Walrasian Equilibrium: Hardness, Approximations and Tractable Instances

Ning Chen* Atri Rudra*

Abstract. We study the complexity issues for Walrasian equilibrium in a special case of combinatorial auction, called single-minded auction, in which every participant is interested in only one subset of commodities. Chen et al. [5] showed that it is NP-hard to decide the existence of a Walrasian equilibrium for a single-minded auction and proposed a notion of approximate Walrasian equilibrium called relaxed Walrasian equilibrium. We show that every single-minded auction has a relaxed Walrasian equilibrium that satisfies at least two-thirds of the participants, proving a conjecture posed in [5]. Motivated by practical considerations, we introduce another concept of approximate Walrasian equilibrium called weak Walrasian equilibrium. We show NP-completeness and hardness of approximation results for weak Walrasian equilibria.

In search of positive results, we restrict our attention to the tollbooth problem [19], where every participant is interested in a single path in some underlying graph. We give a polynomial time algorithm to determine the existence of a Walrasian equilibrium and compute one (if it exists), when the graph is a tree. However, the problem is still NP-hard for general graphs.

Keywords. Walrasian equilibrium, Combinatorial auction, Single-minded auction, NP-hard, Approximation.

1 Introduction

Imagine a scenario where a movie rental store is going out of business and wants to clear out its current inventory. Further suppose that based on their rental records, the store has an estimate of which combinations (or bundles) of items would interest a member and how much they would be willing to pay for those bundles. The store then sets prices for each individual item and allocates bundles to the members (or buyers) with the following "fairness" criterion— given the prices on the items no buyer would prefer to be allocated any bundle other than those currently allocated to her. Further, it is the last day of the sale and it is imperative that no item remain after the sale is completed. A natural way to satisfy this constraint is to give away any unallocated item for free. This paper looks at the complexity of computing such allocation and prices.

The concept of "fair" pricing and allocation in the above example is similar to the concept of market equilibrium which has been studied in economics literature for more than a hundred years starting with the work of Walras [32]. In this model, buyers arrive with an initial endowment of some item and are looking to buy and sell items. A market equilibrium is a vector of prices of the

*Department of Computer Science & Engineering, University of Washington, Seattle, WA, 98105, USA. Email: {ning,atri}@cs.washington.edu.

items such that the market clears, that is, no item remains in the market. Arrow and Debreu [3] showed more than half a century ago that when the buyers' utility functions are concave, a market equilibrium always exists though the proof was not constructive. A polynomial time algorithm to compute a market equilibrium for linear utility functions was given recently by Jain [22]. We note that in this setup the items are divisible while we will consider the case of indivisible items.

Equilibrium concepts [26] are an integral part of classical economic theory. However, only in the last decade or so these have been studied from the computational complexity perspective. This perspective is important as it sheds some light on the feasibility of an equilibrium concept. As Kamal Jain remarks in [22]— "If a Turing machine cannot efficiently compute it, then neither can a market". A small sample of the recent work which investigates the efficient computation of equilibrium concepts appears in the following papers [12, 13, 22, 5, 28, 29, 10, 11, 6, 7].

In this paper, we study the complexity of computing a Walrasian equilibrium [32], one of the most fundamental economic concepts in market conditions. Walrasian equilibrium formalizes the concept of "fair" pricing and allocation mentioned in the clearance sale example. In particular, given an input representing the valuations of bundles for each buyer, a Walrasian equilibrium specifies an allocation and price vector such that any unallocated item is priced at zero and all buyers are satisfied with their corresponding allocations under the given price vector. In other words, if any bundle is allocated to a buyer, the profit/utility[1] made by the buyer from her allocated bundle is no less than the profit she would have made if she were allocated any other bundle under the same price vector.

In the clearance sale example, we assumed that every buyer is interested in some bundles of items— this is the setup for combinatorial auctions which have attracted a lot of attention in recent years due to their wide applicability [31, 9]. In the preceding discussions, we have disregarded one inherent difficulty in this auction model— the number of possible bundles is exponential and thus, even specifying the input makes the problem intractable. In this paper, we focus on a more tractable instance called *single-minded auction* [24] with the linear pricing scheme[2]. In this special case, every buyer is interested in a single bundle. Note that the inputs for these auctions can be specified quite easily. The reader is referred to [2, 1, 27, 5, 17] for more detailed discussions on single-minded auctions.

If bundles can be priced arbitrarily, a Walrasian equilibrium always exists for combinatorial auctions [8, 25]. However, if the pricing scheme is restricted to be linear, a Walrasian equilibrium may not exist. Several sufficient conditions for the existence of a Walrasian equilibrium with the linear pricing scheme have been studied. For example, if the valuations of buyers satisfy the *gross substitutes* condition [23], the *single improvement* condition [18], or the *no complementarities* condition [18], a Walrasian equilibrium is guaranteed to exist. Bikhchandani et al. [4] and Chen et al. [5] gave an exact characterization for the existence of a Walrasian equilibrium— the total value of the optimal allocation of bundles to buyers, obtained from a suitably defined integer program, is equal to the value of the corresponding relaxed linear program. These conditions are, however, not easy to verify in polynomial time even for single-minded auctions. In fact, checking the existence of a Walrasian equilibrium for single-minded auctions is NP-hard [5].

The intractability result raises a couple of natural questions: (1) Instead of trying to satisfy

[1]The difference between how much the buyer values a bundle and the price she pays for it.

[2]The price of a bundle is the sum of prices of the items in the bundle.

all the buyers, what fraction of the buyers can be satisfied ? (2) Can we identify some structure in the demands of the buyers for which a Walrasian equilibrium exists and can be computed in polynomial time? In this paper we study these questions and have the following results:

- *Conjecture on relaxed Walrasian equilibrium.* Chen et al. [5] proposed an approximation of Walrasian equilibrium, called *relaxed Walrasian equilibrium.* This approximate Walrasian equilibrium is a natural approximation where instead of trying to satisfy all buyers, we try to satisfy as many buyers as we can. [5] gave a simple single-minded auction for which no more than 2/3 of the buyers can be satisfied. They also conjectured that this ratio is tight— that is, for any single-minded auction one can come up with an allocation and price vector such that at least 2/3 of the buyers are satisfied. In our first result we prove this conjecture. In fact we show that such an allocation and price vector can be computed in polynomial time.

- *Weak Walrasian equilibrium.* There is a problem with the notion of relaxed Walrasian equilibrium— it does not require all the winners to be satisfied. This is infeasible in practice. For instance in the clearance sale example, the store cannot sell a bundle to a buyer at a price greater than her valuation. In other words, at a bare minimum all the winners have to be satisfied. With this motivation, we define a stronger notion of approximation called *weak Walrasian equilibrium* where we try to maximize the number of satisfied buyers subject to the constraint that all winners are satisfied. In our second result we show that computing a weak Walrasian equilibrium is strongly NP-complete. We achieve this via a reduction from the Independent Set problem. Independently, Huang et al. [21] showed that computing a weak Walrasian equilibrium is NP-hard (but not strongly NP-hard) via a reduction from the hardness of Walrasian equilibrium.

- *Tollbooth problem.* With the plethora of negative results, we shift our attention to special cases of single-minded auctions where computing Walrasian equilibria might be tractable. With this goal in mind, we study the tollbooth problem introduced by Guruswami et al. [19] where the bundle every buyer is interested in is a path in some underlying graph. We show that in a general graph, it is still NP-hard to compute a Walrasian equilibrium given that one exists. We then concentrate on the special case of a tree, and show that we can determine the existence of a Walrasian equilibrium and compute one (if it exists) in polynomial time. Essentially, we prove this result by first showing that the optimal allocation can be computed by a polynomial time dynamic program. In fact, the optimal allocation is equivalent to computing the set of maximum weighted edge-disjoint paths in a tree. A polynomial time divide and conquer algorithm for the latter was given by Tarjan [30]. See Section 4.3 for more details on how this part of our work differs from the existing literature.

