Mathematical Programming (2020) 179:343–384
https://doi.org/10.1007/s10107-018-1334-9

FULL LENGTH PAPER

Series A

Computing Walrasian equilibria: fast algorithms and structural properties

Renato Paes Leme[1] · **Sam Chiu-wai Wong**[2]

Received: 24 July 2017 / Accepted: 20 September 2018 / Published online: 28 September 2018

Abstract

We present the first polynomial time algorithm for computing Walrasian equilibrium in an economy with indivisible goods and *general* buyer valuations having only access to an *aggregate demand oracle*, i.e., an oracle that given prices on all goods, returns the aggregated demand over the entire population of buyers. For the important special case of gross substitute valuations, our algorithm queries the aggregate demand oracle $\widetilde{O}(n)$ times and takes $\widetilde{O}(n^3)$ time, where n is the number of goods and the $\widetilde{O}(\cdot)$ notation denotes an asymptotic bound up to polylogarithmic factors. Both algorithms are randomized. At the heart of our solution is a method for exactly minimizing certain convex functions which cannot be evaluated but for which the subgradients can be computed. We also give the fastest known algorithm for computing Walrasian equilibrium for gross substitute valuations in the *value oracle model*. Our algorithm has running time $\widetilde{O}((mn+n^3)T_V)$ where T_V is the cost of querying the value oracle. A key technical ingredient is to regularize a convex programming formulation of the problem in a way that subgradients are cheap to compute. En route, we give necessary and sufficient conditions for the existence of *robust Walrasian prices*, i.e., prices for which each agent has a unique demanded bundle and the demanded bundles clear the market. When such prices exist, the market can be perfectly coordinated by solely using prices.

Keywords Walrasian equilibrium · Convex optimization · Gross substitutes

Mathematics Subject Classification 90C25 (Convex programming) · 91B26 (Market models)

A shorter conference version of this paper appeared in the ACM-SIAM Symposium on Discrete Algorithms (SODA17) and has been posted on arXiv (https://arxiv.org/abs/1511.04032). The conference version is attached to this submission for the records.

✉ Renato Paes Leme
renatoppl@google.com

Extended author information available on the last page of the article

1 Introduction

1.1 A macroscopic view of the market

As part of our everyday experience, prices reach equilibria in a wide range of economics settings. Yet, markets are complicated and consist of heterogeneous goods and a huge population of buyers can have very diverse preferences that are hard to model analytically. With the sheer amount of information needed to describe the economy, how can the market possibly reach an equilibrium? *Well, perhaps not all this information is needed.*

In this paper, we provide evidence supporting this belief through the lens of algorithms. Specifically, we propose algorithms for computing market equilibrium using very limited amount of information. Our result suggests that information theoretically, it is not necessary to make too many measurements or observations of the market to compute an equilibrium. This may also shed light into how markets operate.

As the first step, we must design a realistic model to represent the economy. The standard approach in Theoretical Computer Science would require the entire input be specified but for a market, it is simply too computationally expensive to model its individual agents in full details. So what should we turn to? If equilibrium represents the collective behavior of the agents, perhaps some kind of aggregate information would be enough. Such information can be average salaries, interest rate, population, fashion trend and so on. An algorithm would ideally process these *macroscopic-scale* information in an efficient manner to compute equilibrium price that allows the market to clear.

We show that it is possible to compute market equilibrium by exploiting the very rudimentary information of *aggregate demand*, i.e. the quantity demanded for each item at a given price aggregated over the entire population of buyers. This result implies, among other things, that a market can be viewed as an aggregate entity. For the sake of reaching equilibrium, detailed knowledge about its individual buyers at the microscopic level may not really be needed. Rather, it should be their collective behavior that dictates the outcome of the market.

The use of aggregate demand by our algorithm also resonates with a common perception of the role played by excess demand/supply. A highly sought-after good would usually see its price soar whereas an unpopular good would be inexpensive. This is similar to our algorithms which, in some sense, operate by increasing the price of overdemanded good and vice versa in an iterative fashion. We note however that by no means are we suggesting that our algorithms closely mirror how a market actually works. While the holy grail of this research direction is to understand how a market reaches equilibrium in practice, perhaps a humble first step is to show that this can be done algorithmically with as little information and assumption as possible.

Our starting point is the Gul and Stachetti's model [15] of an economy of (multi-unit) indivisible goods, but we make no further assumptions on the structure of the valuation functions. The goal is to compute market equilibrium: a set of item prices and allocations of items to buyers such that the market clears and each buyer gets his or her favorite bundle of goods under the current prices.

The market can only be accessed via an *aggregate demand oracle*: given prices for each item, what is the demand for each item aggregated over the entire population. Clearly in this model, it is not possible to compute an allocation of items to buyers, since the oracle access model allows no access to buyer-specific information. Curiously, equilibrium prices are still computable and in a very efficient manner:

Theorem 1 (Informal) *In a consumer market with n goods and m buyers, we can find a vector of equilibrium prices, whenever it exists, using* $\widetilde{O}(n^2)$ *calls to the aggregate demand oracle and* $\widetilde{O}(n^5)$ *time. If valuations are gross substitutes,* $\widetilde{O}(n)$ *calls to the aggregate demand oracle and* $\widetilde{O}(n^3)$ *time suffice.*

Notably, the number of buyers plays no role. Our algorithm has query and time complexity essentially independent of the number of buyers. This feature is especially relevant in practice as markets are usually characterized by a large population and relatively few number of goods. The city of Berkeley, for example, has about 350 restaurants but 120,000+ people!

1.2 From telescopes to augmenting lenses

Aggregate demand oracles are like looking at the economy from a telescope. Having a telescope has its advantages: it is possible to get a very global view of the economy with a few queries. On the other hand, extracting details is hard.

Our second question is how fast equilibrium can be computed with only a *local view of the economy*? Our analogue for augmenting lenses will be the *value oracle model*, in which one can query the value of each buyer for each bundle of items. This again has its advantages: it provides very fine-grained information about the market, but has the shortcoming that many queries are needed to extract any sort of global information.

Can equilibrium prices be computed with small amount of information even at the microscopic level? This quest is clearly hopeless for general valuation functions. But for one of the most important classes of valuation functions in economics, *gross substitute valuations*, there are enough structures to allow us to construct equilibrium prices using microscopic information.

The history of *gross substitutes* is intertwined with the development of theory of Walrasian equilibrium (also called market equilibrium in this paper). Indeed, Kelso and Crawford [21] show that Walrasian equilibrium always exists for gross substitute valuations. Hatfield and Milgrom [18] argue that most important examples of valuation functions arising in matching markets belong to the class of gross substitutes. Gross substitutes were used to guarantee the convergence of salary adjustment processes in the job market [21], to guarantee the existence of stable matchings in a variety of settings [25,43], to show the stability of trading networks [17,20], to design combinatorial auctions [1,30] and even to analyze settings with complementarities [16,46].

Since the oracle access is very local, we clearly need to query each agent at least once, so the dependence on the number of buyers must be at least linear. We show that indeed it is possible to solve this problem with a linear dependence on the number of buyers and cubic dependence in the number of items:

Theorem 2 (Informal) *In a consumer market with n goods and m buyers whose valuation functions satisfy the gross substitute condition, we can find an equilibrium (or Walrasian) price and allocation using* $mn + \widetilde{O}(n^3)$ *calls to the value oracle and* $\widetilde{O}(n^3)$ *time.*[1]

With buyer-specific information, we can also compute the optimal allocation in the same runtime.

Proving this result requires novel insights about the structure of gross substitute valuations. In particular, one of our main structural lemmas answers a question posed by Hsu et al. [19]: *when do prices coordinate markets?* In general, Walrasian prices mean that there is a choice of favorite bundles for each buyer that clears the market. It is far from trivial how to choose those bundles, since each agent can have various favorite bundles (for example, consider the case where all items are identical and all agents have the same valuation for the items). We say that a price vector form *robust Walrasian prices* if each agent demands a unique bundle under those prices and the bundles clear the market. Such vectors would allow prices alone to clear the market without any extra coordination. We show that:

Theorem 3 (Informal) *In a consumer market with n goods and m buyers whose valuation functions satisfy the gross substitute condition, robust Walrasian prices exist if and only if there is a unique Walrasian allocation (i.e. buyers get the same bundle in any Walrasian equilibrium). Whenever they exist, they can be computed in* $\widetilde{O}(mn+n^3)$ *time from the Walrasian allocation.*

1.3 Our algorithms and techniques

We study the Walrasian equilibrium problem in three different settings: (i) general valuations in the aggregate demand oracle model; (ii) gross substitute valuations in the aggregate demand oracle model and (iii) gross substitutes in the value oracle model. In all three settings, the starting point is the linear programming formulation of Bikhchandani and Mamer [4].

General valuations in the aggregate demand oracle model (Sect. 3) The main difficulty in working with the LP in Bikhchandani and Mamer [4] is that the constraints depend on the value of buyers for each bundle, to which we have no access to. In particular, we are not able to test for any given setting of variables of the LP if it is feasible or not. We are in a strange situation that if we knew that a point was infeasible, we could find an appropriate separating hyperplane, and if we knew it was feasible, we could use the objective function as a separating hyperplane, but since we can't test feasibility, we can't decide which to use. Our solution is to move the constraints to the objective function and turn the problem into an unconstrained convex minimization problem. The problem in hand has the feature that we can't evaluate the function but we can compute its subgradients.

Traditional cutting plane algorithms such as the Ellipsoid Method need access to both a separation oracle and functional values. To overcome this issue we use the

[1] The notation $g = \widetilde{O}(f)$ means that $g \leq \alpha f \log^{\beta}(f)$ for some constants $\alpha, \beta > 0$.

fact that the cutting plane of Lee et al. [27] provides strong dual guarantees and that our separation oracle is given by subgradients. Originally, [27] show how to adapt their cutting plane method to convex optimization, however, their algorithm and proof still rely on being able to evaluate the function. We show in our Theorem 6 that their algorithm can be slightly modified to achieve the same guarantee using only subgradients, (i.e., without using functional values).

A second obstacle we face is that algorithms to minimize convex functions only provide approximate guarantees and to find a Walrasian equilibrium we need the exact minimum. In general minimizing a convex function exactly is impossible, but in our case, this can be done by exploiting the connection to the LP. Note that given the very restricted way that we can access the problem (only via the aggregate demand oracle), we can't apply the techniques to perturb and round the linear programming in a black-box fashion. Khachiyan's approach [22] can be adapted to our setting, but can't be applied directly since we don't have a proper LP. We show how to perturb and round the objective function to achieve the desired running time. Our rounding is randomized and based on the Isolation Lemma.

Gross substitutes in the aggregate demand model (Sect. 4) If valuation functions satisfy gross substitutes, then we can exploit the fact that the set of Walrasian prices form an integral polytope with a lattice structure to simplify the algorithm and obtain an improved running time and oracle call complexity. The improvement comes from using structural properties of gross substitutes to show that a simpler and milder perturbation to the objective is enough and that rounding can be done in a simpler manner. This highlights the importance of looking at perturbation and rounding in a non-black-box manner.

Gross substitutes in the value oracle model (Sect. 5) An aggregate demand oracle call can be simulated from $O(mn^2)$ value oracles calls. This can be plugged into the previous algorithm to obtain a running time of $\tilde{O}(mn^3 T_V)$ where T_V is the time required by the value oracle. We use two ideas to improve the running time to $\tilde{O}((mn + n^3)T_V)$. The first one is to regularize the objective function. As with the use of regularizers in other context in optimization, this is to penalize the algorithm for being too aggressive. The bound of $O(mn^2)$ value oracle calls per iteration of the cutting plane algorithm is so costly precisely because we are trying to take an aggressively large step. A second idea is to re-use one of the stages of the subgradient computation in multiple iterations, amortizing its cost per iteration.

Robust Walrasian prices and market coordination (Sect. 6) Still in the value oracle model, we show how to obtain the efficient allocation from the subgradients observed in the optimization procedure. An important by-product of our analysis is that we give necessary and sufficient conditions for the existence of robust Walrasian prices, i.e., Walrasian prices under which each buyer has a unique bundle in their demand set. Whenever such prices exist, we give an $\tilde{O}(mn + n^3)$ algorithm to compute them. This answers an open question in Hsu et al. [19], who ask when it is possible to completely coordinate markets by solely using prices.

Combinatorial algorithms for Walrasian equilibrium (Sect. 7) Murota and Tamura [31] gave combinatorial algorithms for the problem of computing Walrasian equilibria via

a reduction to the M-convex submodular flow problem. It is also possible to obtain combinatorial algorithms for the welfare problem by reducing it to the valuated matroid intersection problem and applying the algorithms in Murota [33,34]. The running time is not explicitly analyzed in [31,33,34]. Here we describe those algorithms for the reader unfamiliar with M-convex submodular flows in terms of a sequence of elementary shortest path computations and analyze its running time. We show that they have running time of $\tilde{O}(mn^3)$ in the value oracle model. We use the same ideas used to speed up the computation of Walrasian prices by regularizing the market potential to speed up those algorithms and improve its running time to $\tilde{O}(mn + n^3)$.

1.4 Comparison to related work

Iterative auctions and subgradient algorithms The first algorithm for computing Walrasian equilibria in an economy of indivisible goods is due to Kelso and Crawford [21] and it is inspired by Walras' tâtonnement procedure [47], which means "trial-and-error". Despite the name, it constitutes a very ingenious greedy algorithm: goods start with any price, then we compute the aggregate demand of the agents, increase the price by one for all goods that were over-demanded and decrease by one the price of all goods that are under-demanded. This gives a very natural and simple algorithm in the aggregate demand oracle model. This algorithm, however, is not polynomial time since it runs in time proportional to the magnitude of the valuations.

The seminal work of [21] originated two lines of attack of the problem of computing Walrasian equilibria: the first line is by applying subgradient descent methods [1,41,42]. Such methods typically either only guarantee convergence to an approximate solution or converge in pseudo-polynomial time to an exact solution. This is unavoidable if subgradient descent methods are used, since their convergence guarantee is polynomial in M/ϵ where M is the maximum valuation of a buyer for a bundle and ϵ is the accuracy of the desired solution. To be able to round to an *exact* optimal solution the running time must depend on the magnitude of the valuations. Another family of methods is based on primal-dual algorithms. We refer to de Vries, Schummer and Vohra [11] for a systematic treatment of the topic. For primal-dual methods to converge exactly, they need to update the prices slowly—in fact, in [11] prices change by one unit in each iteration—causing the running time to be pseudo-polynomial time.

Polynomial time approaches via the Ellipsoid Method The Welfare Problem for gross substitutes was independently shown to be solvable in polynomial by Murota [34] and Nisan and Segal [40]. Remarkably, this was done using completely different methods.

Nisan and Segal's approach is based on a linear programming formulation of Walrasian equilibrium due to Bikhchandani and Mamer [4]. The authors show that the dual of this formulation can be solved using both the value and demand oracles for gross substitutes as a separation oracle for the LP. This can be combined with the fact that demand oracles for gross substitutes can be constructed from value oracles in $O(n^2)$ time [10] to obtain a polynomial-time algorithm in value oracle model.

This is the method that is closer to ours in spirit: since we both approach the problem via a mathematical programming formulation and apply interior point methods. In terms of oracle access, Nisan and Segal crucially rely on value oracles to implement

the separation oracle in their LP—so their solution wouldn't generalize to the aggregate demand oracle model, since neither per-agent demand oracles nor value oracles can be recovered from aggregate demand oracle.[2] The running time in their paper is never formally analyzed, but since their formulation has $m + n$ variables, it would lead to a superlinear dependence in the number of agents.

Nisan and Segal employ the LP to compute a set of Walrasian prices and the value of the Walrasian allocation. In order to compute the allocation itself, they employ a clever technique called *self-reducibility*, commonly used in complexity theory. While it allows for an elegant argument, it is a very inefficient technique, since it requires solving nm linear programs. In total, this would lead to a running time of $O(mn^2(m + n)^3)$ using currently fastest cutting plane algorithms as the LP solver.

