Auction Protocols for Decentralized Scheduling*

Michael P. Wellman, William E. Walsh

Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48109-2110 USA
E-mail: (wellman, wew)@umich.edu

and

Peter R. Wurman

Department of Computer Science
North Carolina State University
Raleigh, NC 27695 USA
E-mail: wurman@csc.ncsu.edu

and

Jeffrey K. MacKie-Mason

School of Information
University of Michigan
Ann Arbor, MI 48109-1092 USA
E-mail: jmm@umich.edu

Decentralized scheduling is the problem of allocating resources to alternative possible uses over time, where competing uses are represented by autonomous agents. Market mechanisms use prices derived through distributed bidding protocols to determine schedules. We investigate the existence of equilibrium prices for some general classes of scheduling problems, the quality of equilibrium solutions, and the behavior of an ascending auction mechanism and bidding protocol. To remedy the potential nonexistence of price equilibria due to complementarities in preference, we introduce additional markets in combinations of basic goods. Finally, we consider direct revelation mechanisms, and compare to the market-based approach.

Journal of Economic Literature Classification Numbers: C62, C70, D44.

* Revised and extended version of "Some economics of market-based distributed scheduling", presented at the *Eighteenth International Conference on Distributed Computing Systems*, Amsterdam, May 1998.

1. INTRODUCTION

Allocating resources with and for distributed computing systems presents particular challenges attributable to the decentralized nature of the computation. Consider, for instance, the problem of scheduling network access for programs representing various users on the Internet. In such an environment, system modules (user programs) represent independent entities (users) with conflicting and competing scheduling requirements, who may possess localized information relevant to their needs (such as the value they place on a particular schedule). To recognize this independence, we treat the modules as *agents*, ascribing each of them autonomy to decide how to deploy resources under their control in service of their interests. We assume that the agents communicate via messages in which they may convey some of their private information.

The challenges for a decentralized solution to the scheduling problem then include: How do we manage message passing, reach closure, and determine the final schedule? Further, since the value of alternatives depends on the information held privately by the agents, how do we elicit messages that contain the information needed to formulate a desirable schedule?

The first problem is fundamental in distributed computing systems, due to the asynchrony of communication. Imagine that Bob, based on what he currently knows, announces "I want to use the conference room at 11 am". Later, Bob's boss Alice announces "I want to hold a manager's meeting to discuss merit raises at 11 am in the conference room". If Bob were permitted to send another message, he might announce: "but any time before 2 pm is acceptable for me". This new message might change what Ted wants to announce, and so forth. A distributed system to solve a scheduling problem based on message-passing needs to specify which messages are admissible (have a well-formed syntax), when they may be sent, and when closure (if ever) will be reached and a schedule formulated.

The second problem is the subject of the theory of mechanism design. Given agents' private information about resources and preferences, and some social welfare critera, some schedules can be considered more desirable than others. Then the problem is to design a *mechanism*: to choose rules for formulating a schedule based on received messages, and possibly for exchanging other resources (e.g., money), that will induce the agents to reveal the private information needed to determine the socially more desirable schedules.

Within this setting, a decentralized scheduling method can be analyzed according to how well it exhibits the following properties:

- Self-interested agents can make effective decisions with local (private) information, without knowing the private information and strategies of other agents.
- The method requires minimal communication overhead.
- The method reaches closure in reasonable time and at reasonable computational expense.

• Solutions do not waste resources. If there is some way to make some agent(s) better off without harming others, it should be done. A solution that cannot be improved in this way is called *Pareto optimal*.

(As suggested above, it might sometimes be appropriate to adopt some stronger optimality criteria, based on a judgment about social value of the various agents.) The four criteria above bring together central concerns of distributed computation and mechanism design.

Straightforward distributed scheduling policies—such as first-come first-served, shortest-job-first, priority-first, and combinations thereof—do not generally possess these properties. For example, queue-position schemes are insensitive to relative value based on the *substance* of the task being performed. On the other hand, priority-based schemes beg the question of how to set priorities so that desirable results follow. If self-interested agents are free to set their own priorities, then without some incentive to the contrary, they will specify maximum priority for whatever they are interested in.

Citing such limitations, several have proposed that distributed resource allocation problems be solved via market mechanisms [6], an approach we have called *market-oriented programming* (MOP) [40]. In MOP, we define agent activities in terms of resources required and produced, reducing an agent's decision problem to evaluating the tradeoffs of acquiring different resources. These tradeoffs are represented in terms of market prices, which define a common scale of value across the various resources. The problem for designers of computational markets is to specify the configuration of resources traded (formally designated *goods* in the market), and the mechanism by which agent interactions determine prices.

Assuming that a scheduling problem must be decentralized, markets can provide several advantages:

• Markets are naturally decentralized. Agents make their own decisions about how to bid based on the prices and their own relative valuations of the goods.

• Communication is limited to the exchange of *bids* and *prices* between agents and the market mechanism. In particular settings, it can be shown that price systems minimize the dimensionality of messages required to determine Pareto optimal allocations [14].

• Since agents must back their representations with exchange offers, some mechanisms can elicit the information necessary to achieve Pareto and global optima (or come within some tolerance of optimal) in some well-characterized situations.

Of course, all of these benefits do not automatically accrue as a result of setting up a market-like environment. Although the First and Second Welfare Theorems [21] guarantee strong performance for some market mechanisms, these results are formally restricted to rather special environments. Scheduling problems often exhibit complementarities and nonconvexities, which violate the ideal conditions for the welfare theorems or for particular market protocols.

Prior work applying market-inspired mechanisms to scheduling [1, 13, 20, 37, 38] and other distributed resource allocation problems [6, 17, 35, 47] has produced promising empirical results. Understanding the scope of these methods, and developing a general design methodology for computational markets, however, requires an analytical characterization of their properties. In our own MOP work, we have adopted the framework of general equilibrium theory [21], and have found that our computational markets behave predictably when conditions of the theory are met [25, 40, 42]. We have also applied the approach to discrete optimization problems—where the conditions guaranteeing desirable outcomes are not satisfied—and have found (not surprisingly) that the methods sometimes work, and other times break down [39, 41].

Since scheduling problems very often involve discrete (indivisible) resource units, we have undertaken to analyze directly the behavior of computational market mechanisms for such problems. We start by defining a general class of discrete allocation problems, and characterizing some distinctions particularly meaningful in the scheduling domain. We show how some recent results in economic theory apply to the scheduling problem, and report our own extensions and analysis.

