $\sum_{i\in[n]_0}(b_i-p_i)\pi(\boldsymbol{b},i) \geq \sum_{i\in[n]_0}(b_i-p_i)\pi'(\boldsymbol{b},i)$ for all budget-feasible allocations π' if, and only if, $\pi(\boldsymbol{b}) \in D_{\boldsymbol{b}}(\boldsymbol{p})$.*

Proof. Let π and π' be two allocations budget-feasible at $\boldsymbol{p}$ with corresponding expenditures e and e', and fix a bid $\boldsymbol{b}$. Note that $e(\boldsymbol{b},[n]_0) = e'(\boldsymbol{b},[n]_0)$, as every bid spends its entire budget (some, or all, of it on the reject good 0). Hence,

$$\begin{aligned}
\sum_{i\in[n]_0}(b_i-p_i)\pi(\boldsymbol{b},i) &= \sum_{i\in I_{\boldsymbol{b}}(\boldsymbol{p})}\left(\frac{b_i}{p_i}-1\right)e(\boldsymbol{b},i)\\
&= \max_{i\in[n]_0}\left(\frac{b_i}{p_i}-1\right)e(\boldsymbol{b},[n]_0)\\
&= \max_{i\in[n]_0}\left(\frac{b_i}{p_i}-1\right)e'(\boldsymbol{b},[n]_0)\\
&\geq \sum_{i\in[n]_0}\left(\frac{b_i}{p_i}-1\right)e'(\boldsymbol{b},i)\\
&= \sum_{i\in[n]_0}(b_i-p_i)\pi'(\boldsymbol{b},i). \qquad \square
\end{aligned}$$

Theorem 6.6 (First welfare theorem). *If $(\boldsymbol{p},\pi)$ is a competitive equilibrium, then π is an efficient allocation.*

Proof. Let $(\boldsymbol{p},\pi)$ be a competitive equilibrium with corresponding expenditure e. Let π' be a feasible allocation at $\boldsymbol{p}$ with expenditure e'. Lemma 6.5 implies

$$\sum_{\boldsymbol{b}\in\mathcal{B},i\in[n]}(b_i-p_i)\pi(\boldsymbol{b},i) \geq \sum_{\boldsymbol{b}\in\mathcal{B},i\in[n]}(b_i-p_i)\pi'(\boldsymbol{b},i).$$

Rearranging and using the fact that $e(\mathcal{B},[n]) = \boldsymbol{p}\cdot\boldsymbol{s} \geq e'(\mathcal{B},[n])$, we have

$$\sum_{\boldsymbol{b}\in\mathcal{B},i\in[n]} b_i\pi(\boldsymbol{b},i) - \sum_{\boldsymbol{b}\in\mathcal{B},i\in[n]} b_i\pi'(\boldsymbol{b},i) \geq e(\mathcal{B},[n]) - e'(\mathcal{B},[n]) \geq 0. \qquad \square$$

Theorem 6.7 (Second welfare theorem). *Let $(\boldsymbol{p}, \pi)$ be a competitive equilibrium and π' be another efficient and feasible allocation. Then $(\boldsymbol{p}, \pi')$ is also a competitive equilibrium.*

Proof. By definition of efficiency, we first note that

$$\sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]} b_i \pi(\boldsymbol{b}, i) = \sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]} b_i \pi'(\boldsymbol{b}, i). \tag{6.1}$$

Secondly, as π is a competitive equilibrium and π' is feasible, we have $\pi(\mathcal{B}) = \boldsymbol{s}$ and $\pi'(\mathcal{B}) \leq \boldsymbol{s}$. Taking the dot product of $\boldsymbol{p}$ with both terms, we get

$$\boldsymbol{p} \cdot \pi(\mathcal{B}) \geq \boldsymbol{p} \cdot \pi'(\mathcal{B}). \tag{6.2}$$

We now show that equality must hold for (6.2) by demonstrating that strict inequality contradicts (6.1). This proves our claim. Indeed, assuming (6.2) with strict inequality and using Lemma 6.5, we obtain

$$\begin{aligned}
\sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]} b_i \pi(\boldsymbol{b}, i) &= \sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]} (b_i - p_i)\pi(\boldsymbol{b}, i) + \boldsymbol{p} \cdot \pi(\mathcal{B}) && (6.3)\\
&> \sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]} (b_i - p_i)\pi'(\boldsymbol{b}, i) + \boldsymbol{p} \cdot \pi'(\mathcal{B}) && (6.4)\\
&= \sum_{\boldsymbol{b}\in\mathcal{B}, i\in[n]} b_i \pi'(\boldsymbol{b}, i). && (6.5)
\end{aligned}$$

□

6.5 Element-wise minimal prices

Recall that prices $\boldsymbol{p}$ are feasible if there exists a valid allocation π at $\boldsymbol{p}$. We show that set of feasible prices form a lower semi-lattice. In particular, there exists a special price vector $\boldsymbol{p}^*$ that is element-wise smaller than all other feasible prices (so that $\boldsymbol{p}^* \leq \boldsymbol{p}$ for all feasible $\boldsymbol{p}$). For any two price vectors $\boldsymbol{p}$ and $\boldsymbol{q}$, we let $\boldsymbol{p} \wedge \boldsymbol{q}$ denote their *element-wise minimum* defined as $(\boldsymbol{p} \wedge \boldsymbol{q})_i = \min\{p_i, q_i\}$. The following lemma is central to our proof.

Lemma 6.8. *If $\boldsymbol{p}$ and $\boldsymbol{q}$ are feasible, then so is their element-wise minimum $\boldsymbol{p} \wedge \boldsymbol{q}$.*

Fix feasible prices $\boldsymbol{p}$ and $\boldsymbol{p}'$ with element-wise minimum $\boldsymbol{r} = \boldsymbol{p} \wedge \boldsymbol{p}'$, and let π and π' respectively denote valid allocations at $\boldsymbol{p}$ and $\boldsymbol{p}'$. In order to prove Lemma 6.8, we construct an allocation τ at $\boldsymbol{r}$ and show that it is valid. We first define the set of goods A in which $\boldsymbol{p}$ is strictly dominated by $\boldsymbol{p}'$, and its complement B, so $A = \{i \in [n] \mid p_i < p'_i\}$ and $B = \{i \in [n] \mid p_i \geq p'_i\}$. Then our allocation τ at $\boldsymbol{r}$ is given by

$$\tau(\boldsymbol{b}, \cdot) = \begin{cases} \pi'(\boldsymbol{b}, \cdot) & \text{if } \boldsymbol{b} \text{ demands some good } i \in B \text{ at } \boldsymbol{r}, \\ \pi(\boldsymbol{b}, \cdot) & \text{otherwise.} \end{cases} \tag{6.6}$$

In order to prove the validity of τ, we first state a technical lemma that establishes the connection between the demand of a bid at $\boldsymbol{p}$, $\boldsymbol{p}'$ and $\boldsymbol{r}$.

Lemma 6.9. *Suppose bid* $\boldsymbol{b}$ *demands good* $i \in A$ *at* $\boldsymbol{r}$*. Then* $\boldsymbol{b}$ *also demands* i *at* $\boldsymbol{p}$ *and* $I_{\boldsymbol{b}}(\boldsymbol{p}) \subseteq I_{\boldsymbol{b}}(\boldsymbol{r})$*. Similarly, suppose* $\boldsymbol{b}$ *demands* $i \in B$ *at* $\boldsymbol{r}$*. Then it also demands* i *at* $\boldsymbol{p}'$*, and* $I_{\boldsymbol{b}}(\boldsymbol{p}') \subseteq I_{\boldsymbol{b}}(\boldsymbol{r})$*. Moreover, we have* $I_{\boldsymbol{b}}(\boldsymbol{p}') \subseteq B$*.*

Proof. Fix a bid $\boldsymbol{b}$ that demands good $i \in A$ at $\boldsymbol{r}$. As $p_i = r_i$, this implies $\frac{b_i}{p_i} = \frac{b_i}{r_i} \geq \frac{b_j}{r_j} \geq \frac{b_j}{p_j}$ for all goods $j \in [n]_0$. The first inequality holds due to the definition of demand, and the second inequality follows from $r_j \leq p_j, \forall j \in [n]_0$. Hence, $\boldsymbol{b}$ demands good i at $\boldsymbol{p}$. For the second claim that $I_{\boldsymbol{b}}(\boldsymbol{p}) \subseteq I_{\boldsymbol{b}}(\boldsymbol{r})$, fix a good $k \in I_{\boldsymbol{b}}(\boldsymbol{p})$. Then we have $\frac{b_k}{r_k} \geq \frac{b_k}{p_k} \geq \frac{b_i}{p_i} = \frac{b_i}{r_i} \geq \frac{b_j}{r_j}$ for all goods $j \in [n]_0$. The first inequality holds due to $r_k \leq p_k$, and the second and third inequalities follow from the fact that $\boldsymbol{b}$ demands k at $\boldsymbol{p}$ and i at $\boldsymbol{r}$. Hence, if bid $\boldsymbol{b}$ demands good k at $\boldsymbol{p}$, then it demands k at $\boldsymbol{r}$.