Combinatorial approaches A second technique was developed by Murota [33,34] and leads to very efficient combinatorial algorithms. Murota's original paper never mentions the term "gross substitutes". They were developed having a different object in mind, called *valuated matroids*, introduced by Dress and Wenzel [12,13] as a generalization of the Grassmann–Plücker relations in p-adic analysis. Murota developed a strongly-polynomial time algorithm based on network flows for a problem called the *valuated matroids assignment problem*. There is a tortuous path connecting gross substitutes to valuated matroids. Valuated matroid turned out to be one aspect of a larger theory, Discrete Convex Analysis, developed by Murota (see his book [36] for a comprehensive discussion). One central object of this theory is the concept of $M^\natural$-concave functions, introduced by Murota and Shioura [29]. It came to many as a surprise when Fujishige and Yang [14] showed that $M^\natural$-concave functions and gross substitutes are the same thing. Their equivalence is highly non-trivial and their definitions are very different to the point it took at least a decade for those concepts to be connected. Murota and Tamura [31] later apply the ideas in discrete convex analysis to give polynomial time algorithms to various equilibrium problems in economics. The running time is never explicitly analyzed in their paper. Here we show that their running time is $\tilde{O}(mn^3)$ and improve it to $\tilde{O}(mn + n^3)$

Market coordination Related to Sect. 6 in our paper is the line of research on Market Coordination. This line of inquiry was initiated by Hsu et al. [19] who pose the question of when prices are enough to coordinate markets. They argue that the minimum vector of Walrasian prices always causes ties and bound how much overdemand can be caused by any given vector of Walrasian prices for unit-demand valuations and matroid based valuations. They ask such questions in both the traditional market model of Gul and Stachetti and in a stochastic setting where valuations are drawn from distributions. In the traditional market model, we answer the question in the title of their paper for the general class of gross substitutes, by giving necessary and sufficient conditions for markets to be coordinated by only using prices (Theorem 13). We also give a simple algorithm for computing those prices whenever they exist. Such necessary and sufficient conditions were given simultaneously and independently by Cohen-Addad et al. [7].

[2] Note that the construction of Blumrosen and Nisan [5] to construct value oracles from demand oracles crucially requires *per-buyer* demand oracles. The same construction doesn't carry over to aggregate demand oracles.

2 Preliminaries

Let $\mathbb{Z}$ be the set of integers, $\mathbb{Z}_{\geq 0}$ be the set of non-negative integers. For any $k \in \mathbb{Z}_{\geq 0}$, we define $[k] := \{1, 2, \ldots, k\}$ and $[\![k]\!] = \{0, 1, \ldots, k\}$. Given $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}^n_{\geq 0}$, we define $[\![\mathbf{s}]\!] = \prod_{j=1}^n [\![s_j]\!]$.

2.1 Market equilibrium: prices, allocations and welfare

Following the classic model of Gul and Stachetti [15], we define a *market of indivisible goods* as a set $[m]$ of buyers, a set $[n]$ of items and a supply $s_j \in \mathbb{Z}_{\geq 0}$ of each item j. Each buyer i has a valuation function $v_i : \mathbb{Z}^n_{\geq 0} \to \mathbb{Z}$ over the multisets of items with $v_i(0) = 0$. For the first part of our paper, we make no further assumptions about the valuation function or the supply.

Given a price vector $\mathbf{p} \in \mathbb{R}^n$, we define the *utility* of agent i for a bundle $x \in [\![\mathbf{s}]\!]$ under price vector $\mathbf{p}$ as: $u_i(x; \mathbf{p}) := v_i(x) - \mathbf{p} \cdot x$, where $\mathbf{p} \cdot x$ refers to the standard dot product $\sum_{j=1}^n p_j x_j$. Notice also that we make no assumptions about the signs of v_i and $\mathbf{p}$.

An *allocation* $\mathbf{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(m)})$ is simply an assignment of items to each agent where i receives $x^{(i)} \in [\![\mathbf{s}]\!]$. An allocation $\mathbf{x}$ is *valid* if it forms a partition of the items, i.e. $\sum_i x^{(i)}_j = s_j$ for every j. The *social welfare* of a valid allocation $\mathbf{x}$ is defined as $\text{Sw}(\mathbf{x}) = \sum_{i \in [m]} v_i(x^{(i)})$. Finally, the optimal social welfare is simply the largest possible social welfare among all valid allocations.

Given prices $\mathbf{p} \in \mathbb{R}^n$, we would expect a rational agent to buy x such that his utility $v_i(x) - \mathbf{p} \cdot x$ is maximized. We call x the demand of i under $\mathbf{p}$, as defined formally below. Note that there may be multiple utility-maximizing subsets.

Definition 1 *(Demand set)* Given prices $\mathbf{p} \in \mathbb{R}^n$ on each item, the demand set $D(v, \mathbf{p})$ for a valuation function v is the collection of vectors (bundles) for which the utility is maximized:

$$D(v, \mathbf{p}) := \arg\max_{x \in [\![\mathbf{s}]\!]} v(x) - \mathbf{p} \cdot x.$$

If v is the valuation function v_i of agent i, we also use the shorthand $D(i, \mathbf{p})$ as the demand set.

Definition 2 *(Equilibrium)* A *Walrasian equilibrium* (also called *competitive equilibrium*) consists of a price vector $\mathbf{p} \in \mathbb{R}^n$ and a valid allocation $\mathbf{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(m)})$ s.t. $x^{(i)} \in D(i, \mathbf{p}) \forall i$. We call $\mathbf{p}$ a *Walrasian/equilibrium price* and $\mathbf{x}$ a *Walrasian/equilibrium allocation* induced by $\mathbf{p}$.

In other words, a competitive equilibrium describes a state where items are sold in such a way that the total demand $\sum_i x^{(i)}_j$ for each item precisely meets its supply s_j, i.e. the market *clears*. The reason for the name *competitive* equilibrium is that it achieves the optimal social welfare. This is known as the first and second welfare theorems in economics.

Lemma 1 (First and second welfare theorems) *Let* $\mathbf{x}$ *be an equilibrium allocation induced by an equilibrium prices* $\mathbf{p}$. *Then* $\mathbf{x}$ *achieves the optimal social welfare. Moreover, if* $\mathbf{p}$ *is any Walrasian price and* $\mathbf{x}$ *is any optimal allocation, then the pair* $(\mathbf{p}, \mathbf{x})$ *form a Walrasian equilibrium.*

This nicely reduces social welfare maximization to finding competitive equilibrium whenever it exists. Note that the definition of social welfare has nothing to do with prices. In a way, this lemma shows that equilibrium prices act as a certificate for optimality of equilibrium allocation.

2.2 Oracles

To study market equilibrium computationally, we must clarify how we can extract information about the market. In this paper we consider three models that access the market in different scales:

Microscopic scale: value oracle model Here the algorithm has access to the value that each agent has for each bundle. This gives the algorithm very fine-grained information, but may require many calls for the algorithm to access any sort of macroscopic information about the market.

Definition 3 *(Value oracle)* The value oracle for agent $i \in [m]$ takes $x \in [\![\mathbf{s}]\!]$ as input and outputs $v_i(x)$. We denote by T_V the time spent by the value oracle to answer a query.

Agent scale: demand oracle model Here the algorithm presents a price vector $\mathbf{p}$ to an agent and obtains how many units are demanded at $\mathbf{p}$. At any given price, the agent could be indifferent between various bundles. The demand oracle can return an arbitrary bundle.

Definition 4 *(Demand oracle)* The demand oracle for agent i takes as input a price vector $\mathbf{p} \in \mathbb{R}^n$ and outputs a demand vector $d_i(\mathbf{p}) \in D(i, \mathbf{p})$. We denote by T_D the time spent by the demand oracle to answer a query.

Macroscopic (market) scale: aggregate demand oracle model The *aggregate demand oracle model* presents a macroscopic view of the market where algorithms cannot observe individual agents but only their aggregate response to any given price $\mathbf{p}$.

Definition 5 *(Aggregate Demand oracle)* The aggregate demand oracle takes as input a price vector $\mathbf{p} \in \mathbb{R}^n$ and outputs a demand vector $d(\mathbf{p}) \in \mathbb{Z}_{\geq 0}$ such that there exist bundles $x^{(i)} \in D(i, \mathbf{p})$ satisfying $d(\mathbf{p}) = \sum_i x^{(i)}$. We denote by T_{AD} the time spent to answer a query.

3 Walrasian equilibrium for general valuations in $\tilde{O}(n^2 \cdot T_{AD} + n^5)$

We show that in the aggregate demand oracle model, whenever a Walrasian equilibrium exists, it can be computed using $\tilde{O}(n^2)$ aggregate demand oracles calls and $\tilde{O}(n^2 \cdot$

$T_{AD} + n^5$) time. We emphasize that our result is in the *aggregate* demand oracle model—which is the typical information available to markets: *which goods are under- and over-demanded by the population of buyers?* Previous polynomial-time algorithms for computing Walrasian equilibria [4,31,34,40] require buyer-level demand oracles or value oracles. Previous algorithms that use only aggregate demand oracles (such as [1, 11,15,21,42] and related methods based on ascending auctions) are pseudopolynomial since they depend linearly on the magnitude of the valuation functions.

First we discuss a linear programming formulation for this problem and a corresponding convex programming formulation. The formulation itself is fairly standard and appears in various previous work. When we try to find an *exact* minimizer of this formulation in polynomial time using only aggregate demand oracles, we encounter a series of obstacles. The main ones are: (i) how to optimize a function we cannot evaluate; and (ii) how to find an exact minimizer using convex optimization (which typically gives only an approximate guarantee). In both cases, we must apply optimization algorithms in a *non-black-box* manner and exploit special structures of the problem.

Linear programming formulation We start from the formulation of Bikhchandani and Mamer [4] (also studied by Nisan and Segal [40]) of the Walrasian equilibrium problem as a linear program. Consider the following primal-dual pair:

$$\begin{aligned}
&\max_{z} \sum_{i\in[m],x\in[\![\mathbf{s}]\!]} v_i(x)\cdot z_{i,x} \text{ s.t.}\\
&\sum_x z_{i,x} = 1, && \forall i \quad (u_i)\\
&\sum_{i,x} x_j \cdot z_{i,x} = s_j, && \forall j \quad (p_j)\\
&z_{i,x} \geq 0, && \forall i, x
\end{aligned}
\qquad\Bigg|\qquad
\begin{aligned}
&\min_{(\mathbf{p},\mathbf{u})} \sum_i u_i + \mathbf{p}\cdot\mathbf{s} \text{ s.t.}\\
&u_i \geq v_i(x) - \mathbf{p}\cdot x, \quad \forall i, x \qquad (z_{i,x})
\end{aligned}$$

Lemma 2 ([4]) *A market equilibrium* $(\mathbf{p}^{eq}, \mathbf{x})$ *exists iff the primal program has an integral solution. In such case, the set of equilibrium prices is the set of solutions to the dual LP projected to the* $\mathbf{p}$*-coordinate.*

We provide a proof of the previous lemma in the appendix for completeness. This lemma reduces the problem of finding a Walrasian equilibrium, whenever it exists, to the problem of solving the dual program. The approach in Nisan and Segal [40] is to use a (per buyer) demand oracle as a separation oracle for the dual program. Given that we care only about the $\mathbf{p}$ variables, we can reformulate the dual as:

$$\begin{aligned}
&\min_{(\mathbf{p},u)} u + \mathbf{p}\cdot\mathbf{s} \text{ s.t.}\\
&u \geq \sum_i v_i(x) - \mathbf{p}\cdot x, \quad \forall x \in [\![\mathbf{s}]\!].
\end{aligned} \tag{D}$$

It is simple to see that for every feasible vector $(\mathbf{p}, \mathbf{u})$ of the original dual, we can find a corresponding point $(\mathbf{p}, \sum_i u_i)$ of the transformed dual with the same value.

Conversely, given a feasible point $(\mathbf{p}, u)$ of the transformed dual, we can come up with a point $(\mathbf{p}, \mathbf{u})$ of the original dual with equal or better value by setting $u_i = \max_x v_i(x) - \mathbf{p} \cdot x$.

Thus it suffices to find an optimal solution to the transformed dual program. The separation problem is now simpler: consider a point $(\mathbf{p_0}, u_0)$ that is infeasible. If some constraint is violated, then it must be the constraint for $x^{(i)}$ maximizing $\sum_i v_i(x^{(i)}) - \mathbf{p}_0 \cdot x^{(i)}$, so $u_0 < \sum_i v_i(x^{(i)}) - \mathbf{p}_0 \cdot x^{(i)}$. For all feasible $(\mathbf{p}, u)$ we know that $u \geq \sum_i v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)}$. Therefore:

$$u - u_0 + (\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{d}(\mathbf{p}_0) \geq 0$$

is a valid separating hyperplane, where $\mathbf{d}(\mathbf{p}_0) = \sum_i x^{(i)}$ is the output of the aggregate demand oracle. If on the other hand $(\mathbf{p_0}, u_0)$ is feasible, we can use the objective function to find a separating hyperplane, since for any optimal solution $(\mathbf{p}, u)$ we know that $u + \mathbf{p} \cdot \mathbf{s} \leq u_0 + \mathbf{p}_0 \cdot \mathbf{s}$. So we can separate it using:

$$u - u_0 + (\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{s} \leq 0.$$

An obstacle The main obstacle to this approach is that since the aggregate demand oracle has no access to the value of $\sum_i v_i(x^{(i)})$, one cannot construct a separation oracle from the aggregate demand oracle.

Convex programming formulation One way to get around this obstacle is to further transform the dual program to get rid of utility variable u altogether. We do so by moving the constraints to the objective function in the form of penalties. The LP then becomes the problem of minimizing a convex function. The same function has been used as a potential function to analyze price adjustment procedures [2] and combinatorial algorithms [30] and is the basis for the family of algorithms known as subgradient algorithms (e.g. [1,6,41,42]).

Given a market with supply $\mathbf{s}$ and agents with valuations $v_1, \ldots, v_m$, we define the market potential function $f : \mathbb{R}^n \to \mathbb{R}$ as:

$$f(\mathbf{p}) = \sum_{i=1}^{m} \left(\max_{x \in [\![\mathbf{s}]\!]} v_i(x) - \mathbf{p} \cdot x \right) + \mathbf{p} \cdot \mathbf{s}. \tag{C}$$

Since f is nothing more than the dual linear program with the constraints moved to the objective function, the set of minimizers of f is exactly the set of optimal solutions of the dual linear program (or more precisely, their projections to the $\mathbf{p}$-variable). Each term $\max_{x \in [\![\mathbf{s}]\!]} v_i(x) - \mathbf{p} \cdot x$ is a convex function in $\mathbf{p}$ since it is a maximum over linear functions. Hence f is convex since it is a sum of convex functions.

One remarkable fact about f is that we can obtain a subgradient from the aggregate demand oracle:

Lemma 3 (Subgradient oracle) *Let* $\mathbf{d}(\mathbf{p})$ *be the aggregate demand, then* $\mathbf{s} - \mathbf{d}(\mathbf{p})$ *is a subgradient of* f *in* $\mathbf{p}$.

Proof From the Envelope Theorem, a subgradient of $\max_{x \in [\![\mathbf{s}]\!]} v_i(x) - \mathbf{p} \cdot x$ is given by the subgradient of $v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)}$ for $x^{(i)} \in \arg\max_x v_i(x) - \mathbf{p} \cdot x$. Since $v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)}$ is a linear function in $\mathbf{p}$, its gradient is simply $-x^{(i)}$. So, for any $x^{(i)}$ in the arg max, the vector $\mathbf{s} - \sum_i x^{(i)}$ is a subgradient o f f. In particular, $\mathbf{s} - \mathbf{d}(\mathbf{p})$. □

A useful fact in studying this function is that we have an initial bound on the set of minimizers:

Lemma 4 *If there exists a Walrasian equilibrium, then the set of minimizers* $P := \arg\min_{\mathbf{p}} f(\mathbf{p})$ *is contained in the box* $[-2M, 2M]^n$ *for* $M = \max_i \max_{x \in [\![s]\!]} |v_i(x)|$.

Proof Let $\mathbf{p} \in P$ be a vector of equilibrium prices and $\mathbf{x}$ an optimal allocation. Since the pair $(\mathbf{p}, \mathbf{x})$ constitute a Walrasian equilibrium, then for any item j, there is some $x_j^{(i)} \geq 1$ and therefore,

$$v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)} \geq v_i(x^{(i)} - \mathbf{1}_j) - \mathbf{p} \cdot (x^{(i)} - \mathbf{1}_j)$$

where $\mathbf{1}_j$ is the unit vector in the j-th coordinate. This gives us: $p_j \leq v_i(x^{(i)}) - v_i(x^{(i)} - \mathbf{1}_j) \leq 2M$.

For the lower bound, if there is more than one buyer then p_j must be larger than $-2M$, otherwise all the supply of item j will be demanded by all buyers and therefore $\mathbf{p}$ cannot be Walrasian. □

Lemma 5 *The set of Walrasian prices is a polytope* P *whose vertices have coordinates of the form* $p_j = a_j / b_j$ *with* $a_j, b_j \in \mathbb{Z}$ *and* $|b_j| \leq (Sn)^n$ *for* $S = \max_j s_j$.

