Ascending-Price Mechanism for General Multi-Sided Markets

Dvir Gilor
The Open University of Israel,
Raanana, Israel
dvir@gilor.com

Rica Gonen
The Open University of Israel,
Raanana, Israel
ricagonen@gmail.com

Erel Segal-Halevi
Ariel University,
Ariel, Israel
erelsgl@gmail.com

ABSTRACT

We present an ascending-price mechanism for a multi-sided market with a variety of participants, such as manufacturers, logistics agents, insurance providers, and assemblers. Each deal in the market may consist of a combination of agents from separate categories, and different such combinations are simultaneously allowed. This flexibility lets multiple intersecting markets be resolved as a single global market. Our mechanism is obviously-truthful, strongly budget-balanced, individually rational, and attains almost the optimal gain-from-trade when the market is sufficiently large. We evaluate the performance of the suggested mechanism with experiments on real stock market data and synthetically produced data.

KEYWORDS

Multi-Sided Markets, Truthful Auctions, Strong Budget Balance

ACM Reference Format:
Dvir Gilor, Rica Gonen, and Erel Segal-Halevi. 2021. Ascending-Price Mechanism for General Multi-Sided Markets. In *Appears at the 3rd Games, Agents, and Incentives Workshop (GAIW 2021). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., London, UK, May 2021*, IFAAMAS, 13 pages.

1 INTRODUCTION

The aim of this paper is to automatically arrange the trade in complex multi-lateral markets. As an example, consider a market for a certain kind of laptop computer, and assume for simplicity that it is made of only two components, e.g. CPU and RAM. Even in this simplified market, there may be several different categories of traders: 1. Buyers, who are interested in a laptop; 2. Laptop producers, who produce whole laptops; 3. CPU producers; 4. RAM producers; 5. Constructors, who construct a laptop from its parts; 6. Transporters, who take a laptop and bring it to an end consumer. A deal in this market can take one of two forms:

- A buyer buys a laptop from a laptop-producer, and asks a transporter to transport it to his/her place. This involves traders of categories 1, 2 and 6.
- A buyer buys CPU, RAMs and a construction service, and has the final product transported. This involves traders of categories 1, 3, 4, 5 and 6.

Appears at the 3rd Games, Agents, and Incentives Workshop (GAIW 2021). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., Aziz, Ceppi, Dickerson, Hosseini, Lev, Mattei, McElfresh, Zick (chairs), May 2021, London, UK. © 2021 Copyright held by the owner/author(s). . . . $ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

In each category there may be many different traders, with potentially different utilities for participating in a deal. Typically, the value of a buyer is positive and the value of a producer or service-provider is negative. The main questions of interest for automatically arranging the trade is *who* will trade and *how much* they will pay (or receive). The answers to these questions should satisfy several natural requirements(see Section **2** for formal definitions):

(1) *Individual rationality (IR)*: no agent should lose from participating: the amount paid by a trading agent should be at most as high as the agent's value (if the value is negative then the agent should receive money). A non-trading agent should pay nothing.

(2) *Weak budget balance (WBB)*: the total amount paid by all agents together should be at least 0, so that the market manager does not lose money. A stronger requirement called *strong budget balance (SBB)* is that the total amount be exactly 0, i.e., the market manager does not take away money from the market, which might drive traders away.

(3) High *gain-from-trade (GFT)*: the GFT is the sum of values of all agents actively participating in the trade. For example, suppose a certain buyer values a laptop at 1000, the laptop-producer values it at -700 (the cost of production is 700), the CPU and RAM producers and constructor value their efforts at -200 each, and the transporter values the deal at -50 (the cost of transportation is 50). Then, the GFT from a deal involving categories 1, 2, 6 is $1000 - 700 - 50 = 250$, and the GFT from a deal involving categories 1, 3, 4, 5, 6 is $1000 - 200 - 200 - 200 - 50 = 350$. Maximizing the GFT implies that the latter deal is preferred.

(4) *Truthfulness*: the agents' values are their private information. We assume that the agents act strategically to maximize their utility (assumed to be their value minus the price they pay). Truthfulness means that such a utility-maximizing agent reports his/her true valuation. A stronger requirement called *obvious truthfulness* [27] is that, for each agent, the lowest utility he may get by acting truthfully is at least as high as the highest utility he may get by acting non-truthfully.

1.1 Previous Work

The study of truthful market mechanisms started with Vickrey [36]. He considered a market with only *one category* of traders (buyers), where the famous *second-price auction* attains all four desirable properties: IR, WBB, maximum GFT and truthfulness.

When there are *two categories* of traders (buyers and sellers), the natural generalization of Vickrey's mechanism is no longer WBB — it may run a deficit. Moreover, Myerson and Satterthwaite [31] proved that *any* mechanism that is IR, truthful and maximizes the GFT must run a deficit. The way out of this impossibility paradox was found by McAfee [28]. In his seminal paper, he presented the first *double auction* (auction for a two-category market) that is IR,

WBB, truthful, and *asymptotically* maximizes the GFT. By asymptotically we mean that its GFT is at least $(1 - 1/k)$ of the optimal GFT, where k is the number of deals in the optimal trade. Thus, when k approaches infinity, the GFT approaches the optimum.

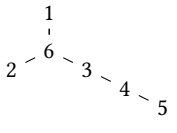
McAfee's mechanism has been extended in various ways. Particularly relevant to our setting is the extension by Babaioff and Nisan [5], with *multiple categories* of traders, arranged in a *linear supply chain*. Their model contains a single *producer* category, a single *consumer* category, and several *converter* categories. Each deal must involve a single producer, a single consumer, and a single agent of each converter category. In our laptop example, their model covers either a market with the chain 1,2,6 or a market with the chain 1,3,4,5,6, but not a market where both chains are possible. For this model, they present a mechanism that is IR, WBB, truthful, and attains asymptotically-optimal GFT

Recently, Gonen and Segal-Halevi [23] considered a multiple-category market in which, like Babaioff and Nisan [5]'s market, all deals must be of the same structure, which they call a "recipe". Their recipes are more general than the linear supply chains of Babaioff and Nisan [5], since they are not restricted to a producer-converters-consumer structure. They present auctions that are IR, SBB, truthful and asymptotically-optimal, but only for a single-recipe market.

Comparison to other supply-chain mechanisms is presented in Appendix **A.1**, and a survey of more recent works on two-sided markets is presented in Appendix **A.2**.

1.2 Our Contribution

We study markets with multiple kinds of supply-chains which, following Gonen and Segal-Halevi [23], we call "recipes". In a general multi-recipe market, computing the optimal trade — even without strategic considerations — is MAX-SNP-hard (see Appendix **C**). In particular, it is NP-hard to compute a trade that attains at least 94/95 of the optimal GFT. Hence, it is unlikely that a mechanism that runs in polynomial time can be asymptotically-optimal. In this paper, we focus on a special case in which the optimal trade can be computed in polynomial-time (Section **3**): the case in which the agent categories can be arranged in a *forest* (acyclic graph), and each recipe is a path from a root to a leaf in that forest. Our laptop market corresponds to a forest with the tree:



We present a randomized ascending mechanism for such markets (Section **4**). Our mechanism is IR, SBB and obviously-truthful. Moreover, all these properties hold *universally* — for every possible outcome of the randomization. The expected GFT of our mechanism is asymptotically-optimal — it approaches the optimum when the optimal number of deals in all recipes approaches infinity (See Section **5** for the formal statements). We evaluate the performance of our mechanism on both real and synthetic data (see Section **6**).

Our mechanism extends [23] in the setting of *binary* recipes, in which each category participates in each recipe either zero or one times. Extending [23] to handle non-binary recipes is beyond the scope of this paper and is the topic of our current research. Some other possible extensions of our mechanism are discussed in Appendix **D**. In particular, we explain why the limitation to acyclic graphs is economically reasonable.

2 FORMAL DEFINITIONS

2.1 Agents and Categories

A *market* is defined by a set of *agents* grouped into different *categories*. N is the set of agents, G is the set of agent categories, and N_g is the set of agents in category $g \in G$. The categories are pairwise-disjoint, that is, each agent belongs to a single category, so $N = \sqcup_{g \in G} N_g$.

Each deal in the market requires a certain combination of traders. We call a subset of agents that can accomplish a single deal a *procurement-set* (PS).

A *recipe* is a vector of size $|G|$, denoted by $\mathbf{r} := (r_g)_{g \in G}$, where $r_g \in \mathbb{Z}_+$ for all $g \in G$. It describes the number of agents of each category that should be in each PS: each PS should contain r_1 agents of category 1, r_2 agents of category 2, and so on. The set of recipes available in the market is denoted by R.

In the market of McAfee [28] each deal requires one buyer and one seller, so there is a single recipe and $R = \{(1, 1)\}$. In our initial laptop-market example there are two recipes and $R = \{(1, 1, 0, 0, 0, 1); (1, 0, 1, 1, 1, 1)\}$. The first one corresponds to deals with a buyer, a producer and a transporter, and the second one corresponds to deals with a buyer, a CPU producer, a RAM producer, a constructor and a transporter. In this paper we assume that recipes are *binary*, i.e., $r_g \in \{0, 1\}$ for every recipe $\mathbf{r}$ and every $g \in G$.