Now suppose that $\boldsymbol{b}$ demands $i \in B$ at $\boldsymbol{r}$. The proof of the first claim is identical to the case $i \in A$. We prove the last claim that $I_{\boldsymbol{b}}(\boldsymbol{p}') \subseteq B$. Suppose, for contradiction, that $\boldsymbol{b}$ demands a good $k \in A$ at $\boldsymbol{p}'$, and good $i \in B$ at $\boldsymbol{r}$. This implies $\frac{b_k}{p'_k} < \frac{b_k}{p_k} = \frac{b_k}{r_k} \leq \frac{b_i}{r_i} = \frac{b_i}{p'_i}$, in contradiction to the fact that k is demanded at $\boldsymbol{p}'$. □

We can now prove Lemma 6.8.

Proof of Lemma 6.8. Let $\boldsymbol{p}$ and $\boldsymbol{p}'$ be two feasible prices and $\boldsymbol{r} = \boldsymbol{p} \wedge \boldsymbol{p}'$ denote their element-wise minimum. As above, π and π' are valid allocations at $\boldsymbol{p}$ and $\boldsymbol{p}'$, and τ is defined as in (6.6). First we see that τ is indeed an allocation, as $\tau(\boldsymbol{b}, j) \geq 0$ for all bids $\boldsymbol{b}$ and goods j. It remains to prove that τ satisfies the three criteria of Definition 6.1 that define validity. We can partition bids into two sets: the set $\mathcal{B}_B \subseteq \mathcal{B}$ of bids that demand a good in B at $\boldsymbol{r}$, and the set $\mathcal{B}_A = \mathcal{B} \setminus \mathcal{B}_B$ of bids that do not. Note that, by Lemma 6.9, the bids in $\mathcal{B}_B$ demand only goods in B at $\boldsymbol{p}'$ and, by definition, the bids in $\mathcal{B}_A$ demand only goods in A at $\boldsymbol{p}$. It follows that, under π, each bid $\boldsymbol{b} \in \mathcal{B}_A$ is only allocated quantities of goods in A (so $\pi(\mathcal{B}_A, j) > 0$ only if $j \in A$), and, similarly, every bid $\boldsymbol{b} \in \mathcal{B}_B$ only receives quantities of goods in B under π' (so $\pi(\mathcal{B}_B, j) > 0$ only if $j \in B$).

First we show that τ is supply-feasible at $\boldsymbol{r}$. Recall that τ is supply-feasible if $\tau(\mathcal{B}, j) \leq s_j$ for all $j \in [n]$. Fix some good $j \in A$. Then by definition of τ, we have $\tau(\mathcal{B}, j) = \pi(\mathcal{B}_A, j) + \pi'(\mathcal{B}_B, j)$. Recalling that $\pi'(\mathcal{B}_B, j) = 0$ and making use of the validity of π at $\boldsymbol{p}$, we get $\tau(\mathcal{B}, j) \leq \pi(\mathcal{B}, j) \leq s_j$. This immediately implies that τ satisfies part (i) of Definition 6.1 at $\boldsymbol{r}$ for all goods $j \in A$. Analogously, we can show that τ satisfies (i) at $\boldsymbol{r}$ for all goods $j \in B$ by recalling that $pi(\mathcal{B}_A, j) = 0$ for any good $j \in B$. As $A \cup B = [n]$, we have shown that τ is supply-feasible.

Next we argue that τ is budget-feasible at $\boldsymbol{r}$. Indeed, note that any bid $\boldsymbol{b} \in \mathcal{B}_A$ only demands goods in A. Moreover, the prices of goods in A are the same at $\boldsymbol{p}$ and $\boldsymbol{r}$, by construction of $\boldsymbol{r}$. It follows that $\boldsymbol{b}$ spends the same under τ at $\boldsymbol{r}$ as it does under π at $\boldsymbol{p}$. As π is budget-feasible at $\boldsymbol{p}$, we see that τ satisfies part (ii) of Definition 6.1 for all bids in $\mathcal{B}_A$. Similarly, as the bids in $\mathcal{B}_B$ only demand good in B, we apply the same argument to see that τ satisfies part (ii) for all bids in $\mathcal{B}_B$. As $\mathcal{B} = \mathcal{B}_A \cup \mathcal{B}_B$, τ is budget-feasible.

Finally, we show the demand-validity of τ at $\boldsymbol{r}$. Lemma 6.9 and the validity of π at $\boldsymbol{p}$ together imply that, for any $j \in [n]$, we have $\tau(\boldsymbol{b}, j) = \pi(\boldsymbol{b}, j) > 0$ only if $j \in I_{\boldsymbol{b}}(\boldsymbol{p}) \subseteq I_{\boldsymbol{b}}(\boldsymbol{r})$, which implies that τ satisfies part (iii) of Definition 6.1 at $\boldsymbol{r}$ for all bids in $\mathcal{B}_A$. Analogously, as $\tau(\boldsymbol{b}, j) = \pi'(\boldsymbol{b}, j)$ for all bids $\boldsymbol{b} \in \mathcal{B}_B$, we see that τ satisfies constraint (iii) at $\boldsymbol{r}$ for all bids in $\mathcal{B}_B$. Hence, τ is demand-valid at $\boldsymbol{r}$, and thus valid. □

Corollary 6.10. *There exists an element-wise minimal price vector $\boldsymbol{p}^*$.*

Proof. Suppose there exists no such price vector. This means that for all feasible $\boldsymbol{p}$, there exists $\boldsymbol{q}$ with $q_i < p_i$ for at least one good $i \in [n]$. Fix some feasible prices $\boldsymbol{p}$ with the property that $\boldsymbol{p}$ cannot be reduced any further in any direction. Such a point must exist, as the feasible region is closed and restricted to $\mathbb{R}^n_+$. By assumption, there exists $\boldsymbol{q}$ with $q_i < p_i$ for some $i \in [n]$. Now consider $\boldsymbol{p}' = \boldsymbol{p} \wedge \boldsymbol{q}$. By Lemma 6.8, we know that $\boldsymbol{p}'$ is feasible. But as $\boldsymbol{p}' \leq \boldsymbol{p}$ with $\boldsymbol{p}'_i < \boldsymbol{p}_i$, this contradicts our assumption that $\boldsymbol{p}$ cannot be reduced further. □

6.6 Maximising revenue

In Section 6.5, we established that the set of feasible prices contain a unique element-wise minimal price vector $\boldsymbol{p}^*$. We now show that revenue is maximised at these prices. This result is intuitive, as bidders wish to continue to spend their entire budget if we reduce prices (even if the goods they spend on might change). In the proof, we take care to show that bidders marginal between the reject good and other goods are able to spend the same on non-reject goods, and demand does not exceed supply. Note that we do not assume $\boldsymbol{p}^*$ to be the *only* prices at which revenue is maximised; indeed, there can be many revenue-maximising prices. However, $\boldsymbol{p}^*$ is 'bidder-optimal' among revenue-maximising prices in the sense that it maximises the quantities of goods allocated.

Lemma 6.11. *For any two distinct feasible price vectors $\boldsymbol{p}$ and $\boldsymbol{p}'$ with $\boldsymbol{p} \leq \boldsymbol{p}'$ we have $R(\boldsymbol{p}) \geq R(\boldsymbol{p}')$. In other words, the maximum obtainable revenue at $\boldsymbol{p}$ is weakly greater than the revenue obtainable at $\boldsymbol{p}'$.*

Proof. Let π and π' be valid allocations that respectively maximise revenue at $\boldsymbol{p}$ and $\boldsymbol{p}'$. Our goal is to determine a valid allocation τ that achieves a weakly greater revenue at $\boldsymbol{p}$ than π' does at $\boldsymbol{p}'$. As $R(\boldsymbol{p}) \geq \sum_{\boldsymbol{b} \in \mathcal{B}, i \in [n]} p_i \tau(\boldsymbol{b}, i)$ and $R(\boldsymbol{p}') = \sum_{\boldsymbol{b} \in \mathcal{B}, i \in [n]} p'_i \pi'(\boldsymbol{b}, i)$, this immediately implies the result.

If $\boldsymbol{p} = \boldsymbol{p}'$, there is nothing to prove. Hence, we assume without loss of generality that $S := \{i \in [n] \mid p_i < p'_i\}$, the set of goods which are priced strictly lower at $\boldsymbol{p}$ than at $\boldsymbol{p}'$, is non-empty. Fix a bid $\boldsymbol{b} \in \mathcal{B}$. In order to define the new allocation $\tau(\boldsymbol{b}, \cdot)$ to $\boldsymbol{b}$ at $\boldsymbol{p}$, we distinguish between the two cases that $I_{\boldsymbol{b}}(\boldsymbol{p})$ is, and is not, a subset of S.

Case 1: Suppose $\boldsymbol{b}$ demands a subset of S at $\boldsymbol{p}$, so $I_{\boldsymbol{b}}(\boldsymbol{p}) \subseteq S$. In this case, we set $\tau(\boldsymbol{b}, \cdot) = \pi(\boldsymbol{b}, \cdot)$. As π is valid and τ and π are both allocations at the

same prices, the bid $\boldsymbol{b}$ spends its entire budget under τ. Moreover, it is only allocated goods that it demands.

Case 2: Suppose $I_{\boldsymbol{b}}(\boldsymbol{p}) \not\subseteq S$. We note that $I_{\boldsymbol{b}}(\boldsymbol{p}') \cap S = \emptyset$ and $\boldsymbol{b}$ still demands all goods in $I_{\boldsymbol{b}}(\boldsymbol{p}')$, so $I_{\boldsymbol{b}}(\boldsymbol{p}') \subseteq I_{\boldsymbol{b}}(\boldsymbol{p})$. In this case, we set $\tau(\boldsymbol{b}, \cdot) = \pi'(\boldsymbol{b}, \cdot)$. As $\boldsymbol{b}$ is only allocated goods not in S, and $p_i = p'_i$ for all goods $i \in [n]_0 \setminus S$, it follows that $\boldsymbol{b}$ spends the same at both prices.