Proof The vertices of P correspond to $\mathbf{p}$-coordinates of the basic feasible solutions of the dual linear program. Given that the coefficients of the $\mathbf{p}$ variables in the linear program are integers from 0 to S, solving the linear system using Cramer's rule (see Section 5 in [3]) we get that every solution must be a fraction with denominator at most $n! \cdot S^n \leq (Sn)^n$. □

Two more obstacles The natural approach at this point is to try to apply convex optimization algorithms as a black box to optimize f. There are two issues with this approach. The first, which is easier to address, is that unlike algorithms for linear programming which are exact, algorithms for general convex optimization give only ϵ-approximation guarantees. We will get around this issue by exploiting the connection between f and the linear program from which it is derived.

A second more serious obstacle is that we don't have access to the functional value of f, only to its subgradient. This is a problem for the following reasons. Traditional cutting plane methods work by keeping in each iteration t a set G_t called a *localizer*. The localizer is a subset of the domain which is guaranteed to contain the optimal solution. We start with a large enough localizer set G_0 that is guaranteed to contain the solution. In each iteration t, a point $\mathbf{p}_t \in G_{t-1}$ is queried for the subgradient $\partial f(\mathbf{p}_t)$. Now, if $\mathbf{p}^*$ is an optimal solution we know that:

$$0 \geq f(\mathbf{p}^*) - f(\mathbf{p}_t) \geq \partial f(\mathbf{p}_t) \cdot (\mathbf{p}^* - \mathbf{p}_t)$$

where the first inequality comes from the fact $\mathbf{p}^*$ is an optimal solution and the second inequality comes from the definition of the subgradient. This in particular means that any optimal solution must be contained in $H_t = G_{t-1} \cap \{\mathbf{p}; \partial f(\mathbf{p}_t) \cdot (\mathbf{p} - \mathbf{p}_t) \leq 0\}$. The method then updates G_t to either H_t or a superset thereof. The Ellipsoid Methods, for example, updates G_t to the smallest ellipsoid containing H_t. So far, all the steps depend only on the subgradient and not on the actual functional values of $f(\mathbf{p}_t)$. The guarantee of the method, however, is that after a sufficient number of steps, one of the iterates is close to the optimal, i.e., $\min_t f(\mathbf{p}_t) - f(\mathbf{p}^*) \leq \epsilon$. To find the approximate minimizer, however, we need to look at the actual functional values, which we have not access to. See Section 2.2 in Nemirovski [39] for a complete discussion on the guarantees provided by cutting plane methods.

A special case: Walrasian prices with non-empty interior The set of Walrasian prices P forms a convex set, since it corresponds to the set of minimizers of the convex function f. If P has non-zero volume, it is possible to find a Walrasian equilibrium using the Ellipsoid Method without needing to know the functional value. If the set is large, the Ellipsoid method cannot avoid querying a point in the interior of P for too long. And when a point in the interior of P is queried, the only subgradient is zero, or in market terms, each agent has a unique favorite bundle and those bundles clear the market. This means that the aggregate demand oracle returns exactly the supply. Next we make this discussion formal:

Theorem 4 *Assume that the set of Walrasian prices P has non-zero volume, then the Ellipsoid method*[3] *is guaranteed to query a Walrasian price $\mathbf{p}^*$ for which the aggregate demand oracle returns $\mathbf{d}(\mathbf{p}^*) = \mathbf{s}$ in $\tilde{O}(n^3)$ iterations.*

Proof By Lemma 4, $P \subseteq [-2M, 2M]^n$, so we can take the initial set of localizers G_0 in the ellipsoid methods to be the ball of radius $2M\sqrt{n}$ around 0, which has volume $O(M^n)$. The Ellipsoid guarantee (see Section 3 of [39]) is that the volume of G_t is at most $e^{-t/2n}$ of the initial volume. So if the method hasn't queries any point in the interior of P, then we must have G_t containing P.

To bound the volume of P we use the fact that if a polytope has vertices $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$ then the volume of the polytope is lower bounded by the volume of the convex hull of those vertices:

$$\text{vol}(P) \geq \frac{1}{n!} \det \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \mathbf{p}_0 & \mathbf{p}_1 & \ldots & \mathbf{p}_n \end{bmatrix} \geq \frac{1}{n!} \left(\frac{1}{O((Sn)^n)} \right)^n = \Omega((Sn)^{-n^2})$$

where the last inequality follows from Lemma 5.

Therefore, if G_t contains P then: $O(M^n)e^{-t/2n} \geq \Omega((Sn)^{-n^2})$ which implies that $t \leq O(n^3 \log(Sn) + n^2 \log(M)) = \tilde{O}(n^3)$. So after so many iterations we are guaranteed to query a point in the interior of P. For a point $\mathbf{p}$ in the interior of P, there is a small ball around it for which f is flat, so zero must be the unique subgradient at that point. Therefore, the aggregate demand oracle has no choice but to return $\mathbf{d}(\mathbf{p}) = \mathbf{s}$. □

[3] One may use cutting plane methods to derive a better running time but for simplicity we omit doing it for this special case.

The main drawback of this method is that it strongly relies on the fact that P has non-empty interior. Some very simple and well-behaved markets have a set of Walrasian prices of zero volume. Consider for example a market with two items with supply one of each and three buyers, each having valuation $v(0) = 0$ and $v(x) = 1$ otherwise. The set of Walrasian prices is $P = \{(1, 1, 1)\}$. Even for $\mathbf{p}^* = (1, 1, 1)$, the subgradient is not unique since the aggregate demand oracle can return any $\mathbf{d}(\mathbf{p}^*) = (d_1, d_2)$ for $0 \leq d_1 + d_2 \leq 3$ and $d_1, d_2 \in [\![3]\!]$. In this case even trying to modify the objective function is hopeless, since there is no point in the domain for which the aggregate demand oracle is guaranteed to return the supply vector. Since there is no point for which the subgradient oracle is guaranteed to return zero, there is no direct way to recognize a vector of Walrasian prices even if we see one!

Approach for the general case Our approach for optimizing f is as follows: first we show how to obtain an ϵ-approximate minimizer of f in time $O(n \log(nM/\epsilon)T_{AD} + n^3 \log^{O(1)}(nM/\epsilon))$. In order to round the ϵ-approximate solution to the optimal solution, we exploit the fact that f came from a linear program and customize the traditional approach of Khachiyan [22] for rounding approximate to exact solutions.

The idea is to use Lemma 5 to argue that if f has a unique minimizer, and we set ϵ small enough, then all ϵ-approximate solutions are in a small neighborhood of the exact optimal. Moreover, for small enough ϵ, there should be only one point in the format indicated by Lemma 5 in a small neighborhood of the approximate minimizer, so we can recognize this as the exact solution by rounding.

For this approach to work, we need f to have a unique minimizer. To achieve that, we perturb the objective function f to $\hat{f}$ in such a way that $\hat{f}$ has a unique minimizer and that this minimizer is still a minimizer of f. There are several ways to implement this approach, the simplest of which uses the isolation lemma (see Lemma 7). We note that the isolation lemma was not available to Khachiyan at the time so he had to resort to something more complicated.

A recent cutting plane method without using functional values The first part of the algorithm consists in obtaining an ϵ-approximate minimizer of f without using its functional value. In order to do so, we use a recent technique introduced by Lee et al. [27]. The authors show an efficient algorithm which either identifies a point inside the desired convex set K or *certifies* that K contains no ball of radius ϵ. We show that their main theorem (Theorem 31 in the full version of their paper [28]) can be used to compute approximate minimizers of convex functions using only the subgradient. We note that their paper also provides an application of Theorem 31 to minimizing convex functions (Theorem 42 in [28]) but their proof relies on using functional values as they did not assume a separation oracle given by subgradients.

Theorem 5 (Lee et al. (Theorem 31 in [28])) *Let $K \subseteq [-M, M]^n$ be a convex set and consider a separation oracle that can be queried for every point in $\mathbf{p} \in [-M, M]^n$ and will either return that $\mathbf{p} \in K$ or return a half-space $H = \{\mathbf{p} \in \mathbb{R}^n; \mathbf{a} \cdot \mathbf{p} \leq b\}$ containing K. Then there exists an algorithm with running time $O(nT \log(nM/\delta) +$*

$n^3 \log^{O(1)}(nM) \log^2(nM/\delta))$[4] *that either produces a point* $\mathbf{p} \in K$ *or finds a polytope* $P = \{\mathbf{p} \in \mathbb{R}^n; \mathbf{a}_i \cdot \mathbf{p} \leq b_i, i = 1, \ldots, k\}$, $k = O(n)$ *such that:*

- *The constraints* $\mathbf{a}_i \cdot \mathbf{p} \leq \mathbf{b}$ *are either from the original box,* $p_i \leq M$ *or* $p_i \geq -M$ *or are constraints returned by the separation oracle normalized such that* $\|\mathbf{a}_i\|_2 = 1$.
- *The width of* P *is small, i.e., there is a vector* $\mathbf{a}$ *with* $\|\mathbf{a}\|_2 = 1$ *such that*

$$\max_{\mathbf{p} \in P} \mathbf{a} \cdot \mathbf{p} - \min_{\mathbf{p} \in P} \mathbf{a} \cdot \mathbf{p} \leq O(n\delta \log(Mn/\delta)).$$

- *The algorithm produces a certificate of the previous fact in the form of a convex combination* $t_1, \ldots, t_k$ *with* $t_i \geq 0$, $\sum_{i=1}^k t_i = 1$ *and* $k = O(1)$ *such that:*
  - $\left\| \sum_{i=1}^k \mathbf{a}_i \right\|_2 \leq O\left(\frac{\delta}{M}\sqrt{n} \log\left(\frac{M}{\delta}\right)\right)$
  - $\left| \sum_{i=1}^k b_i \right| \leq O\left(n\delta \log\left(\frac{M}{\delta}\right)\right)$.

Now, we show that this result can be used to obtain a convex minimization algorithm that uses only subgradients:

Theorem 6 *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *be an L-Lipschitz convex function equipped with a subgradient oracle with running time T. If f has a minimizer in* $[-M, M]^n$ *we can find a point* $\bar{\mathbf{p}}$ *such that* $f(\bar{\mathbf{p}}) - \min_{\mathbf{p}} f(\mathbf{p}) \leq \epsilon$ *in time* $O(nT \log(nML/\epsilon) + n^3 \log^{O(1)}(nML) \log^2(nML/\epsilon))$.

Proof Let $K = \operatorname{argmin}_{\mathbf{p}} f(\mathbf{p})$ and use the subgradient as the separation oracle, since if $\partial f(\mathbf{p}_t)$ is the subgradient at $\mathbf{p}_t$ we know that for all $\mathbf{p} \in K$, it holds that $\partial f(\mathbf{p}_t) \cdot (\mathbf{p} - \mathbf{p}_t) \leq 0$. What we will do is to run the algorithm in Theorem 5 starting from a very large box $[-M', M']$ for $M' = n^{O(1)}M$ instead of starting from $[-M, M]$, which appears to be more natural. The reason we do that is to avoid having the constraints defining the bounding box added to P.

Either the algorithm will return a point $\mathbf{p} \in K$, in which case we are done or will return a set P like in statement of the theorem. If we are lucky and all constraints added to P are of the type $\partial f(\mathbf{p}_t) \cdot \mathbf{p} \leq \partial f(\mathbf{p}_t) \cdot \mathbf{p}_t$, then we can use the certificate $t_1, \ldots, t_k$ to produce a point $\bar{\mathbf{p}} = \sum_{i=1}^k t_i \mathbf{p}_i$. Now:

$$\begin{aligned} f(\bar{\mathbf{p}}) - f(\mathbf{p}^*) &\leq \sum_i t_i f(\mathbf{p}_i) - f(\mathbf{p}^*) \leq \sum_i \partial f(\mathbf{p}_i) \cdot (\mathbf{p}_i - \mathbf{p}^*) \\ &\leq L \cdot \left[\left\| \sum_i t_i \mathbf{a}_i \right\| \cdot \|\mathbf{p}^*\| + \left| \sum_i t_i b_i \right| \right] \\ &\leq L \cdot \left[O\left(\frac{\delta}{M'} \sqrt{n} \log\left(\frac{M'}{\delta} \right) \right) \cdot M' \sqrt{n} + O\left(n\delta \log\left(\frac{M'}{\delta} \right) \right) \right] \\ &= O\left(n\delta L \log\left(\frac{M'}{\delta} \right) \right) \end{aligned}$$

[4] The running time stated in their paper has $\log^{O(1)}(nM/\delta)$ dependence which is, upon a closer examination, actually $\log^{O(1)}(nM) \log^2(nM/\delta))$ (They ignored the difference because both runtimes give the same result for their applications).

Then setting δ such that $\epsilon = O\left(n\delta L \log\left(\frac{M'}{\delta}\right)\right)$ gives us the desired result.

To be done, we just need to worry about the case where some of the box constraints are present in P. In that case, we argue that the weight put by the coefficients t_i on the box constraints must be tiny, so it is possible to remove those and rescale t. Formally, observe that

$$\sum_i t_i(b_i - \mathbf{a}_i \cdot \mathbf{p}^*) \leq \left|\sum_i t_i b_i\right| + \left\|\sum_i t_i \mathbf{a}_i\right\|_2 \cdot \left\|\mathbf{p}^*\right\|_2 = O\left(n\delta \log\left(\frac{M'}{\delta}\right)\right)$$

Now, if i is an index corresponding to a box constraint such as $p_i \leq M'$ or $p_i \geq -M'$ then $b_i - \mathbf{a}_i \cdot \mathbf{p}^* \geq \left|M' - M\right| = \Omega(n^{O(1)}M)$. This implies in particular that:

$$O\left(n\delta \log\left(\frac{M'}{\delta}\right)\right) \geq \sum_i t_i(b_i - \mathbf{a}_i \cdot \mathbf{p}^*) \geq t_i(b_i - \mathbf{a}_i \cdot \mathbf{p}^*) \geq t_i \Omega(n^{O(1)}M)$$

so $t_i \leq O\left(\frac{n^{-O(1)}\delta}{M} \log\left(\frac{M'}{\delta}\right)\right)$. By choosing a very small δ, say $\delta = O\left(\frac{\epsilon}{Ln^{O(1)}M^{O(1)}}\right)$ we guarantee that is B is the set of indices corresponding to box constraints, then $\sum_{i\in B} t_i \leq O\left(\frac{\epsilon}{n^{O(1)}M^{O(1)}}\right) \leq \frac{1}{2}$ and that $\sum_i t_i(b_i - \mathbf{a}_i \cdot \mathbf{p}^*) \leq O\left(\frac{\epsilon}{Ln^{O(1)}M^{O(1)}}\right)$. This allows us to define for $i \notin B$, $t_i' = t_i / \left(1 - \sum_{i\in B} t_i\right)$, so we have:

$$\begin{aligned}
\frac{\epsilon}{2L} \geq O\left(\frac{\epsilon}{Ln^{O(1)}M^{O(1)}}\right) &\geq \sum_i t_i(b_i - \mathbf{a}_i \cdot \mathbf{p}^*) \geq \sum_{i\notin B} t_i(b_i - \mathbf{a}_i \cdot \mathbf{p}^*) \\
&= \left(1 - \sum_{i\in B} t_i\right)\left(\sum_{i\notin B} t_i'(b_i - \mathbf{a}_i \cdot \mathbf{p}^*)\right) \\
&\geq \frac{1}{2}\left(\sum_{i\notin B} t_i'(b_i - \mathbf{a}_i \cdot \mathbf{p}^*)\right)
\end{aligned}$$

Now we can repeat the argument in the beginning of the proof with $\bar{\mathbf{p}} = \sum_{i\notin B} t_i' \mathbf{p}_i$. □

Perturbation, approximation and rounding The next step is to use the algorithm for finding an approximate solution to find an exact solution. This is impossible for generic convex programs, but since the function we are trying to optimize comes from a linear program, we can do that by exploiting this connection. As done for linear programs, we will do this in three steps: first we perturb the function to be optimized, then we find an approximate solution by Theorem 6 and finally we round it to an exact solution in the format of the optimal solution given by Lemma 5.

We can perturb the objective of the dual linear program by changing it to:

$$\min u + \mathbf{p} \cdot (\mathbf{s} + \mathbf{r}) \text{ s.t. } u \geq \sum_i v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)}, \forall x^{(i)} \in [\![s]\!]. \tag{PD}$$

We want to specify the real-valued vector $\mathbf{r}$ in such a way that the optimal solution to this linear program is still the optimal solution to the original program, but also in a way that the optimal solution becomes unique. First we observe the following:

Lemma 6 *If $r_i < (nS)^{-(2n+1)}$ then an optimal solution of the perturbed program is also an optimal solution to the original program.*

Proof Let $\mathcal{C}$ be the set of vertices (basic feasible solutions) of the dual LP. By Lemma 5 we know that the coordinates of those vertices must be of the form a_i/b_i for integers a_i, b_i such that $0 \leq b_i \leq (nS)^n$. For any linear objective function, the optimal solution must be a point in $\mathcal{C}$. Since the original objective has integral coefficients, when evaluated on any vertex, the objective is a fraction with denominator $b_i \leq (nS)^n$. Therefore, the difference between the objective evaluated at an optimal vertex and the objective evaluated at a suboptimal one is at least $\left|\frac{a_i}{b_i} - \frac{a_i'}{b'i}\right| = \left|\frac{a_i b_i' - a_i' b_i}{b_i b_i'}\right| \geq \frac{1}{(nS)^{2n}}$. Therefore, if $r_i < (nS)^{-(2n+1)}$, then its effect in the objective function is at most $nS \cdot (nS)^{-(2n+1)} \leq (nS)^{-2n}$, so it cannot cause a suboptimal solution to the original program to become an optimal solution of the perturbed program. □

Our final ingredient is a lemma by Klivans and Spielman [23] which is in the spirit of the Isolation Lemma of Mulmuley et al. [38].