Each agent $i \in N$ has a *value* $v_i \in \mathbb{Z}$, which represents the material gain of an agent from participating in the trade. It may be positive, negative or zero. In a two-sided market for a certain good, the value of a buyer is typically positive, while the value of a seller is typically negative and represents the cost of producing the good. However, our model is general and allows the values of different agents in the same category to have different signs. For simplicity, we assume that all the v_i are integer numbers, e.g., all valuations may be given in cents. We also assume that there are publicly known bounds on the possible valuations: for some sufficiently large V, $-V < v_i < V$ for all $i \in N$.

The agents are *quasi-linear in money*: the utility of agent i participating in some PS and paying p_i is $u_i := v_i - p_i$.

2.2 Trades and Gains

The *gain-from-trade* of a procurement-set S, denoted $GFT(S)$, is the sum of values of all agents in S:

$$\mathrm{GFT}(S) := \sum_{i \in S} v_i.$$

In a standard two-sided market, the GFT of a PS with a buyer b and a seller s is $v_b - v_s$, since the seller's value is $-v_s$.

Given a market $(N, G, \mathbf{r})$, a *trade* is a collection of pairwise-disjoint procurement-sets. I.e, it is a collection of agent subsets, $S_1, \ldots, S_k \subseteq N$, such that for each $j \in [k]$, the composition of agents in S_j corresponds to some recipe $\mathbf{r} \in R$. The total GFT is the sum of the GFT of all procurement-sets participating in the trade:

$$\mathrm{GFT}(S_1, \ldots, S_k) := \sum_{j=1}^{k} \mathrm{GFT}(S_j)$$

A trade is called *optimal* if its GFT is maximum over all trades.

The *value* of agent i given trade $\mathbf{S} = (S_1, \ldots, S_k)$, denoted $v_i(\mathbf{S})$, is either v_i or 0: it is v_i if $i \in S_j$ for some $j \in [k]$, and 0 otherwise.

2.3 Mechanisms

The definitions below cover only the notions used in the present paper. For a more complete treatment of mechanisms and their properties see [32].

A *deterministic direct mechanism* is a function that takes as input a vector $\mathbf{b}$ containing agent bids, and returns as output a trade $\mathbf{S}(\mathbf{b})$ and a price-vector $\mathbf{p}(\mathbf{b})$. The *utility* of each agent i, given a deterministic mechanism and a bid vector $\mathbf{b}$, is $u_i(\mathbf{b}) := v_i(\mathbf{S}(\mathbf{b})) - p_i(\mathbf{b})$.

A deterministic direct mechanism is *truthful* if the utility of every agent i is maximized when the agent bids v_i, for any fixed bids of the other agents. Formally, for every vector $\mathbf{b} = (b_1, \ldots, b_n)$, denote by $\mathbf{b}|_{b_i \leftarrow x}$ the vector $(b_1, \ldots, b_{i-1}, x, b_{i+1}, \ldots, b_n)$. A mechanism is truthful if for every agent i and $\mathbf{b}$:

$$u_i(\mathbf{b}|_{b_i \leftarrow v_i}) \geq u_i(\mathbf{b}).$$

A deterministic direct mechanism is *individually-rational (IR)* if the utility of every agent i when the agent bids v_i is at least 0, regardless of the bids of the other agents:

$$u_i(\mathbf{b}|_{b_i \leftarrow v_i}) \geq 0.$$

A *randomized direct mechanism* is a lottery over deterministic direct mechanisms. In other words, it is a mechanism in which the functions $\mathbf{S}$ and $\mathbf{p}$ may depend not only on the bids but also on some random variables.

A randomized direct mechanism is called *universally-truthful* if it is a lottery over truthful deterministic direct mechanisms. In a universally-truthful randomized mechanism, the utility of agent i is maximized when the agent bids v_i, regardless of the bids of the other agents, and regardless of the random variable values. Similarly, a randomized direct mechanism is *universally-IR* if it is a lottery over IR deterministic direct mechanisms.

A mechanism is called *obviously truthful* if for every agent i and vectors $\mathbf{b}, \mathbf{b}'$:

$$u_i(\mathbf{b}|_{b_i \leftarrow v_i}) \geq u_i(\mathbf{b}').$$

In other words, the lowest utility the agent can get when reporting truthfully is at least as high as the highest utility the agent can get when reporting untruthfully, where "lowest" and "highest" are w.r.t. all possible reports of the other agents. This is a very strong property that is not satisfied by non-trivial direct mechanisms. However, an analogous property is satisfied by some *sequential mechanisms*.

In a *deterministic sequential mechanism*, at each time, an agent has to choose an *action* from a prespecified set of actions. In order to give meaning to the notion of truthfulness, we assume that the "action" is an answer to a query on the agent's value: at time t, the designer presents a function q_t to some agent i, and the agent is expected to reveal $q_t(v_i)$. Our mechanisms will only use Boolean functions such as "is $v_i > 2$?". Based on the agents' answers so far, the designer may decide to continue asking queries, or to end. When the mechanism ends, the designer examines the vector of answers $\mathbf{a}$, and determines the trade $\mathbf{S}(\mathbf{a})$ and the price-vector $\mathbf{p}(\mathbf{a})$.

Given an answer vector $\mathbf{a}$ and an agent i, denote by $\mathbf{a}|_{a_i \leftarrow x}$ the vector in which the answer of agent i to any function q_t is $q_t(x)$ (and the answers of other agents remain as in $\mathbf{a}$). A deterministic sequential mechanism is called *obviously truthful* if, at any step during the execution, and for any two vectors $\mathbf{a}$ and $\mathbf{a}'$ consistent with the history of answers up to the current step:

$$u_i(\mathbf{a}|_{a_i \leftarrow v_i}) \geq u_i(\mathbf{a}').$$

In other words, the lowest utility the agent can get by answering truthfully, according to v_i, is at least as high as the highest utility he can get by answering untruthfully.

A deterministic direct mechanism is a special case of a deterministic sequential mechanism in which there is only one step of queries and the queries are "what is your value?". If such a mechanism is obviously-truthful, then it is also truthful (set $\mathbf{a} = \mathbf{a}' = \mathbf{b}$ in the definition of obvious-truthfulness).

A *randomized sequential mechanism* is a lottery over deterministic sequential mechanisms; it is called *universally obviously-truthful* if it is a lottery over obviously-truthful deterministic sequential mechanisms.

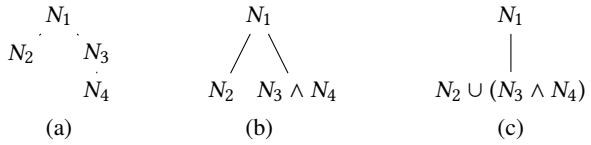


Figure 1: Examples of trees in a recipe-forest.

Table 1: an example market.

Category	Agents' values
N_1: buyers	**17, 14, 13, 9,** 6, 2
N_2: sellers	**-4, -5,** -8, -10
N_3: A-producers	**-1, -3,** -5
N_4: B-producers	**-1, -4,** -6

2.4 Recipe forests

Recall that a *forest* is an acyclic graph, composed of one or more *trees*; a *rooted forest* is a forest in which, in each tree, one vertex is denoted as its *root*.

Definition 2.1. A recipe-set R is called a *recipe-forest* if there exists a rooted forest T in which the set of nodes is G, and each recipe $\mathbf{r} \in R$ corresponds to a path P from the root of some tree in T to a leaf of that tree (i.e., $r_g = 1$ for each $g \in P$ and $r_g = 0$ for each $g \notin P$).

We use the same letter g to denote both the category index and the corresponding node in T. As an example, the set $R = \{(1,1,0,0), (1,0,1,1)\}$ is a recipe-forest with a single tree shown in Figure 1(a). The root category is N_1. The recipe $(1,1,0,0)$ corresponds to a path from N_1 to the leaf N_2. The recipe $(1,0,1,1)$ corresponds to a path from N_1 through N_3 to N_4.

3 COMPUTING OPTIMAL TRADE

We first present an algorithm for computing the optimal trade assuming all values are known. We illustrate the algorithm on the market in

Table 2: Optimal trade in that market.

Procurement sets
Buyer 17, A-producer −1, B-producer −1
Buyer 14, seller −4
Buyer 13, seller −5
Buyer 9, A-producer −3, B-producer −4

the Table 1. The algorithm is based on contracting the recipe-forest down to a single node. Two types of contraction operations are used.

In a **vertical contraction**, a leaf that is a single child is combined with its parent in the following way. Suppose the sets of agent values in the child category are $v_1 \geq v_2 \geq \ldots \geq v_{m_v}$ and the agent values in the parent category are $u_1 \geq u_2 \geq \ldots \geq u_{m_u}$. Replace the parent category by a new category with $m := \min(m_v, m_u)$ values: $u_1 + v_1, u_2 + v_2, \ldots, u_m + v_m$. For example, a vertical contraction on the tree of Figure 1(a) results in the tree of Figure 1(b), where $N_3 \wedge N_4$ denotes the elementwise combination of N_3 and N_4. In the Table 1 market, $N_3 \wedge N_4$ contains the value pairs $\{(-1,-1),(-3,-4),(-5,-6)\}$ whose values are $\{-2,-7,-11\}$.

The rationale is that the unique root-leaf path that passes through the parent passes through its child too, and vice-versa. Therefore, any PS that contains an agent of the parent category must contain an agent of the child category, and vice-versa. In economic terms, these two categories are *complements*. Hence, elementwise combination of the two categories leads to a market with identical optimal GFT.