The paper is organized as follows. In Section 2, we review the basic properties of Walrasian equilibrium for single-minded auctions. In Section 3, we study two different type of approximations— relaxed Walrasian equilibrium and weak Walrasian equilibrium. In Section 4, we study the tollbooth problem— for the different structure of the underlying graphs, we give different hardness or tractability results. Section 5 describes our dynamic program based algorithm to compute the optimal allocation in the tollbooth problem on trees. We conclude with Section 6.

2 Preliminaries

In a *single-minded auction* [24], an auctioneer sells m heterogeneous commodities $\Omega = \{\omega_1, \ldots, \omega_m\}$, with unit quantity each, to n potential buyers. Each buyer i desires a fixed subset[3] of commodities of Ω, called *demand* and denoted by $d_i \subseteq \Omega$, with *valuation* (or *cost*) $c_i \in \mathbb{R}^+$. That is, d_i is the only bundle of commodities that i is interested in and c_i is the maximum amount that i is willing to pay for it.

After receiving the submitted tuple (d_i, c_i) from each buyer i (the *input*), the auctioneer specifies the tuple (X, p) as the *output* of the auction:

- *Allocation vector* $X = \langle x_1, \ldots, x_n \rangle$, where $x_i \in \{0, 1\}$ indicates if i wins d_i ($x_i = 1$) or not ($x_i = 0$). Note that we require $\sum_{i:\ \omega_j \in d_i} x_i \leq 1$ for any $\omega_j \in \Omega$. $X^* = \langle x_1^*, \ldots, x_n^* \rangle$ is said to be an *optimal allocation* if for any allocation X, we have

$$\sum_{i=1}^{n} c_i \cdot x_i^* \ \geq \ \sum_{i=1}^{n} c_i \cdot x_i.$$

  That is, X^* maximizes total valuations of the winning buyers.

- *Price vector* $p = \langle p(\omega_1), \ldots, p(\omega_m) \rangle$ (or simply, $\langle p_1, \ldots, p_m \rangle$) such that $p(\omega_j) \geq 0$ for all $\omega_j \in \Omega$. In this paper, we consider linear pricing scheme, that is, $p(\Omega') = \sum_{\omega_j \in \Omega'} p(\omega_j)$, for any $\Omega' \subseteq \Omega$.

Given the output tuple (X, p), if buyer i is a winner (*i.e.*, $x_i = 1$), her (quasi-linear) *utility* is defined by $u_i(X, p) = c_i - p(d_i)$; otherwise (*i.e.*, $x_i = 0$ and i is a loser), her utility $u_i(X, p) = 0$.

We now define the concept of Walrasian equilibrium.

Definition 2.1 (Walrasian equilibrium) ([18]) *A* Walrasian equilibrium *of a single-minded auction is a tuple (X, p), where $X = \langle x_1, \ldots, x_n \rangle$ is an allocation vector and $p = \langle p_1, \ldots, p_m \rangle$ is a price vector, such that*

1. *(market condition) $p(X_0) = 0$, where $X_0 = \Omega \setminus \left(\bigcup_{i:\ x_i = 1} d_i \right)$ is the set of commodities that are not allocated to any buyer. Note that this also implies that for any $\omega_j \in X_0$, $p_j = 0$.*

2. *(utility condition) The utility of each buyer is maximized. That is, for any winner i, $c_i \geq p(d_i)$, whereas for any loser i, $c_i \leq p(d_i)$. (In this case, we say the buyer is* satisfied.*)*

Chen et al. [5] showed the following hardness result for determining the existence of a Walrasian equilibrium.

Theorem 2.1 ([5]) *Determining the existence of a Walrasian equilibrium in a single-minded auction is NP-complete.*

Lehmann et al. [24] showed that in a single-minded auction, even the computation of the optimal allocation is NP-hard. Indeed, we have the following relation between Walrasian equilibrium and optimal allocation.

[3] That is where the name "single-minded" comes from.

Theorem 2.2 ([5]) *For any single-minded auction, if there exists a Walrasian equilibrium (X, p), then X must be an optimal allocation.*

Due to the above result, we know that computing a Walrasian equilibrium is at least as hard as computing the optimal allocation. However, if we do know an optimal allocation X^*, we can compute the price vector according to the following linear program:

$$\begin{aligned}
&\sum_{j:\ \omega_j \in d_i} p_j \leq c_i, \quad \forall\ x_i^* = 1 \\
&\sum_{j:\ \omega_j \in d_i} p_j \geq c_i, \quad \forall\ x_i^* = 0 \\
&p_j \geq 0, \quad \forall\ \omega_j \in \Omega \\
&p_j = 0, \quad \forall\ \omega_j \in \Omega \setminus \left(\bigcup_{i:\ x_i^* = 1} d_i \right)
\end{aligned}$$

Any feasible solution p defines a Walrasian equilibrium (X^*, p).

It is known that the price vector in a Walrasian equilibrium does not depend on its allocation vector [18]. That is, if (X, p) is a Walrasian equilibrium and Y is another optimal allocation, then (Y, p) is a Walrasian equilibrium as well[4] (for the sake of completeness, we give a proof of this statement in the appendix). Thus, given an optimal solution X^*, one can check if there exists a Walrasian equilibrium by checking if the above linear constraints have a feasible solution. Further, if a Walrasian equilibrium exists, we can compute a Walrasian equilibrium that maximizes the revenue by considering the following objective function

$$\max \sum_{j:\ \omega_j \in \Omega} p_j.$$

Therefore, we have the following conclusion:

Corollary 2.1 *For any single-minded auction, if the optimal allocation can be computed in polynomial time, then we can determine if a Walrasian equilibrium exists or not and compute one (if it exists) that maximizes the revenue in polynomial time.*

3 Approximate Walrasian Equilibrium

Theorem 2.1 says that in general computing a Walrasian equilibrium is hard. In this section, we consider two notions of approximation of a Walrasian equilibrium. The first one, called *relaxed Walrasian equilibrium*, due to Chen et al. [5], tries to maximize the number of buyers that satisfy the utility condition, given that the market condition is satisfied. We also introduce a stronger notion of approximation called *weak Walrasian equilibrium* that in addition to being a relaxed

[4] At a high level, the reason is that the winners who have non zero utility in a Walrasian equilibrium are precisely those that are present in all optimal allocations.

Walrasian equilibrium has the extra constraint that the utility condition is satisfied for *all* winners. We will define these notions formally and record some of their properties in the following two subsections.

3.1 Relaxed Walrasian Equilibrium

We first recall the notion of relaxed Walrasian equilibrium from [5].

Definition 3.1 (relaxed Walrasian equilibrium) ([5]) *A* relaxed Walrasian equilibrium *of a single-minded auction is a tuple* (X, p) *that satisfies the following condition:*

- *(market condition)* $p(X_0) = 0$*, where* $X_0 = \Omega \setminus \left(\bigcup_{i:\ x_i=1} d_i\right)$.

Our goal is to find relaxed Walrasian equilibria that maximize the number of satisfied buyers. Note that Theorem 2.1 implies that it is NP-hard to compute a relaxed Walrasian equilibrium such that the maximum number of buyers are satisfied. Let us now look at the following example from [5].

Example 3.1 Consider the following single-minded auction. Three buyers bid for three commodities, where $d_1 = \{\omega_1, \omega_2\}, d_2 = \{\omega_2, \omega_3\}, d_3 = \{\omega_1, \omega_3\}$, and $c_1 = c_2 = c_3 = 3$. Note that there is at most one winner for any allocation. Assume without loss of generality that buyer 1 wins d_1. Thus, under the condition of $p(\omega_3) = 0$ (since ω_3 is an unallocated commodity), at most two inequalities of $p(\{\omega_1\omega_2\}) \leq 3$, $p(\{\omega_2\omega_3\}) \geq 3$, $p(\{\omega_1\omega_3\}) \geq 3$ can hold simultaneously. Therefore at most two buyers can be satisfied.