Note that, in both cases, $\boldsymbol{b}$ is only allocated goods that it demands. To summarise, we define τ as

$$\tau(\boldsymbol{b}, \cdot) = \begin{cases} \pi(\boldsymbol{b}, \cdot) & \text{if } \boldsymbol{b} \text{ demands a subset of } S \text{ at } \boldsymbol{p}, \\ \pi'(\boldsymbol{b}, \cdot) & \text{otherwise.} \end{cases}$$

We now prove that τ is valid. We have already argued above that all bids satisfy the demand and budget conditions of Definition 6.1. It remains to show that aggregate demand does not exceed supply s_i for any goods $i \in [n]_0$ under τ. Note that a bid is allocated a good i in S under τ if, and only if, it satisfies Case 1 above. Indeed, in this case we set $\tau(\boldsymbol{b}, i) = \pi(\boldsymbol{b}, i)$. Hence, for any $i \in S$, we have $\sum_{\boldsymbol{b} \in \mathcal{B}} \tau(\boldsymbol{b}, i) \leq \sum_{\boldsymbol{b} \in \mathcal{B}} \pi(\boldsymbol{b}, i) \leq s_i$. Similarly, for any $i \notin S$ the bid will satisfy Case 2, and we get $\sum_{\boldsymbol{b} \in \mathcal{B}} \tau(\boldsymbol{b}, i) \leq \sum_{\boldsymbol{b} \in \mathcal{B}} \pi'(\boldsymbol{b}, i) \leq s_i$.

Finally, we see that τ achieves weakly greater revenue at $\boldsymbol{p}$ than π' does at $\boldsymbol{p}'$. To see this, note that each bid satisfying Case 1 spends its entire budget on non-reject goods and thus contributes a weakly greater amount to revenue at $\boldsymbol{p}$ than at $\boldsymbol{p}'$, while a bid satisfying Case 2 contributes the same amount to revenue. □

Corollary 6.12. *Revenue is maximised at element-wise minimal feasible prices $\boldsymbol{p}^*$.*

Proof. Suppose $\boldsymbol{p} \neq \boldsymbol{p}^*$ is a revenue-maximising price vector. As $\boldsymbol{p}^*$ is element-wise minimal, we have $\boldsymbol{p}^* \leq \boldsymbol{p}$. Then by Lemma 6.11, we can obtain weakly greater revenue at $\boldsymbol{p}^*$ than at $\boldsymbol{p}$, or $R(\boldsymbol{p}^*) \geq R(\boldsymbol{p})$. But as $\boldsymbol{p}$ maximises R, so does $\boldsymbol{p}^*$. □

6.7 Market-clearing prices are unique

We now turn from revenue to competitive equilibrium. First we show that market-clearing prices, the first component of a competitive equilibrium, are unique in our arctic auction. This mirrors the case of Fisher markets, which also admit unique market-clearing prices. In the next section, we will show that these prices are the element-wise minimal prices $\boldsymbol{p}^*$ before providing an algorithm for finding $\boldsymbol{p}^*$ in Section 6.9.

Theorem 6.13. *There is a unique price vector $\boldsymbol{p}^*$ at which the market is cleared.*

Proof. Suppose we have a valid allocation π that exhausts supply $\boldsymbol{s}$. We want to find the corresponding prices $\boldsymbol{p}$ that support π. First, we formulate the problem of finding $\boldsymbol{p} \geq 0$ as a system of linear inequalities by writing out the budget constraints as equalities and the demand constraints as inequalities. The latter is achieved by

adding, for every bid $\boldsymbol{b}$ that demands a positive amount of good i, the inequalities $\frac{b_i}{p_i} \geq \frac{b_j}{p_j}, \forall j \in [n]_0$. Our goal is to prove that there is exactly one solution to this system of inequalities. In Section 6.8, we show that the element-wise minimal prices $\boldsymbol{p}^*$ clear the market, so the polytope defined by this system has at least one solution.[3]

For contradiction, suppose there is a second feasible price vector $\boldsymbol{p} \geq \boldsymbol{p}^*$ that also clears the market. Then by convexity of the polytope, it follows that $\boldsymbol{p}' = \boldsymbol{p}^* + \varepsilon(\boldsymbol{p} - \boldsymbol{p}^*)$ is also market-clearing for any $0 \leq \varepsilon \leq 1$. From now on, we assume that $\boldsymbol{p}'$ is obtained by letting ε be infinitesimally small. Let $S := \{i \in [n] \mid p_i^* < p_i\}$ denote the prices that change when we move from $\boldsymbol{p}^*$ to $\boldsymbol{p}'$. As $\boldsymbol{p} \neq \boldsymbol{p}^*$, S is non-empty. We first make a technical observation about the demand of each bid at $\boldsymbol{p}^*$ and $\boldsymbol{p}'$.

Observation 6.14. *For any bid $\boldsymbol{b} \in \mathcal{B}$, we note: (i) If $I_{\boldsymbol{b}}(\boldsymbol{p}^*) \not\subseteq S$, then $I_{\boldsymbol{b}}(\boldsymbol{p}') = I_{\boldsymbol{b}}(\boldsymbol{p}^*) \setminus S$. (ii) If $I_{\boldsymbol{b}}(\boldsymbol{p}^*) \subseteq S$, then $I_{\boldsymbol{b}}(\boldsymbol{p}') \subseteq I_{\boldsymbol{b}}(\boldsymbol{p}^*)$. This holds because ε was chosen to be infinitesimally small.*

Now suppose e' is a valid expenditure at $\boldsymbol{p}'$ that clears the market. Let $\mathcal{B}_S := \{b \in \mathcal{B} \mid I_{\boldsymbol{b}}(\boldsymbol{p}^*) \subseteq S\}$ denote the set of bids satisfying case (ii) of Observation 6.14. We take a look at the revenue contributions from each bid towards goods in S. At $\boldsymbol{p}'$, Observation 6.14 tells us that only the bids in $\mathcal{B}_S$ spend on goods in S, and they each spend their entire budget on S. Hence, the budget and supply feasibility of e' imply $\beta(\mathcal{B}_S) = e'(\mathcal{B}_S, S) = e'(\mathcal{B}, S) = \sum_{i \in S} p_i' s_i$. At $\boldsymbol{p}^*$, the bids of $\mathcal{B}_S$ also spend their entire budget on S, but it may be the case that other bids also spend part of their budget on S. Hence, $\beta(\mathcal{B}_S) = e(\mathcal{B}_S, S) \leq e(\mathcal{B}, S) = \sum_{i \in S} p_i^* s_i$. It follows that $\sum_{i \in S} p_i' s_i \leq \sum_{i \in S} p_i^* s_i$, in contradiction to the fact that $p_i' > p_i^*$ for all $i \in S$. □

Corollary 6.15. *The element-wise minimal prices $\boldsymbol{p}^*$ uniquely induce an efficient allocation and a competitive equilibrium.*

Proof. This is immediate from the welfare theorems and Theorem 6.13. □

6.8 Minimal feasible prices clear the market

Suppose $\boldsymbol{p}$ is a feasible price vector at which no allocation exhausts supply $\boldsymbol{s}$ in all goods. In this case, we present a method to scale a subset of goods uniformly in price by a factor $0 < c < 1$ while retaining feasibility and increasing aggregate demand. As stated in Corollary 6.18, this implies our main result that the element-wise minimal prices $\boldsymbol{p}^*$ clear the market. Our approach bears similarities to a price-increase procedure for solving Fisher markets sketched out in (Adsul et al., 2012).

In order to describe our method of scaling prices, we first introduce the demand and expenditure graphs. Then we present a subroutine that – given prices $\boldsymbol{p}$, an expenditure e and an unsaturated good i under e at $\boldsymbol{p}$ – either returns scaled-down

[3] Indeed, the algorithm in Section 6.9 finds these market-clearing prices.

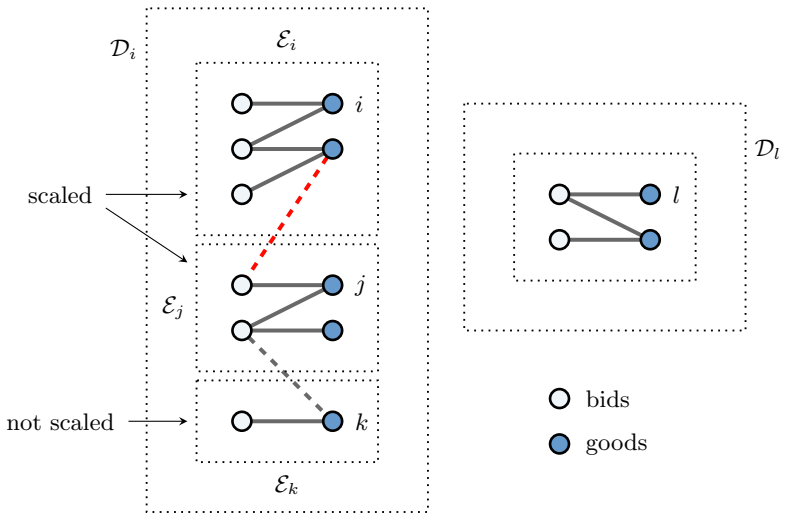


Figure 6.1: An illustration of the demand and expenditure graphs. Solid and dashed lines respectively indicate edges in $\mathcal{E}$ and $\mathcal{D} \setminus \mathcal{E}$. The component $\mathcal{D}_i$ contains three components of $\mathcal{E}$. If we root $\mathcal{D}_i$ at i, we note that there is an arc (dashed red) from a good in $\mathcal{E}_i$ to a bid in $\mathcal{E}_j$, and an arc (dashed grey) from a bid in $\mathcal{E}_j$ to a good in $\mathcal{E}_k$. Hence the tree constructed in Section 6.8.2, which contains all goods to be scaled down in price, consists of $\mathcal{E}_i$ and $\mathcal{E}_j$ but not $\mathcal{E}_k$.

prices or an expenditure at $\boldsymbol{p}$ under which i is saturated. After invoking this subroutine $O(n)$ times, prices will be scaled down. We note that the subroutine defined here is also called in our algorithm (cf. Section 6.9) to find the unique market-clearing prices $\boldsymbol{p}^*$.