In a **horizontal contraction**, two sibling leaves are combined by taking the union of their categories in the following way. Suppose the sets of agent values in the left sibling category are $v_1, \ldots, v_{m_v}$ and in the right sibling category are $u_1, \ldots, u_{m_u}$. Replace both categories by a new category with $m := m_v + m_u$ values: $v_1, \ldots, v_{m_v}, u_1, \ldots, u_{m_u}$. For example, a horizontal contraction on the tree of Figure 1(b) results in the tree of Figure 1(c), where $N_2 \cup (N_3 \wedge N_4)$ denotes the combination of N_2 and $N_3 \wedge N_4$. In the Table 1 market, $N_2 \cup (N_3 \wedge N_4)$ contains the values $\{-4,-5,-8,-10\} \cup \{-2,-7,-11\}$ whose values are $\{-2,-4,-5,-7,-8,-10,-11\}$.

The rationale is that, for every path from the root to one leaf there exists a path from the root to the other leaf, and vice-versa. Therefore, in any PS that contains an agent of one leaf-category, this agent can be replaced with an agent from the other leaf-category. In economic terms, these categories are *substitutes*. Therefore, uniting them leads to a market with the same optimal GFT.

In any tree with two or more vertices, there is a leaf that is either a single child or has a sibling leaf (for example, any leaf farthest from the root). Therefore, any tree admits either a vertical or a horizontal contraction, and it is possible to contract any tree to a single node. For example, a vertical contraction on the tree of Figure 1(c), in the Table 1 market, yields: $\{17-2, 14-4, 13-5, 9-7, 6-8, 2-10\}$. The optimal trade in this market is the set of all deals with positive values, which in this case contains four deals with values $\{15, 10, 8, 2\}$. This corresponds to an optimal trade with $k = 4$ deals, shown at the Table 2.

If the forest has two or more trees, then all contracted trees can be further combined using a horizontal contraction to a single node. The process is shown as **Algorithm 1**.

Algorithm 1 Find the optimal GFT.

Input: A set of categories G, a set of traders N_g for all $g \in G$, and a recipe-forest R based on a forest T.
For each agent $i \in \cup_g N_g$, the value v_i is public knowledge.
Output: Optimal trade in the market.
1. If T has a single vertex g:
   Return all agents in N_g with a positive value: $\{i \in N_g | v_i > 0\}$
2. Else, if T has two roots without children g_l and g_s:
   Do a horizontal contraction of g_l into g_s. Go back to step 1.
3. Else, if there is a leaf g_l that is a single child of its parent g_p:
   Do a vertical contraction of g_l into g_p. Go back to step 1.
4. Else, there is a leaf g_l with a sibling leaf g_s:
   Do a horizontal contraction of g_l into g_s. Go back to step 1.

Algorithm 2 Ascending prices mechanism.

Input: A market N, a set of categories G and a recipe-forest R.
Output: Strongly-budget-balanced trade.
1. *Initialization:* Let $M_g := N_g$ for each $g \in G$.
   Determine initial price-vector $\mathbf{p}$:
   For each non-leaf g, set $p_g := -V$;
   For each leaf g, set: $p_g := -V(\text{MaxDepth} - \text{Depth}(g) + 1)$;
2. Using Algorithm 3, select a set $G^* \subseteq G$ of categories.
3. For each $g^* \in G^*$, ask each agent in $i \in M_{g^*}$ whether $v_i > p_{g^*}$.
   (a) If an agent $i \in M_{g^*}$ answers "no", then
   Remove i from M_{g^*} and go back to step 2.
   (b) If all agents in M_{g^*} for all $g^* \in G^*$ answer "yes", then
   for all $g^* \in G^*$, let $p_{g^*} := p_{g^*} + 1$.
   (c) If after the increase $\sum_{g \in G} p_g \cdot r_g = 0$ for some $\mathbf{r} \in R$, then go on to step 4.
   (d) else go back to step 3.
4. Determine final trade using Algorithm 4.

4 ASCENDING AUCTION MECHANISM

4.1 General Description

The ascending-price auction is a randomized sequential mechanism. The general scheme is presented as **Algorithm 2**. For each category g, the auctioneer maintains a price p_g, and a subset $M_g \subseteq N_g$ of all agents that are "in the market" (that is, their value is higher than the current price of their category). At each iteration, the auctioneer chooses a subset of the prices, and increases each price p_g in this subset by 1. After each increase, the auctioneer asks each agent in turn, in a pre-specified order (e.g. by their index), whether their value is still higher than the price. An agent who answers "no" is permanently removed from the market. After each increase, the auctioneer computes the sum of prices of the categories in each recipe, defined as: Prices-sum$(\mathbf{r}) := \sum_{g \in G} p_g$. When this sum equals 0, the auction ends and the remaining agents trade in the final prices.

To flesh out this scheme, we need to explain (a) how the prices are initialized, (b) how the set of prices to increase is selected, and (c) how the final trade is determined.

(a) An important challenge in determining the prices is that the sum of prices must be the same for all recipes $\mathbf{r} \in R$, so that the price-sum crosses 0 for all recipes simultaneously, and all deals are

Algorithm 3 Find a set of prices to increase.

Input: A set of categories G, a set of remaining traders M_g for all $g \in G$, and a recipe-forest R based on a forest T.
Output: A subset of G denoting categories whose price should be increased.

0. *Initialization*: For each category $g \in G$, let $m_g := |M_g|$ = the number of agents of N_g who are in the market.
1. If T contains two or more trees,
   Recursively run Algorithm 3 on each individual tree T';
   Denote the outcome by $I_{T'}$.
   Return $\bigcup_{T' \in T} I_{T'}$.
2. Let g_0 be the category at the root of the single tree.
   Let $c_{g_0} := \sum_{g' \in \text{CHILDREN}(g_0)} m_{g'}$.
3. If $m_{g_0} > c_{g_0}$ [or g_0 has no children at all],
   then return the singleton $\{g_0\}$.
4. Else ($c_{g_0} \geq m_{g_0}$), for each child g' of g_0:
   Recursively run Algorithm 3 on the sub-tree rooted at g';
   Denote the outcome by $I_{g'}$.
   Return $\bigcup_{g' \in child(g_0)} I_{g'}$.

Algorithm 4 Determine a feasible trade.

Input: A set of categories G,
a set of remaining traders M_g for all $g \in G$,
and a recipe-forest R based on a forest T.
Output: A set of PSs with remaining traders,
each of which corresponds to a recipe in R.

1. If T has a single vertex g:
   Return M_g — the set of traders remaining in category g.
2. If T has two roots without children g_l and g_s:
   Do a horizontal contraction of g_l into g_s. Go back to step 1.
3. Otherwise, pick an arbitrary leaf category $g_l \in T$.
4. If g_l is a single child of its parent $g_p \in T$:
   Perform a randomized vertical contraction of g_l and g_p.
   Go back to step 1.
5. Otherwise, g_l has a sibling $g_s \in T$:
   Perform a horizontal contraction of g_l and g_s.
   Go back to step 1.

simultaneously SBB. For the initial prices, this challenge is handled by the initialization of **Algorithm 2**: the price of each non-leaf category is set to $-V$, and the price of each leaf category is set to a number which is at most $-V$, computed such that the price-sum in each path from a root to a leaf is the same.

(b) Selecting which prices to increase is handled by **Algorithm 3**. It is a recursive algorithm: if the forest contains only a single category (a root with no children), then of course this category is selected. Otherwise, in each tree, either its root category or its children are selected for increase. The selection is based on the number of agents of each category g who are currently in the market. We denote this number by $m_g := |M_G|$.

We denote the root category of a tree by g_0. The algorithm first compares m_{g_0} to the sum of the m_g for all children of g_0 (which is denoted by c_{g_0}). If m_{g_0} is larger, then the price selected for increase is the price of g_0; Otherwise (c_{g_0} is larger or equal), the prices to increase are the prices of children categories: for each child category, Algorithm 3 is used recursively to choose a subset of prices to increase, and all returned sets are combined. It is easy to prove by induction that the resulting subset contains exactly one price for each path from a root to a leaf. Therefore, all prices in the subset are increased simultaneously by one unit, and the price-sum in all recipes remains equal.

Consider again the tree of Figure 1(a), and suppose the numbers of remaining traders in the four categories are $6, 4, 3, 3$. Initially the algorithm compares m_1 to $m_2 + m_3$; since the latter is larger, the algorithm recursively checks the subtrees rooted at $g = 2$ and $g = 3$. In the former there is only one category so it is returned; in the latter, there is one child $g = 4$. Since $m_3 \leq m_4$, the child $g = 4$ is selected. The final set of prices to increase is $\{p_2, p_4\}$. If the counts were $m_1 = 6, m_2 = 3, m_3 = 2, m_4 = 2$ instead, then the set of prices to increase would be $\{p_1\}$. Note that in both cases, a single price is increased in each recipe.

The equality of price-sums is preserved by the price-increase. The price-sum increases by 1 at each step, so at some point it reaches 0. At that point, the auction stops.