	ω_1	ω_2	ω_3	valuation
Buyer 1	$\surd$	$\surd$		3
Buyer 2		$\surd$	$\surd$	3
Buyer 3	$\surd$		$\surd$	3

Chen et al. [5] conjectured that this example actually gives the lower bound on ratio of the number of satisfied buyers in relaxed Walrasian equilibria. We prove this conjecture in the following theorem.

Theorem 3.1 *Any single-minded auction has a relaxed Walrasian equilibrium for which at least* $2n/3$ *buyers are satisfied, where* n *is the number of buyers. Furthermore, such a relaxed Walrasian equilibrium can be computed in polynomial time.*

Proof. To prove the theorem, we need to show that for any single-minded auction, there is a tuple (X, p) such that at least $2n/3$ buyers are satisfied. Our proof is constructive. We start with an initial allocation Y which is chosen greedily. In the next phase, we iteratively build X from Y and set prices p for the commodities while maintaining the invariant that at least $2/3$ of the buyers in X are satisfied.

We first construct an allocation Y as follows. Initially, let $y_1 = 1$ (that is, the first buyer is a winner). For each buyer $i = 2, \ldots, n$, if $d_i \cap d_k = \emptyset$ for all $1 \leq k \leq i-1$ with $y_k = 1$, let $y_i = 1$;

otherwise, let $y_i = 0$. Let $W = \{i \mid y_i = 1\}$ be the set of winners and $L = \{i \mid y_i = 0\}$ be the set of losers in terms of Y. For each winner $i \in W$, define $L_i = \{k \in L \mid d_i \cap d_k \neq \emptyset\}$. For each loser $k \in L$, define $W_k = \{i \in W \mid d_i \cap d_k \neq \emptyset\}$.

We next describe an iterative procedure to compute (X, p) from Y. In the process the sets W and L might change. However, at every stage we will always ensure that the demand of any buyer in W is disjoint from the demands of all other buyers in W (as well as buyers that have been declared winners according to X by that stage).

Repeat the following steps till $W = L = \emptyset$:

1. If there is an $i \in W$ satisfying $|L_i| \geq 2$, set $x_i = 1$ and $x_k = 0$ for each $k \in L_i$. We set the price of commodities in d_i sufficiently large to guarantee $p(d_i \cap d_k) \geq c_k$, for any $k \in L_i$. That is, no matter how we set prices for other commodities, buyers in L_i are always satisfied (note that buyer i may not be satisfied). Then let $W \leftarrow W \setminus \{i\}$ and $L \leftarrow L \setminus L_i$ (note that at least 2/3 of the removed buyers are satisfied), update L_i for the remaining winners, and repeat this step until $|L_i| \leq 1$ for every remaining winner i.
2. For every buyer $i \in W$ such that $L_i = \emptyset$, set $x_i = 1$ and $W \leftarrow W \setminus \{i\}$.
3. Consider any remaining loser k: We know all winners in W_k still remain, and for any $\omega_j \in d_k$, there is at most one remaining winner in W_k with demand including ω_j. There are the following two possible sub-steps:
   (a) Repeat this sub-step as long as there is a loser k such that $c_k \leq \sum_{i \in W_k} c_i$. Set $x_k = 0$ and $x_i = 1$ for each $i \in W_k$. Define $p(d_i \cap d_k) = c_i$ for each $i \in W_k$. All unspecified commodities in d_k are priced at zero. Thus all buyers in W_k as well as k are satisfied. Let $W \leftarrow W \setminus W_k$ and $L \leftarrow L \setminus \{k\}$.
   (b) For any remaining loser k, we have $c_k > \sum_{i \in W_k} c_i$. Pick one (say k') arbitrarily and update the sets W and L by defining $W \leftarrow W \cup \{k'\} \setminus W_{k'}$ and $L \leftarrow L \cup W_{k'} \setminus \{k'\}$. Note that in particular this implies that all buyers in W have disjoint demands. For each $i \in W$ and $k \in L$, define L_i and W_k exactly the same way as above. Go back to step 1.

In the above construction, all unspecified commodities are priced at zero.

We first claim that the allocation produced is a valid one, i.e., no two buyers who share a commodity are both declared as winners. This follows from the fact that only buyers who are present in W are declared as winners. Further, in the procedure above it is always made sure that buyers in W do not have overlapping demands, and the demand of any buyer in W at any stage is disjoint from the demand of any buyer who has been declared a winner (according to the (partial) allocation X) in the previous stages.

Furthermore, at least $2n/3$ buyers are satisfied at the end of the algorithm (Step 1 is the only step where some buyers may not get satisfied). According to the algorithm, if a commodity is not allocated to any buyer, it must be priced at zero. Therefore, (X, p) is a relaxed Walrasian equilibrium for which at least $2n/3$ buyers are satisfied.

Finally, we argue that the algorithm terminates in polynomial time. Divide the sequence of steps of the algorithm into *iterations*, where in an iteration the algorithm starts from Step 1 and ends back at Step 1 (which can happen only if Step 3(b) is executed). To argue the running time,

we claim that in any two consecutive iterations, at least one buyer is removed from $W \cup L$.[5] For the sake of contradiction, assume that there are two consecutive iterations such that $W \cup L$ remains the same. Note that if any of the conditions of Steps 1,2 or 3(a) are satisfied, at least one buyer from $W \cup L$ is removed. Thus, in both iterations none of the conditions of Steps 1,2 or 3(a) are satisfied. In other words, in both iterations, Step 3(b) is executed. However, we claim that this is not possible as in the second iteration, the condition of either Step 1,2 or 3(a) will be satisfied, which is a contradiction. In the rest of the proof, we argue this last claim. To avoid overloading notations, let the sets W and L before the start of Step 3(b) in the first and second iteration be denoted by $W^{(1)}$, $L^{(1)}$ and $W^{(2)}$, $L^{(2)}$, respectively. Further, let $k' \in L^{(1)}$ be the loser that is moved to $W^{(2)}$ in the first iteration. Consider any $i \in W^{(1)}_{k'}$ (that is, $i \in W^{(1)}$ and $d_i \cap d_{k'} \neq \emptyset$). By the construction, the demand of i and every buyer in $W^{(1)} \setminus W^{(1)}_{k'}$ are disjoint. Thus, $W^{(2)}_i$ (recall that in the second iteration $i \in L^{(2)}$) is exactly the set $\{k'\}$. Now, as Step 3(b) was executed in the first iteration, we have $c_i \leq \sum_{j \in W^{(1)}_{k'}} c_j < c_{k'} = \sum_{j \in W^{(2)}_i} c_j$, which implies that the condition of Step 3(a) is satisfied for i in the second iteration. Thus, in the second iteration, Step 1,2 or 3(a) will be executed. □

3.2 Weak Walrasian Equilibrium

The notion of relaxed Walrasian equilibrium does not require all the winners to be satisfied. Indeed, in the proof of Theorem 3.1, to satisfy more buyers, we may require some winner to pay an amount which is higher than her valuation. However, this is infeasible in practice: the auctioneer cannot expect a winner to pay such a high price. Motivated by this observation, we introduce a stronger concept of approximate Walrasian equilibrium, which we call weak Walrasian equilibrium.

Definition 3.2 (weak Walrasian equilibrium) *A* weak Walrasian equilibrium *of a single-minded auction is a tuple (X, p) that satisfies the following conditions:*

- *(market condition) $p(X_0) = 0$, where $X_0 = \Omega \setminus \left(\bigcup_{i:\ x_i=1} d_i\right)$.*
- *(winner utility condition) The utility of each winner is maximized. That is, for any winner i, $c_i \geq p(d_i)$.*

Similar to the relaxed Walrasian equilibrium, our goal is to find weak Walrasian equilibria that maximize the number of satisfied buyers (including losers). Unfortunately, this extra restriction makes the problem much harder as the following theorem shows.

