Algorithmica (2007) 48: 159–172
DOI: 10.1007/s00453-007-0064-9

Algorithmica



# On Walrasian Price of CPU Time[1]

Xiaotie Deng,[2] Li-Sha Huang,[3] and Minming Li[3]

**Abstract.** We study a Walrasian equilibrium model to determine the price of CPU time. The customers have jobs that require a given length of CPU slot allocation with their valuations dependent on the assigned time slots. The owner of the CPU processing time receives compensation for time slots sold to the customers, subject to the condition that the slots sold to a customer are those that the customer most desires, given the price structure for the time slots. We establish conditions for jobs to have Walrasian equilibrium, and obtain complexity results to determine the Walrasian equilibrium. In particular, the issues of excessive supply of CPU time and price dynamics are discussed under our model.

**Key Words.** Walrasian equilibrium, Price sequence, Job scheduling.

**1. Introduction.** Information technology has changed our lifestyles with the creation of many new consumer products, such as word processing software, computer games, search engines, and online communities. Digital goods and services are fast becoming everyone's shopping items. While the new goods of Information Age are enriching the market place with ever-changing products, they pose a great challenge to our understanding of economics. Such a new economy has already demanded many theoretical tools (new and old, of economics and other related disciplines) be applied to their development and production, marketing, and pricing (see, e.g. [10]). At the same time, no theory has been able to paint a clear picture of the reality, far less so in contrast to classic economics.

The lack of a full understanding of the new economy is mainly due to the fact that digital goods can often be re-produced at no additional cost, though multifold other factors could also be part of the difficulty. Not surprisingly, the marketplace practice of digital goods is anything but what is predicted by classic economics for commodities:

- While the price is much influenced by the marginal cost re-production, digital goods are not all sold at zero price but a wide range of possible (and seemingly arbitrary) prices.
- While consumers of the same product usually pay the same price in classic economics, differentiated pricing is a common practice for digital goods.
- While all positive priced goods are cleared in classic economics, digital goods and services with excessive supplies are often not free.

[1] Xiaotie Deng was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 1156/04E). Lia-Sha Huang was supported by Natural Science Foundation of China (No. 60135010,60321002) and the Chinese National Key Foundation Research and Development Plan (2004CB318108).

[2] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong.
[3] State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing, People's Republic of China. hs02@mails.tsinghua.edu.cn.

Received February 15, 2005; revised August 30, 2006. Communicated by L. Whang.
Online publication May 22, 2007.

It is understandable to have some of such contradictions to classic economics because of the special characteristics of digital goods, pointed out above. It may well be treated as a special extreme case for the commodity economy. However, as the Internet economy becomes an indispensable part of everyone's life, it is unavoidable that one may search for a comprehensive theoretical understanding of digital goods pricing mechanism. In this work we take a humble step on such a mission by focusing on CPU time as a product for sale in the market. We study it with the Walrasian pricing model in economics.

CPU time as a commercial product is extensively studied in grid computing (see, e.g. [8]). Singling out CPU time pricing will help us to set aside other complicated issues caused by secondary factors, and a complete understanding of this special digital product (or service) may shed some light on the study of other goods in the digital economy.

The utilization of CPU time by multiple customers has been a crucial issue in the development of the operating system concept. The rise of grid computing proposes to utilize computational resources fully, e.g. CPU time, disk space, bandwidth. Market oriented schemes have been proposed for efficient allocation of computational grid recourses, in [9] and [12]. Later, various practical and simulation systems have emerged in grid resource management, e.g., Spawn [15], Popcorn [14], D'Agents [3], etc. Besides the resource allocation in grids, Feigenbaum et al. [7] gave an example of introducing economic mechanism into routing between Internet domains.

Our approach deals with the relationship of key concepts in the economy: commodity, price, and customer valuation. We are interested in the price equilibrium model in the tradition of Walras [16], Arrow and Debreu [1] and the complexity of computing equilibrium [6]. In most CPU allocation models, CPU time is treated as the same commodity that would reduce the theoretical problem to the classic model of one commodity economy, and the rigidity of customer job makes it further simplified. For such customer jobs, price equilibrium is quite impossible since the CPU time is often not fully utilized. The equilibrium price would be, in such a case of excessive supply, zero. We explore a more general job model that the customer valuation would be dependent on the completion time of its job. We also study the non-increasing property of price sequence in job scheduling models, which can be viewed as a comparative work to Chen et al.'s [5] study in the price sequence of online auctions.

The paper is organized as follows. The relevant definitions and necessary preliminaries are introduced in Section 2. In Section 3 we first establish a theorem on the existence of Walrasian equilibrium for our general CPU job model, in the traditional linear programming and integer programming paradigm. With this theorem, we prove that it is NP-hard to determine the existence of Walrasian equilibrium in the job scheduling model. In Section 4 we focus on a class of linear valuation functions. As a positive result, we prove that Walrasian equilibrium exists in the model if the total available CPU time is a little more than the sum of all required CPU service time. In Section 5 we establishes a non-increasing property of equilibrium price for job scheduling models. We conclude our work in Section 6 with remarks and discussion on the current results and possible future extensions.

**2. Preliminaries.** In this section we introduce the job scheduling problem, a model of valuation functions, and the definition of Walrasian equilibrium.