In the next section, we motivate the work with a concrete example of a simple factory scheduling problem. In Section 3, we provide a formal economic model of a general version of the problem, and in Section 4 we relate some equilibrium and optimality properties associated with the problem. In Section 5, we briefly describe a general framework for auction protocols, and describe and analyze one such protocol in Section 6. To address limitations of the basic market formulation, we present an extended combinatorial market in Section 7, and a direct revelation mechanism in Section 8. Finally, we consider future work in Section 9.

2. A FACTORY SCHEDULING ECONOMY

Consider a factory with an unscheduled day shift. There are eight one-hour time slots, labeled 9:00 to 16:00 according to their respective end times. Slots can be allocated for the production of customer orders. The factory has a *reserve price* for each time slot, representing the minimum price that the factory is willing to accept in exchange for that time slot.

Assume each customer agent has one job it wants completed. An agent's job is defined by its duration (length), its deadline, and the value (expressed in dollars) the agent places on the job. An agent is willing to spend up to this value to complete its job. To do so, the agent must acquire a number of slots no less than the length (not necessarily contiguous), no later than the deadline. The agent gets no value if its job cannot be completed before its deadline. The global value of a solution is the sum of values of the agents holding the goods, which is the sum of the reserve price for each time slot that was not sold, plus the value associated with each customer agent that meets its job deadline.

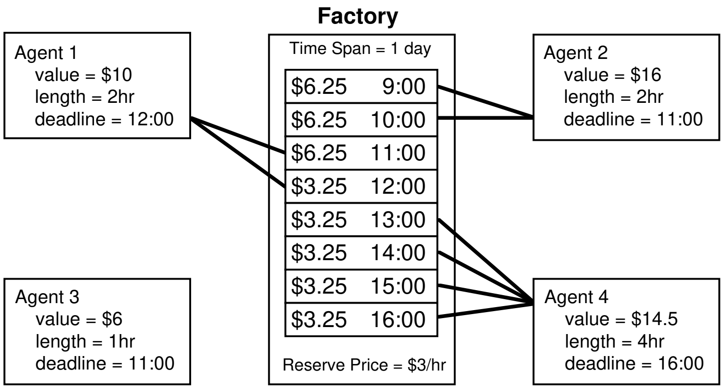

FIG. 1. A factory scheduling economy. Lines connecting the agents to time slots represent one feasible allocation.

EXAMPLE 2.1. The agents are shown in Figure 1.[1] Since the sum of lengths exceeds available factory time, it is not possible for all of the agents to produce their orders. The allocation depicted in Figure 1 represents a global optimum.

Given an assignment of prices to goods, we can define an agent's optimal choice as a set of slots that complete the job at the minimum cost, or the empty set if the the job could not not be completed for less than its value. The reader can verify that at the prices shown in Figure 1, each agent makes a locally optimal choice in the globally optimal allocation.

3. FORMAL MODEL OF THE SCHEDULING ECONOMY

We define a general discrete resource allocation problem in terms of the following elements:

- G, a set of n discrete goods,
- A, a set of m agents, and $\perp$ representing the seller or null agent,
- prices $p = \langle p_1, \ldots, p_n \rangle$.

We assume that agents have quasilinear utility functions, meaning that their valuations can be measured in terms of a common numeraire, which for convenience can be taken to be "money". Therefore, we can directly compare the utility of different agents, and meaningfully treat the sum as a measure of global value. Agent j gets utility $v_j(X) + M_j$ for holding the set of goods X, $X \subseteq G$, and M_j units of money.

[1] An interactive online demonstration of the ascending auction (Section 6) applied to this example can be found at `http://auction.eecs.umich.edu/demos/factory.html`.

Let $H_j(p)$ denote the maximum surplus value achievable by agent j at prices p. That is,

$$H_j(p) \equiv \max_{X \subseteq G} \left[v_j(X) - \sum_{i \in X} p_i \right].$$

Note that for some prices, an agent may maximize its surplus with the empty set.

A *solution* is a mapping $f : G \to A \cup \{\perp\}$, indicating which agent, if any, gets each good. Let $F_j \equiv \{i | f(i) = j\}$ denote the set of goods allocated to agent j, and $F_\perp \equiv \{i | f(i) = \perp\}$ the set of unallocated goods in f.

The seller of good i has utility equal to its *reserve value* q_i if the good is unallocated, or the money it receives for the good if it is allocated. Intuitively, the reserve value denotes the value to the owner, or the "system", of not allocating the good to any agent. Different time slots could potentially have different reserve values; for instance, a factory may have a higher reserve price for evening hours to cover overtime expenses.

The *global value* of a solution, $v(f)$, is the sum of the agent values achieved and the reserve value of goods not used by agents,[2]

$$v(f) \equiv \sum_{i \in F_\perp} q_i + \sum_{j=1}^{m} v_j(F_j).$$

We measure the system value of a solution *ex post*, that is, conditional on knowing all agents' valuations. A solution is *optimal* if no other solution has higher value.

In the remainder of this article, we present market schemes and auction protocols for this very general resource allocation problem. However, the theoretical results and examples we present focus on particular subclasses of scheduling problems where each agent has one job to complete. For these problems, we associate each agent j with a job length λ_j, and $1 \le K_j \le n$ deadlines $d_j^1 < \cdots < d_j^{K_j}$ and value levels $v_j^1 > \cdots > v_j^{K_j}$. The value $v_j(X)$ of a set of goods X is v_j^k if d_j^k is the earliest deadline such that X includes at least λ_j time slots no later than d_j^k. For convenience we represent the time slots as integers, starting from one. Note that although the domain of $v_j(\cdot)$ comprises all $O(2^n)$ possible time-slot bundles, for the scheduling problem this value function can be encoded compactly in terms of $O(K_j)$ deadlines and values.

If $\lambda_j = 1$ for all j, we call the scheduling problem *single unit*. Problems violating this constraint are *multiple unit*. If each agent j has a single deadline ($K_j = 1$), we call the problem *fixed deadline*. If $K_j > 1$ for some j (i.e., j accrues greater value for finishing the job sooner), then we call the problem *variable deadline*.

[2] Because all agents have utility that is linear in money, the total value obtained from money is constant and hence can be ignored.