Theorem 3.2 *Given $h \leq N$, determining the existence of weak Walrasian equilibria for which at least h buyers are satisfied is strongly NP-complete, where N is the number of buyers.*

Proof. It is easy to see the problem is in NP. This is because for any tuple (X, p), we can check the market and winner utility condition efficiently and decide if there are at least h satisfied buyers in the solution (X, p).

[5]This will imply that the procedure has at most $2n$ iterations. Further, it is easy to check that each step in an iteration can be performed in polynomial time in n and m (where m is the number of commodities).

To show the NP-hardness, we give a reduction from the Independent Set problem– given a connected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices, we are asked if there is a subset of vertices $V^* \subseteq V$, $|V^*| \geq k$, such that for any $v_i, v_j \in V^*$, $(v_i, v_j) \notin E$.

Let $G' = (V', E')$ be a complete graph with n vertices, where $|V'| = \{v'_1, v'_2, \ldots, v'_n\}$. Define a new graph

$$H = (V \cup V', E \cup E' \cup E''), \text{ where } E'' = \{(v_1, v'_1), (v_2, v'_2), \ldots, (v_n, v'_n)\}.$$

We say v_i and v'_i are a pair of *counterparts* for each i.

We construct an instance of single-minded auction with $N = 2n$ buyers as follows. For each vertex $v \in V \cup V'$, we attach a buyer v with unit valuation and demand d_v, where $|d_v| = 2n$, such that for any different $v_i, v_j, v_k \in V \cup V'$, we have the following three constraints:

(a) if $(v_i, v_j) \in E \cup E' \cup E''$, then $|d_{v_i} \cap d_{v_j}| = 1$;

(b) if $(v_i, v_j) \notin E \cup E' \cup E''$, then $|d_{v_i} \cap d_{v_j}| = 0$;

(c) $|d_{v_i} \cap d_{v_j} \cap d_{v_k}| = 0$.

Specifically, the demands of buyers can be constructed according to the following rule:

```
1.  Order the vertices arbitrarily by v_1, ..., v_{2n}
2.  Let d_{v_i} = {ω_{i,1}, ..., ω_{i,2n}} be the set of demand of buyer v_i, i = 1, ..., 2n
3.  Let a_1 = a_2 = ··· = a_{2n} = 1
4.  For i = 2 to 2n
5.    For j = 1 to i − 1
6.      If (v_i, v_j) ∈ E ∪ E' ∪ E'', then
7.        ω_{i,a_i} = ω_{j,a_j} (which means it is the same commodity)
8.        a_i ← a_i + 1
9.        a_j ← a_j + 1
```

In the above construction, all unspecified commodities are different. It is easy to see it satisfies the three constraints. To complete the proof, we show that G has an independent set of size at least k if and only if there exists a weak Walrasian equilibrium (X, p) for which the number of satisfied buyers is at least $h = 2k + 2$.

Assume there is an independent set V^* of G with cardinality higher than or equal to k, *i.e.*, $|V^*| \geq k$. Note that as G is connected, $V^* \subset V$. Define the allocation and price vector as follows:

- For any $v \in V^*$, let $x_v = 1$, *i.e.*, v is a winner. Let $p(d_v \cap d_{v'}) = 1$ for each $v \in V^*$, where v' is the counterpart of v in H.
- Select a vertex $v' \in V'$, such that $(v', v'') \notin E''$ for any $v'' \in V^*$, let $x_{v'} = 1$, *i.e.*, v' is a winner. Note that as $V^* \subset V$, such a vertex always exists. Let $p(d_v \cap d_{v'}) = 1$, where v is the counterpart of v' in H.
- For every other $v \in V \cup V'$, let $x_v = 0$, *i.e.*, v is a loser. For any commodity which has no price yet set their price to be zero.

Therefore, the (X, p) constructed above satisfies Definition 3.2— all winners and their counterparts are satisfied (since their valuation is one), which implies there are at least $2k + 2$ satisfied buyers.

For the other direction, assume for the constructed auction, there is a tuple (X, p) such that at least $2k+2$ buyers are satisfied. For these satisfied buyers, we claim there are at least $k+1$ winners. This is because each winner, who is satisfied as required, compensates for at most one unit amount valuation, and each satisfied loser corresponds to at least one unit amount valuation. According to our construction of demands above, no three buyers share the same commodity. Thus, the number of winners is not less than that of satisfied losers (recall that by the market clearing condition, any commodity with a non-zero price is allocated to a winner). Furthermore, there is no common shared commodities among these $k + 1$ winners. Finally, at least k of them are corresponding to vertices in V, which implies these k vertices constitute an independent set. □

If we regard Independent Set and weak Walrasian equilibrium as optimization problems, the construction we showed above is actually a gap-preserving reduction [20]. Therefore we have the following conclusion:

Corollary 3.1 *There is an $\epsilon > 0$ such that the approximation of weak Walrasian equilibrium within a factor N^ϵ is NP-hard, where N is the number of buyers.*

4 The Tollbooth Problem

As we have seen in the preceding discussions, in general the computation of a (weak) Walrasian equilibrium in a single-minded auctions is hard. In the search of tractable instances, we consider a special case of single-minded auctions where the set of commodities and demands can be represented by a graph. Specifically, we consider the *tollbooth problem* [19], in which we are given a graph, the commodities are edges of the graph, and the demand of each buyer is a path in the graph. For the rest of this section, we will use the phrase tollbooth problem for a single-minded auction where the input is from a tollbooth problem.

4.1 Tollbooth Problem in General Graphs

For the general graph, as the following theorem shows, the computations of both the optimal allocation and Walrasian equilibrium (if it exists) are difficult.

Theorem 4.1 *If a Walrasian equilibrium exists in the tollbooth problem, it is NP-hard to compute one.*

Proof. Due to Theorem 2.2, we only need to prove the NP-hardness of computing an optimal allocation. We reduce from Set Packing by 3-sets [15]. That is, given a finite set $S = \{s_1, \ldots, s_m\}$, collection $T \subseteq 2^S$ of subsets, where $|t_i| \leq 3$ for each $t_i \in T$, and integer k, does T contain at least k mutually disjoint sets?

We construct a graph for the tollbooth problem as follows: For each $s_j \in S$, we include two vertices a_j and b_j, and an edge (a_j, b_j). For each element in T, say $t_i = (s_{j_1}, s_{j_2}, s_{j_3}) \in T$, we include two extra vertices $w_{i,1}, w_{i,2}$ and connect the graph as shown in Figure 1. If $t_i = (s_{j_1}, s_{j_2})$, then just add one new vertex $w_{i,1}$ and the edges $(b_{j_1}, w_{i,1})$, $(w_{i,1}, b_{j,2})$. If $t_i = (s_{j_1})$, then just add one new vertex $w_{j,1}$ and the edge $(b_{j_1}, w_{i,1})$.

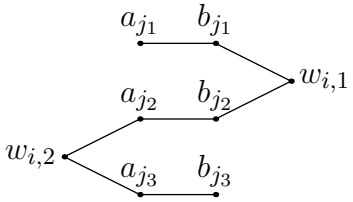


Figure 1: Gadget construction in the reduction.

We associate a buyer corresponding to every t_i with a path at valuation one. If $t_i = (s_{j_1}, s_{j_2}, s_{j_3})$, then the demand path is $(a_{j_1}, b_{j_1}, w_{i,1}, b_{j_2}, a_{j_2}, w_{i,2}, a_{j_3}, b_{j_3})$. For smaller sets, the associated demand path is the one naturally induced by the corresponding gadget.