2.1. *XOR Bids and Valuation Functions*. We adopt the notion of combinatorial auctions [13] in our discussion, which is helpful for us to establish our results. Consider an exchange economy $(\Omega, I, V)$:

- Commodities: The seller sells $m$ kinds of indivisible commodities in the market. Let $\Omega = \{\omega_1 \times \delta_1, \ldots, \omega_m \times \delta_m\}$ denote the set of commodities, where $\delta_j$ is the available quantity of the item $\omega_j$.
- Agents: There are $n$ agents in the market acting as buyers, denoted by $I = \{1, 2, \ldots, n\}$.
- Valuation functions: Each buyer $i \in I$ has a valuation function $v_i$: $2^{\Omega} \to \mathbb{R}^+$ to submit the maximum amount of money he is willing to pay for a certain bundle of items. Let $V = \{v_1, v_2, \ldots, v_n\}$.

Nisan [13] introduced a formalism, the bidding language, to express various valuation functions. Any valuation function can be represented in this form [13].

DEFINITION 1 [13]. An XOR combination of two valuation functions $v_1$ and $v_2$ is defined by

$$(v_1 \text{ XOR } v_2)(S) = \max\{v_1(S), v_2(S)\}.$$

An *atomic bid*, or so-called single-minded bid, is a valuation function $v$ defined by a pair $\{S, q\}$ where $S \subset \Omega$ and $q \in \mathbb{R}^+$:

$$v(T) = \begin{cases} q, & \text{if} \quad S \subset T, \\ 0, & \text{otherwise.} \end{cases}$$

An *XOR bid* is a set of atomic bids combined by XOR operators, written as

$$v = (S_1, q_1) \text{ XOR } (S_2, q_2) \text{ XOR } \cdots \text{ XOR } (S_n, q_n).$$

Given $(\Omega, I, V)$ as the input, the market will determine an *allocation* and a *price vector* as the output:

- An *allocation* $X = \{X_0, X_1, X_2, \ldots, X_n\}$ is a partition of $\Omega$, in which $X_i$ is the bundle of commodities assigned to buyer $i$ and $X_0$ is the set of unallocated commodities.
- A *price* vector $p$ is a non-negative vector in $\mathbb{R}^m$, whose $j$th entry is the price of goods $\omega_j \in \Omega$.

The *social efficiency* of an allocation $X$ is the sum of all buyers' valuation: $\sum_{i=1}^{n} v_i(X_i)$. An allocation $X^* = \{X_0^*, X_1^*, \ldots, X_n^*\}$ is said to be *optimal* if it maximizes social efficiency, i.e. $\sum_{i=1}^{n} v_i(X_i^*) \geq \sum_{i=1}^{n} v_i(X_i)$ for any other allocation $X = \{X_0, X_1, \ldots, X_n\}$.

For any subset $T = \{\omega_1 \times \sigma_1, \ldots, \omega_m \times \sigma_m\} \subset \Omega$, define $p(T)$ as $p(T) = \sum_{j=1}^{m} \sigma_j p_j$. If buyer $i$ is assigned to a bundle $X_i$ and the price vector is $p$, his *utility* is defined to be $u_i(X_i, p) = v_i(X_i) - p(X_i)$.

2.2. *The CPU Job Scheduling Problem*. We consider two types of players in a market-driven CPU resource allocation model: a resource provider and $n$ consumers. The provider sells to the consumers CPU time slots and the consumers each have a job that requires a fixed number of CPU time slots, and its valuation function depends on

the time slots assigned to the job, usually the last assigned CPU time slot. We assume that all jobs are released at time $t = 0$ and the $i$th job needs $s_i$ time units. We denote by $v_i(\cdot)$ the valuation function of agent $i$ on the time slots assigned to it. In general, the jobs may be or may not be interruptible but we focus on jobs that are interruptible without preemption cost, as is often modeled for CPU jobs.

Using the notion of the previous subsection, for the job scheduling problem, there are $m$ commodities (time units), $\Omega = \{\omega_1, \ldots, \omega_m\}$, and $n$ buyers (jobs), $I = \{1, 2, \ldots, n\}$, in the market. Each buyer has a valuation function $v_i$, usually only dependent on the completion time. Moreover, if not explicitly mentioned, every job's valuation function is non-increasing with respect to the completion time. We call such valuation functions *non-increasing valuation functions*.

### 2.3. *Walrasian Equilibrium Price*

DEFINITION 2 [11]. A *Walrasian equilibrium* for an exchange economy $(\Omega, I, V)$ is a tuple $(X, p)$, where $X = \{X_0, X_1, \ldots, X_n\}$ is an allocation and $p$ is a price vector, satisfying

(1) $p(X_0) = 0$;
(2) $u_i(X_i, p) \geq u_i(B, p), \forall B \subset \Omega, \forall 1 \leq i \leq n$.

Such a price vector is also called a market clearing price, or a Walrasian price, or an equilibrium price.