(c) Once the auction ends, the final trade has to be computed. At this stage, it is possible that in some recipes, the numbers of traders remaining in the market are not balanced. In order to construct an integer number of procurement-sets of each recipe, some agents must be removed from the trade. The traders to remove must be selected at random and not by their value, since selecting traders by value might make the mechanism non-truthful. To this end, we replace the vertical contraction operation with a *randomized vertical contraction*. A leaf that is a single child is combined with its parent in the following way. Denote the leaf and parent category by l and p respectively, and let M_i be the set of traders remaining in category i. Let $n_{\min} := \min(|M_l|, |M_p|)$ = the integer number of procurement-sets that can be constructed from the agents in both categories. For each $g \in \{l, p\}$ if $|M_g| > n_{\min}$ then choose $|M_g| - n_{\min}$ agents uniformly at random and remove them from M_g. Then perform a vertical contraction with the remaining agents.

The horizontal contractions can be performed deterministically, as no traders should be removed. The process of determining the final trade is summarized as **Algorithm 4**.

4.2 Example Run

We illustrate Algorithm 2 using the example in Table 1, where the recipe set is $R = \{(1, 1, 0, 0), (1, 0, 1, 1)\}$ and the recipe-forest contains the single tree shown in Figure 1(a). The execution is shown in Table 3.

Step 1. The initialization step ensures that (a) the initial sum of prices is the same in each recipe; (b) the price in each category is lower than the lowest possible value of an agent in this category, which we denoted by $-V$. In the example, the initial prices are $-V, -2V, -V, -V$, and the price-sum of all recipes is $-3V$.

Step 2. The categories whose price should be increased are determined using Algorithm 3. In the example, the numbers of remaining traders are $6, 4, 3, 3$. Since $6 < 4 + 3$, the price of the root category (the buyers) is not increased. In the first branch, the seller-price is selected for increase. In the second branch, there is a tie between

Table 3: Execution of Algorithm 2 on market from Table 1

Category counts	G^*	Price-increase stops when	New prices	Price-sum
		[Initialization]	$-V, -2V, -V, -V$	$-3V$
6, 4, 3, 3	2, 4	B-producer -6 exits	$-V, -V-6, -V, -6$	$-2V-6$
6, 4, 3, 2	2, 3	A-producer -5 exits	$-V, -11, -5, -6$	$-V-11$
6, 4, 2, 2	2, 4	seller -10 exits	$-V, -10, -5, -5$	$-V-10$
6, 3, 2, 2	1	buyer 2 exits	$2, -10, -5, -5$	-8
5, 3, 2, 2	2, 4	B-producer -4 exits	$2, -9, -5, -4$	-7
5, 3, 2, 1	2, 3	seller -8 exits	$2, -8, -4, -4$	-6
5, 2, 2, 1	1	buyer 6 exits	$6, -8, -4, -4$	-2
4, 2, 2, 1	2, 3	A-producer -3 exits	$6, -7, -3, -4$	-1
4, 2, 1, 1	1	price-sum crosses zero	$7, -7, -3, -4$	0

the A-producer and the B-producer, which is broken in favor of the child. Therefore, the chosen set G^* is $\{2, 4\} = \{\text{seller, B-producer}\}$.

Step 3. The auctioneer increases the prices of each category $g^* \in G^*$ by $1/r_{g^*}$, until one agent of some category $g^* \in G^*$ indicates that his value is not higher than the price, and leaves the trade. The price never skips any agent's integer value, because the initial category price was a big negative integer number $(-V)$ and the increment is done always by $1/r_g$ so the category price visits every integer from $-V$ to the current category price. In the example, the first agent who answers "no" is B-producer -6. While p_4 has increased to -6, p_2 has increased to $-V-6$, so the price-sum in all recipes remains the same: $-2V-6$. After B-producer -6 is removed, we return to step 2 to choose a new set of prices to increase. The algorithm keeps executing steps 2 and 3 as described in Table 3. Finally, while the algorithm increases p_1, before buyer 9 exits the trade, the price-sum in all recipes becomes 0 and the algorithm proceeds to step 4.

Step 4. The final trade is determined by Algorithm 4. In the example, a randomized vertical contraction is first done between the A-producers and B-producers. Since there is one A-producer -1 and one B-producer -1, none of them has to be removed, and the combined category now has a single pair. Next, a horizontal contraction is done between the pair of producers and the remaining two sellers. This results in a combined category of size 3. Finally, a randomized vertical contraction is done between this combined category and the buyers' category. Since there are 4 remaining buyers, but only 3 sets in the child category, one of the buyers is chosen at random and removed from trade. Finally, three deals are made: two deals follow the recipe $(1, 1, 0, 0)$ and involve a buyer and a seller, and one deal follows the recipe $(1, 0, 1, 1)$ and involves a buyer, an A-producer and a B-producer.

5 ASCENDING AUCTION PROPERTIES

Due to space constraints, most proofs are in Appendix B.

A crucial feature of our mechanism is that the price-sum along each path from the same node to a leaf is constant.

LEMMA 5.1. *Throughout Algorithm 2, for any category $g \in G$, the price-sum along any path from g to a leaf is the same for all paths.*

The economic properties of the auction are summarized in the following theorems.

THEOREM 5.2. *Algorithm 2 is universally strongly budget balanced, individually-rational and obviously truthful.*

PROOF. Given a fixed priority-ordering on the agents, consider the deterministic variant of the algorithm in which, in step 3 of Algorithm 4, instead of the randomized vertical contraction, the removed agents in each category are selected deterministically by the fixed agent ordering. Algorithm 2 is a lottery on such deterministic mechanisms, where the agent ordering is selected uniformly at random. Therefore, to prove that the randomized mechanism satisfies a property universally, it is sufficient to prove that each such deterministic variant satisfies this property.

Strong budget balance holds since by Lemma 5.1 (applied to the root category), the price-sum for all recipes remains the same throughout the execution, and the algorithm stops whenever this sum becomes 0. Individual rationality holds since $i \in N_g$ may remain in the market only if $v_i \geq p_g$. To prove obvious-truthfulness, we consider an agent $i \in N_g$ who is asked whether $v_i > p_g$, and check the two possible cases:

- Case 1: $v_i > p_g$. If the agent answers truthfully "yes", then his lowest possible utility is 0 (since the mechanism is IR). If the agent answers untruthfully "no", then his highest possible utility is 0 since he is immediately removed from trade and cannot return.
- Case 2: $v_i \leq p_g$. If the agent answers truthfully "no", then his lowest possible utility is 0 (since he is removed from trade immediately). If the agent answers untruthfully "yes", then his highest possible utility is 0, since the utility is $v_i - p_g$ and the price can only increase.

In both cases, the lowest possible utility of a truthful agent is at least the highest possible utility of a non-truthful agent. □

We now show that the ascending auction attains an asymptotically optimal GFT. The analysis assumes that the valuations are *generic* — the sum of valuations in every subset of agents is unique. In particular, the optimal trade is unique. This is a relatively mild assumption, since every instance can be modified to have generic valuations, as explained by Babaioff and Walsh [7].

First, choose a sufficiently large constant $W \geq n+1$ and replace each value v_i by $2^W \cdot v_i$. This scaling obviously has no effect on the optimal or the actual trade. Then, arbitrarily assign a unique integer index $i \in \{1, \ldots, n\}$ to every agent, and set $v_i' := 2^W \cdot v_i + 2^i$.

Now the sum of valuations in every agent subset is unique, since the n least significant bits in its binary representation are unique. Moreover, for every subset $I \subseteq N$, $\sum_{i \in I} v_i' \approx 2^W \sum_{i \in I} v_i$ plus some "noise" smaller than $2^{n+1} \leq 2^W$.

Therefore, the optimal trade in the new instance corresponds to one of the optimal trades in the original instance, with the GFT multiplied by 2^W. If the constant W is sufficiently large, the "noise" has a negligible effect on the GFT.

Definition 5.3. (a) The number of deals in the optimal trade is denoted by k.

(b) For each recipe $\mathbf{r} \in R$, the number of deals in the optimal trade corresponding to $\mathbf{r}$ is denoted by $k_{\mathbf{r}}$ (so $k = \sum_{\mathbf{r} \in R} k_{\mathbf{r}}$).

(c) The smallest positive number of deals of a single recipe in the optimal trade is denoted by $k_{\min} := \min_{\mathbf{r} \in R, k_{\mathbf{r}} > 0} k_{\mathbf{r}}$.

THEOREM 5.4. *The expected GFT of the ascending-price auction of Section 4 is at least $1 - 1/k_{\min}$ of the optimal GFT.*

To prove Theorem 5.4 we need several definitions. For every category $g \in G$:

(*) k_g := the number of deals in the optimal trade containing an agent from N_g (equivalently: the number of deals whose recipe-path passes through g). If g is the root category then $k_g = k$. If g is any non-leaf category then

$$k_g = \sum_{g' \text{ is a child of } g} k_{g'}. \quad (1)$$

In the Table 3 market, k_g for categories 1,2,3,4 equals $4, 2, 2, 2$ respectively.

(*) v_{g,k_g} := the value of the k_g-th highest trader in N_g — the lowest value of a trader that participates in the optimal trade. In the Table 3 market, v_{g,k_g} for categories 1,2,3,4 equals $9, -5, -3, -4$ respectively. Note that, in any path from the root to a leaf, the sum of v_{g,k_g} is positive — otherwise we could remove the PS composed of the agents corresponding to this path, and get a trade with a higher GFT.

(*) v_{g,k_g+1} := the highest value of a trader that does not participate in the optimal trade (or $-V$ if no such trader exists). In the Table 3 market, v_{g,k_g+1} for categories 1,2,3,4 equals $6, -8, -5, -6$ respectively. Note that, in any path from the root to a leaf, the sum of v_{g,k_g+1} is at most 0 — otherwise we could add the corresponding PS and get a trade with a higher GFT.