4. PRICE EQUILIBRIA

DEFINITION 4.1. [price equilibrium] A solution f is in *equilibrium* at prices p iff

1. For all agents j, $v_j(F_j) - \sum_{i \in F_j} p_i = H_j(p)$.
2. For all i, $p_i \geq q_i$.
3. For all $i \in F_\perp$, $p_i = q_i$.

Intuitively, this definition states that in equilibrium, each agent (including the seller) gets an allocation that maximizes its utility given the current prices. Equilibria sometimes exist, and are generally not unique. Consider Example 2.1. The solution shown, with only agent 3 receiving no goods, is in equilibrium at the set of prices suggested, with slots 9:00, 10:00, and 11:00 each having a price of \$6.25, and all other slots having a price of \$3.25. The same solution is also in equilibrium with respective prices of \$6.50 and \$3.35, and many other combinations. The equilibrium solution has value \$40.50, which is optimal. Indeed it had to be, as demonstrated by the following result.

THEOREM 4.1. *For the general discrete resource allocation problem, if there exists a p such that f is in equilibrium at p, then f is an optimal solution.*

Proof. Bikhchandani and Mamer [3] and Gul and Stacchetti [12] provide proofs for an exchange economy without reserve prices. A slight extension accounts for reserve prices.

Let f be in equilibrium at prices p, and let f' be an alternative solution. By the definition of solution value, we have

$$v(f) = \sum_{j=1}^{m} v_j(F_j) + \sum_{i \in F_\perp} q_i.$$

Since in equilibrium the price of unallocated goods is equal to the reserve value,

$$\begin{aligned} v(f) &= \sum_{j=1}^{m} v_j(F_j) + \sum_{i \in F_\perp} p_i \\ &= \sum_{j=1}^{m} v_j(F_j) + \sum_{i \in G} p_i - \sum_{i \in G \setminus F_\perp} p_i. \end{aligned}$$

For all goods, equilibrium prices must be at least as high as reserve values. Therefore,

$$\begin{aligned} v(f') &\leq \sum_{j=1}^{m} v_j(F'_j) + \sum_{i \in F'_\perp} p_i \\ &= \sum_{j=1}^{m} v_j(F'_j) + \sum_{i \in G} p_i - \sum_{i \in G \setminus F'_\perp} p_i. \end{aligned}$$

Let $P = \sum_{i \in G} p_i$. Rearranging the above expressions, we have

$$\begin{aligned} v(f) &= \sum_{j=1}^{m} \left(v_j(F_j) - \sum_{i \in F_j} p_i \right) + P, \\ v(f') &\leq \sum_{j=1}^{m} \left(v_j(F'_j) - \sum_{i \in F'_j} p_i \right) + P. \end{aligned}$$

By the definition of equilibrium, F_j maximizes the term inside the parentheses, for each agent j. Thus, we must have that $v(f) \geq v(f')$. ∎

This result confirms the usual consequence of competitive equilibrium: that no further gains from trade are possible and so the result is Pareto optimal. Since we assume that agent values are expressible in price units, Pareto optimality corresponds to global optimality.

EXAMPLE 4.1. There are two agents as described in Table 1, and the reserve price of each good is zero.

The optimal solution, $f(1) = f(2) = 1$, is not in equilibrium at any prices, and indeed no equilibrium exists in this case. If p were in equilibrium, then $p_1 \geq \$2$ and $p_2 \geq \$2$, otherwise agent 2 would demand one of the goods. But if these inequalities hold then agent 1 would not demand the two time slots it requires.

In this example, the nonexistence of equilibrium prices is due to *complementarities* in agent preferences. Agent 1 considers the two time slots complementary in that it values one iff it has the other. Complementarities cannot arise in the single-unit scheduling problem.

LEMMA 4.1. *For all instances of the single-unit scheduling problem, there exists a unique price equilibrium p^* such that for any other price equilibrium p, $p^*_i \leq p_i$, for all i.*

TABLE 1.
A problem with no equilibrium. Adapted from a demonstration [23] that price equilibria may not exist in the FCC market for radio spectrum.

Name	Job Length	Deadline	Value
Agent 1	2	2	\$3
Agent 2	1	2	\$2

Proof. An exchange economy characterized by quasilinear utilities for single goods always has a unique minimum equilibrium price vector in the sense described [34]. The single-unit scheduling problem is a special case. ▮

THEOREM 4.2. *Any optimal solution to the single-unit scheduling problem (fixed or variable deadline) is supported by a price equilibrium.*

Proof. By Lemma 4.1, the single-unit scheduling problem always has at least one price equilibrium p. By Theorem 4.1, p supports an optimal solution. Since p supports an optimal solution, it can be shown that all optimal solutions must be supported by p [3, 12]. ▮

Together, Theorems 4.1 and 4.2 establish that a solution to the single-unit scheduling problem is optimal iff it is supported by a price equilibrium. Example 4.1 demonstrates that relaxing the single-unit restriction immediately leads to the possibility that an equilibrium will not exist. For the general setting, Milgrom [24] shows that a single complementarity is sufficient to prevent a price equilibrium. In the scheduling case, it is easy to show that whenever there is one agent j with $\lambda_j \geq 2$ and valuation for some deadline exceeding the corresponding reserve prices, we can construct an example without an equilibrium, using $\lambda_j - 1$ additional agents with single-unit jobs.

In addition to the single-unit restriction of Theorem 4.2, we can identify a few other conditions that guarantee the existence of equilibrium. If all agents have additive preferences over goods then an equilibrium exists.[3] Additivity of preference is one sufficient condition for *gross substitutability*—if the price for one good goes up, demand does not go down for any other good—which in turn guarantees the existence of equilibrium [15]. Bikhchandani and Mamer [3] present some other technical conditions for existence of equilibrium, which do not seem to be immediately expressible in scheduling terms.

[3]Note that preferences are not additive in the multiple-unit scheduling problem. However, equilibrium would exist if agents had additive preferences for completing multiple single-unit jobs.

Finally, note that in an equilibrium for the scheduling economy, prices for differently allocated time slots must be nonincreasing with their time indices.