6.8.1 The demand and expenditure graphs

In order to describe demand and spending relationships between bids and goods, we introduce two bipartite graphs, the *demand graph* and the *expenditure graph*. Modulating expenditures in appropriately defined sub-trees of these graphs is the central component in our price reduction procedure.

Definition 6.16 (Demand graph and expenditure graph). Let e be a valid expenditure at feasible prices $\boldsymbol{p}$. We denote by $\mathcal{D}$ the bipartite *demand graph* on vertex sets $\mathcal{B}$ and $[n]_0$. There is an edge $(\boldsymbol{b}, i) \in D$ if, and only if, $\boldsymbol{b}$ demands i at $\boldsymbol{p}$. Moreover, we define the *expenditure graph* $\mathcal{E}$ of $\mathcal{D}$ induced by the edges $(\boldsymbol{b}, i)$ with positive expenditure $e(\boldsymbol{b}, i) > 0$. Finally, $\mathcal{D}_i$ and $\mathcal{E}_i$ respectively denote the connected component of $\mathcal{D}$ and $\mathcal{E}$ that contains good i.

Note that $\mathcal{E}_i$ is a subgraph of $\mathcal{D}_i$ for all goods $i \in [n]_0$, and each $\mathcal{D}_i$ can contain multiple connected components of $\mathcal{E}$. Both graphs are illustrated in Fig. 6.1.

Breaking cycles. Our price reduction method assumes that the demand graph is acyclic. For this reason, we follow Orlin (2010) in perturbing bids by an infinitesimal amount in order to break cycles. Induce an ordering $\mathcal{B} = \{\boldsymbol{b}^1, \boldsymbol{b}^2, \ldots\}$ on the bids,

Algorithm 6.1 Tree-based price reduction

Input: Feasible prices $\boldsymbol{p}$, valid expenditure e and unsaturated good i.

Output: Prices $\boldsymbol{p}' \leq \boldsymbol{p}$ and expenditure e', where either $\boldsymbol{p}' = \boldsymbol{p}$ and i is saturated, or $p'_j < p_j$ is for some goods $j \in [n]$.

1: Construct tree T by rooting $\mathcal{D}_i$ at i and removing subtrees rooted at goods j of arcs $(\boldsymbol{b}, j) \in \mathcal{D}_i \setminus \mathcal{E}$.
2: **if** $0 \notin T$ **then** go to step 9. **end if**
3: Compute the path $i = g^1, \boldsymbol{b}^1, \ldots, g^k, \boldsymbol{b}^k, g^{k+1} = 0$ from i to 0 in T and the minimum m of $p_i s_i - e(\mathcal{B}, i)$ and $\min_{j \in [k]} e(g^j, \boldsymbol{b}^j)$. Update e by $e(\boldsymbol{b}^j, g^j) \mathrel{+}= m$ and $e(\boldsymbol{b}^j, g^{j+1}) \mathrel{-}= m$.
4: **if** good i is now saturated under e **then**
5: **return** $\boldsymbol{p}$ and e.
6: **else**
7: Let j be the smallest such value $j \in [k]$ and remove from T the subtree rooted at g^{j+1}.
8: **end if**
9: Traverse T depth-first to recursively compute $\sum_{k \in T_v} \alpha_k p_k s_k$ for each vertex $v \in T$.
10: Compute $c^{(1)}, c^{(2)}$ and $c^{(3)}$ as specified in Section 6.8.2, and scaling factor $c^* = \max\left\{c^{(1)}, c^{(2)}, c^{(3)}\right\}$.
11: Traverse T depth-first to recursively compute $\Delta(\boldsymbol{b}, j)$ for each arc in T. Compute e' as described in (6.7).
12: Let $p'_j = c^* p_j$ for all $j \in T$ and $p'_j = p_j$ otherwise.
13: **return** $\boldsymbol{p}'$ and e'.

and perturb each entry b_i^k by adding $\varepsilon^{kn} + \varepsilon^i$ for some infinitesimally small $\varepsilon > 0$. In our algorithms, this perturbation can be simulated without a running-time penalty by implementing lexicographic tie-breaking when constructing the demand and expenditure graphs. For details about the perturbation, we refer to (Orlin, 2010).

6.8.2 Reducing prices along a tree

We now describe the subroutine introduced above. The full procedure is stated in Algorithm 6.1 and elucidated below. Correctness is proved in Theorem 6.17.

Identifying goods to reduce in price. Suppose i is an unsaturated good under expenditure e at $\boldsymbol{p}$. We describe a procedure which either identifies a set J of goods to reduce in price, or finds a new expenditure e' at $\boldsymbol{p}$ under which good i is saturated. We first construct a directed tree T by rooting the (undirected) tree $\mathcal{D}_i$ at good i and then remove all subtrees rooted at endpoint goods j of arcs $(\boldsymbol{b}, j) \in \mathcal{D}_i \setminus \mathcal{E}$. This is illustrated in Fig. 6.1. For notational convenience, we also let T_v denote the subtree of T rooted at v for any vertex $v \in T$.

If T does not contain the reject good 0, we let J consist of the goods of T and are done. Otherwise, we consider the directed path $(i = g^1, \boldsymbol{b}^1, \ldots, g^k, \boldsymbol{b}^k, g^{k+1} = 0)$

from i to 0 in T and modify the expenditure e to redirect spending from 0 to i as follows. Let $m > 0$ be the minimum of $p_i s_i - e(\mathcal{B}, i)$ and $\min_{j\in[k]} e(\boldsymbol{b}^j, g^j)$. We let $e'(\boldsymbol{b}^j, g^j) = e(\boldsymbol{b}^j, g^j) + m$ and $e'(\boldsymbol{b}^j, g^{j+1}) = e(\boldsymbol{b}^j, g^j) - m$ for all $j \in [k]$. Note that the resulting expenditure is valid and redirecting spending in this way does not add edges to the expenditure graph, or cause the expenditure on any goods in $[n]$ to reduce. In particular, no saturated good becomes unsaturated.

If $m = p_i s_i - e(\mathcal{B}, i)$, then good i is saturated under the new expenditure e', and we are done. Otherwise, if $m < p_i s_i - e(\mathcal{B}, i)$, we obtain the tree corresponding to the new expenditure e' by identifying the smallest $j \in [k]$ such that $e'(\boldsymbol{b}^j, g^{j+1}) = 0$ and removing the subtree rooted at g^{j+1} from T. As the tree no longer contains the reject good 0, we let J denote the goods in T and are done.

Scaling prices. Now we suppose that we have identified the set of goods J of the tree T not containing 0, and good i is unsaturated. In order to fully specify our scaling method, it remains to determine the factor $0 < c < 1$ by which to uniformly scale the prices of goods in J so that the resulting prices remain feasible. Let $\boldsymbol{p}'$ denote the prices after scaling, so $p'_j = cp_j$ for $j \in J$ and $p'_j = p_j$ otherwise. We proceed by describing an expenditure e' at $\boldsymbol{p}'$ that is derived from e by adding, or removing, a difference term parameterised by c to, or from, each $e(\boldsymbol{b}, j)$. The scaling factor c is then chosen to ensure that e' is valid. Recall that e' needs to satisfy the conditions in Definition 6.1 to be a valid expenditure at prices $\boldsymbol{p}'$. We recall, for instance, that the supply constraint imposes $e'(\mathcal{B}, j) \leq p'_j s_j$ for all goods $j \in [n]$. In the following, we will impose a stronger supply constraint for all goods $j \in [n] \setminus \{i\}$. Suppose that the proportion of good j aggregately demanded under e is given by $\alpha_j \in [0, 1]$, so that $e(\mathcal{B}, j) = \alpha_j p_j s_j$ for good j. Then we require that $e'(\mathcal{B}, j) = \alpha_j p'_j s_j$ for all $j \in [n] \setminus \{i\}$. (So $e'(\mathcal{B}, j) = c\alpha_j p_j s_j$ for $j \in J$ and $e'(\mathcal{B}, j) = \alpha_j p_j s_j$ for $j \notin J$.) Note that this trivially implies the supply constraint, as $\alpha_j \leq 1$ and $c < 1$.

