On Walrasian Price of CPU Time
Extended Abstract

Xiaotie Deng[1,⋆], Li-Sha Huang[2,⋆⋆], and Minming Li[2]

[1] Department of Computer Science, City Univ. of Hong Kong
csdeng@cityu.edu.hk
[2] State Key Laboratory of Intelligent Technology and Systems
Dept. of Computer Science and Technology, Tsinghua Univ., Beijing, China
{hs02,liminming98}@mails.tsinghua.edu.cn

Abstract. We study a Walrasian Equilibrium model to determine the prices of CPU time as merchandise. 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 CPU processing time receives compensation for time slots sold to the customers, subject to the condition that the slots sold to a customer is those that customer most desires, given the price structure for the time slots. We establish conditions for jobs to have Walrasian Equilibrium, and obtain algorithm and complexity results to determine Walrasian equilibrium price and allocation. In particular, the issues of excessive supply of CPU time and price dynamics are discussed under our model.

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 have been able to paint a clear picture of the reality, far less so in comparison with 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 multi-fold 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:

⋆ Supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 1156/04E).
⋆⋆ Supported by Natural Science Foundation of China (No.60135010,60321002) and the Chinese National Key Foundation Research and Development Plan (2004CB318108).

L. Wang (Ed.): COCOON 2005, LNCS 3595, pp. 586–595, 2005.

- 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 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.

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 quest 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 operating system concept. The rise of grid computing proposes to fully utilize computational resources, e.g. CPU time, disk space, bandwidth. Market oriented schemes have been proposed for efficient allocation of computational grid recourses, by [9, 12]. And 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 of 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 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 propert of price sequence in job scheduling models, which can be viewed as a comparative work to Chen *et. al.* [5]'s study in the price sequence of online auctions.

The paper is organized as follows. In Section 2, the relevant definitions and necessary preliminaries are introduced. 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 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 Walrasian Equilibrium price.

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 (Ω, I, V):

- Commodities: The seller sells m kinds of indivisible commodities in the market. Let $\Omega = \{\omega_1 \times \delta_1, ..., \omega_m \times \delta_m\}$ denote the set of commodities where δ_j is the available quantity of the item ω_j.
- Agents: There are n agents in the market acting as buyers, denoted by $I = \{1, 2, ..., n\}$.
- 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. $V = \{v_1, v_2, ..., v_n\}$.

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

Definition 1. *[13] An XOR combination of two valuation functions v_1 and v_2 is defined by:*

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

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 $q \in \mathbb{R}^+$:

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

An XOR bid is a combination of several atomic bids by XOR operators, written as

$$v = (S_1, q_1) \ XOR \ (S_2, q_2) ... \ XOR \ (S_n, q_n)$$

Given (Ω, I, V) as input, the market is to determine an *allocation* and a *price vector* as output:

- An *allocation* $X = \{X_0, X_1, X_2, ..., X_n\}$ is a partition of Ω 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 nonnegative vector in $\mathbb{R}^m$ whose j-th entry is the price of good $\omega_j \in \Omega$.

The *social efficiency* of an allocation X is the sum of all buyers' valuation: $\sum_{i=1}^{n} v_i(X_i)$. An allocation $X^* = \{X_0^*, X_1^*, ..., X_n^*\}$ 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 $X = \{X_0, X_1, ..., X_n\}$.

For any subset $T = \{\omega_1 \times \sigma_1, ..., \omega_m \times \sigma_m\} \subset \Omega$, define $p(T)$ as $p(T) = \sum_{j=1}^{m} \sigma_j p_j$. If buyer i is assigned to a bundle of commodities 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 has a job that requires a fixed number of CPU time, 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 modelled for CPU jobs.

Using the notion of the previous subsection, for the job scheduling problem, there are m commodities (time units), $\Omega = \{\omega_1, ..., \omega_m\}$, and n buyers (jobs) , $I = \{1, 2, ..., n\}$, 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 w.r.t. 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 (Ω, I, V) is a tuple (X, p), where $X = \{X_0, X_1, ..., X_n\}$ is an allocation and p is a price vector, satisfying that:*

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

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

There is a well-known proposition of Walrasian equilibrium:

Proposition 1. *[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 $\{\omega_2, \omega_3\}$, 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 strong NP-hard even when restricted to job scheduling models.

3.1 Relation to Linear Programming

Bikhchandani *et. al.* [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, $\Omega = \{\omega_1 \times \delta_1, ..., \omega_m \times \delta_m\}$ is the set of commodities, $I = \{1, 2, ..., n\}$ is the set of buyers. Each buyer i has a valuation function which can be represented by r_i pairs:

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

Maximization of social efficiency is equivalent to solving the following linear programming problem when items are divisible:

LPR:

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

Denote its integer restriction as **IP**. Now we can reach the main theorem of this subsection:

Theorem 1. *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 LP problem is linear to the total number of XOR bids.*

The proof of this theorem can be found in the full version of this paper.

3.2 Reducing LP Size for Non-increasing Valuation Function

The main difficulty of directly applying Theorem 1 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 must be or may not be consecutive. We call the former one *consecutive allocation*, and the latter *general allocation*.