There is a well-known proposition of Walrasian equilibrium:

PROPOSITION 3 [4]. *Walrasian equilibrium maximizes the social efficiency, i.e. if* $(X, p)$ *is a Walrasian equilibrium, then* $X$ *is an optimal allocation.*

EXAMPLE 1. Two non-interruptible jobs compete for four time units $\{\omega_1, \omega_2, \omega_3, \omega_4\}$. Their valuation functions are

$$v_1 = (\{\omega_1, \omega_2\}, 7) \text{ XOR } (\{\omega_2, \omega_3\}, 4) \text{ XOR } (\{\omega_3, \omega_4\}, 1),$$
$$v_2 = (\{\omega_1\}, 7) \text{ XOR } (\{\omega_2\}, 5) \text{ XOR } (\{\omega_3\}, 3) \text{ XOR } (\{\omega_4\}, 1).$$

The equilibrium price is $(3, 1, 0, 0)$. Job 1 gets $\{\omega_2, \omega_3\}$ and job 2 gets $\{\omega_1\}$. It is an example of Walrasian equilibrium for the job scheduling problem.

## 3. Existence and Complexity.

In this section we propose a sufficient and necessary condition for the existence of Walrasian equilibrium in an exchange economy with indivisible commodities. As its application, we show that deciding the existence of Walrasian equilibrium is strongly NP-hard even when restricted to job scheduling models.

### 3.1. *Relation to Linear Programming.*

Bikhchandani and Mamer [2] proved that Walrasian equilibrium exists in an exchange economy with indivisible commodities if and only if the buyers' welfare cannot be improved by making the commodities divisible.

In other words, Bikhchandani's theorem [2] claims that Walrasian equilibrium exists if and only if the optimum of an integer programming problem equals the optimum of its linear relaxation.

However, the size of their linear programming problem is exponential to the total number of commodities which is unacceptable computationally. Chen et al. [4] obtained a similar result for atomic bids while the complexity is linear to the number of commodities and buyers. We extend their result to XOR bids and show that the size of the linear programming problem is linear to the number of items and XOR clauses.

Assume in an economy, $\Omega = \{\omega_1 \times \delta_1, \ldots, \omega_m \times \delta_m\}$ is the set of commodities, $I = \{1, 2, \ldots, n\}$ is the set of buyers. Each buyer $i$ has a valuation function which can be represented by $r_i$ pairs:

$$(S_{i1}, q_{i1}) \text{ XOR } (S_{i2}, q_{i2}) \text{ XOR } \cdots \text{ XOR } (S_{ir_i}, q_{ir_i}).$$

When items are divisible, maximizing the social efficiency is equivalent to solving the following linear programming problem:

**LPR**:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \sum_{j=1}^{k_i} q_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i,j|\omega_k \in S_{ij}} x_{ij} \le \delta_k, \qquad \forall \omega_k \in \Omega, \\ & \sum_{j=1}^{r_i} x_{ij} \le 1, \qquad \forall 1 \le i \le n, \\ & 0 \le x_{ij} \le 1, \qquad \forall i, j. \end{aligned}$$

Denote its integer restriction by **IP**. The following theorem shows that the gap between the integer programming problem **IP** and its linear relaxation implies the non-existence of the Walrasian equilibrium.

THEOREM 4. *In an economy with indivisible commodities and XOR valuation functions, the Walrasian equilibrium exists if and only if the optimum of* **IP** *equals the optimum of* **LPR**. *The size of the LP problem is linear to the total number of XOR bids.*

PROOF. We can write **LPR** in a more convenient form. Define a counting function $f(i, j)$ by $f(i, j) = \sum_{k=1}^{i-1} r_k + j$ for $i = 1, \ldots, n$ and $j = 1, \ldots, r_i$. Let $N = \sum_{i=1}^{n} r_i$ be the total number of XOR clauses. Construct an $m \times N$ matrix $\Delta = (\Delta_{kl})$ where $\Delta_{kf(i,j)} = 1$ if and only if the item $k \in S_{ij}$. Let the column vector $\delta = (\delta_1, \ldots, \delta_m)^T$ represent the available amount of each commodity. Construct an $n \times N$ matrix $A$ to represent the constraints that each buyer $i$ can get at most one bundle of commodities: $A = (a_{kl})$, where $a_{kl} = 1$ if and only if $f(k, 1) \le l \le f(k, r_k)$. Let $x = (x_{11}, \ldots, x_{1r_1}, x_{21}, \ldots, x_{nr_n})^T$ be the allocation: if buyer $i$ gets the faction $v$ of $S_{ij}$, then $x_{ij} = v$. Define the valuation vector $q = (q_{11}, \ldots, q_{1r_1}, q_{21}, \ldots, q_{nr_n})^T$. Finally, let $I_n$ be the $n$-dimensional identity matrix and let $1_n$ be the $n$-dimensional vector of all 1's. Then **LPR** can be formulated

as:

**LPR**:

$$\max q^T x$$
$$\text{s.t.} \begin{pmatrix} \Delta \\ A \\ I_N \\ -I_N \end{pmatrix} x \le \begin{pmatrix} \delta \\ 1_n \\ 1_N \\ 0_N \end{pmatrix};$$

and its dual problem:

**DLPR**:

$$\min (y, u, z, w) \begin{pmatrix} \delta \\ 1_n \\ 1_N \\ 0_N \end{pmatrix}$$
$$\text{s.t.} (y, u, z, w) \begin{pmatrix} \Delta \\ A \\ I_N \\ -I_N \end{pmatrix} = q^T,$$
$$(y, u, z, w) \ge 0.$$

Here $y = (y_1, \ldots, y_m)$ is an $m$-dimensional row vector; $u = (u_1, \ldots, u_n)$ is an $n$-dimensional row vector; $z = (z_{11}, \ldots, z_{1r_1}, z_{21}, \ldots, z_{nr_n})$ is an $N$-dimensional row vector; and $w = (w_{11}, \ldots, w_{1r_1}, w_{21}, \ldots, w_{nr_n})$ is an $N$-dimensional row vector.