The stronger supply condition uniquely determines an expenditure e' for any given c, which can be expressed by adding, or subtracting, the differential terms $\Delta(\boldsymbol{b}, j)$ to, or from, $e(\boldsymbol{b}, j)$. For every arc in T between some bid $\boldsymbol{b}$ and good j with endpoint $v \in \{\boldsymbol{b}, j\}$, we define $\Delta(\boldsymbol{b}, j) := (1 - c) \sum_{k\in T_v} \alpha_k p_k s_k$. For all other $\boldsymbol{b}$ and j, we let $\Delta(\boldsymbol{b}, j) = 0$. Finally, we define the new expenditure e' as

$$e'(\boldsymbol{b}, j) = \begin{cases} e(\boldsymbol{b}, j) - \Delta(\boldsymbol{b}, j), & \text{if } \boldsymbol{b} \text{ precedes } j \text{ in } T, \\ e(\boldsymbol{b}, j) + \Delta(\boldsymbol{b}, j), & \text{otherwise.} \end{cases} \tag{6.7}$$

It remains to find the smallest $0 < c < 1$ such that e' remains valid. Conceptually, we choose c by reducing from 1 until one of three scenarios occurs. For each scenario i, we determine the scaling factor $c^{(i)}$ at which this scenario occurs. Our scaling factor is then chosen by computing $c^* = \max\{c^{(1)}, c^{(2)}, c^{(3)}\}$. The scenarios are as follows.

1. Spending $e(\mathcal{B}, i)$ on good i increases until it becomes saturated. This strictly reduces the number of unsaturated goods in $\mathcal{D}$.

2. Some $e'(\boldsymbol{b}, j)$ decreases to 0. This breaks up a component in $\mathcal{E}$.

3. Some bid not in T starts demanding a good in T (along with goods outside T). This connects two components in $\mathcal{D}$.

Scenario 1: We note that c must be chosen such that $e'(\mathcal{B}, i) = e(\mathcal{B}, i) + \Delta(\mathcal{B}, i) \leq cp_i s_i$. Substituting $\Delta(\mathcal{B}, i) = \sum_{k \in J \setminus \{i\}} (1-c)\alpha_k p_k s_k$ and solving for c, we see that this holds for scaling factor

$$c^{(1)} = \frac{e(\mathcal{B}, i) + \sum_{k \in J \setminus \{i\}} \alpha_k p_k s_k}{p_i s_i + \sum_{k \in J \setminus \{i\}} \alpha_k p_k s_k}. \tag{6.8}$$

As $0 \leq e(\mathcal{B}, i) \leq p_i s_i$, factor $c^{(1)}$ is well-defined and $0 < c^{(1)} < 1$.

Scenario 2: Note that $\Delta(\boldsymbol{b}, j) > 0$ for any scaling factor c that is strictly less than 1. Hence, $e'(\boldsymbol{b}, j)$ is non-negative for all arcs from a good j to a bid $\boldsymbol{b}$, for any $c < 1$. Now fix an arc from bid $\boldsymbol{b}$ to good j. In this case, we need to ensure that $e(\boldsymbol{b}, j) \geq \Delta(\boldsymbol{b}, j)$ to guarantee $e'(\boldsymbol{b}, j) \geq 0$. Substituting $\Delta(\boldsymbol{b}, j) = (1-c)\sum_{k \in T_j} \alpha_k p_k s_k$, and solving for c, we get

$$c \geq 1 - \frac{e(\boldsymbol{b}, j)}{\sum_{k \in T_j} \alpha_k p_k s_k}. \tag{6.9}$$

Hence $c^{(2)}$ as chosen below guarantees $e'(\boldsymbol{b}, j) \geq 0$ for every bid $\boldsymbol{b}$ and good j with equality for at least one pair $(\boldsymbol{b}, j)$.

$$c^{(2)} = \max_{(\boldsymbol{b}, j) \in T} \left(1 - \frac{e(\boldsymbol{b}, j)}{\sum_{k \in T_j} \alpha_k p_k s_k}\right). \tag{6.10}$$

Note that $0 < e(\boldsymbol{b}, j) \leq \alpha_j p_j s_j \leq \sum_{k \in T_j} \alpha_k p_k s_k$ for all $(\boldsymbol{b}, j) \in T$, so $c^{(2)}$ is well-defined and satisfies $0 < c^{(2)} < 1$.

Scenario 3: Fix a bid $\boldsymbol{b}$. Then $\boldsymbol{b}$ continues to demand goods outside T only if $\frac{1}{c} \max_{j \in T} \frac{b_j}{p_j} \leq \max_{k \notin T} \frac{b_k}{p_k}$. Hence, by solving for c and taking the maximum over all bids not in T, we get

$$c^{(3)} = \max_{\boldsymbol{b} \notin T} \left[\max_{j \in T} \frac{b_j}{p_j} \min_{k \notin T} \frac{p_k}{b_k}\right]. \tag{6.11}$$

Note that $1 \leq \max_{j \in T} \frac{b_j}{p_j} < \max_{k \in [n]_0} \frac{b_k}{p_k}$ for all $\boldsymbol{b} \notin T$, as these bids do not demand any goods in T at $\boldsymbol{p}$. Hence, $c^{(3)}$ is well-defined and $0 < c^{(3)} < 1$.

We now show correctness of our price-reduction routine.

Theorem 6.17. *Suppose $\boldsymbol{p}$ is not market-clearing, and e is a valid expenditure at $\boldsymbol{p}$ with an unsaturated good i. The price-reduction procedure above (Algorithm 6.1), when given $\boldsymbol{p}$, e and i, either returns a valid expenditure e' at $\boldsymbol{p}$ that sates good i, or it returns reduced prices $\boldsymbol{p}'$ and a valid expenditure e' at $\boldsymbol{p}'$, in polynomial time.*

Proof. Suppose first that the subroutine terminates in line 5. Then the resulting expenditure e' that it returns is trivially valid, and we are done.

Now suppose that the subroutine does not terminate in line 5. Then we note that $\boldsymbol{p}'$ returned by the procedure is strictly smaller than $\boldsymbol{p}$ for all goods $j \in J$,

as the scaling factor is strictly less than 1. Hence it suffices to show that the expenditure e' returned by Algorithm 6.1 after scaling prices of goods J by factor c^* is valid in the sense of Definition 6.1. By construction of c^*, $e'(\boldsymbol{b}, j)$ is non-negative for all bids $\boldsymbol{b}$ and goods j. We also note that all arcs in T coincide with edges in the demand graph by construction, and if $e(\boldsymbol{b}, j)$ changes, then there is an edge between $\boldsymbol{b}$ and j in the demand graph. Meanwhile, the choice of $c^{(3)}$ guarantees that all bids $\boldsymbol{b} \notin T$ demand, at $\boldsymbol{p}'$, a superset of the goods they demand at $\boldsymbol{p}$. It follows e is demand-valid.

Next, we verify that e is budget-feasible. Suppose first that $\boldsymbol{b}$ is a bid not in T. Then, by construction of T, none of the prices of goods that $\boldsymbol{b}$ demands are scaled, and the spending of $\boldsymbol{b}$ is unchanged. Next suppose that bid $\boldsymbol{b}$ is a leaf in T, and good j is its parent. Then $\Delta(\boldsymbol{b}, j) = \sum_{k \in T_{\boldsymbol{b}}} \alpha_k p_k s_k = 0$, as the tree $T_{\boldsymbol{b}}$ contains no goods. Hence, $e'(\boldsymbol{b}, j) = e(\boldsymbol{b}, j) = \beta(\boldsymbol{b})$ and the condition for budget feasibility holds. Finally, suppose bid $\boldsymbol{b}$ is not a leaf in T. Let g^1 be its parent and $g^2, \ldots, g^k$ be its children. We need to verify that $e'(\boldsymbol{b}, [n]_0) = \beta(\boldsymbol{b})$. As $e(\boldsymbol{b}, [n]_0) = \beta(\boldsymbol{b})$, it suffices to see that $-\Delta(\boldsymbol{b}, g^1) = \sum_{j=2}^{k} \Delta(\boldsymbol{b}, g^j)$. But this follows from the definition of Δ.

Next, we show that e is supply-feasible. For the originally unsaturated good i, this follows immediately from the choice of our scaling factor $c^* \geq c^{(1)}$ in (6.8). For all other goods, we verify the stronger supply condition introduced above. Suppose first that j is a good not in T. Then by construction of T, the amount that each bid spends on j remains unchanged. Next suppose good j is a leaf in T, and bid $\boldsymbol{b}$ is its parent. Then the stronger supply constraint requires that $e'(\boldsymbol{b}, j) = ce(\boldsymbol{b}, j) = c\alpha_j p_j s_j$, which agrees with our definition of $\Delta(\boldsymbol{b}, j) = (1 - c)\alpha_j p_j s_j$. Finally, suppose good j is not a leaf in T, and let $\boldsymbol{b}$ be its parent and $\mathcal{B}'$ be its children. Then the stronger supply constraint is satisfied, as

$$e'(\mathcal{B}, j) = e(\mathcal{B}, j) - \Delta(\boldsymbol{b}, j) + \Delta(\mathcal{B}', j) = \alpha_j p_j s_j - (1 - c)\alpha_j p_j s_j = c\alpha_j p_j s_j.$$

To see that the subroutine runs in polynomial time, note that the running time is dominated by traversing the tree with $n + |\mathcal{B}|$ nodes depth-first, which is polynomial in n and $|\mathcal{B}|$. □

6.8.3 The procedure for reducing prices

Recall that, given prices $\boldsymbol{p}$ that are not market-clearing, and an expenditure e, the subroutine described above either modifies e in order to ensure that an unsaturated good becomes saturated, or reduces prices of some goods. We claim that after $O(n)$ repetitions of this subroutine with a new unsaturated good, prices of some goods are reduced. Indeed, we start with at most n unsaturated goods. Each application of the subroutine that fails to reduce prices strictly reduces the number of unsaturated goods, as no saturated goods can revert to being unsaturated. In particular, this implies that the element-wise minimal prices $\boldsymbol{p}^*$ clear the market.