THEOREM 4.3. *Let f be a solution for the scheduling economy, in equilibrium at prices p. If $i \in F_j$ for any j, then $p_{i'} \geq p_i$ for all $i' < i$, $i' \notin F_j$. If $i, i'' \in F_j$, $i < i''$, but $p_i < p_{i''}$, then the price vector $\hat{p} = \langle \ldots, p_{i-1}, p_{i''}, \ldots, p_{i''-1}, p_i, \ldots \rangle$ is also an equilibrium.*

Proof. First, if $p_{i'} < p_i$, then agent j could obtain greater surplus by replacing i with i'. Second, swapping the prices clearly does not affect j's surplus. Moreover, it does not open any opportunities for improvement by other agents, since the cost of obtaining any number of slots by a deadline in the $[i, i'']$ interval can only have increased. ▌

5. AUCTION PROTOCOLS

We use the term *protocol* to refer to a *mechanism*, along with agent *bidding policies*. The mechanisms we consider are generically called *auctions*. McAfee and McMillan provide the following definition [22]:

> An auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants.

This definition includes the well known English open-outcry and first-price sealed bid auctions—commonly used to sell art and to award procurement contracts, respectively—as well as a broad range of other mechanisms, including fixed pricing, Dutch auctions, Vickrey auctions, commodities markets, and the ascending, combinatorial, and Generalized Vickrey auction schemes described in Sections 6 through 8.

In order to place some structure on the space of possibilities, and also to provide a common interface to agents, we define a somewhat restricted, but still very general auction mechanism.

1. Agents send bids to the mechanism to indicate their willingness to exchange goods.
2. The auction may post *price quotes* to provide summarized information about the status of the price-determination process.

Steps 1 and 2 may be iterated.

3. The auction determines an allocation and notifies the agents as to who purchases what from whom at what price.

The above sequence may be performed once or repeated any number of times.

Auctions can be differentiated across many parameters including, but not limited to, those concerning: matching algorithm, price determination algorithm, event timing, bid restrictions, and intermediate price revelation [26, 31, 46]. One of the most important distinctions is whether an individual auction allocates a single resource, or several at once. The latter type, called *combinatorial auctions* (Section 7), accept bids referring to combinations of basic goods.

We have implemented the Michigan Internet AuctionBot[4] [45], a configurable auction server that implements a broad class of mechanisms, defined by a parametric characterization of auction design space. The AuctionBot provides interfaces for human and software agents to create and participate in auctions. Currently the AuctionBot supports the major classical (single-resource) auction types, including the mechanism for the ascending auction protocol described in Section 6.

In order to predict auction outcomes, we must consider the agents' presumed bidding policies, which in turn we might base on some model of their beliefs and preferences. In some auction contexts we are able to determine analytically that a particular bidding policy is part of a Bayesian-Nash equilibrium, or even the dominant strategy. In other settings we rely on experimentation and rules of thumb based on economic principles to determine reasonable bidding policies.[5]

The auction mechanisms we discuss are decentralized in the sense that each agent calculates its own bidding strategy, based on local information. Single-resource auctions, as in the ascending auction protocol, are further distributed in that allocations for each good can be computed separately.

6. ASCENDING AUCTION

We define the ascending auction protocol for the general discrete resource allocation problem. Separate auctions determine prices for each of the goods. Agents submit successively higher bids to the auctions, and auctions immediately report price quotes to all interested agents upon receiving a bid. We allow that agents and auctions operate asynchronously, that is, we impose no bound on the relative times that agents take to compute and send bids, or auctions to compute and send price quotes. Nor do we assume any ordering on bid actions, other than explicitly stated. (As a consequence of this asynchrony and flexible ordering, the protocol is nondeterministic.) When the bidding stops (i.e., the protocol reaches *quiescence* [43]), each auction allocates its respective good to the highest bidder at the price the agent bid, or the good is retained by the seller if there are no bids.

6.1. Bidding Rules

At any point in time, the *bid price* in the auction for good i, denoted β_i, is the highest bid in the auction thus far. If auction i has received no bids, β_i is

[4] `http://auction.eecs.umich.edu`

[5] Our analysis is from the standard noncooperative perspective, which assumes that agents do not directly coordinate their bidding. *Collusion* has been an issue in the FCC spectrum auctions; anti-collusion measures are considered in that context, for example, by Milgrom [24].

undefined. Auction i's *ask price*, denoted α_i, is $\beta_i + \epsilon$, for some fixed ϵ, if β_i is defined. Otherwise, the ask price is q_i.

The ascending auction rejects any bid less than its ask price. Agents are not allowed to withdraw bids. An agent may replace its bid with another, but the new bid must be at least the current ask price. These rules guarantee that prices do not decrease and that the bidding process terminates.

6.2. Agent Bidding Policies

When an agent j enters the market, it bids the ask prices for the set of goods, X, that maximizes its surplus H_j, based on the current ask prices (breaking ties arbitrarily). As other agents continue to bid, agent j may lose some of its bids. When this occurs, j bids the ask price on the set of goods that maximizes its surplus, assuming that it can obtain the goods it is currently winning at their bid prices. For the single-unit scheduling problem, whenever an agent is not already winning a bid, it simply bids the ask price for the single good that maximizes its surplus at the ask prices. If no good would provide it with a positive surplus, then the agent "drops out" of the auction. For the general multi-unit variable-deadline problem, the surplus-maximizing bids can be computed by a straighforward algorithm in $O(n(\log n + K_j))$ time.

This bidding strategy is quite simple, involving no anticipation of other agents' strategies. For the single-unit problem, such anticipation is unnecessary, as the agent would not wish to change its bid even after observing what the other agents did. This is called the *no regret* property [3], and means that from the agent's perspective, no bidding policy would have been a better response to the other agents' bids. The no-regret property does *not* hold, however, for the ascending auction in the multiple-unit scheduling problem, regardless of the bidding strategy [3]. In general, an agent might perform better, for example, through accurate prediction of the other agents' behavior. In the absence of a basis for prediction, however, the simple strategy proposed may indeed be reasonable.

6.3. Analysis of the Ascending Auction

Let p_i denote the price for i. Under the ascending auction rules, when the protocol reaches quiescence, $p_i = \beta_i$ if defined, otherwise $p_i = q_i$.

It is possible that the ascending auction can determine prices that differ from an equilibrium of a multiple-unit scheduling economy by arbitrarily large amounts.

EXAMPLE 6.1. The bid increment is $\epsilon = \$1$ and the reserve prices are zero. The agents are described in Table 2.

Although there are many equilibrium price sets (one of which is $p_1 = \$8$, $p_2 = \$8$, and $p_3 = \$1$), the ascending auction may not find an equilibrium. Agent 2 could bid up good 3 until $\alpha_3 > \$2$ while it and agent 1 both bid up the prices on 1 and 2. The reader can verify that any equilibrium must have agent 3 winning good 3 at a price no greater than \$2.