Recall that, during the auction, $m_g := |M_g|$ = the number of agents of category g currently in the market (whose value is larger than p_g), and

$$c_g := \sum_{g' \text{ is a child of } g} m_{g'}. \quad (2)$$

When the algorithm starts, $m_g \geq k_g$ for all $g \in G$, since all participants of the optimal trade are in the market. Similarly, $c_g \geq k_g$. In contrast to equation (1), m_g and c_g need not be equal. By adding dummy agents with value $-V + 1$ to some categories, we can guarantee that, when the algorithm starts, $m_g = c_g$ for all non-leaf categories $g \in G$. For example, in the Table 3 market it is sufficient to add a buyer with value $-V + 1$. This addition does not affect the optimal trade, since no PS in the optimal trade would contain agents with such low values. It does not affect the actual trade either, since the price-sum is negative as long as there are dummy agents in the market. Once $m_g = c_g$, we show that these values remain close to each other throughout the algorithm:

LEMMA 5.5. *For all non-leaf categories $g \in G$,*

$$c_g \leq m_g \leq c_g + 1.$$

Definition 5.6. Given a price-vector $\mathbf{p}$, a subset $G' \subseteq G$ is called:
(a) *Cheap* — if $p_g \leq v_{g,k_g+1}$ for all $g \in G'$;
(b) *Expensive* — if $p_g \geq v_{g,k_g}$ for all $g \in G'$.

We apply Definition 5.6 to paths in trees in the recipe-forest T. Intuitively, in a cheap path, the prices are sufficiently low to allow the participation of agents not from the optimal trade, while in an expensive path, the prices are sufficiently high to allow the participation of agents only from the optimal trade.

Lemmas B.1 - B.5 show some cases when Cheap and Expensive paths can and cannot exist in certain forest-trees. These lemmas are then used to prove Lemma B.6: *when Alg. 2 ends, $m_g \in \{k_g, k_g - 1\}$ for all $g \in G$.* With Lemma B.6, we prove our main theorem. Lemmas B.1 - B.6 and their proofs appear in Appendix B.3.

PROOF OF THEOREM 5.4. By Lemma B.6, each recipe $\mathbf{r} \in R$ with $k_\mathbf{r} = 0$ does not participate in the trade at all. For each recipe $\mathbf{r} \in R$ with $k_\mathbf{r} > 0$, for each category g in $\mathbf{r}$, all k_g optimal traders of g, except maybe the lowest-valued one, participate in the final trade. Therefore, in the random selection of the final traders (Algorithm 4), at least $k_g - 1$ random deals are performed out of the k_g optimal deals. Hence, the approximation ratio of the GFT coming from recipe $\mathbf{r}$ alone is at least $1 - 1/k_\mathbf{r}$ of the optimum. Taking the minimum over all recipes yields the ratio claimed in the theorem. □

6 EXPERIMENTS

We evaluated the performance of our ascending auction using simulation experiments. For these preliminary experiments, we used the recipe-forest $\mathbf{R} = \{(1, 1, 0, 0), (1, 0, 1, 1)\}$, which contains a single tree with only binary recipes and two paths ($N_1 \longrightarrow N_2$ and $N_1 \longrightarrow N_3 \longrightarrow N_4$). In the future we plan to do experiments with larger forests and non-binary recipes. For several values of $n \leq 1000$, we constructed a market with n agents of each category g. The agents' values were chosen at random as explained below. For each n, we made 1000 runs and averaged the results. We split the values among the categories uniformly at random, so each category has n values.

6.1 Agents' Values

We conducted two experiments. In the first experiment, the value of each buyer (root category) was selected uniformly at random from $[1, 1000]$, and the value of each trader from the other three categories was selected uniformly at random from $[-1, -1000]$.

In the second experiment, the values were selected based on real stocks prices on Yahoo's stock market site using 33 stocks. For each stock, we collected the prices from every day from the inception of the stock until September 2020. Every day the stock has 4 values: Open, Close, High and Low. All price values are multiplied by 1000, so they can be represented as integers, to avoid floating-point rounding errors. On each stock, we collected all the price values and used those price values as agents' values at random. For the non-root categories, the values were multiplied by -1. There were more than 1000 values for each category.

6.2 Number of Deals and Gain From Trade

In each run, we calculated k (the number of deals in the optimal trade), $k_{\min}$, $k_{\max}$ (recipe minimum and maximum number of deals in the optimal trade), $1 - \frac{1}{k_{\min}}$ (the theoretical lower bound ratio) and *OGFT* (the optimal gain-from-trade). We found that the average value of k was approximately $0.6n$. For the ascending-price mechanism, we calculated k' (the actual number of deals done by the mechanism), $k_{\min}{}'$, $k_{\max}{}'$ (the actual recipe minimum and maximum number of deals done by the mechanism) and the *GFT* (the actual gain-from-trade of deals done by the mechanism).

6.3 Results and Conclusions

The results from the stock-prices experiment are presented in Table 4 and in Figure 2. The results from the uniform-random experiment

Table 4: Results with stock-market prices and the recipe-forest $\mathrm{R} = \{(1,1,0,0),(1,0,1,1)\}$.

	Optimal					Ascending Price					
n	k	$k_{\min}$	$k_{\max}$	$1-\frac{1}{k_{\min}}$	$OGFT$	k'	${k_{\min}}'$	${k_{\max}}'$	$\%k'$	GFT	$\%GFT$
2	1.11	1.00	1.00	6.592	106323.0	0.48	0.48	0.48	37.24	60563.5	43.473
4	2.27	1.56	1.83	24.051	231902.7	1.49	1.35	1.38	65.02	194877.0	80.655
6	3.43	1.89	2.65	30.710	348117.4	2.56	1.93	2.22	74.39	322155.4	90.531
10	5.78	2.46	4.31	42.567	594618.4	4.84	2.63	3.89	83.53	578495.9	96.269
16	9.32	3.36	6.79	56.172	953490.8	8.34	3.42	6.37	89.46	943732.1	98.473
26	15.19	5.05	10.91	69.904	1563658.4	14.20	4.88	10.50	93.45	1557792.5	99.401
50	29.34	9.49	20.77	83.685	3010702.2	28.33	9.13	20.37	96.56	3007474.9	99.831
100	58.77	18.86	41.35	91.395	6037200.9	57.74	18.50	40.95	98.25	6035460.2	99.953
500	294.16	90.59	206.05	96.664	30271280.7	293.03	91.92	205.66	99.61	30270801.8	99.997
1000	588.46	176.99	411.96	97.440	60588937.5	587.23	178.03	411.59	99.79	60588643.6	99.999

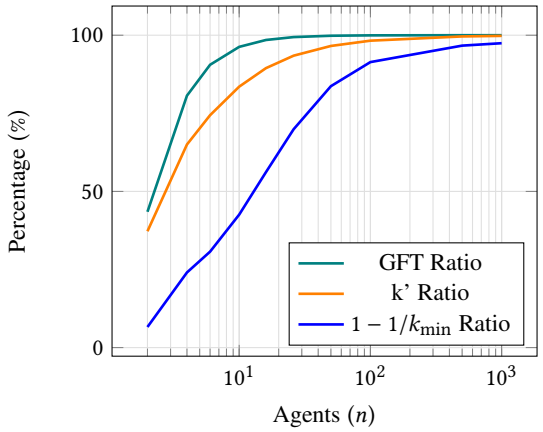


Figure 2: Graph of results from Table 4.

are presented in Appendix in Table 5 and in Figure 3. The highlights of both experiments are similar:

- The actual number of maximum trades (${k_{\max}}'$) is very near $k_{\max}-0.5$, and the actual number of trades (k') is very near $k-1$. That is, the mechanism loses, on average, half a deal per recipe, and one deal overall. In contrast, theoretically, the mechanism might lose up to one deal per recipe (see the proof of Theorem 5.4)[1].
- For the optimal trade, we have $k \approx 3n/5$ where approximately $n/5$ optimal deals use one recipe and $2n/5$ optimal deals use the other recipe, i.e., $k_{\min} \approx n/5$ and $k_{\max} \approx 2n/5$. Note that for small values of n, sometimes there are optimal deals only in one path of the tree. In such cases $k_{\min} = k_{\max} = k$. Therefore, for small n the average of $k_{\min}$ plus the average of $k_{\max}$ may be larger than the average of k.
- The actual GFT of the ascending auction is much higher than the theoretical lower bound $1-1/k_{\min}$ of the optimum. For example, when $n = 10$ (and $k_{\min} \le 3$), the theoretical lower bound is less than 50%, but the ascending-price auction attains more than 95%. It surpasses 99.9% already for $n \ge 100$.

[1]The ${k_{\min}}'$ might be higher than $k_{\min}$. This happens because in some iterations there might be zero actual deals in one recipe. In that case, the ${k_{\min}}'$ gets the value of ${k_{\max}}' = k'$ which is the only recipe that has a number of deals greater than 0.

REFERENCES

[1] Lawrence M Ausubel. 2004. An efficient ascending-bid auction for multiple objects. *American Economic Review* 94, 5 (2004), 1452–1475.

[2] Lawrence M Ausubel and Paul R Milgrom. 2002. Ascending auctions with package bidding. *Advances in Theoretical Economics* 1, 1 (2002).