Lemma 7 (Klivans and Spielman (Lemma 4 in [23])) *Let $\mathcal{C}$ be a set of points in $\mathbb{R}^n$ where all coordinates assume at most K distinct values. Then if $\mathbf{r}$ is a vector with coordinates sampled at random from $\{0, 1, \ldots, Kn/\epsilon\}$, then with probability $1 - \epsilon$, there is a unique $\mathbf{p} \in \mathcal{C}$ minimizing $\mathbf{r} \cdot \mathbf{p}$.*

We note that although Lemma 4 in [23] is stated for $\mathcal{C}$ having coordinates in $\{0, 1, \ldots, K-1\}$, the proof only uses the fact that the coordinates of $\mathcal{C}$ assume at most K distinct values.

Lemma 8 *If $r_j = z_j / \left[Mn(nS)^{2n+1}\right]$ where z_j is drawn uniformly from $\{0, \ldots, nM(nS)^{2n} - 1\}$, then with probability $\frac{1}{2}$, the dual program has a unique minimizer and this minimizer is a minimizer of the original program.*

Proof Let $\mathcal{C}$ be the set of vertices (basic feasible solutions) of the dual linear program in $[-M, M]^n$. All the minimizers of the original dual program are guaranteed to be in this set because of Lemma 4. Lemmas 4 and 5 tell us that the coordinates of points in $\mathcal{C}$ are of the form a_i/b_i where a_i and b_i are integers such that $1 \leq b_i \leq (nS)^n$ and $|a_i| \leq Mb_i$. So there are at most $\sum_{b=1}^{(nS)^n} b \cdot 2M \leq 2M(nS)^{2n}$ coordinates.

Now, Lemma 6 guarantees that the magnitude of the perturbation will prevent suboptimal vertices in the original program to become optimal vertices in the perturbed program. Lemma 7 guarantees that with half probability the vertex minimizing the objective is unique. □

The perturbed dual program translates to the following perturbed objective function:

$$\hat{f}(\mathbf{p}) = \sum_{i=1}^{m} \left(\max_{x \in [\![\mathbf{s}]\!]} v_i(x) - \mathbf{p} \cdot x \right) + \mathbf{p} \cdot (\mathbf{r} + \mathbf{s}). \tag{PC}$$

Since the subgradient of $\hat{f}$ can be computed from the aggregate demand oracle $\partial \hat{f}(\mathbf{p}) = \mathbf{s} + \mathbf{r} - \mathbf{d}(\mathbf{p})$ we can use Theorem 6 to find an ϵ-minimizer.

Lemma 9 *For $\epsilon = (nMS)^{-O(n)}$, if $\mathbf{p}$ is an ϵ-approximation to the optimal value of $\hat{f}$, i.e., $\hat{f}(\mathbf{p}) - \hat{f}(\mathbf{p}^*) \leq \epsilon$ then the optimal solution is the only point $\mathbf{p}^*$ such that $\|\mathbf{p} - \mathbf{p}^*\|_2 \leq \frac{1}{(nMS)^{O(n)}}$ and has coordinates of the form described in Lemma 5.*

Proof Let $\mathcal{C}$ be the set of vertices of the dual linear program, which are points of the form $(u_i, \mathbf{p}_i)$ and let $\mathbf{p}$ be an ϵ-approximation to $\hat{f}$. Now, if $u = \sum_i (\max_x v_i(x) - \mathbf{p} \cdot x)$ then $(u, \mathbf{p})$ is feasible in the dual linear program, and in particular, it can be written as a convex combination of points in $\mathcal{C}$, i.e., $(u, \mathbf{p}) = \sum_i t_i (u_i, \mathbf{p}_i)$ for $t_i \geq 0$ and $\sum_i t_i = 1$. There is only one vector in $\mathcal{C}$, call it $(u^*, \mathbf{p}^*)$ for which the objective evaluates to $\hat{f}^*$. For all other vertices, the objective evaluates to at least $\hat{f}^* + \frac{1}{Mn(nS)^{2n+1}} \cdot \frac{1}{(nS)^n} = \hat{f}^* + \frac{1}{Mn(nS)^{3n+1}}$ due to Lemmas 5 and 8. Therefore, by evaluating $(u, \mathbf{p}) = \sum_i t_i (u_i, \mathbf{p}_i)$ on the linear objective of the perturbed dual program, we get:

$$\hat{f}^* + \epsilon \geq \hat{f}(\mathbf{p}) \geq t_* \hat{f}^* + (1 - t_*) \left(\hat{f}^* + \frac{1}{Mn(nS)^{3n+1}} \right)$$

where t_* is the weight put on $(u^*, \mathbf{p}^*)$ by the convex combination, therefore, if $\epsilon = (nMS)^{-O(n)}$ then $t^* \geq 1 - (nMS)^{-O(n)}$. In particular:

$$\left\| \mathbf{p} - \mathbf{p}^* \right\|_2 \leq \left\| \sum_{i \neq *} t_i (\mathbf{p}_i - \mathbf{p}^*) \right\|_2 \leq (1 - t_*) \cdot \max_{i \neq *} \left\| \mathbf{p}_i - \mathbf{p}^* \right\| \leq (nMS)^{-O(n)}$$

Therefore, $\mathbf{p}^*$ is in a ball of radius $(nMS)^{-O(n)}$ around $\mathbf{p}$. Also, any other point $\mathbf{p}' \neq \mathbf{p}^*$ with coordinates of the form a_i / b_i with $b_i \leq (nS)^n$ can be in this ball, since two distinct points of this type differ in at least one coordinate by at least $(nS)^{-n}$ so their distance is at least that much. □

Putting it all together:

Theorem 7 *There is an algorithm of running time $O(n^2 T_{AD} \log(SMn) + n^5 \log^{O(1)}(SMn))$ to compute an exact vector of Walrasian prices whenever it exists using only access to an aggregate demand oracle.*

Proof From the potential function f of the market, use Lemma 8 to construct a perturbed potential function $\hat{f}$. Then use Theorem 6 to optimize $\hat{f}$ with $\epsilon = (nMS)^{-O(n)}$. Finally, use the guarantee to argue that the exact optimal value is the only point in the format of Lemma 5 in the ball of radius $(nMS)^{-O(n)}$ around the ϵ-approximate minimizer. At this point, we can round the solution to the exact point by using the method of continuous fractions in Kozlov et al. [24] (see [44] for a complete exposition of this and related methods of rounding). □

4 Walrasian equilibrium for Gross substitutes in $\tilde{O}(nT_{AD} + n^3)$

In the previous section we discussed how Walrasian equilibria can be computed without any assumptions on the valuation function using only an aggregate demand oracle. The focus was to address various difficulties in applying optimization tools for this problem.

Here we remain in the aggregate demand oracle model and focus on computing Walrasian equilibria for markets typically studied in economics, which are those where buyers have *gross substitute valuations*. The development of the theory of gross substitute valuations is intertwined with the development of the theory of Walrasian equilibrium for markets with discrete goods. In particular, it is the largest class of valuation functions that is closed under perturbation by an additive function[5] for which Walrasian equilibria always exist.

Gross substitutes play a central role in economics. Hatfield and Milgrom [18] argue that most important examples of valuation functions arising in matching markets belong to the class of gross substitutes. Gross substitutes have been used to guarantee the convergence of salary adjustment processes in the job market [21], to guarantee the existence of stable matchings in a variety of settings [25,43], to show the stability of trading networks [17,20], to design combinatorial auctions [1,30] and even to analyze settings with complementarities [16,46].

The concept of gross substitutes has been re-discovered in different areas from different perspectives: Dress and Wenzel [13] propose the concept of *valuated matroids* as a generalization to the Grassmann–Plücker relations in p-adic analysis. Dress and Terhalle [9] define the concept of *matroidal maps* which are the exact class of functions that can be optimized by greedy algorithms. Murota [32] generalized the concept of convex functions to discrete lattices, which gave birth to the theory known as Discrete Convex Analysis. One of the central objects in the Discrete Convex Analysis are M-concave and $M^\natural$-concave functions (the latter class was introduced by Murota and Shioura [29]). We refer to a recent survey by Murota [37] for comprehensive discussion on application of Discrete Convex Analysis in economics.

Surprisingly, gross substitutes, valuated matroids, matroidal maps and $M^\natural$-concave functions are all equivalent definitions—despite being originated from very different motivations. We refer to [26] for a detailed historic description and a survey on their equivalence.

Before presenting our algorithms, we first give a quick summary of the standard facts about gross substitutes that are needed. For a more comprehensive introduction, please see [26].

4.1 A crash course on gross substitutes

First we define gross substitute valuations on the hypercube $\{0,1\}^{[n]}$ and then we extend the definition to $[\![\mathbf{s}]\!]$ for any supply vector $\mathbf{s}$. When talking about functions defined on the hypercube, we will often identify vectors $x \in \{0,1\}^{[n]}$ with the set

[5] A class $\mathcal{C}$ of valuation functions is closed under perturbations by additive functions if for every $v \in \mathcal{C}$ and every vector $\mathbf{w} \in \mathbb{R}^n$, the function $w(x) = v(x) + \mathbf{w} \cdot x$ is also in $\mathcal{C}$.

$S = \{i \in [n] : x_i = 1\}$, so we write $v(S)$ where $S \subseteq [n]$ meaning $v(\mathbf{1}_S)$ where $\mathbf{1}_S$ is the indicator vector of S.

Next we define three classes of valuation functions. The reader who saw the spoilers in the previous subsection will already suspect that they are equivalent.

Definition 6 (*Gross substitutes, Kelso and Crawford* [21]) A function $v : 2^{[n]} \to \mathbb{Z}$ is a gross substitute (GS) if for any price $\mathbf{p} \in \mathbb{R}^n$ and $S \in D(v, \mathbf{p})$, any price $\mathbf{p}' \geq \mathbf{p}$ (entry-wise) has some $S' \in D(v, \mathbf{p}')$ satisfying $S \cap \{j : p_j = p'_j\} \subseteq S'$.

In other words, price increases for some items can't affect the purchasing decisions for the items whose price stayed the same.

The second definition concerns when the demand oracle problem $\max v(S) - p(S)$ can be solved efficiently:

Definition 7 *(Matroidal Maps, Dress and Terhalle* [10]) A function $v : 2^{[n]} \to \mathbb{Z}$ is called matroidal if for any price $\mathbf{p} \in \mathbb{R}^n$, the set S obtained by the following greedy algorithm solves $\max_{S \subseteq [n]} v(S) - p(S)$:

Greedy algorithm: Start with the empty set $S = \varnothing$. While $|S| < n$ and there exists $i \notin S$ such that $v(S \cup i) - p_i - v(S) > 0$, then update $S \leftarrow S \cup i^*$ where $i^* = \operatorname{argmax}_{i \in [n]-S} v(S \cup i) - p_i - v(S)$ (break ties arbitrarily).

The third definition generalizes the concept of convexity and concavity to the hypercube. We can define convexity for continuous functions $f : \mathbb{R}^n \to \mathbb{R}$ as being function such that for all vectors $\mathbf{v} \in \mathbb{R}^n$, if x^* is a local minimum of $f(x) - \mathbf{v} \cdot x$, then x^* is also a global minimum. This definition generalizes naturally to the hypercube as follows:

Definition 8 *(Discrete Concavity, Murota* [36]*, Gul and Stachetti* [15]) A function $v : 2^{[n]} \to \mathbb{Z}$ is discrete concave function if local optimality implies global optimality for all price vectors $\mathbf{p}$. Formally: if $S \subseteq [n]$ is such that for all $i \in S$ and $j \notin S$, $v(S) \geq v(S \cup j) - p_j$, $v(S) \geq v(S - i) + p_i$ and $v(S) \geq v(S \cup j - i) + p_i - p_j$, then $v(S) - \mathbf{p}(S) \geq v(T) - \mathbf{p}(T), \forall T \subseteq [n]$.

Discrete concave functions have two important properties: (i) the demand oracle has a succinct certificate of optimality and (ii) the function can be optimized via local search for any given prices.

Those definitions arose independently in different communities and, amazingly enough, those definitions turned out to be equivalent. The equivalence of discrete concavity and gross substitutes is due to Fujishige and Yang [14] and the equivalence of discrete concavity and matroidal maps appears explicitly in Paes Leme [26].

Theorem 8 *A valuation function is in gross substitutes iff it is a matroidal map and iff it is a discrete concave valuation.*

The concept of gross substitutes generalizes naturally to multi-unit valuations: given any function $v : [\![\mathbf{s}]\!] \to \mathbb{Z}$ we can translate it to a single-unit valuation function $\tilde{v} : 2^{[\sum_i s_i]} \to \mathbb{Z}$ by treating each of the s_i copies of item i as a single item. We say that a multi-unit valuation function v is gross substitutes if its corresponding single

item version $\tilde{v}$ is in gross substitutes. We refer to the excellent survey by Shioura and Tamura [45] on gross substitutability for multi-unit valuations.

An important property of gross substitutes is:

Theorem 9 ([8,21]) *If all buyers have gross substitute valuations, then a Walrasian equilibrium always exists.*

The original theorem showing the existence of Walrasian equilibrium for gross substitutes in the single-unit case ($s_j = 1$ for all j) is due to Kelso and Crawford [21]. The extension to multi-units is due to Danilov et al. [8].

4.2 Walrasian Prices form an integral polytope

By Theorem 9, the set of Walrasian prices is non-empty. We also know that it forms a convex polyhedral set, since they are the set of minimizers of a convex function that comes from a linear program. Perhaps a more direct way to see it is that given an optimal allocation $\mathbf{x}$, the set of Walrasian prices can be characterized by the following set of inequalities:

$$P = \{\mathbf{p} \in \mathbb{R}^n \mid v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)} \geq v_i(x) - \mathbf{p} \cdot x, \forall i \in [m], x \in [\![s]\!]\}.$$

Lemma 5 provided a good characterization of the vertices of this polytope in the general case. For gross substitutes, however, there is an even nicer characterization. The next lemma is a special case of a result by Murota (Theorem 11.16 in [36]) that shows that the set of Walrasian prices form an $L^\natural$-convex polytope and hence is an integral polyhedron with a lattice structure. To keep the paper self-contained we give a proof from first principles.

Lemma 10 *If buyer valuations are gross substitutes, then all the vertices of the feasible region of the dual program D (defined in Sect. 3) have integral coordinates. In particular, the set of Walrasian prices form an integral polytope.*

Proof Let $(u, \mathbf{p})$ be a non-integral feasible point of the dual program D. We will write it as a convex combination of integral points. Let $x^{(i)} \in \operatorname{argmax} v_i(x) - \mathbf{p} \cdot x$ and $w = u - \sum_i v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)}$.

Now, we define a distribution over integral points in the following way: sample a random threshold $\theta \in [0, 1]$. If $\theta < p_i - \lfloor p_i \rfloor$, set $\hat{p}_i = \lceil p_i \rceil$, otherwise set $\hat{p}_i = \lfloor p_i \rfloor$. Similarly, if $\theta < w - \lfloor w \rfloor$, set $\hat{w} = \lceil w \rceil$ otherwise set $\hat{w} = \lfloor w \rfloor$. Set $\hat{u} = \hat{w} + \sum_i v_i(x^{(i)}) - \hat{\mathbf{p}} \cdot x^{(i)}$. It is easy to see that $(\hat{u}, \hat{\mathbf{p}})$ are integral and that $\mathbb{E}[(\hat{u}, \hat{\mathbf{p}})] = (u, \mathbf{p})$. We are only left to prove that $(\hat{u}, \hat{\mathbf{p}})$ are feasible.