TABLE 2.

A multiple-unit problem (Example 6.1).

Name	Job Length	Deadline	Value
Agent 1	2	2	\$20
Agent 2	2	3	\$8
Agent 3	1	3	\$2

TABLE 3.

A multiple-unit problem (Example 6.2).

Name	Job Length	Deadline	Value
Agent 1	1	1	\$3
Agent 2	2	2	\$11

In the multiple-unit scheduling problem, the ascending auction can produce allocations that are arbitrarily far from optimal.

EXAMPLE 6.2. There are two agents as shown in Table 3. Reserve prices are $q_1 = \$1$ and $q_2 = \$9$, and the bid increment is $\epsilon = \$1$.

If agent 2 places its bids first, it will bid \$1 for 1 and \$9 for 2. Agent 1 will then bid \$2 for 1. The bidding will stop with good 1 allocated to agent 1 and good 2 allocated to agent 2. This solution has a value of \$3 yet the optimal solution, with 2 unallocated, has a value of \$12. It is easy to see—by increasing q_2 and v_2 by the same amount—that the ascending auction can produce a solution that is arbitrarily far from optimal.

If we restrict each agent's requirement to a single time slice, then by Theorem 4.2 an equilibrium exists. However, the ascending auction protocol is not guaranteed to reach an equilibrium even with this restriction.

EXAMPLE 6.3. The bid increment is $\epsilon = \$1$. The reserve prices are $q_1 = \$4$, $q_2 = \$3$, and $q_3 = \$3$. The agents are described in Table 4.

It is possible that agent 2 may bid first, for 2. Then $\alpha_2 = \$4$. Agent 1 will then bid \$4 for either 1 or 2. If it bids for 1 then the bidding will stop and agent 1 will win 1 for \$4 and agent 2 will win 2 for \$3. But since $p_2 = \$3 < p_1$, agent 1 would maximize its surplus by demanding 2 at the final prices. However, the bidding rules prohibit any readjustment towards an equilibrium. The auction does

TABLE 4.

A single-unit problem (Example 6.3).

Name	Job Length	Deadline	Value
Agent 1	1	2	\$6
Agent 2	1	3	\$7

not allow agent 1 to withdraw its bid for 1, and hence the final allocation violates condition 1 of the definition of equilibrium.

It is not hard to see that the potential failure to reach equilibrium can be demonstrated for any positive value of ϵ, no matter how small. Nevertheless, unlike the multiple-unit problem, we can bound the distance from the equilibrium price vector by $\kappa\epsilon$, where $\kappa = \min(n, m)$.

THEOREM 6.1. *For the variable-deadline, single-unit scheduling problem, the final price of any good determined by ascending auction protocol will differ from the unique minimum equilibrium prices by at most $\kappa\epsilon$.*

Proof. Demange et al. prove this result for the ascending auction protocol in an exchange economy where buyers want no more than a single item from a set of available goods [7]. Such is the case for the single-unit scheduling problem. ▮

Consider again Example 6.3. The solution shown has a value of \$16. If agent 1 had received good 2 and agent 2 had received good 3 then the value of the solution would be \$17, which is optimal. However, the solution can be suboptimal by only a bounded amount.

THEOREM 6.2. *The ascending auction protocol with a given ϵ produces a solution to the variable-deadline, single-unit scheduling problem that is suboptimal by at most $\kappa\epsilon(1 + \kappa)$.*

Proof. Let f be the allocation reached by the ascending auction and f^* an optimal allocation. p_i is the price found for i in the ascending auction, and p_i^* the unique minimum equilibrium price for i (recall that Lemma 4.1 and Theorem 4.2 established that a unique minimum price vector exists and supports f^*). Let $e_i = p_i^* - p_i$. From Theorem 6.1 we know that $\mid e_i \mid \leq \kappa\epsilon$.

Let F and F^* be the set of goods allocated in f and f^*, respectively. The degree of suboptimality is

$$\begin{aligned} v(f^*) - v(f) &= \left(\sum_{i \in F_\perp^*} q_i + \sum_{j=1}^{m} v_j(F_j^*) \right) - \left(\sum_{i \in F_\perp} q_i + \sum_{j=1}^{m} v_j(F_j) \right) \\ &= \sum_{i \in F_\perp^* \setminus F_\perp} q_i - \sum_{i \in F_\perp \setminus F_\perp^*} q_i + \sum_{j=1}^{m} v_j(F_j^*) - \sum_{j=1}^{m} v_j(F_j). \quad (1) \end{aligned}$$

In the single-unit problem, an agent bids for the good that maximizes its surplus. In the solution allocation, this surplus must be at least the surplus it would get from any other good at the ask price, otherwise the agent would have bid for that good instead. In particular, it is at least that it would get from its good in f^* at the ask price. Therefore, when the ascending auction stops, aggregating over agents,

$$\begin{aligned} \sum_{j=1}^{m} v_j(F_j) - \sum_{i \in F} p_i &\geq \sum_{j=1}^{m} v_j(F_j^*) - \sum_{i \in F^*} \alpha_i \\ &\geq \sum_{j=1}^{m} v_j(F_j^*) - \sum_{i \in F^*} (p_i + \epsilon), \end{aligned}$$

since $\alpha_i \leq p_i + \epsilon$. Rearranging, and using the facts that $F \setminus F^* = F_\perp^* \setminus F_\perp$ and $F^* \setminus F = F_\perp \setminus F_\perp^*$, we have

$$\begin{aligned} \sum_{j=1}^{m} v_j(F_j^*) - \sum_{j=1}^{m} v_j(F_j) &\leq \sum_{i \in F^*} (p_i + \epsilon) - \sum_{i \in F} p_i \\ &= \sum_{i \in F^* \setminus F} p_i - \sum_{i \in F \setminus F^*} p_i + \sum_{i \in F^*} \epsilon \\ &= \sum_{i \in F_\perp \setminus F_\perp^*} p_i - \sum_{i \in F_\perp^* \setminus F_\perp} p_i + \sum_{i \in F^*} \epsilon. \quad (2) \end{aligned}$$

Goods unallocated in f must have prices equal to their reserve prices,

$$\sum_{i \in F_\perp \setminus F_\perp^*} p_i = \sum_{i \in F_\perp \setminus F_\perp^*} q_i. \quad (3)$$