Note that by the above construction, for any two distinct buyers i and j, their demand paths are disjoint if and only if $t_i \cap t_j = \emptyset$. In other words, T contains at least k mutually disjoint sets if and only if the value of the optimal allocation in the resulting graph is at least k. Thus the theorem follows. □

4.2 Tollbooth Problem in a Tree

In this subsection, we consider the tollbooth problem in a tree. Note that even for this simple structure, Walrasian equilibrium may not exist, as shown by Example 3.1. In this special case, however, we can determine whether a Walrasian equilibrium exists or not and compute one (if it exists) in polynomial time. Due to Corollary 2.1, we only describe how to compute an optimal allocation efficiently. To this end, we give a polynomial time dynamic program algorithm (for clarity, the details are deferred to Section 5). Therefore, we have the following conclusion:

Theorem 4.2 *For any tollbooth problem in a tree, there exists polynomial time algorithms to compute an optimal allocation, determine if a Walrasian equilibrium exists or not and compute one (if it exists).*

4.3 Comparison with Related Work

As was noted in the introduction, computing an optimal allocation (for the tollbooth problem in a tree) is the same as computing the maximum weighted edge-disjoint paths in a tree for which a polynomial time algorithm already exists [30]. Another dynamic program algorithm for computing edge-disjoint paths with maximum cardinality in a tree was shown by Garg et al. [16]. However, their algorithm crucially depends on the fact that all paths have the same valuation while our setup is the more general weighted case.

The algorithm of Tarjan [30] uses a divide and conquer approach and has a faster running time than our dynamic program based algorithm. However, we are not aware of any dynamic program based algorithm for this problem (except for the algorithm of Garg et al. [16] that only works for a special case as noted above). In particular, it will be interesting to study the generalization of our dynamic program based algorithm for graphs with constant tree width.

5 Dynamic Program based Algorithm

In this section, we present a dynamic program based polynomial time algorithm to compute an optimal allocation for the tollbooth problem on trees. We begin with the case of binary trees, then show how to generalize the dynamic program for general trees.

5.1 Basic Idea Behind the Algorithm

Before presenting the details of the algorithm, in this subsection, we will go over the basic ideas behind the algorithm (for the binary trees case).

First, we arbitrarily designate a vertex in the tree as the root– this defines a natural notion of children (and parent) for every node in the tree. The first crucial observation is the following: every edge in the tree has at most one demand from the optimal solution that passes through it. The essential idea behind the dynamic program is to guess this demand for every edge (one of the possible guesses will be to not pick any demand). Let us now consider how choosing one demand (say d) for an edge $(u, pr(u))$, where $pr(u)$ is the parent of the node u, affects the rest of the solution. Assume that u has two children v and w (u might have less than two children but those cases are simpler). Now comes the second crucial observation– the chosen demand d cannot pass through *both* v and w (as all the demands are paths in the tree). Now, if d passes through (say) v then the choice of demand for the edge (u, v) is fixed– it has to be d. Also any demand that is chosen within the subtree rooted at w has to lie within the subtree (plus the edge (u, w)). Further, apart from the constraints mentioned above, the choices made in the subtrees rooted at v and w are independent. Thus, the best choices for the subtrees at v and w can be made using a dynamic program.

5.2 The Setup

We will consider a tree $T = (V, E)$ rooted at some arbitrary node $r \in V$. For any two nodes $s, t \in V$, the unique path between them in T would be denoted by $\mathcal{P}(s, t)$. We will consider $\mathcal{P}(s, t)$ to be an ordered set of nodes $\{s = v_0, v_1, \cdots, v_{k-1}, v_k = t\}$ for some k such that $(v_i, v_{i+1}) \in E$ for $0 \leq i \leq k-1$. Further, for any node $v \in V$, its *parent* node is denoted by $pr(v)$. That is, $u = pr(v)$ if and only if $(u, v) \in E$ and $|\mathcal{P}(u, r)| < |\mathcal{P}(v, r)|$.

A demand is identified with a pair (s, t), $s, t \in V$, with the implicit understanding that the demand requires all the edges in $\mathcal{P}(s, t)$. The set of all demands is denoted by $\mathcal{D}$. Every demand $d \in \mathcal{D}$ has a valuation denoted by $c(d)$. We say two demands are *disjoint* if their corresponding paths do not have any edge in common. Thus, an allocation of $\mathcal{D}$ in T is subset of $\mathcal{D}$ such that any two demands are disjoint. Recall that our objective is to find an allocation with the maximum valuation.

For any node $u \in V$, we denote the set of demands which need the edge $(u, pr(u))$ by

$$\mathcal{D}(u) = \{(s, t) \in \mathcal{D} \mid \{u, pr(u)\} \subseteq \mathcal{P}(s, t)\}.$$

Also define the indicator functions $\alpha(\cdot, \cdot)$ and $\beta(\cdot, \cdot)$ as follows: for any $u \in V$ and $d = (s, t) \in \mathcal{D}$,

$$\begin{aligned} \alpha(u, d) = 1 \quad &\text{iff} \quad \mathcal{P}(s, t) \cap \mathcal{P}(u, r) = \{u\} \ \wedge \ u \notin \{s, t\} \\ \beta(u, d) = 1 \quad &\text{iff} \quad \mathcal{P}(s, t) \cap \mathcal{P}(u, r) = \{u, pr(u)\} \ \wedge \ pr(u) \in \{s, t\} \end{aligned}$$

Figure 2 gives examples for these two indicator functions.

$\alpha(u, d) = 1$ $\beta(u, d) = 1$

Figure 2: Illustration of $\alpha(\cdot, \cdot)$ and $\beta(\cdot, \cdot)$. (The bold edges form the demand d.)

Further let the subtree rooted at u be denoted by T_u, and define

$$\mathcal{D}(T_u) = \{(s, t) \in \mathcal{D} \mid \mathcal{P}(s, t) \cap V(T_u) \neq \emptyset\},$$

where $V(T_u)$ is the set of vertices of T_u. Note that $\mathcal{D}(u) \subseteq \mathcal{D}(T_u)$.

Finally, for any node $u \in V$ and $d \in \mathcal{D}(u)$, let $C(u, d)$ denote the maximum valuation of any allocation of $\mathcal{D}(T_u)$ in T_u that contains d. Define a special value $C(u, 0)$ as the maximum of the valuations of any allocation of $\mathcal{D}(T_u)$ in T_u that contains no demand that uses any node outside of $V(T_u)$. Recall that we are interested in $C(r, 0)$.

Before we spell out the algorithm for computing the optimal allocation, here is another definition which we will use frequently. For any $u \in V$, define

$$d^*(u) = \arg \max_{d \in \mathcal{D}(u) \ \wedge \ \beta(u,d)=1} C(u, d).$$

We will overload notation and say $d^*(u) = 0$ if $\mathcal{D}(u) = \emptyset$ or for all $d \in \mathcal{D}(u)$, $\beta(u, d) = 0$ or $C(u, 0) \geq \max_{d \in \mathcal{D}(u) \ \wedge \ \beta(u,d)=1} C(u, d)$.

5.3 The Binary Tree Case

We will first consider the case when degree of T is bounded by 2. This would give all the main ideas behind the dynamic program based algorithm.

We recall the two key observations about the structure of the problem. First, for the optimal solution OPT, every edge $e = (u, pr(u)) \in E$ appears in at most one demand $d \in OPT$. Note that if such a demand d exists then $d \in \mathcal{D}(u)$. Thus, the guesses for d are reflected in the entries $\{C(u, d)\}_{d \in \mathcal{D}(u)}$ and $C(u, 0)$ — the latter being for the case when OPT does not use any demand in $\mathcal{D}(u)$ or $\mathcal{D}(u) = \emptyset$. Second, as all the demands are paths in T, for any demand $d \in \mathcal{D}(u)$ for some $u \in V$, among the possible children v and w of u (there might be no children or just one child), at most one of v and w has the demand d in their $\mathcal{D}(\cdot)$.