We know that $\hat{u} \geq \sum_i v_i(x^{(i)}) - \hat{\mathbf{p}} \cdot x^{(i)}$ since $\hat{w} \geq 0$. If we show that $x^{(i)} \in \arg\max_x v_i(x) - \mathbf{p} \cdot x$, then we are done, since it automatically implies that all other constraints in the dual program D are satisfied. To see this notice that since $x^{(i)} \in \operatorname{argmax} v_i(x) - \mathbf{p} \cdot x$, then for all items j and k such that $x_j^{(i)} < s_j$ and $x_k^{(i)} > 0$ we have that:

$$v_i(x^{(i)}) \geq v_i(x^{(i)} + \mathbf{1}_j) - p_j,\ v_i(x^{(i)}) \geq v_i(x^{(i)} - \mathbf{1}_k) + p_k$$
$$v_i(x^{(i)}) \geq v_i(x^{(i)} - \mathbf{1}_k + \mathbf{1}_j) + p_k - p_j$$

Since the valuations are integer valued, it is simple to check that rounding using a threshold won't violate any of the above inequalities. Thus:

$$v_i(x^{(i)}) \geq v_i(x^{(i)} + \mathbf{1}_j) - \hat{p}_j,\ v_i(x^{(i)}) \geq v_i(x^{(i)} - \mathbf{1}_k) + \hat{p}_k$$
$$v_i(x^{(i)}) \geq v_i(x^{(i)} - \mathbf{1}_k + \mathbf{1}_j) + \hat{p}_k - \hat{p}_j$$

Therefore under price vector $\hat{\mathbf{p}}$, a buyer can't improve his utility from $x^{(i)}$ by adding, removing or swapping an item. Since gross substitutes are equivalent to discrete concavity (Definition 8), local optimality implies global optimality, i.e., $x^{(i)} \in \arg\max_x v_i(x) - \hat{\mathbf{p}} \cdot x$. □

Another important property proved by Gul and Stachetti [15] is that the set of Walrasian prices forms a lattice. This property is also a consequence of the $L^\natural$-convex structure in Murota [36].

Theorem 10 (Gul and Stachetti [15], Murota [36]) *If buyer valuations are gross substitutes, then the set of Walrasian prices form a lattice, i.e., if* $\mathbf{p}$ *and* $\mathbf{p}'$ *then* $\bar{\mathbf{p}}$ *and* $\underline{\mathbf{p}}$ *are also Walrasian prices for* $\bar{p}_i = \max(p_i, p'_i)$ *and* $\underline{p}_i = \min(p_i, p'_i)$.

4.3 A simpler and faster algorithm for gross substitutes

Using the fact that the set of Walrasian prices is an integral polytope with a lattice structure we simplify the algorithm described in the previous section and improve its running time. First, since we have a lattice structure we no longer need to randomly perturb the objective function to make the solution unique. A simple and deterministic perturbation suffices. Integrality also allows us to round to an optimal solution from an approximate solution of smaller accuracy (i.e. larger ϵ).

Lemma 11 *If valuations are gross substitutes, then by taking* $r_j = \frac{1}{2Sn}$ *in the perturbed dual program* PD*, its optimal solution is unique and also optimal for the original dual program* D*.*

Proof Since all the vertices of the polytope are integral and the coefficients are at most S, a perturbation of $r_j = \frac{1}{2Sn}$ can't affect the objective by more than half. So it can't cause a suboptimal vertex to become optimal. Also, since the set of Walrasian prices form a lattice, there is a Walrasian price $\bar{\mathbf{p}}$ such that $\bar{\mathbf{p}} \geq \mathbf{p}$ for every Walrasian price $\mathbf{p}$. Therefore this must be the unique optimal solution to the perturbed program. □

The previous lemma allows us to prove a better version of Lemma 9:

Lemma 12 *For* $\epsilon < 1/(4nMS)$*, if* $\mathbf{p}$ *is an* ϵ*-approximation to the optimal value of* $\hat{f}$*, i.e.,* $\hat{f}(\mathbf{p}) - \hat{f}(\mathbf{p}^*) \leq \epsilon$ *then the optimal solution is the only integral point* $\mathbf{p}^*$ *such that* $\|\mathbf{p} - \mathbf{p}^*\|_2 < \frac{1}{2}$.

Proof The proof can be obtained following the proof of Lemma 9, replacing the generic guarantees in Lemmas 5 and 8 by the better guarantees provided by Lemma 11 for gross substitutes. □

Putting it all together we have:

Theorem 11 *There is an algorithm of running time* $O(nT_{AD}\log(SMn) + n^3\log^{O(1)}(SMn))$ *to compute an exact vector of Walrasian prices in a market where buyers have gross substitute valuations.*

Proof Same proof as in Theorem 7 with $\epsilon = 1/(5nMS)$. Also, since the optimal price vector is integral, instead of using the method of continuous fractions to round a solution, it is enough to round each component to the nearest integer. □

5 Walrasian equilibrium for gross substitutes in $\tilde{O}((mn + n^3) \cdot T_V)$

We now move from the macroscopic view of the market to a microscopic view. We assume access to the market via a *value oracle*, i.e, given a certain buyer i and a bundle $S \subseteq [n]$ of goods, we can query the value of $v_i(S)$. We also assume from this point on that the supply of each good is one, or in other words, that the valuation functions are defined on the hypercube.

The fact that the demand of each buyer for any given price can be computed by the greedy algorithm (Definition 7) let us simulate the aggregate demand oracle by the value oracle model.

Lemma 13 *The outcome of the aggregate demand oracle can be computed in time* $O(mn^2T_V)$*, where* T_V *is the running time of the value oracle.*

Proof The number of queries required for the greedy algorithm described in Definition 7 to compute $S_i \in \arg\max_S v_i(S) - \mathbf{p}(S)$ is $n \cdot (|S_i| + 1)T_V \leq O(n^2T_V)$. Since there are m buyers, the total time to compute the demand of all buyers is $O(mn^2T_V)$. The aggregate demand oracle simply outputs $d(\mathbf{p})$ where $d_j(\mathbf{p}) = \#\{i : j \in S_i^*\}$. □

Now we can plug Lemma 13 directly into Theorem 11 and obtain a running time of $\tilde{O}(mn^3T_V)$. In the rest of this section, we show how to improve this to $\tilde{O}((mn+n^3)T_V)$.

5.1 Faster Algorithm via regularization

The idea to improve the running time from $\tilde{O}(mn^3T_V)$ to $\tilde{O}((mn + n^3)T_V)$ is to regularize the objective function. As with the use of regularizers in other context in optimization, this is to penalize the algorithm for being too aggressive. The bound of $O(mn^2)$ value oracle calls per iteration of the cutting plane algorithm is so costly precisely because we are trying to take an aggressively large step.

To provide some intuition, imagine that we have a price $\mathbf{p}$ that is very close to optimal and that S_i are the set of items demanded by the buyers at price $\mathbf{p}$. Intuitively, if $\mathbf{p}$ is close to a Walrasian price then the sets $S_1, \ldots, S_m$ should be almost disjoint, which means that the total cost of the greedy algorithm should be $\sum_i n(|S_i|+1) \approx mn+n^2$. So when the prices are good, oracle calls should be cheaper. This tells us that when prices are good, fewer calls to the value oracle suffice to compute the aggregate demand oracle. When prices are far from equilibrium, perhaps a more crude approximation to the aggregate demand oracle is enough.

Based on this idea we define the following regularized objective function:

$$\tilde{f}(\mathbf{p}) = \max_{\sum_i |S_i| = n} \left[\sum_{i=1}^{m} v_i(S_i) - \mathbf{p}(S_i) \right] + \mathbf{p}([n]). \tag{RC}$$

The regularization consists of taking the maximum over all sets $(S_1, \ldots, S_m)$ such that $\sum_i |S_i| = n$. Without this restriction, we have the original market potential function f. The new function $\tilde{f}$ has three important properties: (i) it is still convex, since it is a maximum over linear functions in $\mathbf{p}$ and therefore we can minimize it easily; (ii) its set of minimizers is the same as the set of minimizers of f and (iii) subgradients are cheaper to compute. Intuitively, $\tilde{f}$ is very close to f when $\mathbf{p}$ is close to equilibrium prices but only a crude estimate when $\mathbf{p}$ is far from equilibrium. Next we show those statements formally and present an algorithm for computing the subgradient of $\hat{f}$.

We give an alternate form of $\tilde{f}$ which is nicer to work with algorithmically. A consequence of the next lemma, is that for every $\mathbf{p}$ there is a constant γ such that $\tilde{f}(\mathbf{p}) = f(\mathbf{p} + \gamma \cdot \mathbf{1}_{[n]})$ and that such parameter γ can be found algorithmically. In other words:

$$\tilde{f}(\mathbf{p}) = \sum_{i=1}^{m} \left(\max_{S \subseteq [n]} v_i(S) - \mathbf{p}(S) - \gamma \cdot |S| \right) + \mathbf{p}([n]) + \gamma \cdot n$$

for some γ that depends on $\mathbf{p}$ (and the tiebreaking rule used by the greedy algorithm). Among other things, this formulation of $\tilde{f}$ resembles common regularizers used in optimization better. One can think of it as if $\mathbf{p}$ is changed to $\mathbf{p} + \gamma \cdot \mathbf{1}_{[n]}$.

Lemma 14 *Suppose $v_i(j)$ are given and stored as n lists sorted in decreasing order $L_j = \{v_i(j)\}_{i=1..m}$. With a running time[6] of $n^2 \cdot T_V + \tilde{O}(n^2)$, given price $\mathbf{p}$, there is an algorithm, which we call* AllGreedy*, that finds*

1. $S_1^*, \ldots, S_m^*$ *maximizing* $\max_{\sum_i |S_i| = n} \left(\sum_{i=1}^{m} v_i(S_i) - \mathbf{p}(S_i) \right)$.
2. γ *such that for all i,* $S_i^* \in D(i, \mathbf{p} + \gamma \cdot \mathbf{1}_{[n]})$. *Moreover, for any* $\gamma' > \gamma$ *and* $S_i' \in D(i, \mathbf{p} + \gamma' \cdot \mathbf{1}_{[n]})$, *we have* $\sum_i |S_i'| < n$.
3. $\tilde{f}(\mathbf{p}) = f(\mathbf{p} + \gamma \cdot \mathbf{1}_{[n]})$.

Proof First, we define the algorithm AllGreedy. The algorithm starts with a very large value of γ such that $D(i, \mathbf{p} + \gamma \cdot \mathbf{1}_{[n]}) = \{\varnothing\}$ for all agents i. Then we gradually decrease γ keeping at each step a set $S_i^*(\gamma) \in D(i, \mathbf{p} + \gamma \cdot \mathbf{1}_{[n]})$ that monotonically grow as γ decreases, in the sense that for $\gamma_1 > \gamma_2$, $S_i^*(\gamma_1) \subseteq S_i^*(\gamma_2)$. We stop the algorithm as $\sum_i \left| S_i^*(\gamma) \right|$ reaches n.

The algorithm is best visualized as a continuous process, although it admits a very efficient discrete implementation as we will see in a moment. Before, we note that we can use the Greedy algorithm to compute $S_i^*(\gamma) \in D(i, \mathbf{p} + \gamma \cdot \mathbf{1}_{[n]})$ and if we fix the same tie breaking, the order in which we add the elements is the same for every γ, the only thing that changes is the stopping criteria (the larger γ is, the later we stop).

[6] Assuming the cost of initializing $S_i^* = \varnothing$ is not needed. This is acceptable here because our algorithm would only use S_i^* to compute the subgradient which $S_i^* = \varnothing$ has no effect on.

So this procedure can be implemented by running a greedy algorithm in parallel for each agent i. Initially γ is very large and all $S_i^*(\gamma) = \varnothing$. Then in any given moment, we can compute what is the largest value of γ for which it is possible to add one more item to the demanded set of i. This is simply the largest marginal of an i for the next item:

$$\max_i \max_{j \notin S_i^*} v_i(S_i^* \cup j) - v_i(S_i^*) - p_j$$

We can decrease γ to this value and advance one of the agent's greedy algorithm one step further.

We need to argue that it satisfies the three properties in the lemma and that it can be implemented in $n^2 T_V + O(n^2)$ time. The algorithm achieves this running time by updating lists $\mathtt{L}_j$ such that in each iteration, it is a sorted list of $v_i(S_i^* \cup j) - v_i(S_i^*)$. Since all sets start as the empty set, this is correct in the beginning of the algorithm. Now, in each iteration, we can scan all the lists to find the next largest marginal, taking $O(n)$ to inspect the top of each list. This gives us the next value of γ and the pair i, j to update $S_i^* \leftarrow S_i^* \cup j$. Now, after the update, we go through each list $\mathtt{L}_k$ updating the value of the marginal of agent i for k, since S_i^* was updated. This takes $O(\log(m))$ for each list, so in total, this process takes $nT_V + \tilde{O}(n)$. Since there are at most n iterations, the overall running time is $n^2 T_V + \tilde{O}(n^2)$.

Now, for three properties in the lemma, property 2 is true by construction. For properties 1 and 3, consider the following chain of inequalities:

$$\begin{aligned}
\tilde{f}(\mathbf{p}) &= \max_{\sum_i |S_i| = n} \left(\sum_{i=1}^m v_i(S_i) - \mathbf{p}(S_i) \right) + \mathbf{p}([n]) \\
&= \max_{\sum_i |S_i| = n} \left(\sum_{i=1}^m v_i(S_i) - \mathbf{p}(S_i) - \gamma \cdot |S_i| \right) + \mathbf{p}([n]) + \gamma \cdot n \\
&\leq \max_{S_i \subseteq [n]} \left(\sum_{i=1}^m (v_i(S_i) - \mathbf{p}(S_i) - \gamma \cdot |S_i| \right) + \mathbf{p}([n]) + \gamma \cdot n = f(\mathbf{p} + \gamma \cdot \mathbf{1}_{[n]}) \\
&= \sum_{i=1}^m \left[v_i(S_i^*) - \mathbf{p}(S_i^*) - \gamma \cdot |S_i^*| \right] + \mathbf{p}([n]) + \gamma \cdot n \\
&= \sum_{i=1}^m \left[v_i(S_i^*) - \mathbf{p}(S_i^*) \right] + \mathbf{p}([n]) \\
&\leq \max_{\sum_i |S_i| = n} \left(\sum_{i=1}^m v_i(S_i) - \mathbf{p}(S_i) \right) + \mathbf{p}([n]) = \tilde{f}(\mathbf{p}).
\end{aligned}$$

Hence, all inequalities hold with equality, which means in particular that $\tilde{f}(\mathbf{p}) = f(\mathbf{p} + \gamma \cdot \mathbf{1}_{[n]})$ and $S_1^*, \ldots, S_m^*$ maximize $\max_{\sum_i |S_i| = n} \left(\sum_{i=1}^m v_i(S_i) - \mathbf{p}(S_i) \right)$. □

Corollary 1 *Suppose $v_i(j)$ are given and stored as n sorted lists $\{v_i(j)\}_j$ each of which has m elements. Then the greedy algorithm computes a subgradient of $\tilde{f}$ in $n^2 \cdot T_V + \tilde{O}(n^2)$ time.*

Proof This follows directly from Lemma 14 as the gradient of $\sum_{i=1}^{m} \left(v_i(S_i^*) - \mathbf{p}(S_i^*)\right) - \mathbf{p}([n])$ is a subgradient of $\tilde{f}$. □

Corollary 2 *If $\mathbf{p}^*$ minimizes $\tilde{f}$, then $\mathbf{p}^* + \gamma \cdot \mathbf{1}_{[n]}$ is an equilibrium price. Here γ is defined as in Lemma 14 with respect to $\mathbf{p}^*$. Conversely, any Walrasian price $\mathbf{p}^{eq}$ is a minimizer of $\tilde{f}$.*

The proof of the previous corollary is given in the appendix.

Theorem 12 *For gross substitutes, we can find an equilibrium price in time $mn \cdot T_V + O(mn \log m + n^3 \log(mnM) \cdot T_V + n^3 \log^{O(1)}(mnM))$.*

Proof Corollary 2 says that it is enough to find a minimizer of $\tilde{f}$. The algorithmic procedure in Theorem 11 can be used to solve $\tilde{f}$ approximately and then round it to an optimal solution.

To bound the overall running time, we note that: computing $v_i(j)$ and storing them as n sorted lists takes $mn \cdot T_V + O(mn \log m)$ time. By Corollary 1, the separation oracle for $\tilde{f}$ can be implemented in $n^2 \cdot T_V + O(n^2 \log(m))$ time. □

6 Robust Walrasian prices, market coordination and Walrasian allocations

So far we focused on computing Walrasian prices. Now we turn to the other component of Walrasian equilibrium, which is to compute the optimal allocation. If we have only access to an aggregate demand oracle, then computing prices is all that we can hope for, since we have no per-buyer information (in fact, we don't even know the number of buyers). If we have access to a value oracle, computing the optimal allocation is possible.