If the optimum of **IP** equals the optimum of **LPR**, let $x^*$ be the optimal solution to **IP**, and let $(y^*, u^*, z^*, w^*)$ be the optimal solution to **DLPR**. By the slackness condition, we have

$$(y^*, u^*, z^*, w^*) \left( \begin{pmatrix} \Delta \\ A \\ I_N \\ -I_N \end{pmatrix} x^* - \begin{pmatrix} \delta \\ 1_n \\ 1_N \\ 0_N \end{pmatrix} \right) = 0. \tag{3.1}$$

We can construct a Walrasion equilibrium $(X, p)$ by $x^*$ and $y^*$. If $x^*_{ij} = 1$, then define $X_i = S_{ij}$; if $x_{ij} = 0$ for all $j$, then define $X_i = \emptyset$. $X_0 = \Omega - \bigcup_{i=1}^n X_i$. The price vector $p$ is simply set to be $y^*$. We show that $(X, p)$ is a Walrasian equilibrium in the following three steps.

(1) If $x_{ij} = 1$, i.e., buyer $i$ get his $j$th bid, by (3.1), $w^*_{ij} = 0$. So we have

$$\sum_{k=1}^{m} y^*_k \Delta_{kf(i,j)} + u^*_i + z^*_{ij} = q_{ij}.$$

This implies $u^*_i \le q_{ij} - \sum_{k=1}^m y^*_k \Delta_{kf(i,j)} = u_i(S_{ij}, p) = u_i(X, p)$. For all $1 \le j' \le$

$r_i$, $j' \neq j$, we have $z^*_{ij'} = 0$ by (3.1). Hence,

$$\sum_{k=1}^{m} y^*_k \Delta_{kf(i,j')} + u^*_i - w^*_{ij'} = q_{ij'}$$

$$\Rightarrow u_i(S_{ij'}, p) = q_{ij'} - \sum_{k=1}^{m} y^*_k \Delta_{kf(i,j')} \leq u^*_i \leq u_i(S_{ij}, p).$$

(2) If $x_{ij} = 0$ for some $i$ and all $1 \leq j \leq r_i$, that is, buyer $i$ loses all of his bids, by the slackness condition, $z^*_{ij} = 0$ for all $1 \leq j \leq r_i$. Hence,

$$\sum_{k=1}^{m} y^*_k \Delta_{kf(i,j)} + u^*_i - w^*_{ij} = q_{ij} \quad \Rightarrow \quad u_i(S_{ij}, p) = q_{ij} - \sum_{k=1}^{m} y^*_k \Delta_{kf(i,j)} \leq 0.$$

(3) If item $k$ is not sold out, that is, $(\Delta x^* - \delta)_k \neq 0$, then by the slackness condition (3.1), $p_k = y^*_k = 0$.

On the other hand, if a Walrasian equilibrium $(X, p)$ exists, we construct the vector $x^*$ by $X$ in the converse process of the above proof. By Proposition 3, $x^*$ is the optimal solution to **IP**. We show that it is also the optimum of **LPR**. Define $(y^*, u^*, z^*, w^*)$ by $y^* = p^T$, $u^* = (u_i(X, p))_{i=1}^n$, $z^* = (z_{ij}) = (p(S_{ij}) + u_i - q_{ij})$, and $w^* = 0$. Then we can verify that

$$(y^*, u^*, z^*, w^*) \left( \begin{pmatrix} \Delta \\ A \\ I_N \\ -I_N \end{pmatrix} x^* - \begin{pmatrix} \delta \\ 1 \\ 1 \\ 0 \end{pmatrix} \right) = 0.$$

By the duality theory of linear programming, $x^*$ is also the optimum of **LPR**. □

3.2. *Reducing the LP Size for the Non-Increasing Valuation Function*. The main difficulty of directly applying Theorem 4 to job scheduling problems is that the number of XOR clauses is exponential to the number of available time units if the jobs are allowed to be interrupted. More precisely, if the number of total time units is $m$ and job $i$'s time span is $s_i$, then there are $\binom{m}{s_i}$ XOR clauses in the valuation function $v_i$. In this subsection we try to overcome this obstacle.

In a feasible allocation, the allocated time units to every job may or may not be consecutive. We call the former one *consecutive allocation*, and the latter *general allocation*.

LEMMA 5. *General allocation cannot gain more social efficiency than consecutive allocation.*

PROOF. We only need to show that we can adjust any allocation to a consecutive allocation without reducing social efficiency. Assume job $i$ is the last one that is not allocated consecutive time units. We can adjust the allocation as follows: move all the time units allocated to $i$ to the end of time sequence just before $i$'s completion time. This adjustment will not cause any job to finish later. According to the non-increasing property of valuation functions, the total social efficiency will not decrease. The lemma follows by induction. □

Lemma 5 shows that general allocation cannot gain more social efficiency than consecutive allocation, when the valuation functions are non-increasing with respect to the completion time. What happens if we stands by the Walrasian equilibrium's point of view?