We now formally define the recurrence relations for $C(\cdot, \cdot)$. For any, $u \in V$, we have the following cases:

- Case 1: u is a leaf of T.

$$C(u, 0) = 0$$

For any $d \in \mathcal{D}(u)$,

$$C(u,d) = \beta(u,d) \cdot c(d)$$

- Case 2: u has one child v.

$$C(u,0) = C(v, d^*(v))$$

For any $d \in \mathcal{D}(u)$,

$$C(u,d) = \begin{cases} \beta(u,d) \cdot c(d) + C(v,d), & \text{if } d \in \mathcal{D}(v) \\ \beta(u,d) \cdot c(d) + C(u,0), & \text{otherwise} \end{cases}$$

- Case 3: u has two children v and w.

$$C(u,0) = \max \left\{ C(v, d^*(v)) + C(w, d^*(w)), \max_{d \in \mathcal{D} \ \wedge \ \alpha(u,d)=1} (C(v,d) + C(w,d) + c(d)) \right\}$$

For any $d \in \mathcal{D}(u)$,

$$C(u,d) = \begin{cases} \beta(u,d) \cdot c(d) + C(v,d) + C(w, d^*(w)), & \text{if } d \in \mathcal{D}(v) \\ \beta(u,d) \cdot c(d) + C(v, d^*(v)) + C(d,w), & \text{if } d \in \mathcal{D}(w) \\ \beta(u,d) \cdot c(d) + C(u,0), & \text{otherwise} \end{cases}$$

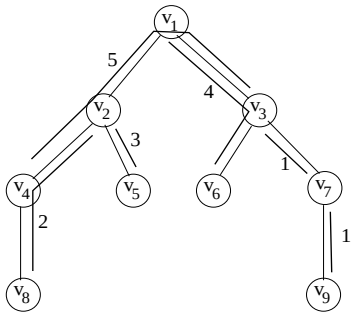
Given the above recurrence relation, the dynamic program algorithm is obvious. The algorithm maintains the table $C(\cdot,\cdot)$. It starts filling in the table from the leaves and then works its way up to the root using the recurrence relations given above. We also need to store the choices of demands made by the dynamic program in some auxiliary table $A(\cdot,\cdot)$. For the sake of completeness, the expressions for $A(u,d)$ and $A(u,0)$ are presented in Appendix B.1. It is not hard to see that one can reconstruct the solution from $A(\cdot,\cdot)$.

Example 5.1 In Figure 3, the demands are $d_1 = \{v_4, v_2, v_1, v_3\}$, $d_2 = \{v_1, v_3, v_6\}$, $d_3 = \{v_2, v_5\}$, $d_4 = \{v_8, v_4, v_2\}$, $d_5 = \{v_3, v_7\}$, $d_6 = \{v_7, v_9\}$, respectively. The value of each demand is the number beside it in the figure. We compute the optimal allocation in terms of the above dynamic program algorithm from leaves v_8, v_9, v_5, v_6 to root v_1. The entries in $C(\cdot,\cdot)$ which are never evaluated are indicated by $-$. The final allocation corresponding to $C(v_1, 0)$ is the set $\{d_1, d_3, d_5, d_6\}$.

We mention some technical points before proving the correctness of the dynamic program. First, one must be careful that the value of a demand is added only once in the computations– this is achieved by using the indicator functions $\alpha(\cdot,\cdot)$ and $\beta(\cdot,\cdot)$. Second, the values of $d^*(\cdot)$ cannot be computed as a pre-processing phase but would be computed as the algorithm proceeds. This separate notation was used to simplify the recursion expressions. Finally note that for any $u \in V$, the value $C(u, d^*(u))$ will correspond to the value of the optimal solution in the subtree $T_u \cup \{pr(u)\}$.

Lemma 5.1 *$C(r,0)$ gives the value of the optimal allocation and can be computed in time $O(mn^2)$, where m is the number of edges and n is the number of demands.*

Proof. The correctness of Case 1 is obvious. We will now prove the correctness of Case 3: the argument for Case 2 is similar and is omitted.



$C(\cdot,\cdot)$	0	d_1	d_2	d_3	d_4	d_5	d_6
v_1	12	–	–	–	–	–	–
v_2	5	3	–	–	–	–	–
v_3	2	2	6	–	–	–	–
v_4	0	0	–	–	2	–	–
v_5	0	–	–	3	–	–	–
v_6	0	–	0	–	–	–	–
v_7	1	–	–	–	–	2	–
v_8	0	–	–	–	0	–	–
v_9	0	–	–	–	–	–	1

Figure 3: The evaluation of the table $C(\cdot,\cdot)$ on the input graph on the left.

Let us first consider $C(u,0)$. Note that this quantity should compute the optimal allocation in T_u. There are two cases now. In the first case, there is some demand d such that $d \in \mathcal{D}(v) \cap \mathcal{D}(w)$. Thus, one feasible allocation in T_u is to use d and then use the optimal allocations in T_v and T_w which use d (the valuation of which are given by $C(v,d)$ and $C(w,d)$ respectively). The second term in the expression for $C(u,0)$ picks the best among such choices. In the second case, the optimal solution can combine the optimal allocations in $T_v \cup \{u\}$ and $T_w \cup \{u\}$ (which are given by $C(v,d^*(v))$ and $C(w,d^*(w))$ respectively). Note that the optimal solution has to choose one among the choices mentioned above, which we have argued above is the one computed by the expression for $C(u,0)$.

Now let us consider $C(u,d)$ for some $d \in \mathcal{D}(u)$. First consider the case when $d \in \mathcal{D}(v)$. Again after a moment's thought, it is clear that the optimal solution would combine the optimal solution in T_v using d (which is computed by $C(v,d)$) and the optimal solution in $T_w \cup \{u\}$ (which is computed by $C(w,d^*(w))$). The argument is similar for the case when $d \in \mathcal{D}(w)$. In the final case, d is of the form $\{u, p(u), \cdots\}$, in which case the optimal solution would choose d and the optimal solution in T_u (which is given by $C(u,0)$). Thus, we have argued the correctness of Case 3.

We finally do a crude bound on the running time of the algorithm. Let $m = |E|$ and $n = |\mathcal{D}|$. The algorithm has to compute at most mn entries, and for each entry we have need $O(n)$ time. Thus, in all the algorithm requires time $O(mn^2)$. □

5.4 The General Tree Case

We now consider the case when the tree T can have degree greater than two. Most of the details are the same as the binary case so we will just focus on the recurrence relations for $C(u,0)$ and $C(u,d)$ for $u \in V$ and $d \in \mathcal{D}(u)$. First note that Case 1 and Case 2 remain the same. So let us focus on the case when the degree of u is at least two.

Let u have $t \geq 2$ children given by the set $Z = \{v_1, v_2, \cdots, v_t\}$. We will now construct a weighted graph G_Z on the set of vertices $Z \cup Z'$ where Z' is a set of t new nodes: Each $v_i \in Z$ has a corresponding copy $v_i' \in Z'$. We now define the edges and their weights. For any $1 \leq i \leq t$, put in an edge (v_i, v_i') whose weight is $C(v_i, d^*(v_i))$. Finally for any $i, j \in \{1, \cdots, t\}$, include the edge

(v_i, v_j) if there exists a demand $d \in \mathcal{D}$ such that $d \in \mathcal{D}(v_i) \cap \mathcal{D}(v_j)$ with the weight

$$\max_{\alpha(u,d)=1 \ \wedge \ d \in \mathcal{D}(v_i) \cap \mathcal{D}(v_j)} (C(v_i, d) + C(v_j, d) + c(d)).$$

Now consider the maximum weighted matching $\mathcal{M}(G_Z)$ of G_Z, and let its weight be denoted by $w(\mathcal{M}(G_Z))$. Define

$$C(u, 0) = w(\mathcal{M}(G_Z)).$$

We now consider $C(u, d)$ for any $d \in \mathcal{D}(u)$. If $d \in \mathcal{D}(v_i)$, for $1 \leq i \leq t$, then define

$$C(u, d) = \beta(u, d) \cdot c(d) + C(d, v_i) + w(\mathcal{M}(G_{Z - \{v_i\}})).$$

If $d \notin \cup_{i=1}^{t} \mathcal{D}(v_i)$, then define

$$C(u, d) = \beta(u, d) \cdot c(d) + C(u, 0).$$

The expressions for $A(\cdot, \cdot)$ are given in Appendix B.2. Now we are ready to prove the following result.