To convince the reader that this is a non-trivial problem, we show that computing an optimal allocation from Walrasian prices is at least as hard as solving the *Matroid Union Problem*. In the Matroid Union problem we are given m matroids defined over the same ground set $M_i = ([n], \mathcal{B}_i)$ and a promise that there exist basis $B_i \in \mathcal{B}_i$ such that $[n] = \cup_i B_i$. The goal is to find those bases. Now, consider the following mapping to the problem of computing an optimal allocation: consider m agents with valuations over a set $[n]$ of items such that $v_i = r_{M_i}$, i.e. the rank of matroid M_i (matroid rank functions are known to be gross substitute valuations [26]). The price vector $\mathbf{1}$ is clearly a vector of Walrasian prices. Finding the optimal allocation, however, involves finding $\mathbf{S} = (S_1, \ldots, S_m)$ maximizing $\sum_i r_{M_i}(S_i)$.

The previous paragraph hopefully convinced the reader that finding an allocation is not always a simple task even if we know the prices. One approach to solve this problem is based on a modification of standard matroid union algorithms.

The second approach, which we discuss here in details, is based on convex programming and reveals an important structural property of gross substitute valuations

that might be of independent interest. Incidentally, this also answers an open question of Hsu et al. [19] who asked what are the conditions for markets to be perfectly coordinated using prices. More precisely, they showed that under some genericity condition the minimal Walrasian price *for a restricted class of gross substitutes* induces an overdemand at most 1 for each item. On the other hand, our argument in this section says that under the same condition *almost every* Walrasian prices *for any gross substitutes* have *no* overdemand, i.e. the market is perfectly coordinated. This follows from the fact that the polytope of Walrasian prices have nonempty interior and that interior Walrasian price induces no overdemand (see Sect. 6.2).

Next, we review two combinatorial lemmas that will be fundamental for the rest of this section and the next one:

6.1 Two combinatorial lemmas

One of the most useful (and largely unknown) facts about gross substitutes is the following analogue to the Unique Matching Theorem for matroids. The version of this theorem for gross substitutes is due to Murota [33,34] and it was originally proved in the context of valuated matroids, which are known to be equivalent to gross substitutes under a certain transformation. We refer the reader to Lemma 10.1 in [26] for a proof of this lemma in the language of gross substitute valuations. We also refer to chapter 5 of [35] for a comprehensive treatment of such exchange properties.

Lemma 15 (Unique Minimum Matching Lemma) *Let $v : 2^{[n]} \to \mathbb{R}$ be a valuation satisfying gross substitutes, $S \subseteq [n]$, $A = \{a_1, \ldots, a_k\} \subseteq S$, $B = \{b_1, \ldots, b_k\} \subseteq [n] - S$. Consider weighted bipartite graph G with left set A, right set B and edge weights $w_{a_i,b_j} = v(S) - v(S \cup b_j - a_i)$. If $M = \{(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\}$ is the unique minimum matching in G, then:*

$$v(S) - v(S \cup B - A) = \sum_{j=1}^{k} v(S) - v(S \cup b_j - a_j).$$

Lastly, we state a purely combinatorial lemma that is commonly used in conjunction with the previous lemma. We present a sketch of the proof in the appendix and refer to [26,35] for a complete proof.

Lemma 16 *Let $G = (V, E, w)$ be a weighted directed graph without negative weight cycles. Let C be the cycle of minimum number of edges among the cycles with minimum weight. Let $M := \{(u_1, v_1), \ldots, (u_t, v_t)\}$ be a set of non-consecutive edges in this cycle, $U = \{u_1, \ldots, u_t\}$ and $V = \{v_1, \ldots, v_t\}$. Construct a bipartite graph G' with left set U, right set V and for each edge from $u \in U$ to $v \in V$ in the original graph, add an edge of the same weight to G'. Under those conditions, M forms a unique minimum matching in G'. The same result holds for a path P of minimum length among all the minimum weight paths between a pair of fixed nodes.*

6.2 Robust Walrasian prices and market coordination

Hsu et al. [19] raise the following important question: when is it possible to find Walrasian prices that coordinate the market? A vector of Walrasian prices $\mathbf{p}$ is said to coordinate the market if each agent has a unique demanded bundle under $\mathbf{p}$ and those bundles clear the market. If this happens, we say that this vector is *robust*.

Definition 9 *(Robust Walrasian Prices)* A price vector $\mathbf{p}$ is said to be a vector of *robust Walrasian prices* for a certain market if $D(i, \mathbf{p}) = \{S_i\}$ and $\mathbf{S} = (S_1, \ldots, S_m)$ form a partition of the items.

Notice that by the Second Welfare Theorem (Lemma 1), if the optimal allocation is not unique, then no vector of Walrasian prices is robust, since each vector of Walrasian prices support all the allocations. If the optimal allocation is unique, on the other hand, then we show that a vector of robust Walrasian prices always exists. Moreover, the set of Walrasian prices forms a full-dimensional convex set in which all interior points are robust.

Theorem 13 *If there is a unique partition $\mathbf{S} = (S_1, \ldots, S_m)$ maximizing $\sum_i v_i(S_i)$, then there exist a vector $\mathbf{p}$ such for all $\mathbf{p}' \in \prod_j [p_j - \frac{1}{2n}, p_j + \frac{1}{2n}]$ are Walrasian. In particular, the set of Walrasian prices is a full-dimensional convex set and all price vectors in its interior are robust Walrasian prices.*

The proof involves the concept of the *exchange graph*, which was first introduced by Murota in [34] and characterizes the set of all Walrasian prices as the dual variables of the shortest path polytope for a certain graph. Given an optimal allocation $\mathbf{S} = (S_1, \ldots, S_m)$, the Second Welfare Theorem (Lemma 1) combined with the characterization of gross substitute functions from Discrete Convex Analysis (Definition 8) tells us that the set of Walrasian prices can be characterized by:

$$P = \left\{ \mathbf{p} \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} v_i(S_i) \geq v(S_i - j) + p_j, & \forall i \in [m],\ j \in S_i \\ v_i(S_i) \geq v(S_i \cup k) - p_k, & \forall i \in [m],\ k \notin S_i \\ v_i(S_i) \geq v(S_i \cup k - j) - p_k + p_j, & \forall i \in [m],\ j \in S_i, k \notin S_i \end{array} \right\}$$

Which is clearly a convex set defined by $O(\sum_i |S_i|\, n) = O(n^2)$ inequalities. A nice combinatorial interpretation of this polytope is that it corresponds to the set of potentials in a graph.

To make the construction nicer, augment the items with m dummy items, one for each buyer. The set of items becomes $[n] \cup [m]$, and the valuations are extended to the subsets of $[n] \cup [m]$ in a way that agents completely ignore the dummy items, i.e., for $T \subset [n] \cup [m]$, $v_i(T) = v_i(T \cap [n])$. Also, augment S_i to contain the dummy item for buyer i. Under this transformation we can simplify the definition of P to:

$$P = \left\{ \mathbf{p} \in \mathbb{R}^n \;\middle|\; v_i(S_i) \geq v(S_i \cup k - j) - p_k + p_j, \forall i \in [m],\ j \in S_i, k \notin S_i \right\}$$

since we can represent adding and removing an item as a swap with a dummy item. Under this transformation, construct a directed graph with one node for each item in

$[n]$. For each $i \in [m]$, $j \in S_i$ and $k \notin S_i$, add an edge (j,k) with weight $w_{j,k} = v_i(S_i) - v_i(S_i \cup k - j)$.

Since the allocation $\mathbf{S} = (S_1, \ldots, S_m)$ is optimal, there exists at least one vector of Walrasian prices $\mathbf{p} \in P$. This guarantees that the exchange graph has no negative cycles, since for any cycle $C = \{(j_1, j_2), \ldots, (j_t, j_1)\}$, we can bound the sum of weights by $\sum_r w_{j_r, j_{r+1}} \geq \sum_r p_{j_r} - p_{j_{r+1}} = 0$, where the inequality follows from the definition of P and the definition of the weights. Now we argue that the exchange graph can't contain any cycles of zero weight:

Lemma 17 *If* $\mathbf{S}$ *is the unique optimal allocation, then the exchange graph has no cycles of zero weight.*

Proof If there were cycles of zero weight, let C be the cycle of zero weight of minimum length. Now, let $C_i = \{(j_1, t_1), \ldots, (j_a, t_a)\}$ be the edges (j,t) in C with $j \in S_i$ and (consequently) $t \notin S_i$. Now, define $S_i' = S_i \cup \{t_1, \ldots, t_a\} - \{j_1, \ldots, j_a\}$. Notice that we performed the swaps prescribed by the cycle, so each moved item was removed from one set and added to another and as a result, $\mathbf{S}' = (S_1', \ldots, S_m')$ is still a partition of the items. Using Lemmas 15 and 16 we get that:

$$v_i(S_i') - v_i(S_i) = \sum_{r=1}^{a} v_i(S_i \cup t_r - j_r) - v_i(S_i) = \sum_{r=1}^{a} w_{j_r t_r}$$

therefore:

$$\sum_i v_i(S_i') - \sum_i v_i(S_i) = \sum_{e \in C} w_e = 0$$

and so $\mathbf{S}' = (S_1', \ldots, S_m')$ is an alternative optimal allocation, contradicting the uniqueness of $\mathbf{S}$. □

Now we are ready to prove the Theorem 13:

Proof of Theorem 13 Since we know there are no zero weight cycles and all the edge weights are integral, all cycles have weight at least 1. Now perform the following operation: for each vertex $j \in [n]$ in the directed graph, split it into j_1 and j_2 with an edge between j_1 and j_2 with weight $-\frac{1}{n}$. Make all incoming edges to j be incoming to j_1 and all outgoing edges from j to be outgoing from j_2. The resulting graph has again no cycles of negative weight, since the new edges can decrease each cycle by at most 1.

Therefore, it is possible to find a potential in this graph. A potential of a weighted graph with edge weights w_{jt} is a function ϕ from the nodes to $\mathbb{R}$ such that $\phi(t) \leq \phi(j) + w_{jt}$. It can be easily computed by running a shortest path algorithm from any fixed source node and taking the distance from source node to j as $\phi(j)$. For the particular case of the graph constructed, it is useful to take the source as one of the dummy nodes.

After computing a potential from the distance from a dummy node to each node, define the price of j as $p_j = \phi(j_2)$. By the definition of the potential for each edge (j,t) in the graph:

$$p_t = \phi(t_2) \leq \phi(t_1) - \frac{1}{n} \leq \phi(j_2) + w_{jt} - \frac{1}{n} = p_j + w_{jt} - \frac{1}{n}.$$

This means in particular that all inequalities that define P are valid with a slack of $\frac{1}{n}$. Therefore, changing any price by at most $\frac{1}{2n}$ still results in a valid Walrasian equilibrium. This completes the proof of the first part of the theorem.

Since P contains a cube, then it must be a full-dimensional convex body. Finally, let's show that every price vector in the interior of P is a vector of robust Walrasian prices. By the second welfare theorem (Lemma 1), $S_i \in D(i, \mathbf{p})$ for all $\mathbf{p} \in P$. Now, assume that for some point in the interior, there is $S_i' \in D(i, \mathbf{p})$, $S_i' \neq S_i$. Then either: (i) There is $j \in S_i' - S_i$. We decrease the price of j by ϵ so that S_i' becomes strictly better than S_i, i.e. $S_i \notin D(i, \mathbf{p} - \epsilon \mathbf{1}_j)$, which contradicts the second welfare theorem since $\mathbf{p} - \epsilon \mathbf{1}_j \in P$. (ii) There is $j \in S_i - S_i'$. We increase the price of j by ϵ so that S_i' becomes strictly better than S_i, i.e. $S_i \notin D_i(i, \mathbf{p} + \epsilon \mathbf{1}_j)$, which again contradicts the second welfare theorem since $\mathbf{p} + \epsilon \mathbf{1}_j \in P$. □

6.3 Computing Optimal Allocation

Theorem 13 guarantees that if the optimal allocation is unique, then the set of Walrasian prices has large volume. Since the set of Walrasian prices corresponds to the set of minimizers of the market potential $f(\mathbf{p})$, then there is a large region where zero is the unique subgradient of f. In such situations, convex optimization algorithms are guaranteed to eventually query a point that has zero subgradient. The point $\mathbf{p}$ queried corresponds to a set of Walrasian prices and the optimal allocation can be inferred from the subgradient (recall Theorem 4).

Our strategy is to perturb the valuation functions in such a way that the optimal solution is unique and that it is still a solution of the original problem. It is important for the reader to notice that this is a different type of perturbation than the one used in previous sections. While previously we perturbed the objective of the dual program, here we are effectively perturbing the objective of the primal program. One major difference is that if we perturb the objective of the dual, we can still compute the subgradient of $\hat{f}$ in PC using the aggregate demand oracle. If we perturb the objective of the primal, we no longer can compute subgradients using the aggregate demand oracle. With value oracles, however, this is still possible to be done.

To perturb the primal objective function, we use the isolation lemma, a standard technique to guarantee a unique optimum for combinatorial problems.

Lemma 18 (Isolation Lemma [38]) *Let* $\mathbf{w} \in [N]^n$ *be a random vector where each coordinate* w_i *is chosen independently and uniformly over* $[N]$. *Then for any arbitrary family* $\mathcal{F} \subseteq 2^{[n]}$, *the problem* $\max_{S \in \mathcal{F}} \sum_{i \in S} w_i$ *has a unique optimum with probability at least* $1 - n/N$.

For our application, we would like a unique optimum to the problem of maximizing $\sum_{i=1}^{m} v_i(S_i)$ over the partition $(S_1, \ldots, S_m)$. To achieve this, we first replace v_i by $\tilde{v}_i(S) = B v_i(S) + w_i(S)$ where B is some big number to be determined and $w_i(j)$ is set as in the isolation lemma (with N to be determined as well).

Lemma 19 *By setting $B = 2nN$, $N = mn^{O(1)}$ and sampling $w_i(j)$ uniformly from $[N]$ for each $i \in [m]$ and $j \in [n]$, then with probability $1 - 1/n^{O(1)}$, there is a unique partition $(S_1, \ldots, S_m)$ maximizing $\sum_{i=1}^{m} \tilde{v}_i(S_i)$, for $\tilde{v}_i(S) = B \cdot v_i(S) + w_i(S)$.*

Proof Let $(S'_1, \ldots, S'_m)$ be an optimal solution w.r.t the original problem. We first show that any suboptimal partition $(S_1, \ldots, S_m)$ cannot be optimal for the perturbed problem. Since v_i assume integer values, we have $\sum_{i=1}^{m} v_i(S'_i) \geq \sum_{i=1}^{m} v_i(S_i) + 1$. Now:

$$\begin{aligned}\sum_{i=1}^{m} \tilde{v}_i(S'_i) &\geq B + \sum_{i=1}^{m} \tilde{v}_i(S_i) + \sum_{i=1}^{m} w_i(S_i) - \sum_{i=1}^{m} w_i(S'_i) \\ &\geq B + \sum_{i=1}^{m} \tilde{v}_i(S_i) - nN > \sum_{i=1}^{m} \tilde{v}_i(S_i).\end{aligned}$$

which shows that a suboptimal solution to the original problem cannot be optimal for the perturbed one.

Now, consider all partitions $(S'_1, \ldots, S'_m)$ that are optimal for the original problem and identify each optimal partition with a subsets of $[mn] = \{(i, j); i \in [m], j \in [n]\}$ in the natural way: add (i, j) to the subset if $j \in S'_i$. This family of subsets corresponds to $\mathcal{F} \subseteq 2^{[mn]}$ in the statement of the Isolation Lemma and $w_i(j)$ corresponds to $\mathbf{w}$. The result then follows from applying that lemma. □

The strategy to find an optimal allocation is to perturb the valuation functions, then search for an interior point in the set of minimizers of the market potential function f. When we find such a point we can obtain a vector of Walrasian prices for the original market by rounding and the optimal allocation by inspecting the subgradient. To get the desired running time, we need to apply those ideas to the regularized potential function $\tilde{f}$ (see RC in Sect. 5) instead of the original one. To apply this to the regularized potential we need an extra lemma:

Lemma 20 *Let $\mathbf{p}$ be an interior point of the set of minimizers of the regularized potential function $\tilde{f}$, then the allocation $(S_1^*, \ldots, S_m^*)$ produced by the* AllGreedy *algorithm in Lemma 14 is an equilibrium allocation.*

Proof If $(S_1^*, \ldots, S_m^*)$ is a partition over the items, then it is an optimal allocation by part 2 of Lemma 14. To show that it is a partition, observe that since $(S_1^*, \ldots, S_m^*)$ is a maximizer of $\sum_i v_i(S_i) - \mathbf{p}(S_i)$ subject to $\sum_i |S_i| = n$, then we can use it to build a subgradient $\mathbf{g}$ of $\tilde{f}$ such that $g_j = -1 + \left|\{i; j \in S_i^*\}\right|$. Since $\mathbf{p}$ is an interior point of the set of minimizers, the subgradient must be zero and therefore $\left|\{i; j \in S_i^*\}\right| = 1$ for all j. □

Theorem 14 *For gross substitute valuation, we can find a Walrasian equilibrium, i.e. allocation and prices, in time $O((mn + n^3)T_V \log(nmM) + n^3 \log^{O(1)}(mnM))$.*

Proof Use Lemma 19 to perturb the valuation functions and obtain $\tilde{v}_i$ which are still gross substitutes (since they are the sum of a gross substitute valuation and an

additive valuation) and there is a unique optimal allocation. By Lemma 13, the set of minimizer of the market potential function f contains a box of width $1/n$. Since the set of minimizers of the market potential f is contained in the set of minimizers of the regularized potential $\tilde{f}$, then its set of minimizers also contains a box of width $1/n$. We also know that it is contained in the box $[-Mmn^{O(1)}, Mmn^{O(1)}]^n$ since $|\tilde{v}_i(S)| \leq O(Mmn^{O(1)})$.