[3] Moshe Babaioff, Yang Cai, Yannai A Gonczarowski, and Mingfei Zhao. 2018. The Best of Both Worlds: Asymptotically Efficient Mechanisms with a Guarantee on the Expected Gains-From-Trade. In *Proceedings of the 2018 ACM Conference on Economics and Computation*. ACM, 373–373. arXiv preprint 1802.08023.

[4] Moshe Babaioff, Kira Goldner, and Yannai A Gonczarowski. 2020. Bulow-Klemperer-Style Results for Welfare Maximization in Two-Sided Markets. In *Proceedings of SODA'20*. 2452–2471. arXiv preprint arXiv:1903.06696.

[5] Moshe Babaioff and Noam Nisan. 2004. Concurrent Auctions Across the Supply Chain. *Journal of Artificial Intelligence Research (JAIR)* 21 (2004), 595–629. https://doi.org/10.1613/jair.1316

[6] Moshe Babaioff and WilliamE Walsh. 2006. Incentive Compatible Supply Chain Auctions. In *Multiagent based Supply Chain Management*. Vol. 28. Springer Berlin Heidelberg, 315–350.

[7] Moshe Babaioff and William E. Walsh. 2005. Incentive-compatible, budget-balanced, yet highly efficient auctions for supply chain formation. *Decision Support Systems* 39, 1 (March 2005), 123–149. https://doi.org/10.1016/j.dss.2004.08.008

[8] Liad Blumrosen and Shahar Dobzinski. 2014. Reallocation mechanisms. In *EC*. 617–640.

[9] Liad Blumrosen and Shahar Dobzinski. 2018. (Almost) Efficient Mechanisms for Bilateral Trading. In *Working paper*.

[10] L. Blumrosen and Y. Mizrahi. 2016. Approximating gains-from-trade in bilateral trading. In *WINE*. 400–413.

[11] Johannes Brustle, Yang Cai, Fa Wu, and Mingfei Zhao. 2017. Approximating Gains from Trade in Two-sided Markets via Simple Mechanisms. arXiv:1706.04637 http://arxiv.org/abs/1706.04637

[12] Brahim Chaib-Draa and Jörg Müller. 2006. *Multiagent based supply chain management*. Vol. 28. Springer Science & Business Media.

[13] Rachel R Chen, Robin O Roundy, Rachel Q Zhang, and Ganesh Janakiraman. 2005. Efficient auction mechanisms for supply chain procurement. *Management Science* 51, 3 (2005), 467–482.

[14] Miroslav Chlebík and Janka Chlebíková. 2006. Complexity of approximating bounded variants of optimization problems. *Theoretical Computer Science* 354, 3 (2006), 320–338.

[15] Riccardo Colini-Baldeschi, Bart de Keijzer, Stefano Leonardi, and Stefano Turchetta. 2016. Approximately efficient double auctions with strong budget balance. In *SODA*. 1424–1443.

[16] Riccardo Colini-Baldeschi, Paul W Goldberg, Bart de Keijzer, Stefano Leonardi, Tim Roughgarden, and Stefano Turchetta. 2017. Approximately efficient two-sided combinatorial auctions. In *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM, 591–608.

[17] Marek Cygan. 2013. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 509–518.

[18] Sven de Vries, James Schummer, and Rakesh V Vohra. 2007. On ascending Vickrey auctions for heterogeneous objects. *Journal of Economic Theory* 132, 1 (2007), 95–118.

[19] Gabrielle Demange, David Gale, and Marilda Sotomayor. 1986. Multi-item auctions. *Journal of political economy* 94, 4 (1986), 863–872.

[20] P. Dütting, T. Roughgarden, and I. Talgam-Cohen. 2017. Modularity and greed in double auctions. *Games and Economic Behavior* 105(C) (2017), 59–83.

[21] Matthias Gerstgrasser, Paul W Goldberg, Bart de Keijzer, Philip Lazos, and Alexander Skopalik. 2019. Multi-unit bilateral trade. In *Proceedings of the AAAI'19*, Vol. 33. 1973–1980. arXiv preprint 1811.05130.

[22] Mira Gonen, Rica Gonen, and Pavlov Elan. 2007. Generalized trade reduction mechanisms. In *Proceedings of EC'07*. 20–29.

[23] Rica Gonen and Erel Segal-Halevi. 2020. Strongly Budget Balanced Auctions for Multi-Sided Markets.. In *AAAI*. 1998–2005.

[24] Atsushi Iwasaki, Etsushi Fujita, Taiki Todo, Miao Yao, and Makoto Yokoo. 2013. VCG-equivalent in Expectation Mechanism: General Framework for Constructing Iterative Combinatorial Auction Mechanisms. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)* (Department of Informatics, Kyushu University Fukuoka, 819-0395, Japan) *(AAMAS 2013)*. International Conference on Autonomous Agents and Multiagent Systems, Japan, 699–706.

[25] Viggo Kann. 1991. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Inform. Process. Lett.* 37, 1 (1991), 27–35.

[26] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 85–103.

[27] Shengwu Li. 2017. Obviously strategy-proof mechanisms. *American Economic Review* 107, 11 (2017), 3257–87.

[28] R. Preston McAfee. 1992. A dominant strategy double auction. *Journal of Economic Theory* 56, 2 (April 1992), 434–450.

[29] R. P. McAfee. 2008. The Gains from Trade Under Fixed Price Mechanisms. *Applied Economics Research Bulletin* 1 (2008).

[30] Debasis Mishra and David C Parkes. 2007. Ascending price Vickrey auctions for general valuations. *Journal of Economic Theory* 132, 1 (2007), 335–366.

[31] Roger B. Myerson and Mark A. Satterthwaite. 1983. Efficient mechanisms for bilateral trading. *Journal of Economic Theory* 29, 2 (April 1983), 265–281.

[32] Noam Nisan. 2007. Introduction to Mechanism Design (For Computer Scientists). In *Algorithmic Game Theory*, Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani (Eds.). Cambridge University Press, 209–241.

[33] Erel Segal-Halevi, Avinatan Hassidim, and Yonatan Aumann. 2016. SBBA: A Strongly-Budget-Balanced Double-Auction Mechanism. In *Algorithmic Game Theory*, Martin Gairing and Rahul Savani (Eds.). Lecture Notes in Computer Science, Vol. 9928. Springer Berlin Heidelberg, 260–272. https://doi.org/10.1007/978-3-662-53354-3_21

[34] Erel Segal-Halevi, Avinatan Hassidim, and Yonatan Aumann. 2018a. MUDA: A Truthful Multi-Unit Double-Auction Mechanism. In *Proceedings of AAAI'18*. AAAI Press. arXiv preprint 1712.06848.

[35] Erel Segal-Halevi, Avinatan Hassidim, and Yonatan Aumann. 2018b. Double Auctions in Markets for Multiple Kinds of Goods. In *Proceedings of IJCAI'18*. AAAI Press. Previous name: "MIDA: A Multi Item-type Double-Auction Mechanism". arXiv preprint: 1604.06210.

[36] William Vickrey. 1961. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance* 16, No. 1 (Mar., 1961) (1961), 8–37.

APPENDIX

A MORE RELATED WORK

A.1 Supply Chain Management

Chaib-Draa and Müller [12] provide a comprehensive survey of multiagent methods related to supply-chain management. The most general supply-chain auction we are aware of is the trade-reduction mechanism of Babaioff and Walsh [6, 7]. They allow procurement sets of multiple recipes. Their model differs from ours in several respects:

(a) They distinguish between "producer markets" and "consumer markets", with "goods" moving between markets, and impose constraints on the demand and supply of agents in each market. In contrast, our model is abstract and considers only the general notion of a "category", with no specific distinction between producers and consumers, and does not require the notion of a "good".

(b) Their "Unique Manufacturing Technology" requirement forbids some markets that are covered by our model, such as a market with one consumer-category and two producer-categories with the two recipes $(1, 1, 0)$ and $(1, 0, 1)$; see [6] Section 6.

(c) On the other hand, they allow recipes with multiple units, such as $(2, 1, 1, 0, 0, 0)$ and $(2, 1, 0, 2, 1, 1)$; see [6] Figure 1.

(d) Their auction is WBB and truthful, while ours is SBB and obviously-truthful.

A.2 Two-Sided Markets

Two-sided markets have been extensively studied since the seminal work of [28]. Recently, [33] presented a SBB variant of McAfee's mechanism, with similar GFT guarantees. Their mechanism may remove up to one *buyer* from the optimal trade, and it is the buyer with the lowest value among the buyers in the optimal trade. [11] presented two simple mechanism that are IR, truthful, and WBB, and obtain, in expectation, at least half of the expected GFT of the second-best efficiency benchmark.

[3] present approximation results for the double-auction setting and for double-auction with some added constraints on the pairs of agents who can trade with each other. [20] have studied similar double-sided setting to that of [3] however their constraints are applied on each side of the market separately. [4] show a strong analogue to the result of Bulow-Klemperer for welfare in two-sided markets rather than revenue in auctions.

[13] consider a supply-chain auction with a sole buyer and single item-kind, but there are different producers in different supply-locations. The buyer needs a different quantity of the item in different demand-locations. The buyer conducts a reverse auction and has to pay, in addition to the cost of production, also the cost of transportation from the supply-locations to the demand-locations. They do not guarantee SBB. [22] generalized the above settings to a unified trade reduction procedure.