In the next theorem, if jobs are all interruptible, we call the scheduling problem *general scheduling*, and *restricted scheduling* if jobs are all non-interruptible.

THEOREM 6. *Walrasian equilibrium exists in the general scheduling model if and only if Walrasian equilibrium exists in the restricted scheduling version.*

PROOF. Denote *IPG* and *IPR* as the integer programming problem of general scheduling and its restricted version respectively. Denote $M_{IPG}$ and $M_{IPR}$ as their optimal value. Lemma 5 implies that $M_{IPG} = M_{IPR}$. So it is enough to prove that $M_{LPG} = M_{LPR}$ for their linear relaxation.

$M_{LPG} \geq M_{LPR}$ stands trivially.

Without loss of generality, all variables are assumed to be integral. So the solutions of *LPG* and *LPR* are rational. Let $x^*$ be a solution of *LPG*, then there exists a large integer $N$ such that $Nx^*$ is an integral vector. So $x^*$ can be decomposed into a sum of feasible allocations, i.e. $x^* = \sum_{i=1}^{N}(1/N)x_i$ where each $x_i$ is a feasible integral solution of *LPG*. There is at least one $x_k$ such that $V(x_k/N) \geq V(x^*)/N = M_{LPG}/N$, where $V(x)$ denote the social efficiency in the allocation $x$. By Lemma 5, we can rearrange $x_k$ into a consecutive allocation $\hat{x}$ without loss of social efficiency. Hence $V(\hat{x}) \geq M_{LPG}$ which implies $M_{LPR} \geq M_{LPG}$.

This completes the proof. □

Compared with $\binom{m}{s_i}$ clauses in one valuation function of the general scheduling problem, there are only $(m - s_i + 1)$ XOR clauses in the restricted version. Hence Theorem 6 dramatically reduces the scale of the corresponding linear programming problem in Theorem 4. The following example illustrates an application of Theorems 4 and 6.

EXAMPLE 2. Suppose there are three time units for two buyers:

$$v_1 = (\{1, 2\}, 15) \text{ XOR } (\{2, 3\}, 2),$$
$$v_2 = (\{1\}, 20) \text{ XOR } (\{2\}, 20) \text{ XOR } (\{3\}, 8).$$

It is easy to see that optimal integer allocation produces the social efficiency 23. However, the linear program can yield a better solution 27.5 by distributing $\{1, 2\} \times 0.5$ to buyer 1 and $\{1\} \times 0.5 + \{2\} \times 0.5$ to buyer 2. Therefore, the example does not admit a Walrasian equilibrium.

3.3. *Strong NP-Hardness.* Although Theorem 4 can help us to determine the existence of Walrasian equilibrium in job scheduling problems, it is still a hard problem because integer programming is hard. In this subsection we show that it is strongly NP-hard to decide whether Walrasian equilibrium exists in a job scheduling problem.

DWE PROBLEM. Given $m$ time units and altogether $n$ jobs. The $i$th job needs $s_i$ time units. Each job's valuation on its allocated time units only depends on its completion time and is non-increasing with respect to this time. Determine the existence of Walrasian equilibrium in this job scheduling problem.

We will reduce a strongly NP-hard problem, 3-Partition, to a *DWE* problem.

3-PARTITION PROBLEM. Given a set of $3N$ integer numbers $S = \{s_1, s_2, \ldots, s_{3N}\}$, and an integer $B$, which satisfy $\sum_{i=1}^{3N} s_i = NB$, and $B/4 < s_i < B/2$. Determine whether there exists a partition of $S$ into $P_1, P_2, \ldots, P_N$, such that

$$\sum_{j \in P_i} s_j = B \qquad \text{for all} \quad 1 \le i \le N. \tag{3.2}$$

For the preceding 3-Partition problem, we construct a *DWE* problem in which there are altogether $NB$ time units, with $3N$ jobs each applying for $s_i$ time units, respectively. If job $i$'s completion time is $t_i$, its valuation is set to be $s_i(N - \lceil t_i/B \rceil + 1)$.

This job scheduling problem naturally induces an integer programming problem and its linear relaxation. Denote the optimum of these two programming problems by $M_{IP}$ and $M_{LP}$, respectively. By Theorem 4, $M_{IP} = M_{LP}$ if and only if *DWE* admits a Walrasian equilibrium.

LEMMA 7. $M_{LP} \le BN(N+1)/2$.

PROOF. For a time slot $j$ falling in the interval $(kB, (k+1)B]$, consider its *average profit*. Assume it is assigned to the first $r$ jobs in fraction $\{T_1 \times \sigma_1, \ldots, T_r \times \sigma_r\}$, i.e. $j \in T_i$ for all $1 \le i \le r$ and $\sum_{i=1}^{r} \sigma_i = 1$. Then its average profit is defined by $AP_j = \sum_{i=1}^{r} \sigma_i v_i(T_i)/s_i$. It is easy to see that the social efficiency of a (fractional) allocation is the sum of all time units' average profits. By the construction of $v_i$, we have $v_i(T_i) \le s_i(N-k)$. This implies $AP_j \le N - k = N - \lceil j/B \rceil + 1$ and the equality holds if and only if $T_i$ terminates before $(k+1)B + 1$ for all $i$. So the proof completes. □

LEMMA 8. *The three* 3*-Partition problem has a solution if and only if the maximal social efficiency in the corresponding job scheduling problem equals* $BN(N+1)/2$.