Corollary 6.18. *Competitive equilibrium is uniquely achieved at element-wise minimal prices* $\boldsymbol{p}^*$.

Algorithm 6.2 Finding element-wise minimal feasible prices $\boldsymbol{p}^*$

Input: List of bids $\mathcal{B}$ in general position.
Output: Element-wise smallest feasible prices $\boldsymbol{p}'$.

1: Let $\boldsymbol{p}$ be the element-wise maximum over the bids in $\mathcal{B}$ and e be the expenditure at which all bids spend their entire budget on the reject good 0.
2: **while** there are unsaturated goods **do**
3: Identify an unsaturated good i under e at $\boldsymbol{p}$.
4: **while** good i is unsaturated **do**
5: Apply tree-based subroutine stated in Algorithm 6.1 with good i to update $(\boldsymbol{p}, e)$.
6: **end while**
7: **end while**
8: **return** prices $\boldsymbol{p}$ and expenditure e.

Proof. In Section 6.7, we prove that market-clearing prices are unique in our market. Suppose that $\boldsymbol{p}^*$ is not market-clearing. Then by Theorem 6.17, and our discussion above, we can reduce prices of some goods. But this contradicts our assumption that $\boldsymbol{p}^*$ is the element-wise minimal feasible price vector. □

6.9 Finding market-clearing prices

Here we present an algorithm to find market-clearing prices, as well as a running time analysis that shows the algorithm terminates in exponential time. The algorithm uses the tree-based price scaling subroutine introduced in Section 6.8. Note that each call to Algorithm 6.1 makes progress in the sense that prices decrease or a good is saturated.

For a given bid list $\mathcal{B}$, let $\boldsymbol{p} = \bigvee_{\boldsymbol{b}\in\mathcal{B}} \boldsymbol{b}$ be the element-wise maximum over all bids. Then all bids demand the reject good, while possibly being indifferent between 0 and other goods. We define the expenditure e at $\boldsymbol{p}$ under which all bids spend their entire budget on the reject good, so for each bid $\boldsymbol{b} \in \mathcal{B}$ we let $e(\boldsymbol{b}, 0) = \beta(\boldsymbol{b})$ and $e(\boldsymbol{b}, i) = 0$ for all $i \in [n]$. This expenditure is trivially valid.

Of course, no market clears at this initial allocation. We then pick any unsaturated good i and call the subroutine of Algorithm 6.1 with good i at $(\boldsymbol{p}, e)$ to reduce prices until the market for good i clears (good i becomes saturated). Subsequently, we select another unsaturated good, reduce prices by repeatedly calling Algorithm 6.1, and repeat this step until all goods are saturated.

Theorem 6.19. *Algorithm 6.2 finds the market-clearing prices $\boldsymbol{p}^*$ with a running time of $O(n|\mathcal{B}|2^{n+1+|\mathcal{B}|})$.*

Proof. We argue that the number of calls to Algorithm 6.1 is bounded. Note that the outer loop in Algorithm 6.2 iterates $O(n)$ times. Indeed, invoking Algorithm 6.1 does not cause any saturated goods to become unsaturated. Every iteration of the outer loop reduces the number of unsaturated goods by one, and we start with at most n unsaturated goods.

We now show that the inner loop of Algorithm 6.2 applies Algorithm 6.1 $O(|\mathcal{B}|2^{n+1+|\mathcal{B}|})$ times before good i is saturated. Suppose, for contradiction, that the subroutine is called more often. In this case, we note that on each iteration, the scaling factor is chosen such that scenario 2, scenario 3 or both scenarios occur. Let $\mathcal{D}_i^{(k)}$ denote the connected component of $\mathcal{D}$ containing good i when scenario 2 occurs for the k-th time. In particular, we note that any two consecutive occurrences of scenario 2 can be separated by at most $|\mathcal{B}|$ iterations of scenario 3's. To see this, realise that every time only scenario 3 occurs, T grows through the addition of a bid, and we have $|\mathcal{B}|$ in total.

We now argue that all $\mathcal{D}_i^{(l)}$ are distinct. As there are $O(2^{n+1+|\mathcal{B}|})$ possible configurations of $\mathcal{D}_i$, this implies the result. Note that whenever the k-th iteration of scenario 2 occurs, we have $e'(\boldsymbol{b}, j) = 0$ for some arc $(\boldsymbol{b}, j)$, and good $j \in T$ is not in T in the next iteration. Hence, the price of good j is not scaled down in the next iteration. Meanwhile, the price of good i is scaled down on every iteration of Algorithm 6.1. Now suppose, for any k, that $\mathcal{D}_i^{(k)}$ reoccurs at a later stage. This implies that p_i and p_j have been scaled down the same; indeed, this follows by considering the path from i to j in $\mathcal{D}_i^{(k)}$ and applying the conditions of demand. But we have just argued that p_i and p_j have not been scaled down the same, a contradiction.

Finally, we see that the algorithm finds market-clearing prices. To see this, note that it terminates only once all goods are saturated. This concludes the proof. □

6.10 Conclusion and outlook

This chapter presents a new coincidence of social welfare and revenue for the arctic product-mix auction. In particular, we show that both objectives are achieved at 'special' element-wise minimal feasible prices $\boldsymbol{p}^*$. As the arctic auction has been studied only from the perspective of maximising revenue (Fichtl, 2020; Klemperer, 2010), our results form a novel contribution to the literature. The coincidence of revenue-optimality and efficiency makes our market compelling in theory, as well as highly attractive for sellers, buyers, and market platforms in practice.

The complexity of solving arctic auctions. In Section 6.9, we provided initial algorithmic results to compute the efficient and optimal price vector. This algorithm has not yet been implemented, and it is unknown whether it runs efficiently in practice. Indeed, a major open question is whether the version of the arctic auction (without seller costs) that we study here can be solved for social welfare (and thus revenue) in polynomial time.

In light of this question, it may be of interest to evaluate our algorithm theoretically and practically, although the apparent intractability of the algorithm of Adsul et al. (2012) with a conceptually similar price-scaling routine is discouraging from a theoretical perspective. Instead, a potential first step towards establishing membership of our problem in P is to apply the ellipsoid method, before finding algorithms that are viable in practice. Towards this latter goal, it may be of value

to investigate whether existing polynomial-time algorithms for Fisher markets (such as (Devanur and Vazirani, 2004; Devanur et al., 2008; Jain, 2007; Orlin, 2010)) can lead to insights for the arctic auction.

Moreover, the addition of seller costs (as permitted in the original arctic auction implemented for the Icelandic government and not studied here) changes the problem considerably: revenue is no longer maximised at market-clearing prices, and prices that maximise social welfare are no longer unique. (Recall that social welfare sums the utilities of all agents, and the seller's utility is now the difference between their revenue and their costs.) Given these observations, it is likely that new techniques are required to make the auction tractable with seller costs.

Increasing expressivity for bidders. Finally, we note that the arctic language currently only permits bids with positive weight, which restricts the expressivity of the language. Instead, we could allow arctic bids with negative budgets, and carefully define the demand correspondence of 'positive' and 'negative' bids accordingly (analogously to positive and negative bids in the tropical language). In this more general setting, we conjecture that the coincidence of social welfare and revenue continues to hold, but it is an open question whether finding the unique element-wise minimal prices remains tractable.

7

Concluding remarks

This thesis presents algorithms and complexity-theoretic results that shed light on how the tropical and arctic product-mix auctions can be implemented efficiently in practice, and illuminate the (in)tractability of the tropical product-mix bidding language with respect to learning and validating demand expressed in the language. We show that both learning and validating is computationally hard for the tropical bidding language in the general case but can be performed efficiently when the language is restricted. For both auctions, we propose algorithms that allow the auctioneer to solve the auction with the goal of maximising social welfare or maximising envy-free revenue. Notably, while we determine that these two objectives are computationally equivalent for the arctic auction, this does not hold for the tropical auction.

In the case of the tropical auction, Chapter 3 presents the first algorithm for finding a competitive equilibrium allocation (which, as the welfare theorems hold, also maximises social welfare) that makes direct use of the information provided by the tropical bids of all bidders, and runs in fully polynomial time (measured in the size of the bid list). Current implementations of the auction only permit bidders to submit preferences that can be encoded using positive bids, and for such preferences it is well-known that the auction can be solved efficiently using standard linear programming techniques. Allowing bidders to submit arbitrary strong-substitutes preferences by means of positive *and* negative bids required the development of a novel 'combinatorial' allocation algorithm; and we show that our algorithm is efficient in theory and in numerical simulations. In contrast, we show in Chapter 4 that the problem of finding envy-free revenue-maximising prices is APX-hard and provide a bound on the approximation constant, even when bidders are only permitted to submit significantly constrained preferences. Specifically, bidders are allowed to submit preferences that correspond to the aggregate of multiple bidders who each have unit-demand for at most two goods. This implies that there (probably) does not exist a polynomial-time algorithm for finding envy-free revenue-maximising prices.