If we apply the algorithm in Theorem 5 with $\delta = O(1/n^{O(1)})$ to optimize the regularized potential $\tilde{f}$ then we are guaranteed to query a point in the interior of minimizers as otherwise the algorithm would certify that there is $\mathbf{a}$ with $\|\mathbf{a}\|_2 = 1$ such that $\max_{\mathbf{p}\in P} \mathbf{a}\cdot\mathbf{p} - \min_{\mathbf{p}\in P} \mathbf{a}\cdot\mathbf{p} \leq 1/n^{O(1)}$, where P is the set of minimizers.

Finally, we can obtain the optimal allocation for the perturbed using Lemma 14, which is an optimal allocation to the original market according to Lemma 19. □

7 Combinatorial approach to Walrasian Equilibrium for Gross Substitutes

In a sequence of two foundational papers [33,34], Murota shows that the assignment problem for *valuated matroids*, a class of functions introduced by Dress and Wenzel [12] can be solved in strongly polynomial time. We show how this algorithm can be used to obtain an $\tilde{O}(nm + n^3)$ strongly polynomial time algorithm for problem of computing a Walrasian equilibrium for gross substitute valuations. Our contribution is two-fold: first we map the Walrasian equilibrium problem on gross substitute valuations to the assignment problem on valuated matroids and analyze its running time. The straightforward mapping allows us to obtain a strongly polynomial time algorithm with running time $O(mn^3 \log(m+n))$. We note that a different way to reduce the Walrasian equilibrium problem to a standard problem in discrete convex analysis is to map it to the M-convex submodular flow problem as done in Murota and Tamura [31]. We choose to reduce it to the assignment problem on valuated matroids since its running time is simpler to analyze.

Inspired by our $\widetilde{O}(mn + n^3)$ algorithm, we revisit Murota's algorithm and propose two optimizations that bring the running time down to $O((mn + n^3)\log(m+n))$. Murota's algorithm works by computing augmenting paths in a structure known as the *exchange graph*. First we show that for the Walrasian equilibrium problem, this graph admits a more succinct representation. Then we propose a data structure to amortize the cost of some operations across all iterations.

In Sect. 7.1 we define valuated matroids and the assignment problem for valuated matroids. Then we describe Murota's algorithm for this problem. We also discuss the relation between the assignment problem for valuated matroids and the welfare problem for gross substitutes. The goal of Sect. 7.1 is to provide the reader with the historical context for this result.

The reader interested solely in the welfare problem is welcome to skip to Sect. 7.2 which can be read independently, without any mention to valuated matroids or the assignment problem. A complete proof is given in that section.

7.1 The assignment problem for valuated matroids

A *valuated matroid* is an assignment of weights to basis of a matroid respecting some valuated analogue of the exchange property.

Definition 10 *(Valuated matroid)* Let $\mathcal{B}$ be the set of basis of a matroid $\mathcal{M} = (V, \mathcal{B})$. A valuated matroid is a function $\omega : \mathcal{B} \to \mathbb{R} \cup \{\pm\infty\}$ such that for all $B, B' \in \mathcal{B}$ and $u \in B - B'$ there exists $v \in B' - B$ such that:

$$\omega(B) + \omega(B') \leq \omega(B \cup v - u) + \omega(B' \cup u - v)$$

Valuated matroids are related to gross substitutes by the following one-to-one correspondence. We refer the reader to Lemma 7.4 in [26] for a proof.

Proposition 1 *A valuation function $v : 2^{[n]} \to \mathbb{R}$ is a gross substitutes valuation function iff $\omega : \binom{[2n]}{n} \to \mathbb{R}$ defined by $\omega(S) = v(S \cap [n])$ is a valuated matroid defined over the basis of the n-uniform matroid on $2n$ elements.*

Now, we are ready to define the assignment problem for valuated matroids:

Definition 11 *(Valuated matroid assignment problem)* Given two sets V_1, V_2, matroids $\mathcal{M}_1 = (V_1, \mathcal{B}_1)$ and $\mathcal{M}_2 = (V_2, \mathcal{B}_2)$ of the same rank, valuated matroids $\omega_1 : \mathcal{B}_1 \to \mathbb{R}$ and $\omega_2 : \mathcal{B}_2 \to \mathbb{R}$ and a weighted bipartite graph $G = (V_1 \cup V_2, E, w)$, find a matching M from V_1 to V_2 maximizing:

$$w(M) + \omega_1(M_1) + \omega_2(M_2)$$

where M is a subset of edges of E forming a matching and M_1 and M_2 are the sets of endpoints of M in V_1 and V_2 respectively.

Murota gives two strongly-polynomial time algorithms based on network flows for the problem above in [34]—the first based on cycle-cancellations and the second based on flow-augmentations. Although the running time is not formally analyzed in his paper, it is possible to see that his algorithm (more specifically the algorithm *Augmenting algorithm with potentials* in Section 3 of [34]) has running time $\tilde{O}(R \cdot (|E| + R \cdot (|V_1| + |V_2|)))$ for $R = \text{rank}(\mathcal{M}_1) + \text{rank}(\mathcal{M}_2)$.

First, we show a simple reduction from the welfare problem for gross substitutes to this problem. Given m gross substitute valuation functions $v_i : 2^{[n]} \to \mathbb{R}$, define the following instance of the valuated matroid assignment problem: define the first matroid as $\binom{[mn]}{n}$ i.e. the n-uniform matroid on mn elements. Interpret the elements of $[mn]$ as "the allocation of item j to agent i" for each pair (i, j). Following this interpretation, each $S \subseteq [mn]$ can be seen as $S = \cup_{i=1}^{m} S_i$ where S_i are the elements assigned to agent i. This allows us to define for each $S \in \binom{[mn]}{n}$, $\omega_1(S) = \sum_i v_i(S_i)$. For the second matroid, let $\mathcal{M}_2 = ([n], 2^{[n]})$ and $\omega_2(S) = 0$ for all S. Finally, define the edges of E such that for each $j \in [n]$, the j-th element of $[n]$ are connected to the element (i, j) in $[mn]$ for each i.

One needs to prove that ω_1 satisfies the properties defining a valuated matroid, but this can be done using the transformations described in [26]. We omit this proof since we are giving a self-contained description of the algorithm in the next section.

In the construction shown below, $|V_1| = |E| = mn$, $|V_2| = n$ and $\text{rank}(\mathcal{M}_1) = \text{rank}(\mathcal{M}_2) = n$. This leads to an $\tilde{O}(m \cdot n^3)$ strongly polynomial time algorithm for the welfare problem.

7.2 Gross substitutes welfare problem in $\tilde{O}(mn + n^3)$ time

In this section we give a self-contained description of a specialized version of Murota's algorithm for the gross substitutes welfare problem and show that the running time of $\tilde{O}(m \cdot n^3)$ can be improved to $\tilde{O}(mn + n^3)$. Murota's algorithm for the case of generic valuated matroids can be quite complicated. Since the underlying matroids of our problem are simple (uniform matroids) and the functions being optimized have additional structure, it is possible to come up with a simpler algorithm. Our presentation also makes the algorithms accessible to the reader not familiar with discrete convex analysis and the theory of valuated matroids.

We consider the setting in which a set $[n]$ of items needs to be allocated to m agents with monotone valuation functions $v_i : 2^{[n]} \to \mathbb{R}$ satisfying the gross substitutes condition. Consider the intermediary problem of computing the optimal allocation for the first k items (in some arbitrary order):

$$\max_i v_i(S_i) \quad \text{s.t.} \quad \cup_i S_i \subseteq [k] := \{1, 2, \ldots, k\} \quad \text{and} \quad S_i \cap S_j = \varnothing \text{ for } i \neq j. \tag{I_k}$$

The central idea of the algorithm is to successively solve $(I_1), (I_2), \ldots, (I_n)$ using the solution of (I_{k-1}) to compute (I_k). We will show how a solution to (I_k) can be computed from a solution of (I_{k-1}) via a shortest path computation in a graph with $\tilde{O}(m + n^2)$ edges.

A solution for problem (I_{k-1}) consists of an allocation $S = (S_1, \ldots, S_m)$ of items in $[k-1]$ to agents $1, \ldots, m$ and a price vector $p_1, \ldots, p_{k-1}$ that certifies that the allocation is optimal. Since optimality for gross substitutes can be certified by checking that no agent wants to add an item, remove an item or swap an item (Definition 8) then S, p need to satisfy the following conditions for every agent i and every $j \notin S_i$ and $j' \in S_i$:

$$v_i(S_i \cup j) - p_j \leq v_i(S_i) \tag{ADD}$$

$$v_i(S_i - j') + p_{j'} \leq v_i(S_i) \tag{REMOVE}$$

$$v_i(S_i \cup j - j') - p_j + p_{j'} \leq v_i(S_i). \tag{SWAP}$$

7.2.1 Exchange graph

The first step to solve (I_k) is to build a combinatorial object called the *exchange graph* using the solution of (I_{k-1}), expressed as a pair S, p. We define a weighted directed graph on $V = [k] \cup \{\phi_1, \ldots, \phi_m\}$. Intuitively, we can think of ϕ_i as an "empty spot" in the allocation of agent i. We add edges as follows:

- (t, j) for all items t and j not acquired by the same agent under S. If i is the agent holding item t, then the edge represents the change in utility (under price p) for agent i to swap his item t by j:

$$w_{tj} = v_i(S_i) - v_i(S_i \cup j - t) + p_j - p_t$$

- (ϕ_i, j) for all items $j \notin S_i$. It represents the change in utility for i to add item j:

$$w_{\phi_i j} = v_i(S_i) - v_i(S_i \cup j) + p_j.$$

Notice that problem (I_{k-1}) only defines prices for $1, \ldots, k-1$. So for the construction above to be well defined we need to define p_k. We will set p_k in a moment, but before that, note that for all the edges not involving k, $w \geq 0$ which follow by the fact that p is a certificate of optimality for S and therefore conditions ADD and SWAP hold. Finally, notice that there are only directed edges from k to other nodes, so p_k always appear with positive sign in w. So, we can set p_k large enough such that all edges have non-negative weights, in particular, set:

$$p_k = \max \left\{ \max_{i \in [m]; t \in S_i} [v_i(S_i \cup k - t) - v_i(S_i) + p_t], \max_{i \in [m]} [v_i(S_i \cup k) - v_i(S_i)] \right\}.$$

7.2.2 Updating prices and allocations via shortest path

After the exchange graph is built, the algorithm is trivial: compute the minimal-length shortest path from some ϕ_i (i.e. among all paths of minimum weight between ϕ_i and k, pick the one with minimum length). Since the edges are non-negative, the shortest path can be computed in the order of the number of edges using Dijkstra's algorithm in $O(|E| + |V| \cdot \log |V|)$ where $|E|$ is the number of edges in the graph and $|V|$ is the number of vertices. Dijkstra's algorithm can be easily modified to compute the minimum weight path with shortest length using the following idea: if weights are integers, then substitute weights w_{ij} by $w_{ij} - \epsilon$. In the end, round the solution up. Or more formally, run Dijkstra in the ordered ring $(\mathbb{Z}^2, +, <)$ with weights $(w_{ij}, 1)$ where $+$ is the componentwise sum and $<$ is the lexicographic order.

Let P be the path output by Dijkstra. Update the allocation by performing the swaps prescribed by P. In other words, if edge $(t, j) \in P$ and $t \in S_i$, then we swap t by j in S_i. Also, if edge $(\phi_i, j) \in P$ we add j in S_i. Formally, let $(t_r, j_r)_{r=1..a}$ be all the edges in P with $t_r \in S_i$ or $t_r = \phi_i$. Then we update S_i to $S_i \cup \{j_1, \ldots, j_a\} - \{t_1, \ldots, t_a\}$.

The execution of Dijkstra also produces a certificate of optimality of the shortest path in the form of the minimum distance from some ϕ_i to any given node. So, there is a distance function d such that

$$d(\phi_i) = 0, \qquad d(j) \leq d(\phi_i) + w_{\phi_i j}, \qquad d(j) \leq w_{tj} + d(t).$$

Moreover, for all edges (t, j) and (ϕ_i, j) in the shortest path P, this holds with equality, i.e.: $d(j) = d(\phi_i) + w_{\phi_i j}$ and $d(j) = w_{jt} + d(t)$. Update the price of each item j from p_j to $p_j - d(j)$.

7.2.3 Running time analysis

Before we show that each iteration produces an optimal pair of allocation S and prices p for problem (I_k) we analyze the running time.

The exchange graph for problem (I_k) as previously described has $O(mk + k^2)$ edges. Running Dijkstra's algorithm on this graph has running time $\tilde{O}(mk + k^2)$ for (I_k), which corresponds to $\tilde{O}(mn^2 + n^3)$ time overall.

In order to get the overall running time down to $\tilde{O}(mn + n^3)$ we need one extra observation. Since we want to compute the shortest path from any of the ϕ_i nodes to k, we can collapse all ϕ_i nodes to a single node ϕ. Now, for any given node j:

$$w_{\phi j} = \min_i w_{\phi_i j} = p_j + \min_i [v_i(S_i) - v_i(S_i \cup j)].$$

Now, the graph is reduced to $O(k^2)$ edges for problem (I_k). So, Dijkstra can be computed in $\tilde{O}(k^2)$. We are only left with the task to compute $w_{\phi j}$. Our task is to compute $\min_i [v_i(S_i) - v_i(S_i \cup j)]$. This can be divided in two parts:

1. *active agents*: the minimum among the agents i for which $S_i \neq \varnothing$. There are at most k of those, so we can iterate over all of them and compute the minimum explicitly;
2. *inactive agents*: the minimum over all agents with $S_i = \varnothing$. In order to do so we maintain the following data structure: in the first iteration, i.e. in (I_1), we compute $v_i(\{j\})$ for every i, j (which takes $O(mn)$ time) and keep for each item j a sorted list $\mathtt{L}_j$ in decreasing order of $v_i(j)$ for all i.

   In the end of each iteration, whenever an inactive agent i becomes active (i.e. we allocate him an item), we remove them from $\mathtt{L}_j$ for all j. This operation takes $O(n)$ time to go over all lists.

   Now, once we have this structure, we can compute the minimum among the inactive buyers $\min_i [v_i(S_i) - v_i(S_i \cup j)] = -\max_i v_i(\{j\})$ by looking at the minimum element of the list $\mathtt{L}_j$. Therefore, even though we need to pay $O(mn)$ time in (I_1). In each subsequent iteration we pay only $O(n)$ to update lists $\mathtt{L}_j$ and then we can make query the value of $w_{\phi j}$ in constant time.

This leads to a running time of $O(mn)$ in (I_1) and $\tilde{O}(n + k^2)$ in each subsequent iteration, leading to an overall running time of $\tilde{O}(mn + n^3)$. We also note that for each edge of the graph, we query the value oracle once. So the oracle complexity is $O(nm + n^3)$ value oracle calls.

7.2.4 Correctness

We are left to argue that the solution (S, p) produced by the algorithm is indeed a valid solution for problem (I_k). This can be done by checking that the price vector p obtained certifies the optimality of S. The main ingredients for the proof are Lemmas 15 and 16. We encourage the reader to revisit the statement of those lemmas before reading the proof of the following theorem.