PROOF. If 3-Partition has a solution $P_1, P_2, \ldots, P_N$, then distribute the first $B$ time units to jobs in $P_1$, $t_{B+1}$ to $t_{2B}$ to jobs in $P_2$, and so on. The social efficiency under such allocation is $BN(N+1)/2$ which is optimal guaranteed by Lemma 7.

On the other hand, if the maximal social efficiency reaches $BN(N+1)/2$ under an allocation $\{T_1, T_2, \ldots, T_{3N}\}$, then the average profit $AP_j$ equals $(N - \lceil j/B \rceil + 1)$ for all time units $1 \le j \le NB$. From the proof of Lemma 7, this happens only if every $T_i$ is restricted in some interval $(kB, (k+1)B]$. The solution of 3-Partition follows from such an allocation. □

THEOREM 9. *Determination of Walrasian Equilibrium Existence* (*DWE*) *is strongly NP-hard.*

PROOF. With an oracle of the *DWE* problem, we can solve 3-Partition in polynomial time:

If the oracle declares that Walrasian equilibrium does not exist in the job scheduling problem, then due to Theorem 4 and Lemma 7, $M_{IP} < M_{LP} \leq BN(N+1)/2$ which means that the social efficiency of the best allocation is less than $BN(N+1)/2$. Thus Lemma 8 ensures that 3-Partition does not have a solution.

If the oracle declares that there exists a Walrasian equilibrium, due to Theorem 4, we can obtain in polynomial time the maximal social efficiency by solving the linear programming problem. A simple comparison will yield whether 3-Partition has a solution. □

**4. Excessive CPU Time and Equilibrium in the MWCT Model.** In this section we concentrate on a special scheduling problem with linear valuation functions. Assume $n$ jobs are released at the time $t = 1$ for a single machine, the $j$th job's time span is $s_j \in \mathbb{N}^+$ and weight $w_j \geq 0$. The goal of the scheduling is to minimize the weighted completion time: $\sum_{i=1}^{n} w_i t_i$, where $t_i$ is the completion time of job $i$. Such a problem is called the MWCT (Minimal Weighted Completion Time) problem in this section.

We can convert an MWCT problem to an exchange economy: the market sells $m$ commodities $T = \{t_1, \ldots, t_m\}$ (time slots) to $n$ buyers $I = \{1, 2, \ldots, n\}$ (jobs). The valuation of buyer $i$ to a bundle $T_i$ is $v_i(T_i) = w_i(m - t)$, where $|T_i| = s_i$ and $t$ is the largest item in $T_i$ (the completion time). Due to the nice form of the valuation functions, we immediately have the following proposition:

PROPOSITION 10. *The social optimum in the economy is equivalent to the optimum in its corresponding MWCT problem.*

By Theorem 6, we can assume without loss of generality that buyer $i$ only applies for consecutive time units. Note that there is a trivial $O(n \log n)$-time algorithm to find the optimum of MWCT when $m \geq \sum_{i=1}^{n} s_i$. It just simply executes the jobs in a *heavier average weight earlier* order, i.e. if $w_i/s_i > w_j/s_j$, then job $i$ must be executed before job $j$.

Though the universal problem for MWCT is intractable both at optimization and determination of the existence of equilibrium, we do have some promising results when the total number of processor times is large enough:

THEOREM 11. *In a single machine MWCT job scheduling problem, Walrasian equilibrium always exists when $m \geq EM + \Delta$, where $m$ is the total number of processor times, $EM = \sum_{i=1}^{n} s_i$, and $\Delta = \max_k \{s_k\}$.*

PROOF. First, when $m$ is large enough to process all jobs, the optimal solution of a single machine MWCT is easily constructed: process the jobs in a sequence in which $\{w_j/s_j \mid j = 1, 2, \ldots\}$ is non-increasing.

Without loss of generality, assume that $w_1/s_1 \geq w_2/s_2 \geq \cdots \geq w_n/s_n$ and $d_i = 0$ for buyer $i$. The bundle of time slots assigned to buyer $i$ is

$$X_i = \{\alpha_i, \alpha_i + 1, \ldots, \beta_i = \alpha_i + s_i - 1\},$$

where $\alpha_i$ and $\beta_i$ denote the start time and the end time of job $i$, respectively. We denote by $EM$ the sum of all job's time span, i.e. $\sum_{i=1}^{n} s_i$.