Theorem 5.1 *For any tollbooth problem in a tree, there exists a polynomial time algorithm to compute an optimal allocation, determine if a Walrasian equilibrium exists or not and compute one (if it exists).*

Proof. We will show that the dynamic program discussed above computes the optimal allocation in polynomial time. The other claims follow from Corollary 2.1.

We need to prove that $C(u, 0)$ returns the value of the optimal allocation. Before we argue the correctness, note that for $t = 2$, the only possible maximal matchings in $G_{\{v_1, v_2\}}$ are $\{(v_1, v_1'), (v_2, v_2')\}$ and $\{(v_1, v_2)\}$. The weights of the two matchings are $C(v, d^*(v)) + C(w, d^*(w))$ and

$$\max_{d \in \mathcal{D} \ \wedge \ \alpha(u,d)=1} (C(v, d) + C(w, d) + c(d)),$$

respectively, and the expression for $C(u, 0)$ in the binary case is indeed $w(\mathcal{M}(G_{\{v_1, v_2\}})$.

For the general case note that for any allocation in T_u, at most one demand can be common between two different nodes v_i and v_j. And if such a demand exists, v_i and v_j cannot share a demand with any other node. If v_i does not share a demand with any other node, then some allocation in $T_{v_i} \cup \{u\}$ is a subset of the allocation. After a moment's reflection this implies that the allocation corresponds to a matching in G_Z (the edges (v_i, v_j) correspond to v_i and v_j sharing a demand in some allocation while the edges (v_i, v_i') correspond to the case when v_i does not share a demand with any other node in Z). Further, the optimal allocation OPT chooses the maximum weighted matching in G_Z.

The argument for correctness for $C(u, d)$ is essentially the same as the one given for $C(u, 0)$ and is omitted. For the special case of $t = 2$ if $d \in \mathcal{D}(v_1)$ then note that $G_{\{v_2\}}$ has its maximum (and only) matching as $\{(v_2, v_2')\}$ and the expression for $C(u, d)$ in the previous section is indeed a special case of the expression given above.

We will now analyze the running time of the dynamic program for the general tree case. For each entry in $C(\cdot, \cdot)$, we might have to create the graph G_Z which takes $O(m^2 n)$ where $m = |E|$ and $n = |\mathcal{D}|$, and computing the maximum matching takes $O(m^3 \log m)$ [14]. Recall that there are at most mn entries to be filled, thus the algorithm overall takes $O(m^3 n(n + m \log m))$. □

6 Conclusions

In the notions of approximate Walrasian equilibrium that we studied in this paper, we are concerned with relaxing the utility condition of Walrasian equilibrium (Definition 2.1). That is, we guarantee the prices of the approximate Walrasian equilibria clear the market (Definition 3.1, 3.2). Relaxing the market condition is called *envy-free pricing* and is well studied, e.g., in [19]. Unlike the general Walrasian equilibrium, an envy-free pricing always exists, and thus, a natural non-trivial goal is to compute an envy-free pricing that maximizes the revenue [19]. Similarly, trying to compute a revenue-maximizing Walrasian equilibrium is an important goal. However, even checking if a Walrasian equilibrium exists or not is NP-hard [5], which makes the task of finding a revenue-maximizing Walrasian equilibrium an even more ambitious task. Note that Corollary 2.1 shows that it is as hard as computing an optimal allocation.

Our polynomial time algorithm for determining the existence of a Walrasian equilibrium and computing one (if it exists) in a tree generalizes the result for the line case, where a Walrasian equilibrium always exists [4, 5] and can be computed efficiently.

Our work leaves some open questions. In the approximation notion of relaxed Walrasian equilibrium, we showed that any single-minded auction has a relaxed Walrasian equilibrium so that at least 2/3 of the buyers are satisfied. An interesting question is how to approximate the maximum number of satisfied buyers within a ratio better than 2/3 (note that the tight example of ratio 2/3 does not imply that 2/3 is the best possible approximation ratio). In addition, we showed that for the tollbooth problem on a general graph, it is NP-hard to compute the optimal allocation, which implies that given that a Walrasian equilibrium exists, computing one is also hard. A natural question is to resolve the complexity of determining the existence of a Walrasian equilibrium (not compute one if it exists) in a general graph.

Acknowledgments

We thank Neva Cherniavsky, Xiaotie Deng, Venkatesan Guruswami and Anna Karlin for helpful discussions and suggestions. We thank an anonymous reference for pointing out an error in an earlier version of the proof of Theorem 3.1. We thank Yossi Azar for pointing out the references [30, 16] and Xiaotie Deng for pointing out the reference [21]. We also thank the anonymous reviewers for helpful suggestions that have helped improve the presentation of the paper.

References

[1] A. Archer, C. H. Papadimitriou, K. Talwar, E. Tardos, *An Approximate Truthful Mechanism For Combinatorial Auctions with Single Parameter Agents*, Proceedings of the Symposium on Discrete Algorithms (SODA), 205-214, 2003.

[2] A. Archer, E. Tardos, *Truthful Mechanisms for One-Parameter Agents*, Proceedings of the Symposium on Foundations of Computer Science (FOCS), 482-491, 2001.

[3] K. K. Arrow, G. Debreu, *Existence of An Equilibrium for a Competitive Economy*, Econometrica, V.22, 265-290, 1954.

[4] S. Bikhchandani, J. W. Mamer, *Competitive Equilibrium in an Economy with Indivisibilities*, Journal of Economic Theory, V.74, 385-413, 1997.

[5] N. Chen, X. Deng, X. Sun, *On Complexity of Single-Minded Auction*, Journal of Computer and System Sciences, V.69(4), 675-687, 2004.

[6] X. Chen, X. Deng, *3-NASH is PPAD-Complete*, ECCC Technical Report No. TR05-134.

[7] X. Chen, X. Deng, *Settling the Complexity of 2-Player Nash-Equilibrium*, To appear in the Proceedings of the Symposium on Foundations of Computer Science (FOCS), 2006.

[8] W. Conen, T. Sandholm, *Coherent Pricing of Efficient Allocations in Combinatorial Economies*, National Conference on Artificial Intelligence (AAAI), Workshop on Game Theoretic and Decision Theoretic Agents (GTDT), 2002.

[9] P. Cramton, Y. Shoham, R. Steinberg (editors), *Combinatorial Auctions*, MIT Press, 2005.

[10] C. Daskalakis, P. W. Goldberg, C. H. Papadimitriou, *The Complexity of Computing a Nash Equilibrium*, Proceedings of the Symposium on the Theory of Computation (STOC), 71-78, 2006.

[11] C. Daskalakis, C. H. Papadimitriou, *Three-Player Games Are Hard*, ECCC Technical Report No. TR05-139.

[12] X. Deng, C. H. Papadimitriou, S. Safra, *On the Complexity of Equilibria*, Proceedings of the Symposium on the Theory of Computing (STOC), 67-71, 2002. Full version appeared in Journal of Computer and System Sciences, V.67(2), 311-324, 2003.

[13] N. Devanur, C. H. Papadimitriou, A. Saberi, V. V. Vazirani, *Market Equilibrium via a Primal-Dual-Type Algorithm*, Proceedings of the Symposium on Foundations of Computer Science (FOCS), 389-395, 2002.