Chapter 4 also presents super-polynomial algorithms for this objective, paving the way for possible future work on fixed-parameter tractability for this problem.

In the arctic auction, by contrast, the two objectives of welfare and revenue are computationally equivalent. Indeed, in Chapter 6 – which examines the related arctic auction under the objectives of revenue, welfare and competitive equilibrium – our main result is to establish a novel coincidence of these three objectives: the welfare theorems hold, and the unique market-clearing prices also maximise revenue. We also provide an algorithm to find the market-clearing prices. This coincidence result is surprising and, to the best of our knowledge, the first result of this kind; we do not know of any other non-trivial auction settings in which welfare and revenue coincide. Auctions, or markets, with this feature can be of particular interest in many practical settings, as they unify the priorities of both buyers (welfare) and sellers (revenue). For this reason, it is a particularly pertinent question to identify whether there exist other, more general, preferences and market settings that exhibit the same coincidence. Future work in this direction includes exploring the possibility of coincidence when bidders have preferences captured by 'valid' arctic bid lists with positive *and* negative budgets, in analogy to the positive and negative tropical bids.

The tropical bidding language utilised by bidders in the tropical auction in order to submit their preferences bears conceptual similarities to the structure of logical languages. In Chapter 2, we highlight these connections and show that the tropical language can be understood as a generalisation of the well-known language OXS extended to the setting in which multiple units of each good can be demanded. To the best of our knowledge, the tropical language is the only existing concise bidding language that captures precisely the class of SS demand. Nevertheless, the tropical language also creates computational challenges as we outline in Chapter 2. One result of ours, for instance, is that verifying the validity of a bidder's preferences expressed as positive and negative tropical bids is coNP-hard. This obstacle can be circumvented in practice by syntactically restricting the kind of bid lists a bidder is permitted to submit; for example, we describe a polynomial time algorithm for verifying validity when the number of negative bids to a constant. It is still an open question whether the verification problem is fixed parameter tractable in the number of negative bids. Moreover, it remains unknown whether other concise bidding languages that capture precisely SS demand can be designed that are either always valid by construction, or admit tractable verification algorithms for validity.

While the tropical and arctic product-mix bidding languages are intuitive geometrically, bidders may find it difficult to express non-trivial preferences in these, especially when the number of goods available is high. This motivates the procedures that we develop in Chapter 5 to facilitate agents in participating by systematically learning their preferences in terms of the tropical language. We present algorithms to do so assuming access to a demand or valuation oracle, both when demand can be expressed using only positive bids, and when it requires positive and negative bids. Complementing our algorithmic results, we also show super-polynomial lower bounds on the query complexity of computing the list of bids in the general case where bids may be positive and negative. The same lower bound construction technique can also be used to show an exponential lower bound

on learning an *invalid* bid list (that is, a list of tropical bids that does not encode a valid quasilinear demand correspondence). Moreover, while our lower bound holds for deterministic learning algorithms, the same techniques may also lead to lower bounds for randomised algorithms. To the best of our knowledge, this work is the first to study the tractability of expressing demand in a bidding language. Moreover, as our algorithms for learning demand first learn the polyhedral complex associated with the bidder's demand (cf. Section 1.4.3), our techniques may be of independent interest, and can be easily adapted to learning demands with indivisible and certain demand types with divisible goods (such as demand in the arctic auction). Extending the techniques used to learn strong-substitutes demand to broader classes such as ordinary substitutes or (limited) complements would be of particular interest for auction settings that wish to offer bidders more flexibility in their preferences.

Bibliography

B. Adsul, C. S. Babu, J. Garg, R. Mehta, and M. Sohoni. A simplex-like algorithm for linear Fisher markets. *Current Science*, 103:1033–1042, 2012.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

K. J. Arrow and G. Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, 22(3):265–290, 1954.

I. Ashlagi, M. Braverman, Y. Kanoria, and P. Shi. Clearing matching markets efficiently: informative signals and match recommendations. *Management Science*, 66(5):2163–2193, 2020.

L. M. Ausubel. An efficient ascending-bid auction for multiple objects. *American Economic Review*, 94:1452–1475, 12 2004.

L. M. Ausubel. An efficient dynamic auction for heterogeneous commodities. *American Economic Review*, 96:602–629, 6 2006.

E. Baldwin and P. Klemperer. Understanding preferences: "demand types", and the existence of equilibrium with indivisibilities. *Econometrica*, 87:867–932, 5 2019.

E. Baldwin and P. Klemperer. Proof that the product-mix auction bidding language can represent any substitutes preferences. preprint, 2021. URL `http://www.nuffield.ox.ac.uk/users/klemperer/subsproof.pdf`.

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

E. Baldwin, M. Bichler, M. Fichtl, and P. Klemperer. Strong substitutes: Structural properties, and a new algorithm for competitive equilibrium prices. under review, 2 2021a. URL `https://ideas.repec.org/p/nuf/econwp/2102.html`.

E. Baldwin, P. W. Goldberg, P. Klemperer, and E. Lock. Solving strong-substitutes product-mix auctions, 2021b. URL `https://arxiv.org/abs/1909.07313`.

M. Bichler. *The simultaneous multi-round auction format*, pages 85–95. Cambridge University Press, 2017.

M. Bichler and J. K. Goeree. *Handbook of spectrum auction design*. Cambridge University Press, 2017.

M. Bichler, M. Fichtl, and G. Schwarz. Walrasian equilibria from an optimization perspective: A guide to the literature. *Naval Research Logistics (NRL)*, 68(4): 496–513, 2021.

S. Bikhchandani and J. W. Mamer. Competitive equilibrium in an exchange economy with indivisibilities. *Journal of Economic Theory*, 74(2):385–413, 1997. ISSN 0022-0531. doi: https://doi.org/10.1006/jeth.1996.2269. URL `https://www.sciencedirect.com/science/article/pii/S0022053196922693`.

S. Bikhchandani and J. M. Ostroy. The package assignment model. *Journal of Economic Theory*, 107(2):377–406, 2002.

S. Bikhchandani, S. de Vries, J. Schummer, and R. V. Vohra. Linear programming and vickrey auctions. *IMA Volumes in Mathematics and its Applications*, 127: 75–116, 2001.

A. Blum, J. C. Jackson, T. Sandholm, and M. Zinkevich. Preference elicitation and query learning. *Journal of Machine Learning Research*, 5:649–667, 2004.

C. Boutilier and H. H. Hoos. Bidding languages for combinatorial auctions. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 1211–1217, 2001.

W. C. Brainard and H. E. Scarf. How to compute equilibrium prices in 1891. *American Journal of Economics and Sociology*, 64:57–83, 2005.

S. Branzei, Y. Chen, X. Deng, A. Filos-Ratsikas, S. K. S. Frederiksen, and J. Zhang. The fisher market game: Equilibrium and welfare. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 587–593. AAAI, 2014. URL `https://eprints.soton.ac.uk/402583/`.

R. Cassady. *Auctions and auctioneering*. University of California Press, 1967.

D. Chakrabarty, P. Jain, and P. Kothari. Provable submodular minimization via Fujishige-Wolfe's algorithm. *ArXiv e-prints*, 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*, pages 1220–1231. ACM, 2017.

N. Chen and X. Deng. Envy-free pricing in multi-item markets. *ACM Transactions on Algorithms (TALG)*, 10(2):1–15, 2014.

N. Chen, X. Deng, P. W. Goldberg, and J. Zhang. On revenue maximization with sharp multi-unit demands. *Journal of Combinatorial Optimization*, 31:1174–1205, 2016.

R. Colini-Baldeschi, S. Leonardi, and Q. Zhang. Revenue maximizing envy-free pricing in matching markets with budgets. In *International Conference on Web and Internet Economics*, pages 207–220, 2016.

R. Colini-Baldeschi, P. W. Goldberg, B. D. Keijzer, S. Leonardi, T. Roughgarden, and S. Turchetta. Approximately efficient two-sided combinatorial auctions. *EC 2017 - Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 591–608, 2017.

W. Conen and T. Sandholm. Preference elicitation in combinatorial auctions. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 256–259, 2001.

W. Conen and T. Sandholm. Differential-revelation VCG mechanisms for combinatorial auctions. In J. A. Padget, O. Shehory, D. C. Parkes, N. M. Sadeh, and W. E. Walsh, editors, *Agent-Mediated Electronic Commerce IV, Designing Mechanisms and Systems, AAMAS 2002 Workshop on Agent Mediated Electronic Commerce, Bologna, Italy, July 16, 2002, Revised Papers*, volume 2531, pages 34–51. Springer, 2002a.

W. Conen and T. Sandholm. Preference elicitation in combinatorial auctions. In *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy*, pages 168–169. ACM, 2002b.

V. Conitzer, C. Kroer, E. Sodomka, and N. E. Stier-Moses. Multiplicative pacing equilibria in auction markets. *arXiv preprint arXiv:1706.07151*, 2017.

V. Conitzer, C. Kroer, D. Panigrahi, O. Schrijvers, E. Sodomka, N. E. Stier-Moses, and C. Wilkens. Pacing equilibrium in first-price auction markets. In *Proceedings of the 2019 ACM Conference on Economics and Computation*. Association for Computing Machinery, 2019.