Given the optimal allocation $X = \{X_0, X_1, \ldots, X_n\}$, it is enough to prove the existence of a market clearing price. We can achieve this by solving the inequalities derived from Definition 2. That is, a market clearing price vector $p$ should satisfy

$$(4.1)\quad \sum_{k=\alpha_i}^{\beta_i} p_k \le v_i(X_i) = w_i(m-\beta_i), \qquad \text{for all } i$$

$$(4.2)\quad v_i(X_i) - p(X_i) \ge v_i(T) - p(T), \qquad \text{for all } i \text{ and } \quad T = \{\omega_j, \omega_{j+1}, \omega_{j+s_i-1}\}$$

$$(4.3)\quad p_k \ge 0, \qquad \forall 1 \le k \le m.$$

Define an $m$ vector $\Delta$ by $\Delta_k = w_i/s_i$ in which $\omega_k \in X_i$. By the algorithm of optimization, $\Delta_k$ is non-increasing. Construct an $m$ vector $\hat{p}$ by (1) $\hat{p}_1 = 0$; (2) $\hat{p}_{k+1} = \hat{p}_k - \Delta_k$.

For a buyer $i$, with another $X' = \{k - s_i + 1, \ldots, k\}$ and $k > \beta_i$, we have

$$\begin{aligned} p(X) - p(X') &= \sum_{l=\beta_i - s_i + 1}^{\beta_i} p_l - \sum_{l=k-s_i+1}^{k} p_l = \sum_{l=\beta_i - s_i + 1}^{\beta_i} p_l - p_{l+k-\beta_i} \\ &\le \sum_{l=\beta_i - s_i + 1}^{\beta_i} w_i/s_i(k-\beta_i) \le w_i(k-\beta_i) = v_i(X) - v_i(X'). \end{aligned}$$

If $k < \beta_i$, similarly

$$\begin{aligned} p(X') - p(X) &= \sum_{l=k-s_i+1}^{k} p_l - p_{l+\beta_i-k} \ge \sum_{l=k-s_i+1}^{k} w_i/s_i(\beta_i - k) \\ &\ge w_i(k-\beta_i) = v_i(X') - v_i(X). \end{aligned}$$

Hence $\hat{p}$ satisfies (4.2). Moreover, $p = \hat{p} - \hat{p}_m = (\hat{p}_1 - \hat{p_m}, \hat{p}_2 - \hat{p_m}, \ldots, 0)$ will satisfy both (4.2) and (4.3).

By the construction of $p$, we have $p_l - p_{EM} \le (EM - l)w_i/s_i$ and $p_{EM} = 0$. So if $m \ge EM + \max_k \{s_k\}$, for any buyer $i$, we verify (4.1):

$$\begin{aligned} p(X_i) &= \sum_{l=\alpha_i}^{\beta_i} p_l \le \sum_{l=\alpha_i}^{\beta_i} p_{EM} + w_i/s_i(EM - l) \\ &= \sum_{l=\alpha_i}^{\beta_i} w_i/s_i(EM - l) = w_i(EM - (\beta_i + \alpha_i)/2) \\ &\le w_i(m - \beta_i) = v(X_i). \end{aligned}$$

This completes the proof. □

The following example shows that the enough excessive CPU time is necessary.

EXAMPLE 3. There are two jobs $\{1, 2\}$ and five CPU time slots $\{t_1, t_2, t_3, t_4, t_5\}$ in the market. $w_1 = 3, s_1 = 2$; $w_2 = 4, s_2 = 3$. Allocating $(t_1, t_2)$ to job 1 and $(t_3, t_4, t_5)$ to job 2 will produce social efficiency at 9. However, allocating $(t_1, t_2) \times 0.5$ and $(t_3, t_4) \times 0.5$ to job 1 and $(t_1, t_2, t_3) \times 0.5$ to job 2 will produce social efficiency at 10. Hence, by Theorem 4, the Walrasian equilibrium does not exist.

**5. Price Sequence.** If Walrasian equilibrium exists, then we can find not only an optimal schedule but also a price vector for all time units. In this section we prove the existence of a non-increasing price sequence if Walrasian equilibrium exists under the assumption that the valuation functions are non-increasing with respect to completion time.

The proofs of the following lemmas can be found in the full version of this paper.

LEMMA 12. *If Walrasian equilibrium exists with a general allocation and an equilibrium price, then there exists a Walrasian equilibrium with a consecutive allocation and a corresponding equilibrium price.*

PROOF. Suppose there exists a general allocation and a corresponding market clearing price. We will show that we can always adjust the allocation to a consecutive allocation while preserving the equilibrium property. Assume the buyer $k$ is the last buyer not allocated consecutive time units, we can adjust the allocation as follows: allocate the time slot $j$ which is the completion time of buyer $k$'s preceding buyer $k'$ to buyer $k$. Return some time slot $j'$ before $j$ which is formerly allocated to buyer $k$ back to buyer $k'$. We claim that $p_j = p_{j'}$. Otherwise, either buyer $k$ or $k'$ will gain more utility by such an exchange which contradicts the equilibrium property. With $p_j = p_{j'}$, this exchange will not affect the equilibrium property. So the allocation after the exchange still preserves Walrasian equilibrium. The lemma follows by induction. □

LEMMA 13. *Equilibrium price in a consecutive allocation must satisfy the following property*: *if buyer $i$ is allocated before buyer $i'$, then the price of any time unit allocated to $i$ will be higher than or equal to the price of every time unit allocated to buyer $i'$.*

PROOF. Otherwise buyer $i'$ will have a higher utility by choosing the time unit in $i$'s allocation which violates the property mentioned in the lemma. □