[29] designs fixed price SBB double auction under the assumptions that the buyer's distribution dominates the seller's distribution or that there is exponential distribution. Our result does not assume knowledge of the distribution of participating categories. Additionally, we also allow for any number of categories in each recipe, as opposed to two as well as for multiple recipes to simultaneously trade.

[9, 15, 16] also presents SBB auctions. [9, 15] auctions target double-sided and [16] target combinatorial markets. However, their goal is to maximize social welfare as opposed to our goal which is maximizing gain from trade[2]. Thus their mechanisms are not asymptotically-optimal for gain from trade. They also require a prior on the agents' valuations. Similarly to [16], [8] present two-sided combinatorial market solution. [8]'s solution is WBB unlike our SBB solution and maximize social welfare as opposed to our goal which is maximizing gain from trade.

[10] present a mechanism that obtains in expectation at least $1/e$ of the first-best GFT. However, they assume the buyer's valuation is drawn from a distribution satisfying the monotone hazard rate condition. In contrast our result does not assume knowledge of the distribution of participating categories, let alone a specific distribution.

The present work handles multiple categories of agents, but each agent is single-parametric. An orthogonal line of work ([21], [34], [35]) remains with two agent categories (buyers and sellers), but aims to handle multi-parametric agents.

A.3 Iterative One-Sided Auctions

Ascending-price auctions have been developed for various subsets of combinatorial valuations, such as unit demand [19], homogeneous goods [1], submodular buyers [2], and general valuations [18, 30]. However, all these auctions are for a single-sided market. This means that each deal requires only a single strategic agent (the seller is not strategic and not considered a part of the mechanism). The challenge in our multi-sided market is that each deal requires multiple strategic agents. Moreover, in combinatorial ascending auctions, the efficient outcome is attained only in equilibrium, see e.g. Mishra and Parkes [30] and Iwasaki et al. [24]. In contrast, our auction attaina an almost-efficient outcome in dominant strategies.

B PROOFS OF ASCENDING AUCTION PROPERTIES

For the following proofs, the following notation is used:

- $\textsc{Children}(g)$:= the child nodes of the node g in its tree.
- $\textsc{Path}(g_1 \to g_2)$:= the nodes in the path from g_1 to g_2, inclusive.
- $\textsc{Depth}(g)$:= the distance between the node g and the root of its tree. The depth of a root is defined as 0.
- $\textsc{Height}(g)$:= the largest distance between the node g and a leaf of its tree. The height of a leaf is defined as 0.
- $\textsc{Depth}(g)$:= the distance between g and the root of its tree.
- $\textsc{MaxDepth} := \max_{g \text{ is a leaf}} \textsc{Depth}(g)$.

[2]When optimizing GFT we optimize the difference between the total value of the sold items for the buyers and the total value of these items for the sellers. When optimizing social welfare in a market we optimize the sum of the buying agents' valuations plus the sum of the unsold items' value held by selling agents at the end of trade. Despite their conceptual similarity, the two objectives are rather different in approximation. In many cases the social welfare approximation is close to the optimal social welfare solution; however, the gain from trade approximation may not be within any constant factor of the optimal gain from trade.

B.1 Lemma 5.1

LEMMA 1. *Throughout Algorithm 2, for any category $g \in G$, the price-sum along any path from g to a leaf is an integer, and it is the same for all paths.*

PROOF. After the initialization step, the price-sum in all paths from g to a leaf is equal: $-V \cdot (\text{MAXDEPTH} - \text{DEPTH}(g) + 1)$. The selection of prices to increase (Algorithm 3) guarantees that, for any $g \in G$, one of the following holds: either (a) no descendant of g is selected, or (b) exactly one node is selected in any path from g to a leaf. Algorithm 2 increases all selected prices simultaneously by the same amount; therefore the price-sum remains equal. □

B.2 Lemma 5.5

LEMMA 2. *For all non-leaf categories $g \in G$,*

$$c_g \le m_g \le c_g + 1.$$

PROOF. The proof is by induction on the algorithm steps. Before the first step $m_g = c_g$ so the claim holds.

In each step, if $m_g = c_g$ then Algorithm 3 never selects p_g for increase. Hence, Algorithm 2 never removes agents from M_g, so $c_g \le m_g$ still holds. It may remove an agent from a child of g, but since at most one agent is removed in each step, $m_g \le c_g + 1$ still holds after the removal.

If $m_g = c_g + 1$, then the algorithm never increases prices and never removes agents from children of g, so $m_g \le c_g + 1$ still holds; it may remove at most one agent from M_g, so $c_g \le m_g$ holds. □

B.3 Lemmas B.1 - B.6

LEMMA B.1. *Let g_1, g_2 be two children of the same node $g_p \in T$. There cannot be simultaneously a cheap path from g_1 to a leaf and an expensive path from g_2 to a leaf.*

PROOF. Let q_1 be the price-sum along the cheap path from g_1 to a leaf, and q_2 the price-sum along the expensive path from g_2 to a leaf. By definition of cheap and expensive paths, q_1 is the GFT of a part of non-optimal PS, and q_2 is the GFT of a part of an optimal PS; therefore $q_1 < q_2$. In particular, $q_1 \neq q_2$. But since both paths are children of the same node g, this contradicts Lemma 5.1. □

LEMMA B.2. *If $m_g \le k_g - 1$ for some $g \in G$, then there is an expensive path from g to a leaf.*

PROOF. The fact that $m_g \le k_g - 1$ means that $p_g \ge v_{g,k_g}$, so the condition for an expensive path holds for g itself. To show that it holds for a path from g to a leaf, we apply induction on $\text{HEIGHT}(g)$. If $\text{HEIGHT}(g) = 0$ (i.e., g itself is a leaf), then the claim is obvious. Otherwise, by Lemma 5.5,

$$\left(\sum_{g' \text{ is a child of } g} m_{g'}\right) = c_g \le m_g \le k_g - 1 = \left(\sum_{g' \text{ is a child of } g} k_{g'}\right) - 1$$

Therefore, there is at least one child g' of g for which $m_{g'} \le k_{g'} - 1$. Since $\text{HEIGHT}(g') < \text{HEIGHT}(g)$, by the induction assumption there is an expensive path from g' to a leaf. Prepending g to this path yields an expensive path from g to a leaf. □

LEMMA B.3. *If $m_g \ge k_g + 1$ for some $g \in G$, then there is a cheap path from g to a leaf.*

PROOF. The fact that $m_g \ge k_g + 1$ means that $p_g \le v_{g,k_g+1}$, so the condition for a cheap path holds for g itself. To show that it holds for a path from g to a leaf, we apply induction on $\text{HEIGHT}(g)$. If $\text{HEIGHT}(g) = 0$ then the claim is obvious. Otherwise, there are two cases.

Case #1: g has a child g' for which $m_{g'} \ge k_{g'} + 1$. Then by the induction assumption there is a cheap path from g' to a leaf; prepending g to this path yields a cheap path from g to a leaf.

Case #2: $m_{g'} \le k_{g'}$ for all children g' of g. Then $c_g \le k_g \le m_g - 1$. By Lemma 5.5 we must in fact have $c_g = m_g - 1 = k_g$, so $m_{g'} = k_{g'}$ for all children g' of g. Now, let us look back at the history of price-increases made by the algorithm, and identify the most recent price-increase in a descendant of g (a category in the subtree below g). Before this price-increase, $c_g = m_g$ had necessarily held (since otherwise Algorithm 3 would not have chosen a descendant of g for increase). So this price-increase must have been in a child g' of g, which before this increase had $m_{g'} = k_{g'} + 1$. Since $\text{HEIGHT}(g') < \text{HEIGHT}(g)$, by the induction assumption there was an cheap path from g' to a leaf. The price-increase of g' stopped at the moment when agent $k_{g'} + 1$ was removed from $M_{g'}$, i.e., it stopped at $p_{g'} = v_{g',k_{g'}+1}$; therefore, the same path from g' to a leaf is still cheap. Prepending g to this path yields a cheap path from g to a leaf. □

LEMMA B.4. *If $\lceil m_g \rceil \ge k_g + 1$ for some $g \in G$, then there is a cheap path from the root to a leaf (through g).*

PROOF. By Lemma B.3 there is a cheap path from g to a leaf. Therefore, it is sufficient to prove that there is a cheap path from the root to g. The proof is by induction on $\text{DEPTH}(g)$. If $\text{DEPTH}(g) = 0$ (i.e., g itself is the root), then the claim is obvious. Otherwise, let g_p be the parent of g. Otherwise, let g_p be the parent of g.

Case #1: $m_{g_p} \ge k_{g_p} + 1$. Since $\text{DEPTH}(g_p) < \text{DEPTH}(g)$, by the induction assumption there is a cheap path from the root to g_p; appending g to this path yields a cheap path from the root to g.