Theorem 15 *Let S_i^k, p_j^k be the solution of problem (I_k), then for all agent i and all $S' \subseteq [k]$,*

$$v_i(S_i^k) - p^k(S_i^k) \geq v_i(S') - p^k(S').$$

Proof Let S_i^{k-1}, p_j^{k-1} be the solution of problem (I_{k-1}) and let S_i^k, p_j^k be the solution of problem (I_k). If $d(\cdot)$ is the distance function returned by Dijkstra in (I_k), then $p_j^k = p_j^{k-1} - d(j)$ and we also know that $\tilde{w}_{jt} = w_{jt} + d(t) - d(j) \geq 0$ and $\tilde{w}_{\phi_i j} = w_{\phi_i j} - d(j) \geq 0$ by the observation in the end of the Sect. 7.2.2. This implies that for all i, for all $j \notin S_i^{k-1}$ and $t \in S_i^{k-1}$, it holds that:

$$\tilde{w}_{tj} = v_i(S_i^{k-1}) - v_i(S_i^{k-1} \cup j - t) + p_j^k - p_t^k \geq 0$$
$$\tilde{w}_{\phi_i j} = v_i(S_i^{k-1}) - v_i(S_i^{k-1} \cup j) + p_j^k \geq 0$$

This means that properties ADD and SWAP are satisfied. To see that REMOVE is also satisfied for $j < k$ since $v_i(S_i^{k-1}) \geq v_i(S_i^{k-1} - j) + p_j^{k-1}$ for all $j \in S_i^{k-1}$. Since $p_j^k \leq p_j^{k-1}$, this condition must continue to hold.

This means that under the price vector p^k the bundles S_i^{k-1} is still the demanded bundle by agent i (by Definition 8). This means in particular that for all $S' \subseteq [k]$:

$$v_i(S_i^{k-1}) - p^k(S_i^{k-1}) \geq v_i(S') - p^k(S')$$

Now, let $(t_r, j_r)_{r=1..a}$ be the set of edges in the path P output by Dijkstra where $t_r \in S_i^{k-1}$. Then $S_i^k = S_t^{k-1} \cup \{j_1, \ldots, j_a\} - \{t_1, \ldots, t_a\}$.

Using Lemma 16, we note that $(t_1, r_1), \ldots, (t_a, r_a)$ is a unique minimum matching in the sense of Lemma 15. Therefore:

$$v_i(S^i k - 1_i) - v_i(S_i^k) = \sum_{r=1}^{a} v_i(S_i^{k-1}) - v_i(S_i^{k-1} \cup j_r - t_r)$$

Summing $-p^k(S_i^{k-1}) + p^k(S_i^k)$ on both sides, we get:

$$[v_i(S_i^{k-1}) - p^k(S_i^{k-1})] - [v_i(S_i^k) - p^k(S_i^{k-1})] = \sum_{r=1}^{a} \tilde{w}_{t_r j_r} = 0$$

Therefore:

$$v_i(S_i^k) - p^k(S_i^{k-1}) = v_i(S_i^{k-1}) - p^k(S_i^{k-1}) \geq v_i(S') - p^k(S')$$

as desired. □

7.2.5 Descending Auction View

One can reinterpret the procedure above as a descending auction. Initially all items very large price (say like the price set for p_k in the beginning of phase k). Each shortest path computation produces a distance function d that dictates how each price should decrease. Indeed, they monotonically decrease until we reach a Walrasian equilibrium.

We note that it is important in this algorithm that we compute in each step both a primal and a dual solution. Without the dual solution (the price vector), it is still possible to carry out the shortest path computation, but since the edges in the path can have mixed signs, Dijkstra's algorithm is no longer available and one needs to pay an extra factor of n to run Bellman-Ford's algorithm.

A Missing Proofs

Proof of Lemma 1 Let $\mathbf{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(m)})$ be a valid allocation that achieves the optimal social welfare. Then since $x^{(i)} \in D(i, \mathbf{p})$, we have

$$v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)} \geq v_i(y^{(i)}) - \mathbf{p} \cdot y^{(i)}.$$

Summing up, we get

$$\sum_i \left(v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)} \right) \geq \sum_i \left(v_i(y^{(i)}) - \mathbf{p} \cdot y^{(i)} \right).$$

the crucial observation is that $\sum_i \mathbf{p} \cdot x^{(i)} = \sum_i \mathbf{p} \cdot y^{(i)} = \sum_j p_j s_j$. herefore the inequality above simplifies to

$$\sum_i v_i(x^{(i)}) \geq \sum_i v_i(y^{(i)}),$$

i.e. the social welfare of $\mathbf{x}$ is at least that of $\mathbf{y}$. But since $\mathbf{y}$ gives the optimal social welfare, we must then have equality and $\mathbf{x}$ also achieves the optimum.

For the second part, notice that since the last equation holds with equality, then the previous equations should also hold, therefore: $\sum_i v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)} = \sum_i v_i(y^{(i)}) -$

$\mathbf{p} \cdot y^{(i)}$, which says that x_i is also a favorite bundle of i under price vector $\mathbf{p}$. Therefore, $(\mathbf{x}, \mathbf{p})$ form a Walrasian equilibrium. □

Proof of Lemma 2 If $(\mathbf{p}^{\text{eq}}, \mathbf{x})$ is a Walrasian equilibrium it is straightforward to check that setting $\mathbf{p} = \mathbf{p}^{\text{eq}}$, $u_i = \max_{x_i \in [\![s]\!]} v_i(x) - \mathbf{p}^{\text{eq}} \cdot x$ and $z_{i,x} = 1$ when $x = x^{(i)}$ and zero otherwise, we have a primal dual pair of feasible solutions with the same value. Conversely, if the primal program has an integral solution, the definition of Walrasian equilibrium can be obtained from the complementarity conditions.

If the primal program has an optimal integral solution $\mathbf{x}$, then for every solution $(\mathbf{p}, \mathbf{u})$ to the dual program: $\sum_i v_i(x^{(i)}) = \sum_i u_i + \mathbf{p} \cdot \mathbf{s} \geq \sum_i v_i(x^{(i)}) + \mathbf{p} \cdot x_i \geq \sum_i v_i(x^{(i)})$ and therefore all inequalities must hold with equality, so in particular $x^i \in D(v_i, \mathbf{p})$, so $\mathbf{p}$ is a vector of Walrasian prices. Conversely, if $\mathbf{p}$ is a vector of Walrasian prices then $(\mathbf{x}, \mathbf{p})$ is Walrasian equilibrium by the Second Welfare Theorem (Lemma 1). Therefore by setting $u_i = v_i(x^{(i)}) - \mathbf{p} \cdot x^{(i)}$ we obtain a dual feasible solution such that $\sum_i u_i + \mathbf{p} \cdot \mathbf{s} = \sum_i v_i(x^{(i)})$ and therefore $(\mathbf{p}, \mathbf{u})$ is an optimal dual solution. □

Proof of Corollary 2 Let $\mathbf{p}^{\text{eq}}$ and $\mathbf{S} = (S_1, S_2, \ldots, S_m)$ be an equilibrium price and allocation. Consider the following chain of inequalities:

$$\sum_i v_i(S_i) \leq \tilde{f}(\mathbf{p}^*) \leq \tilde{f}(\mathbf{p}^{\text{eq}}) \leq f(\mathbf{p}^{\text{eq}}) = \sum_i v_i(S_i)$$

Where the first inequality follows from the definition of $\tilde{f}$, the second from the fact that $\mathbf{p}^*$ is a minimizer of $\tilde{f}$, the third follows from the fact that $\tilde{f} \leq f$ for all prices $\mathbf{p}$, since f is a maximization over all $S_i \subseteq [n]$ and $\tilde{f}$ is a maximization over all subsets whose cardinality is exactly $[n]$. The last inequality follows from the fact that $\mathbf{p}^{\text{eq}}$ is an equilibrium. This implies that all inequalities should hold with equality, in particular, since $\tilde{f}(\mathbf{p}^*) = \sum_i v_i(S_i)$, then it must be that:

$$\max_{S \subseteq [n]} v_i(S) - \mathbf{p}^*(S) - \gamma \cdot |S| = v_i(S_i) - \mathbf{p}^*(S_i) - \gamma \cdot |S_i|$$

In particular, $S_i \in D(i, \mathbf{p}^* + \gamma \cdot \mathbf{1}_{[n]})$.

The other direction is similar. We have $\tilde{f}(\mathbf{p}^{\text{eq}}) = \sum_i v_i(S_i) = f(\mathbf{p}^{\text{eq}})$ and for any price p,

$$\tilde{f}(\mathbf{p}) = f(\mathbf{p} + \gamma \cdot \mathbf{1}_{[n]}) \geq f(\mathbf{p}^{\text{eq}}) = \tilde{f}(\mathbf{p}^{\text{eq}})$$

which shows that any Walrasian price $\mathbf{p}^{\text{eq}}$ minimizes $\tilde{f}$. □

Proof of Lemma 16 (sketch) Assume that the bipartite graph has a different matching with total weight not larger than the one presented. Then it is possible to construct either a cycle of weight less than C or a cycle of the same weight with smaller number of edges.

Let M' be an alternative matching between U and V of weight at most the weight of M. If M' has smaller weight, replace M by M' and $C \cup M' - M$ is a collection

of cycles with total weight smaller than the weight of C. Since all cycles have non-negative weight, one of the cycles must have weight less than C, contradicting the fact that C is a minimum weight cycle.

Now, if M and M' have the same weight, consider the following family of cycles: for each edge $e = (u', v') \in M'$, construct a cycle C_e composed of edge e and the path from v' to u' in C (in other words, we use e to shortcut C). There is an integer $k \leq t$ such that the collection of cycles C_e uses in total: one of each edge from M', $k-1$ of each edge from M and k of each edge from $C - M$. So the average weight is at most the weight of C. Since the C_e cycles have strictly less edges than C, there should be a cycle with fewer edges than C and weight at most C, which again contradicts the choice of C.

The argument for paths is analogous. □

References

1. Ausubel, L., Milgrom, P.: Ascending auctions with package bidding. Front. Theor. Econ. **1**(1) (2002). https://doi.org/10.2202/1534-5963.1019
2. Ausubel, L.M.: An efficient dynamic auction for heterogeneous commodities. Am. Econ. Rev. **96**, 602–629 (2006)
3. Bland, R.G., Goldfarb, D., Todd, M.J.: The ellipsoid method: a survey. Oper. Res. **29**(6), 1039–1091 (1981)
4. Bikhchandani, S., Mamer, J.W.: Competitive equilibrium in an exchange economy with indivisibilities. J. Econ. Theory **74**(2), 385–413 (1997)
5. Blumrosen, L., Nisan, N.: On the computational power of demand queries. SIAM J. Comput. **39**(4), 1372–1391 (2009)
6. Cheung, Y.K., Cole, R., Devanur, N.R.: Tatonnement beyond gross substitutes?: Gradient descent to the rescue. In: Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1–4, 2013, pp. 191–200 (2013)
7. Cohen-Addad, V., Eden, A., Feldman, M., Fiat, A.: The invisible hand of dynamic market pricing. arXiv:1511.05646 (2015)
8. Danilov, V., Koshevoy, G., Murota, K.: Discrete convexity and equilibria in economies with indivisible goods and money. Math. Soc. Sci. **41**(3), 251–273 (2001)
9. Dress, A.W.M., Terhalle, W.: Rewarding maps: on greedy optimization of set functions. Adv. Appl. Math. **16**(4), 464–483 (1995)
10. Dress, A.W.M., Terhalle, W.: Well-layered mapsa class of greedily optimizable set functions. Appl. Math. Lett. **8**(5), 77–80 (1995)
11. de Vries, S., Schummer, J., Vohra, R.V.: On ascending Vickrey auctions for heterogeneous objects. J. Econ. Theory **132**(1), 95–118 (2007)
12. Dress, A.W.M., Wenzel, W.: Valuated matroids: a new look at the greedy algorithm. Appl. Math. Lett. **3**(2), 33–35 (1990)
13. Dress, A.W.M., Wenzel, W.: Valuated matroids. Adv. Math. **93**(2), 214–250 (1992)
14. Fujishige, S., Yang, Z.: A note on Kelso and Crawford's gross substitutes condition. Math. Oper. Res. **28**(3), 463–469 (2003)
15. Gul, F., Stacchetti, E.: Walrasian equilibrium with gross substitutes. J. Econ. Theory **87**(1), 95–124 (1999)
16. Hatfield, J.W., Kominers, S.D.: Hidden substitutes. In: Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15–19, 2015, p. 37 (2015)
17. Hatfield, J.W., Kominers, S.D., Nichifor, A., Ostrovsky, M., Westkamp, A.: Full substitutability in trading networks. In: Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15–19, 2015, pp. 39–40 (2015)
18. Hatfield, J.W., Milgrom, P.R.: Matching with contracts. Am. Econ. Rev. **95**(4), 913–935 (2005)
19. Hsu, J., Morgenstern, J., Rogers, R., Roth, A., Vohra, R.: Do prices coordinate markets? In: Proceedings of the forty-eighth annual ACM symposium on Theory of Computing. ACM, pp. 440–453 (2016)

20. Ikebe, Y.T., Sekiguchi, Y., Shioura, A., Tamura, A.: Stability and competitive equilibria in multi-unit trading networks with discrete concave utility functions. Jpn. J. Ind. Appl. Math. **32**(2), 373–410 (2015)
21. Kelso Jr., A.S., Crawford, V.P.: Job matching, coalition formation, and gross substitutes. Econometrica **50**(6), 1483–1504 (1982)
22. Khachiyan, L.G.: Polynomial algorithms in linear programming. USSR Comput. Math. Math. Phys. **20**(1), 53–72 (1980)
23. Klivans, A.R., Spielman, D.: Randomness efficient identity testing of multivariate polynomials. In: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, pp. 216–223. ACM (2001)
24. Kozlov, M.K., Tarasov, S.P., Khachiyan, L.G.: The polynomial solvability of convex quadratic programming. USSR Comput. Math. Math. Phys. **20**(5), 223–228 (1980)
25. Kojima, F., Tamura, A., Yokoo, M.: Designing matching mechanisms under constraints: an approach from discrete convex analysis. Mpra paper, University Library of Munich, Germany (2014)
26. Leme, R.P.: Gross substitutability: an algorithmic survey. Games Econ. Behav. **106**, 294–316 (2017)
27. Lee, Y.T., Sidford, A., Wong, S.C.: A faster cutting plane method and its implications for combinatorial and convex optimization. In: 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2015) (2015)
28. Lee, Y.T., Sidford, A., Wong, S.C.: A faster cutting plane method and its implications for combinatorial and convex optimization. arXiv:1508.04874 (2015)
29. Murota, K., Shioura, A.: M-convex function on generalized polymatroid. Math. Oper. Res. **24**(1), 95–105 (1999)
30. Murota, K., Shioura, A., Yang, Z.: Computing a walrasian equilibrium in iterative auctions with multiple differentiated items. In: International Symposium on Algorithms and Computation, pp. 468–478. Springer, Berlin (2013)
31. Murota, K., Tamura, A.: Application of M-convex submodular flow problem to mathematical economics. Jpn. J. Ind. Appl. Math. **20**(3), 257–277 (2003)
32. Murota, K.: Convexity and Steinitz's exchange property. Adv. Math. **124**(2), 272–311 (1996)
33. Murota, K.: Valuated matroid intersection I: optimality criteria. SIAM J. Discrete Math. **9**(4), 545–561 (1996)
34. Murota, K.: Valuated matroid intersection II: algorithms. SIAM J. Discrete Math. **9**(4), 562–576 (1996)
35. Murota, K.: Matrices and Matroids for Systems Analysis, vol. 20. Springer, Berlin (2000)
36. Murota, K.: Discrete Convex Analysis. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia (2003)
37. Murota, K.: Discrete convex analysis: a tool for economics and game theory. J. Mech. Inst. Des. **1**(1), 151–273 (2016)
38. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, pp. 345–354. ACM (1987)
39. Nemirovski, A.: Efficient methods in convex programming (2005)
40. Nisan, N., Segal, I.: The communication requirements of efficient allocations and supporting prices. J. Econ. Theory **129**(1), 192–224 (2006)
41. Parkes, D.C.: iBundle: an efficient ascending price bundle auction. In: Proceedings of the 1st ACM Conference on Electronic Commerce, pp. 148–157. ACM (1999)
42. Parkes, D.C., Ungar, L.H.: An ascending-price generalized Vickrey auction (manuscript). Harvard University (2002)
43. Roth, A.E.: Stability and polarization of interests in job matching. Econometrica **52**(1), 47–57 (1984)
44. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, New York (1998)
45. Shioura, A., Tamura, A.: Gross substitutes condition and discrete concavity for multi-unit valuations: a survey. J. Oper. Res. Soc. Jpn. **58**(1), 61–103 (2015)
46. Sun, N., Yang, Z.: Equilibria and indivisibilities: gross substitutes and complements. Econometrica **74**(5), 1385–1402 (2006)
47. Walras, L.: Éléments d'économie politique pure; ou, Théorie de la richesse sociale. Corbaz (1874)

Affiliations

Renato Paes Leme[1] · **Sam Chiu-wai Wong**[2]

Sam Chiu-wai Wong
samcwong@berkeley.edu

1 Google Research NY, New York, NY, USA

2 UC Berkeley, Berkeley, CA, USA