Goods unallocated in f^* must have minimum equilibrium prices equal to their reserve prices,

$$\sum_{i \in F_\perp^* \setminus F_\perp} p_i^* \equiv \sum_{i \in F_\perp^* \setminus F_\perp} (p_i + e_i) = \sum_{i \in F_\perp^* \setminus F_\perp} q_i. \quad (4)$$

Substituting (2), (3), and (4) into (1) gives

$$\begin{aligned} v(f^*) - v(f) \;&\le\; \sum_{i \in F_\perp^* \setminus F_\perp} (p_i + e_i) - \sum_{i \in F_\perp \setminus F_\perp^*} p_i \\ &\quad + \left(\sum_{i \in F_\perp \setminus F_\perp^*} p_i - \sum_{i \in F_\perp^* \setminus F_\perp} p_i + \sum_{i \in F^*} \epsilon \right) \\ &= \sum_{i \in F_\perp^* \setminus F_\perp} e_i + \sum_{i \in F^*} \epsilon. \end{aligned}$$

The total error is maximized when $e_i = \kappa\epsilon$ for all $i \in F_\perp^* \setminus F_\perp$. Since there can be at most κ goods in $F_\perp^* \setminus F_\perp$ and F^*, this yields an upper bound on the total error: $\kappa\epsilon(1 + \kappa)$. ▮

Computing the clearing and price quotes is trivial in the ascending auction. Communication costs dominate the run time, which can therefore be measured in terms of the bids required. Because bids increase by a fixed increment, the number of iterations is inversely proportional to ϵ. Hence, in choosing the value for ϵ, we trade off solution value for communication efficiency.

We have shown that the simple bidding policy is reasonable for individual agents, and produces allocations with desirable system properties in the single-unit problem. The results do not provide strong support for this simple policy in the multiple-unit problem. Other strategies, such as jump bidding—where an agent bids in large increments for sets of goods to signal its willingness to aggressively pursue that set—may provide potential advantages to individuals or the system. However, it is an open question as to whether there exists a policy for the ascending auction (or any complete protocol) that always finds (within some tolerance) an equilibrium when it exists.

6.4. Incremental Auction Closing

In the basic version of the ascending auction mechanism, we close the auctions simultaneously, once the bidding process reaches quiescence. In a variant, we close one or a few at a time, reopening the bidding process after each close. Once an auction closes, the commitment of the winning bidder to buy the good is finalized, and the price paid constitutes a sunk cost. This may cause the bidder to reassess its decisions about other goods, and bid in auctions it had previously dropped out of.

EXAMPLE 6.4. Reconsider Example 6.2, with agents described by Table 3, and $q_1 = \$1$ and $q_2 = \$9$. As pointed out above, the ascending auction may reach a solution with good 1 allocated to agent 1 for \$2 and good 2 allocated to agent 2 for \$9. This solution has value \$3.

If we close the auction for good 2, then agent 2 treats its payment as sunk, and so now values good 1 at \$11. Therefore, with bidding on good 1 reopened, it will clearly outbid agent 1. If we instead close good 1 and reopen bidding on good 2, nothing changes.

In this example, the resulting allocation has value \$11—still suboptimal, but an improvement over the original allocation. Indeed, it can be shown that incrementally reopening bidding on the last good can only improve solution value. With one good left, the agents value the good according to its marginal contribution to overall value, and so the situation is as for a single-good English auction.

It is also easy to see that incremental auction closing can have no effect in the following cases:

1. For single-unit problems, sunk costs are irrelevant, and so no agents bid in reopened auctions.
2. If the allocation represents a price equilibrium, no agent will change bids after auctions are closed.

Thus, in situations where the ascending protocol is known to work well, we do not expect that incremental closing will degrade quality. In general, however, reopening bidding after some auctions close can have positive or negative effects.

EXAMPLE 6.5. Consider two agents as described by Table 5. Let reserve prices be $q_1 = \$8$ and $q_2 = q_3 = \$2$, and the bid increment be \$1. After the initial bidding process, the ascending auction may reach a result where agent 1 obtains slot 3, and agent 2 obtains slot 2, at prices of \$2 each. At this point agent 2 drops out, as the minimum cost to complete its job would exceed its value. The value of this solution is \$13.

If we close auction 2, agent 2 treats this cost as sunk, and now compares the incremental cost favorably to its job's valuation. It then may enter bids on goods 1 and 3 at \$8 and \$3, respectively. However, agent 1 will rebid, offering \$4 for good 3. The result at this point is a solution with value \$5. This is the final result if we close auction 3. If instead we close auction 1, then agent 2 will again reopen bidding in light of the sunk cost, this time ultimately winning, for a solution value of \$12.

It is clear from the examples above that changes in solution quality depend critically on the order that auctions close. Unfortunately, in general we cannot tell which order will be advantageous, without knowing the agents' private information. Consider Example 4.1, with $\epsilon = 0.5$. It might well result with identical prices for both goods, at prices 1.5, with one agent winning each. Closing the auction in which agent 1 is winning would lead to an improvement after sunk costs are discounted, whereas the other would not. However, there is no way to tell which this is, based solely on the quiescent state.

TABLE 5.

Reopening bidding after one auction closes can degrade solution quality (Example 6.5).

Name	Job Length	Deadline	Value
Agent 1	1	3	\$5
Agent 2	3	3	\$12

7. COMBINATORIAL MARKETS

The ascending auction performs well for single-unit allocation problems. At the end of Section 3 we note that the single-unit restriction is only one sufficient condition for existence of a price equilibrium. However, even when equilibria exist for a multiple-unit problem, the ascending auction may not find one, as shown by Example 6.2. Further, as Example 4.1 demonstrates, many scheduling problems cannot support allocations with any price equilibrium.

In light of these limitations, several have proposed *combinatorial auction* mechanisms, where agents submit bids for combinations of goods [2, 4, 18, 30, 32, 44]. Such mechanisms operate in a variety of ways, typically calculating allocations and prices as a function of bids for all the combinations. Prices may refer to individual goods, or to entire bundles.

One of the drawbacks of combinatorial auctions is their potential computational complexity. With n goods, there are 2^n combinations, which can entail complex calculations for both the agents and the mechanism. Typical formulations of the mechanism's computational problem are NP-Complete, but may admit heuristic search procedures effective in practice in certain environments [9, 33]. As Rothkopf et al. [32] point out, moreover, restricting the set of allowed combinations can preserve computational tractability.