Case #2: $m_{g_p} \le k_{g_p}$. Lemma 5.5 implies:

$$\sum_{g' \text{ is a child of } g_p} m_{g'} = c_{g_p} \le m_{g_p} \le k_{g_p} = \sum_{g' \text{ is a child of } g_p} k_{g'}$$

Removing from both sides the term corresponding to g (which is a child of g_p) yields:

$$\left(\sum_{g' \neq g \text{ is a child of } g_p} m_{g'}\right) \le \left(\sum_{g' \neq g \text{ is a child of } g_p} k_{g'}\right) - 1$$

Therefore, there is at least one child g' of g_p for which $m_{g'} \le k_{g'} - 1$. By Lemma B.2, there is an expensive path from g' to a leaf. But by Lemma B.3, there is a cheap path from g — which is a sibling of g' — to a leaf. By Lemma B.1, these two paths cannot exist simultaneously; hence, Case #2 is not possible. □

LEMMA B.5. *If $m_g \le k_g - 1$ for some $g \in G$, and Algorithm 3 decides to increase the price of g or a descendant of g, then (even before the increase) there is an expensive path from the root to a leaf (through g).*

PROOF. By Lemma B.2, $m_g \le k_g - 1$ implies that there is an expensive path from g to a leaf. Therefore, it is sufficient to prove that there is an expensive path from the root to g. The proof is by

induction on $\text{DEPTH}(g)$. If $\text{DEPTH}(g) = 0$ (i.e., g itself is the root), then the claim is obvious. Otherwise, let g_p be the parent of g. There are two cases.

Case #1: $m_{g_p} \leq k_{g_p} - 1$. Since $\text{DEPTH}(g_p) < \text{DEPTH}(g)$, by the induction assumption there is an expensive path from root to g_p; appending g to this path yields an expensive path from root to g.

Case #2: $m_{g_p} \geq k_{g_p}$. The fact that Algorithm 3 decides to increase the price of g' or a descendant of it implies that that $c_{g_p} \geq m_{g_p}$. Therefore:

$$\left(\sum_{g' \text{ is a child of } g_p} m_{g'}\right) = c_{g_p} \geq m_{g_p} \geq k_{g_p} = \left(\sum_{g' \text{ is a child of } g_p} k_{g'}\right)$$

Removing from both sides the term corresponding to g (which is a child of g_p) yields:

$$\left(\sum_{g' \neq g \text{ is a child of } g_p} m_{g'}\right) \geq \left(\sum_{g' \neq g \text{ is a child of } g_p} k_{g'}\right) + 1$$

Therefore, there is at least one child g' of g_p for which $m_{g'} \geq k_{g'} + 1$. By Lemma B.3, there is a cheap path from g' to a leaf. But by Lemma B.2, there is an expensive path from g — which is a sibling of g' — to a leaf. By Lemma B.1, these two paths cannot exist simultaneously; hence, Case #2 is not possible. □

LEMMA B.6. *When Alg. 2 ends, $m_g \in \{k_g, k_g - 1\}$ for all $g \in G$.*

PROOF. The proof is by contradiction.

First, suppose that $m_g \geq k_g + 1$ for some $g \in G$. By Lemma B.4, there is a cheap path from the root to a leaf; denote the set of categories along this path by G'. The sum of prices of categories $g \in G'$ is the GFT of a non-optimal PS, which is negative. As long as the price-sum is negative, the algorithm does not terminate.

Second, suppose that $m_g \leq k_g - 2$ for some $g \in G$. Since at most a single agent is removed in each iteration, this means that the algorithm decided to increase the price of g while m_g was equal to $k_g - 1$. By Lemma B.5, at that point there existed an expensive path from the root to a leaf; denote the set of categories along this path by G'. The sum of prices of categories $g \in G'$ is the GFT of an optimal PS, which is positive. However, the price-sum increases by a single unit each round, and the algorithm terminates when the price-sum hits zero, so the price-sum can never be positive. □

C HARDNESS OF GENERAL RECIPE SETS

To illustrate the difficulty of handling general recipe-sets, we prove that calculating the optimal trade, even without strategic considerations, is MAX-SNP-hard. This implies that, for some constant, it is hard to compute a constant-factor approximation of the optimal GFT.

THEOREM C.1. *The following problem is MAX-SNP-hard. Given a set N of agents with known integer valuations, a set G of categories, a set R of recipes, and an integer C, decide whether there exists a feasible trade in which the GFT is at least C.*

PROOF. The proof is by reduction from *3-dimensional matching*, which is the following decision problem: given a 3-uniform hypergraph $H = (V, E)$ (a hypergraph in which each edge in E contains 3 vertices of V) and a positive integer C, decide whether H has a matching that contains at least C edges. This problem is known to be NP-hard [26] and MAX-SNP-hard [25].

Given an instance $H = (V, E)$ of 3-D matching, construct an instance of the GFT problem as follows.

- There is a category for each vertex: $G = V$.
- Each category contains a single agent.
- The value of every agent is $1/3$.
- There is a recipe $\mathbf{r}^e$ for each edge $e \in E$, defined as follows:

$$\mathbf{r}^e_i := \begin{cases} 1 & i \in e, \\ 0 & \text{otherwise.} \end{cases}$$

Since H is 3-uniform, each recipe contains exactly 3 ones and the other elements are zero. Therefore, the GFT of every PS is $3 \cdot 1/3 = 1$, and the GFT of every trade equals the number of trading PS. Since each category contains a single agent, each category must appear in at most one PS. Therefore, every feasible trade corresponds to a matching in H and vice-versa, so the problems are equivalent. □

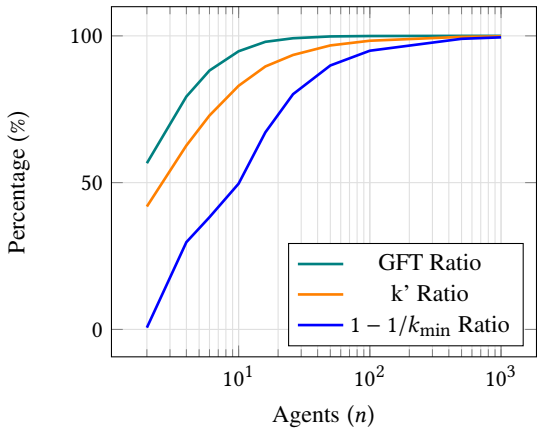


Figure 3: Graph of results from Table 5.

Note that the best known polytime algorithm for 3-D matching attains $3/4$ of the optimum [17], and it is NP-hard to attain $94/95$ of the optimum [14]. this illustrates the kind of approximation to the GFT that we can hope to obtain for general recipe-sets. Developing obviously-truthful mechanisms that attain such constant-factor approximations is an interesting future work direction.

D DISCUSSION AND FUTURE WORK

The dependence of Theorem 5.4 on $k_{\min}$ may appear weak, but it is the best possible. Consider a recipe-forest with five categories and two recipes: (dummy, buyer1, seller1) and (dummy, buyer2, seller2). The dummy category contains infinitely-many agents with value 0; the (buyer1, seller1) categories contain $k_{\max}$ pairs with a GFT of 1; the (buyer2, seller2) categories contain $k_{\min}$ pairs with a GFT of ${k_{\max}}^2$. Here, $OPT = k_{\max} + k_{\min} * {k_{\max}}^2$. It is clearly equivalent to two independent two-sided markets: the (buyer1 ,seller1) market with $OPT = k_{\max}$ and the (buyer2, seller2) market with $OPT \gg k_{\max}$. The approximation ratio of any mechanism is dominated by the ratio on the (buyer2, seller2) market, which by McAfee [28] is at most $1 - 1/k_{\min}$ (while our approximation ratio is $1 - 2/(k_{\min} + 1)$).

Table 5: Results with random prices and the recipe-forest $\mathbf{R} = \{(1,1,0,0),(1,0,1,1)\}$.

	Optimal					Ascending Price					
n	k	$k_{\min}$	$k_{\max}$	$1-\frac{1}{k_{\min}}$	$OGFT$	k'	${k_{\min}}'$	${k_{\max}}'$	$\%k'$	GFT	$\%GFT$
2	1.13	1.00	1.00	0.596	449.9	0.47	0.47	0.47	41.86	254.7	56.614
4	2.38	1.42	1.79	29.725	1051.0	1.49	1.30	1.33	62.65	833.8	79.329
6	3.58	1.61	2.63	38.233	1665.4	2.61	1.77	2.17	72.87	1468.7	88.191
10	5.92	1.98	4.17	49.647	2776.1	4.91	1.94	3.74	82.99	2630.5	94.756
16	9.52	3.04	6.51	67.148	4581.6	8.53	2.63	6.10	89.58	4488.8	97.974
26	15.48	5.03	10.45	80.139	7640.8	14.48	4.41	10.06	93.49	7578.8	99.188
50	29.97	9.91	20.06	89.911	14847.0	29.00	9.33	19.66	96.74	14818.1	99.805
100	59.65	19.75	39.89	94.939	29613.3	58.66	19.16	39.49	98.34	29597.9	99.948
500	300.02	99.59	200.42	98.995	149554.5	299.02	98.99	200.02	99.66	149551.3	99.997
1000	600.15	199.13	401.01	99.497	299470.6	599.13	198.54	400.59	99.82	299468.9	99.999

Our ascending auction requires acyclic graphs. Economically, the case where cycles are allowed is not sensible. Consider a recipe-tree with one parent node and two child nodes. The tree represents two different recipes, each with a parent node category that complements a child node category. The two recipes have mutually substitutable child nodes. Introducing a cycle, i.e., connecting the two child nodes in the graph to form a recipe, would create categories that simultaneously complement and substitute each other. Economically, such a recipe model is unavailing.

From a theoretical perspective, it may be interesting to study other natural settings in which the optimal GFT can be computed efficiently. For example, if every recipe in R has exactly two 1-s and the other elements are 0, then the optimal GFT can be found using a maximum-weight matching algorithm. Is it possible to construct an efficient ascending auction for such cases?