DEFINITION 14. Given two sequences $P = \{p_1, p_2, \ldots, p_m\}$, $Q = \{q_s, q_{s+1}, \ldots, q_m\}$ $(0 < s \leq m)$, define the Minimum $s$-Sum of $P$ at position $i$ as

$$ms_i(P) = \min_{T \subset |i|, |T| = s} \left\{ \sum_{k \in T} p_k \right\}$$

and the Maximal Difference of $Q$ to $P$ as

$$MD(Q, P) = \max_{s \leq i \leq m} \{q_i - ms_i(P)\}.$$

LEMMA 15. *For a permutation* $\pi$: $|m| \to |m|$ *and a vector* $P = (p_1, \ldots, p_m) \in \mathbb{R}^m$, *define* $\pi(P) = (p_{\pi(1)}, \ldots, p_{\pi(m)})$. *Given two non-increasing sequences* $P = \{p_1, p_2, \ldots, p_m\}$ *and* $Q = \{q_s, q_{s+1}, \ldots, q_m\}$, *we have* $MD(Q, P) \leq MD(Q, \pi(P))$.

PROOF. Suppose $MD(Q, P) = q_i - ms_i(P)$. Notice that for any permutation $\pi$: $|m| \to |m|$, we always have $ms_i(P) \geq ms_i(\pi(P))$ since $P$ is a non-increasing sequence. □

LEMMA 16. *If Walrasian equilibrium exists with consecutive allocation, then for every consecutive segment, sorting the prices in non-increasing order will still yield an equilibrium price sequence.*

PROOF. Assume the equilibrium is broken after sorting one segment, then there must be some unsatisfied buyers who can gain more utility by another allocation.

First, if a buyer $i$ can gain more utility after sorting, then the completion time of its better allocation must be situated in the sorted segment. Otherwise the buyer cannot get more utility compared with the unsorted price vector.

Assume at this moment there are $m$ units of time in the sorted segment and the unsatisfied buyer prefers to choose $s$ time units in this segment to gain more utility. Then we denote this buyer's valuation according to the completion time in the adjusted interval as $Q = \{q_s, q_{s+1}, \ldots, q_m\}$, the non-increasing price sequence after adjustment as $P = \{p_1, p_2, \ldots, p_m\}$, and the price sequence before adjustment as $P'$. The unsatisfied buyer can only get better utility by so doing if $MD(V, P) > MD(V, P')$, which contradicts Lemma 15. So, no matter what combination of time units the buyer chooses, he cannot obtain better utility after the price adjustment, which means the equilibrium property still holds. □

By Lemmas 12, 13, and 16, we reach the main theorem of the section:

THEOREM 17. *If there exists a Walrasian equilibrium in a job scheduling problem, we can always let it be an equilibrium with consecutive allocation and a non-increasing equilibrium price vector.*

**6. Conclusion and Discussion.** In the paper we have shown in Theorem 4 the relation of the duality theory of linear programming and the existence of Walrasian equilibrium with indivisible commodities. Theorems 6 and 9 are examples of its direct application in algorithmic complexity issues. With similar technique to the proof of Theorem 9, we prove the NP-hardness of determining the existence of Walrasian equilibrium in various job scheduling models.

In Section 4 we study the MWCT model and show that enough excessive CPU time always admits Walrasian equilibrium. In Section 5 we prove the existence of a non-increasing equilibrium price in the job scheduling model.

## References


[1] K. J. Arrow and G. Debreu. Existence of an equilibrium for competitive economy. *Econometrica*, 22:265–290, 1954.

[2] S. Bikhchandani and J. W. Mamer. Competitive equalibrium in an exchange economy with indivisibilities. *Journal of Economy Theory*, 74:385–413, 1997.

[3] J. Bredin, D. Kotz, and D. Rus. Market-based resource control for mobile agents. In *Proceeding of the* 2*nd International Conference on Autonomous Agents*. ACM Press, New York, 1998.

[4] N. Chen, X. Deng, and X. Sun. On complexity of single-minded auction. *Journal of Computer and System Sciences*, 69(4):675–687, 2004.

[5] N. Chen, X. Deng, X. Sun, and A. C. Yao. Dynamic price sequence and incentive compatibility. In *Proceedings of ICALP*, 2004.

[6] X. Deng, C. Papadimitriou, and S. Safra. On the complexity of price equilibria. *Journal of Computer and System Sciences*, 67(2):311–324, 2003.

[7] J. Feigenbaum, C. H. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of PODC*, 2002.

[8] D. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic models for allocating resources in computer systems. 1996.

[9] D. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic algorithms for load balancing in distributed computer systems. In *Proceedings of DCS*, 1988.

[10] A. V. Goldberg, J. D. Hartline, and A. Wright. Competitive auctions and digital goods. 2001.

[11] F. Gul and E. Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economy Theory*, 87:95–124, 1999.

[12] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, 1989.

[13] N. Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 1–12, 2000.

[14] N. Nisan, L. London, O. Regev, and N. Camiel. Globally distributed computation over the internet—the POPCORN project. In *Proceedings of the International Conference on Distributed Computing Systems*, 1998.

[15] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: a distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992.

[16] L. Walras. *Elements d'Economique Politique Pure*. Corbaz, Lausanne, Switzerland, 1874.