We pursue a strategy similar in spirit to the restriction approach, presenting a reformulation of the problem that extends the price system in a controlled way, without admitting an exponential number of markets. The reformulation expands the class of problems solved by price equilibria, and suggests corresponding auction protocols for determining these prices.

7.1. Problem Formulation

As in the original formulation, we posit

- G, a set of n discrete *basic goods*, and
- A, a set of m agents, and $\perp$ representing the seller or null agent.

Rather than impose a price system over the basic goods, however, in the revised combinatorial formulation we introduce an expanded set of *market goods*, G'. A market good is a pair, (y, z), denoting a bundle of y time slots no later than time z.

More specifically, this bundle contains the time slot z (i.e., the basic good indexed z), and an indeterminate set of $y-1$ slots strictly before z.

The configuration G' consists of all (y, z) pairs such that $1 \leq y \leq z \leq n$, and $y \leq l$. The price system for this formulation assigns prices to all $\sum_{i=1}^{l}(n-i+1) = l(n - \frac{l}{2} + \frac{1}{2}) = O(ln)$ market goods in G'. We denote the price of (y, z) by $p(y, z)$. We generally require that prices be *monotone* in the number of time slots,

$$\text{for all } y \leq y' \leq z,\ p(y', z) \geq p(y, z). \tag{5}$$

A solution is defined as in the original formulation, a mapping from basic goods to agents. A *market allocation*, $\phi : G' \to A \cup \{\perp\}$, is an assignment of market goods to agents. Let $\Phi_j \equiv \{(y, z) | \phi((y, z)) = j\}$ denote the set of market goods allocated to agent j. We say that a market allocation ϕ *is consistent with* a solution f if f gives each agent what it is promised by ϕ. That is, for all $j \in A$, $k \leq n$,

$$|G_k \cap F_j| = \sum_{i \leq k} \begin{cases} y & \text{if } (y, i) \in \Phi_j \\ 0 & \text{otherwise.} \end{cases}$$

where $G_k \equiv \{1, \ldots, k\}$ is the set of basic goods with index less than or equal to k.

Note that although a scheduling agent j obtaining a market good (y, z) cannot be sure exactly *which* time slots it will receive, its utility is completely determined by whether it obtains enough time slots to finish its job, and if so, by what deadline. Specifically, the value j achieves by using market good (y, z) is

$$v'(y, z) \equiv \begin{cases} v_j^{k(z)}, \text{ where } k(z) = \min\{k | d_j^k \geq z\} & \text{if } y \geq \lambda_j \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Let $Y(\Phi, d)$ denote the number of slots guaranteed by deadline d by a set of market goods Φ, and $Y(\Phi) \equiv Y(\Phi, n) = \sum_{(y,z) \in \Phi} y$ the number of slots guaranteed overall. The maximum surplus that j can *ensure* by purchasing market goods at prices p is given by[6]

$$H_j'(p) \equiv \max_{\Phi} \left[\max_{d} v'(Y(\Phi, d), d) - \sum_{(y,z) \in \Phi} p(y, z) \right]. \tag{7}$$

DEFINITION 7.1. [temporally consistent prices] A monotone price function p is *temporally consistent*[7] if

[6]Note that agents have a chance of doing better—or worse—by purchasing goods with $y > \lambda_j$ and higher z values.

[7]Cf. the related notion of temporal consistency we observe in equilibria of the original formulation, by Theorem 4.3.

1. For all $y \leq z \leq z'$, $p(y, z) \geq p(y, z')$.
2. For all $y' \leq z' \leq z, y = y' + y'' \leq z$, $p(y, z) \leq p(y', z') + p(y'', z)$.

LEMMA 7.1. *If p is a temporally consistent price function, the maximum surplus can be achieved with an allocation containing at most one market good,*

$$H'_j(p) = \max\left(0, \max_z [v'(\lambda_j, z) - p(\lambda_j, z)]\right).$$

Proof. Let d and Φ be the deadline and set of market goods, respectively, maximizing the surplus term in (7). If $d > 0$, either $Y(\Phi, d) \geq \lambda_j$ or all applicable prices are zero. In the latter case, or when $d = 0$, the lemma is satisfied trivially. Thus, let us suppose $d \geq Y(\Phi, d) \geq \lambda_j$. The value of the goods received is then $v'(\lambda_j, d)$. With temporally consistent prices, any Φ ensuring this value must cost at least $p(\lambda_j, d)$. ∎

7.2. Equilibrium and Efficiency

DEFINITION 7.2. [combinatorial price equilibrium] A market allocation ϕ is in *equilibrium* at prices p iff

1. For all agents j, $\max_d v'(Y(\Phi_j, d), d) - \sum_{(y,z)\in\Phi_j} p(y, z) = H'_j(p)$.
2. For all (y, z), $p(y, z) \geq \min_{\{B\subseteq G_z:|B|=y\}} \sum_{i\in B} q_i$.
3. There exists a solution f consistent with ϕ such that
   (i) for all j, $\sum_{(y,z)\in\Phi_j} p(y, z) \geq \sum_{i\in F_j} q_i$,
   (ii) for all (y, z) such that $|G_z \cap F_\perp| \geq y$, $p(y, z) \leq \min_{\{B\subseteq G_z\cap F_\perp:|B|=y\}} \sum_{i\in B} q_i$.

We call any solution serving the role of f in the definition above an *implementing solution* for ϕ.

The first, central requirement for equilibrium is that agents maximize surplus at the given prices. Here we dictate that the allocations the agents get maximize their surplus given the market-good prices.

The conditions relating market prices to reserve prices are complicated by the indeterminate relationship between market and basic goods. We require that the price of a market good be at least the minimum consistent reserve price, else the sellers would not part with the constituent basic goods. For market goods actually allocated, we require the price to exceed that of basic goods comprising it in a consistent solution. And when a market good could be satisfied by basic goods unallocated in this solution, the reserves of those goods define an upper bound on its price.

TABLE 6.

A problem with both optimal and suboptimal combinatorial equilibria.

Name	Job Length	Deadline	Value
Agent 1	2	2	\$3
Agent 2	1	2	\$2
Agent 3	1	2	\$2

EXAMPLE 7.1. Reconsider Example 4.1, with parameters illustrated by Table 1, and zero reserve prices. Although no price equilibrium exists for the original formulation, we can support the optimal solution with a combinatorial price equilibrium. Let $l = 2$, and consider prices $p(1,1) = p(1,2) = 2.1$, and $p(2,2) = 2.9$. The allocation $\Phi_1 = \{(2,2)\}$, $\Phi_2 = \emptyset$ can be implemented by the solution giving agent 1 both basic goods, which satisfies the combinatorial equilibrium conditions at these prices. Note that no combinatorial equilibrium can support any other allocation.