[14] Z. Galil, S. Micali, H. Gabow, *An $O(EV \log V)$ Algorithm for Finding a Maximal Weighted Matching in General Graphs*, SIAM Journal on Computing, V.15, 120-130, 1986.

[15] M. R. Garey, D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[16] N. Garg, V. V. Vazirani, M. Yannakakis, *Primal-Dual Approximation Algorithms for Integral Flow and Multicut in Trees*, Algorithmica, V.18, 3-20, 1997.

[17] A. V. Goldberg, J. D. Hartline, *Collusion-Resistant Mechanisms for Single-Parameter Agents*, Proceedings of the Symposium on Discrete Algorithms (SODA), 620-629, 2005.

[18] F. Gul, E. Stacchetti, *Walrasian Equilibrium with Gross Substitutes*, Journal of Economic Theory, V.87, 95-124, 1999.

[19] V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, F. McSherry, *On Profit-Maximizing Envy-Free Pricing*, Proceedings of the Symposium on Discrete Algorithms (SODA), 1164-1173, 2005.

[20] D. S. Hochbaum (editor), *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, 1997.

[21] S. L. Huang, M. Li, B. Zhang, *Approximation of Walrasian Equilibrium in Single-Minded Auctions*, Theoretical Computer Science, V.337(1-3), 390-398, 2005.

[22] K. Jain, *A Polynomial Time Algorithm for Computing the Arrow-Debreu Market Equilibrium for Linear Utilities*, Proceedings of the Symposium on Foundations of Computer Science (FOCS), 286-294, 2004.

[23] A. S. Kelso, V. P. Crawford, *Job Matching, Coalition Formation, and Gross Substitutes*, Econometrica, V.50, 1483-1504, 1982.

[24] D. Lehmann, L. I. O'Callaghan, Y. Shoham, *Truth Revelation in Approximately Efficient Combinatorial Auctions*, ACM Conference on E-Commerce 1999, 96-102. Full version appeared in $JACM$, V.49(5), 577-602, 2002.

[25] H. B. Leonard, *Elicitation of Honest Preferences for the Assignment of Individual to Positions*, Journal of Political Economy, V.91(3), 461-479, 1983.

[26] A. Mas-Collel, W. Whinston, J. Green, *Microeconomic Theory*, Oxford University Press, 1995.

[27] A. Mu'alem, N. Nisan, *Truthful Approximation Mechanisms for Restricted Combinatorial Auctions*, Proceedings of the National Conference on Artificial Intelligence (AAAI), 379-384, 2002.

[28] C. H. Papadimitriou, T. Roughgarden, *Computing Equilibria in Multi-Player Games*, Proceedings of the Symposium on Discrete Algorithms (SODA), 82-91, 2005.

[29] C. H. Papadimitriou, *Computing Correlated Equilibria in Multi-Player Games*, Proceedings of the Symposium on the Theory of Computing (STOC), 49-56, 2005.

[30] R. E. Tarjan, *Decomposition by clique separators*, Discrete Mathematics, V.55, 221-231, 1985.

[31] S. de Vries, R. Vohra, *Combinatorial Auctions: A Survey*, INFORMS Journal on Computing, V.15(3), 284-309, 2003.

[32] L. Walras, *Elements d'economie politique pure; ou, Theorie de la richesse sociale*, (Elements of Pure Economics, or the Theory of Social Wealth), Lausanne, Paris, 1874. (Translated by William Jaffé, Irwin, 1954).

A Missing Details in Section 2

Theorem A.1 ([18]) *If (X, p) is a Walrasian equilibrium and Y is another optimal allocation, then (Y, p) is also a Walrasian equilibrium.*

Proof. As (X, p) is a Walrasian equilibrium, $p(X_0) = 0$ (where $X_0 = \Omega \setminus \left(\bigcup_{i:\ x_i=1} d_i\right)$) and $u_i(X, p) \geq u_i(Y, p)$ for any buyer i. Since Y is an optimal allocation, we have

$$\begin{aligned}
\sum_{i=1}^{n} c_i \cdot y_i - p(\Omega) &\geq \sum_{i=1}^{n} c_i \cdot x_i - p(\Omega) \\
&= \sum_{i\ :\ x_i=1} (c_i - p(d_i)) + \sum_{i\ :\ x_i=0} (0 - p(X_0)) \\
&= \sum_{i=1}^{n} u_i(X, p) \\
&\geq \sum_{i=1}^{n} u_i(Y, p) \\
&= \sum_{i\ :\ y_i=1} (c_i - p(d_i)) + \sum_{i\ :\ y_i=0} 0 \\
&= \sum_{i=1}^{n} c_i \cdot y_i - p(\Omega) + p(Y_0)
\end{aligned}$$

where $Y_0 = \Omega \setminus \left(\bigcup_{i:\ y_i=1} d_i\right)$. This implies that $p(Y_0) = 0$ and both inequalities are actually equalities. Thus, $u_i(Y, p) = u_i(X, p)$. In particular, this implies that (Y, p) defines a Walrasian equilibrium. □

B Expressions for $A(\cdot, \cdot)$

B.1 The Binary Tree Case

For any, $u \in V$, we have the following cases:

- Case 1: u is a leaf of T.

  $$A(u, 0) = \emptyset$$

  For any $d \in \mathcal{D}(u)$,

  $$A(u, d) = \emptyset$$

- Case 2: u has one child v.

  $$A(u, 0) = (v, d^*(v))$$

  For any $d \in \mathcal{D}(u)$,

  $$A(u, d) = \begin{cases} \{(v, d)\}, & \text{if } d \in \mathcal{D}(v) \\ A(u, 0), & \text{otherwise} \end{cases}$$

- Case 3: u has two children v and w.

  Let

  $$d^*(v, w) = \arg \max_{d \in \mathcal{D}\ \wedge\ \alpha(u,d)=1} (C(v, d) + C(w, d) + c(d)).$$

Then

$$A(u,0) = \begin{cases} \{(v, d^*(v)), (w, d^*(w))\}, & \text{if } C(v, d^*(v)) + C(w, d^*(w)) \geq \\ & \quad C(v, d^*(v,w)) + C(w, d^*(v,w)) + c(d^*(v,w)) \\ \{(v, d^*(v,w)), (w, d^*(v,w))\}, & \text{otherwise} \end{cases}$$

For any $d \in \mathcal{D}(u)$,

$$A(u,d) = \begin{cases} \{(v,d), (w, d^*(w))\}, & \text{if } d \in \mathcal{D}(v) \\ \{(v, d^*(v)), (w,d)\}, & \text{if } d \in \mathcal{D}(w) \\ A(u,0), & \text{otherwise} \end{cases}$$

B.2 The General Tree Case

We record the choices in $A(u,0)$ as follows: If $(v_i, v_i') \in \mathcal{M}(G_Z)$, add $(v_i, d^*(v_i))$ to $A(u,0)$; otherwise if $(v_i, v_j) \in \mathcal{M}(G_Z)$, then add $(v_i, d^*(v_i, v_j))$ and $(v_j, d^*(v_i, v_j))$ to $A(u,0)$, where

$$d^*(v_i, v_j) = \arg \max_{\alpha(u,d)=1 \ \wedge \ d \in \mathcal{D}(v_i) \cap \mathcal{D}(v_j)} (C(v_i, d) + C(v_j, d) + c(d)).$$

If $d \in \mathcal{D}(v_i)$, for $1 \leq i \leq t$, then add (v_i, d) to $A(u,d)$ and entries corresponding to $\mathcal{M}(G_{Z-\{v_i\}})$ to $A(u,d)$ as one did for the case of $A(u,0)$. If $d \notin \cup_{i=1}^{t} \mathcal{D}(v_i)$, $A(u,d) = A(u,0)$.