P. Cramton. How Best to Auction Natural Resources. Papers of peter cramton, University of Maryland, Department of Economics - Peter Cramton, 2009. URL `https://ideas.repec.org/p/pcc/pccumd/09anr.html`.

P. Cramton, Y. Shoham, and R. Steinberg. An overview of combinatorial auctions. *ACM SIGecom Exchanges*, 7(1):3–14, 2007.

P. Cramton et al. Spectrum auctions. *Handbook of telecommunications economics*, 1:605–639, 2002.

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

A. De, E. Mossel, and J. Neeman. Majority is stablest: Discrete and sos. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 477–486, 2013.

S. De Vries and R. V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on computing*, 15(3):284–309, 2003.

N. R. Devanur and V. V. Vazirani. The spending constraint model for market equilibrium: Algorithmic, existence and uniqueness results. In *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 519–528, 2004.

N. R. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani. Market equilibrium via a primal-dual algorithm for a convex program. *Journal of the ACM*, 55:1–18, 2008.

S. Dobzinski and J. Vondrák. Communication complexity of combinatorial auctions with submodular valuations. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1205–1215. SIAM, 2013.

B. C. Eaves. A finite algorithm for the linear exchange model. *Journal of Mathematical Economics*, 3:197–203, 1976.

M. Feldman, A. Fiat, S. Leonardi, and P. Sankowski. Revenue maximizing envy-free multi-unit auctions with budgets. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 532–549, 2012.

M. Fichtl. Computing candidate prices in budget-constrained product-mix auctions. note, 2020. URL `https://dss.in.tum.de/files/staff-files/fichtl/BudgetConstrainedPMA.pdf`.

M. Fichtl. On the expressiveness of assignment messages. *Economics Letters*, 208: 110051, 2021.

S. Finster. Strategic bidding in product-mix, sequential, and simultaneous auctions. working paper, 2020a. URL `https://www.nuffield.ox.ac.uk/economics/Papers/2020/2020W03_pma_strategic_5.pdf`.

S. Finster. Selling multiple complements with packaging costs. working paper, 2020b. URL `https://drive.google.com/file/d/18be3dAa0-A-lY7GtATgDCiGG2n2Bpfo8/view`.

S. Finster, P. Goldberg, and E. Lock. The arctic product-mix market: unifying revenue and welfare. working paper, 2021. URL `https://edwinlock.com/pdfs/arctic_auction.pdf`.

S. Friederich and R. Payne. Dealer Liquidity in an Auction Market: Evidence from the London Stock Exchange. *The Economic Journal*, 117(522):1168–1191, 7 2007.

Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1*, pages 548–553. Morgan Kaufmann Publishers Inc., 1999.

J. Garg, R. Mehta, M. Sohoni, and N. K. Vishnoi. Towards polynomial simplex-like algorithms for market equilibria. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1226–1242, 2013.

R. Garg and S. Kapoor. Auction algorithms for market equilibrium. *Mathematics of Operations Research*, 31(4):714–729, 2006.

P. W. Goldberg, E. Lock, and F. Marmolejo-Cossío. Learning strong substitutes demand via queries. In *International Conference on Web and Internet Economics*, pages 401–415, 2020.

F. Gul and E. Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87:95–124, 1999.

V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *SODA*, volume 5, pages 1164–1173, 2005.

J. Hasbrouck, G. Sofianos, and D. Sosebee. New York stock exchange systems and trading procedures. *Director*, 212(998):0310, 1993.

J. W. Hatfield and P. Milgrom. Matching with contracts. *American Economic Review*, 95:913–935, 9 2005.

P. Holmberg, K. Ruddell, and B. Willems. Multi-product supply function equilibria, 2019. URL `https://www.tse-fr.eu/sites/default/files/TSE/documents/conf/2019/Energy_Climate2019/willems_slides.pdf`. slides.

B. Hudson and T. Sandholm. Effectiveness of query types and policies for preference elicitation in combinatorial auctions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 386–393, 2004.

J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 7 2001.

K. Jain. A polynomial time algorithm for computing an Arrow-Debreu market equilibrium for linear utilities. *SIAM Journal on Computing*, 37:303–318, 2007.

M. Kaleta. Bidding languages for continuous auctions. In *New Trends in Databases and Information Systems*, pages 211–220. Springer, 2013.

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

S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37 (1):319–357, 2007.

P. Klemperer. A New Auction for Substitutes: Central Bank Liquidity Auctions, the U.S. TARP, and Variable Product-Mix Auctions. Nuffield College, 2008.

P. Klemperer. The product-mix auction: a new auction design for differentiated goods. *Journal of the European Economic Association*, 8:526–536, 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`.

C. Kroemer, M. Bichler, and A. Goetzendorff. (Un)expected bidder behavior in spectrum auctions: About inconsistent bidding and its impact on efficiency in the combinatorial clock auction. *Group Decision and Negotiation*, 25:31–63, 1 2016.

S. M. Lahaie and D. C. Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 180–188, 2004.

Y. T. Lee, A. Sidford, and S. C. W. Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. *56th IEEE Annual Symposium on Foundations of Computer Science*, pages 1049–1065, 2015.

B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55:270–296, 5 2006.

D. Lehmann, L. I. Oćallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM (JACM)*, 49:577–602, 2002.

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

F. M. Menezes and P. K. Monteiro. *An introduction to auction theory*. OUP Oxford, 2004.

G. Mikhalkin. Decomposition into pairs-of-pants for complex algebraic hypersurfaces. *Topology*, 43:1035–1065, 9 2004.

P. Milgrom. Auction theory. In *Advances in economic theory: Fifth world congress*, volume 1, page 32. Cambridge University Press Cambridge, 1987.

P. Milgrom. Assignment messages and exchanges. *American Economic Journal: Microeconomics*, 1:95–113, 2009.

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

H. Mizuno and S. Takahashi. An envy-free pricing algorithm for multi item market with covering constraints. In *8th International Congress on Advanced Applied Informatics, IIAI-AAI 2019*, pages 746–749. Institute of Electrical and Electronics Engineers Inc., 7 2019.

G. Monaco, P. Sankowski, and Q. Zhang. Revenue maximization envy-free pricing for homogeneous resources. *IJCAI International Joint Conference on Artificial Intelligence*, January:90–96, 2015.

K. Murota. *Discrete Convex Analysis: Monographs on Discrete Mathematics and Applications 10*. Society for Industrial and Applied Mathematics, USA, 2003. ISBN 0898715407.

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

K. Murota and A. Shioura. Exact bounds for steepest descent algorithms of L-convex function minimization. *Operations Research Letters*, 42:361–366, 7 2014.

K. Murota, A. Shioura, and Z. Yang. Computing a Walrasian equilibrium in iterative auctions with multiple differentiated items. In Siu-Wing, L. T.-W. C. Leizhen, and Cheng, editors, *Algorithms and Computation*, pages 468–478. Springer Berlin Heidelberg, 2013.

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.

N. Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 1–12. Association for Computing Machinery, 2000.

N. Nisan. Bidding languages for combinatorial auctions. In *Combinatorial Auctions*, pages 215–232. The MIT Press, 12 2005.

N. Nisan and I. Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129:192–224, 2006.

N. Nisan, T. Roughgarden, Éva Tardos, and V. V. Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

J. B. Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78:109–129, 1997.

J. B. Orlin. Improved algorithms for computing Fisher's market clearing prices. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 291–300, 2010.

M. Ostrovsky and R. Paes Leme. Gross substitutes and endowed assignment valuations. *Theoretical Economics*, 10:853–865, 9 2015.

R. Paes Leme. Gross substitutability: An algorithmic survey. *Games and Economic Behavior*, 106:294–316, 11 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*, pages 632–651. Society for Industrial and Applied Mathematics, 2017.

C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 542–547. Morgan Kaufmann Publishers Inc., 1999.

A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

A. Shioura. Fast scaling algorithms for M-convex function minimization with application to the resource allocation problem. *Discrete Applied Mathematics*, 134:303–316, 2004.

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*, 60:216–243, 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*, 58:61–103, 2015.

N. Sun and Z. Yang. Equilibria and indivisibilities: Gross substitutes and complements. *Econometrica*, 74(5):1385–1402, 2006.

N. Sun and Z. Yang. An efficient and incentive compatible dynamic auction for multiple complements. *Journal of Political Economy*, 122(2):422–466, 2014.

A. Tamura. Coordinatewise domain scaling algorithm for M-convex function minimization. *Mathematical Programming*, 102:339, 2004.

R. E. Tarjan and J. van Leeuwen. Worst-case analysis of set union algorithms. *Journal of the ACM*, 31:245–281, 3 1984.

A. Teytelboym. Gross substitutes and complements: A simple generalization. *Economics Letters*, 123(2):135–138, 2014.

A. Teytelboym. Natural capital market design. *Oxford Review of Economic Policy*, 35(1):138–161, 2019.

S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13:22–30, 3 2011.

H. R. Varian. Online ad auctions. *The American Economic Review*, 99(2):430–434, 2009.

L. Walras. *Éléments d'économie politique pure, ou, Théorie de la richesse sociale (Elements of Pure Economics, or the theory of social wealth)*. F. Rouge, 1896.

H. Zhang and V. Conitzer. Learning the valuations of a k-demand agent. In *International Conference on Machine Learning*, pages 11066–11075, 2020.

M. A. Zinkevich, A. Blum, and T. Sandholm. On polynomial-time preference elicitation with value queries. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 176–185, 2003.