Lemma 1. *General allocation can not gain more social efficiency than consecutive allocation.*

Lemma 1 shows that general allocation cannot gain more social efficiency than consecutive allocation, when the valuation functions are non-increasing w.r.t completion time. What happens if we stands on 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 2. *Walrasian Equilibrium exists in the general scheduling model if and only if Walrasian Equilibrium exists in the restricted scheduling version.*

Compared to $\binom{m}{s_i}$ clauses in one valuation function of general scheduling problem, there are only $(m - s_i + 1)$ XOR clauses in the restricted version. Hence Theorem 2 dramatically reduces the scale of the corresponding linear programming problem in Theorem 1. The following example illustrates an application of Theorem 1 and Theorem 2.

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 the 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, Walrasian Equilibrium does not exist in the example.

3.3 Strong NP-Hardness

Although Theorem 1 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 will show that it is strong 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 w.r.t. this time. Determine the existence of Walrasian Equilibrium in this job scheduling problem.

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

3-Partition problem: Given a set of $3N$ integer numbers $S = \{s_1, s_2, ...s_{3N}\}$, 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 $P_1, P_2, ...P_N$, such that

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

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 $s_i(N - \lceil \frac{t_i}{B} \rceil + 1)$.

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}. By Theorem 1, $M_{IP} = M_{LP}$ if and only if DWE admits a Walrasian Equilibrium.

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

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

Theorem 3. *Determination of Walrasian Equilibrium Existence(DWE) is strong NP-hard.*

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

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

If the oracle declares that there exists a Walrasian Equilibrium, due to Theorem 1, we can in polynomial time obtain 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 MWCT Model

In the section, we concentrate on a 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 MWCT (Minimal Weighted Completion Time) in this section.

We can convert an MWCT problem to an exchange economy: the market sells m commodities $T = \{t_1, ..., t_m\}$ (time slots) to n buyers $I = \{1, 2, ..., n\}$ (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 (completion time). Due to the nice form of the valuation functions, we immediately have the proposition:

Proposition 2. *The social optimum in the economy is equivalent to the optimum in its corresponding MWCT problem.*

By Theorem 2, we can w.o.l.g. assume that buyer i only applies for consecutive time units. Note that there is a classical $O(n \log n)$-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 existence of equilibrium, we do have some promising result when the total number of processor time is large enough:

Theorem 4. *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 time, $EM = \sum_{i=1}^{n} s_i$ and $\Delta = \max_k \{s_k\}$.*

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. $w_1 = 3, s_1 = 2; w_2 = 4, s_2 = 3$. Allocate (t_1, t_2) to job 1 and (t_3, t_4, t_5) to job 2 will produce social efficiency at 9. However, allocate $(t_1, t_2) \times 0.5$ 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 1, 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 non-increasing price sequence if Walrasian Equilibrium exists under the assumption that the valuation functions are non-increasing w.r.t. completion time.

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

Lemma 4. *If Walrasian Equilibrium exists with a general allocation and an equilibrium price, then there exists a Walrasian Equilibrium with a consecutive allocation and corresponding equilibrium price.*

Lemma 5. *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'.*

Definition 3. *Given two sequences $P = \{p_1, p_2, ...p_m\}$, $Q = \{q_s, q_{s+1}, ...q_m\}$ $(0 < s \leq m)$, 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 6. *For a permutation $\pi : |m| \to |m|$ and a vector $P = (p_1, ..., p_m) \in \mathbb{R}^m$, define $\pi(P) = (p_{\pi(1)}, ..., p_{\pi(m)})$. Given two non-increasing sequences $P = \{p_1, p_2, ...p_m\}$ and $Q = \{q_s, q_{s+1}, ...q_m\}$, we have $MD(Q, P) \leq MD(Q, \pi(P))$.*

Lemma 7. *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.*

By Lemma 4, Lemma 5 and Lemma 7, we reach the main theorem of the section:

Theorem 5. *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 1 the relation of the duality theory of linear programming and the existence of Walrasian Equilibrium with indivisible commodities. Theorem 2 and Theorem 3 are the examples of its direct application in algorithmic complexity issues. With similar technique in the proof of Theorem 3, we prove the NP-hardness of determining existence of Walrasian Equilibrium in various job scheduling models.

In Section 4, we study MWCT model and show that enough excessive CPU time always admit Walrasian Equilibrium. We believe that the equilibrium price will lead to an incentive compatible pricing mechanism in the 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 *the Proceeding of the 2nd International Conference on Autonomous Agents*. ACM Press, 1998.
4. Ning Chen, Xiaotie Deng, and Xiaoming Sun. On complexity of single-minded auction. *Journal of Computer and System Sciences*, 69(4):675–687, 2004.
5. Ning Chen, Xiaotie Deng, Xiaoming Sun, and Andrew C. Yao. Dynamic price sequence and incentive compatibility. In *the 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 *the 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 *the 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 Trans. on Computers*, 38(5):705–717, 1989.
13. N. Nisan. Bidding and allocation in combinatorial auctions. In *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 *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.