Unlike in the basic configuration, however, combinatorial price equilibria are not necessarily efficient.

EXAMPLE 7.2. Consider an extension of the previous example, described by Table 6 (with zero reserve prices). The problem has a basic equilibrium, with $p_1 = p_2 = 1.6$, and agents 2 and 3 each getting one of the slots. This optimal solution is also supported by the combinatorial equilibrium prices $p(1,1) = p(1,2) = 1.6$, and $p(2,2) = 3.2$. However, the nonoptimal solution where agent 1 gets both slots is also in equilibrium, at prices $p(1,1) = p(1,2) = 2.1$, and $p(2,2) = 2.9$.

Moreover, the degree of suboptimality is not usefully bounded—even without reserve prices. We can extend Example 7.2 to obtain n-agent problems where equilibrium solutions are a factor of $n - 1$ worse than optimal. On the positive side, optimal solutions supported by price equilibria in the original formulation are retained (albeit not uniquely) in the combinatorial formulation.

THEOREM 7.1. *If in the original formulation, f is in equilibrium at prices p, then in a combinatorial formulation with $l = 1$, the allocation $\phi((1,z)) = f(z)$ is in equilibrium at prices $p(1,z) = p_z$.*

Proof. Let f be the implementing solution for ϕ. For the case of $l = 1$, the surplus maximization criterion and conditions comparing prices to reserve prices are identical to those in the original formulation (Definition 4.1).[8] ∎

LEMMA 7.2. *If ϕ is in equilibrium at temporally consistent prices p, then the market allocation $\hat{\phi}$ defined by*

$$\hat{\Phi}_j = \begin{cases} \emptyset & \text{if } Y(\Phi_j) < \lambda_j \\ \{(\lambda_j, \min\{d : Y(\Phi_j, d) \geq \lambda_j\})\} & \text{otherwise} \end{cases}$$

is also in equilibrium at these prices. Moreover, if f is an implementing solution for Φ, then the solution $\hat{f}$ defined by

$$\hat{F}_j = \begin{cases} \emptyset & \text{if } |F_j| < \lambda_j \\ F_j & \text{if } |F_j| = \lambda_j \\ \arg\max_{F \subseteq F_j : |F| = \lambda_j} \sum_{i \in F} q_i & \text{otherwise} \end{cases}$$

has the same value as f.

Proof. By Lemma 7.1, each agent can maximize its surplus with a single market good of the form specified for $\hat{\Phi}$, with surplus no less than that obtained from Φ. Since Φ is in equilibrium, the surplus must be *exactly* the same. The implementing solution for $\hat{\Phi}$ is $\hat{f}$, obtained from f by deleting the minimum-reserve-price extraneous goods (if any) from each agent's allocation. By construction, if these goods really are extraneous, they must have zero reserve prices, and by temporal price consistency the third condition for equilibrium (Definition 7.2) must hold for $\hat{\phi}$ and $\hat{f}$. By the same token, deallocating goods with zero reserve prices has no effect on solution value. ∎

DEFINITION 7.3. [monotone reserve prices] A scheduling problem exhibits *monotone reserve prices* iff $q_i \geq q_{i'}$ for all $i \leq i'$.

LEMMA 7.3. *If ϕ is in equilibrium at monotone prices p for a scheduling problem with monotone reserve prices, then ϕ is also in equilibrium at temporally consistent prices $\hat{p}$, defined by*

$$\begin{aligned} \hat{p}(1,1) &= p(1,1) \\ \hat{p}(1,z) &= \min(p(1,z), \hat{p}(1,z-1)),\ 2 \leq z \leq n \\ \hat{p}(y,z) &= \min(p(y,z), \hat{p}(y-1,z-1) + \hat{p}(1,z)),\ 2 \leq y \leq z \leq n. \end{aligned} \tag{8}$$

[8] We can extend this result to allow $l > 1$, by setting prices for combinations to the maximum allowable by temporal consistency.

Proof. The transform described lowers prices only when an alternative way of achieving the same task value exists, hence it provides agents no opportunity to improve their surplus. By monotone reserve prices and the anchoring $\hat{p}(1,1) = p(1,1)$, the reduction in single-unit prices (8) does not violate the restriction that goods be priced above their minimum reserve. ∎

By Lemmas 7.2 and 7.3, and given monotone reserve prices, we can restrict attention to allocations of at most one market good per agent, at temporally consistent prices.

THEOREM 7.2. *If G' is a market configuration for a scheduling problem with monotone reserve prices, and $l \geq \max_{j \in A} \lambda_j$, then there exists an equilibrium allocation that is optimal for G'.*

Proof. Let f be an optimal allocation with all unallocated slots as early as possible, and no extraneous slots allocated. That is, if $i \notin F_\perp$, then the solution obtained by removing i from the allocation is not optimal, nor is the solution obtained by swapping i and i', for any $i' \in F_\perp$, $i < i'$. Define the market allocation $\Phi_j = \{(|F_j|, \max_{i \in F_j} i)\}$ if $F_j \neq \emptyset$, $\Phi_j = \emptyset$ otherwise.

Construct a version of the *assignment problem* [16, 19] as follows. The set of source entities to be assigned, S, consists of agents, the seller, and a dummy seller for each unallocated basic good:

$$S = A \cup \{\perp\} \cup \{\sigma_i | i \in F_\perp\}.$$

These entities are to be matched with a target set, T, consisting of goods assigned in the optimal allocation (including nulls), one for the seller agent, and unallocated slots:

$$T = \{\Phi_j | j \in A\} \cup \{\perp\} \cup F_\perp.$$

Agents in S have preferences for elements of T based on their value functions, with value for market goods given by (6), for single (unallocated) slots as if a single-unit market good, and zero value for null goods. Sellers in S value elements of T based on reserve prices, according q_i to (unallocated) slot i, $\sum_{i \in F_j} q_i$ for j's market good in the optimal allocation, and zero to null goods.

The assigment that maps agent j to Φ_j, $\perp$ to $\perp$, and σ_i to slot i is clearly an optimal solution to this problem; any superior assignment would correspond immediately to an improved allocation to the original problem. As Koopmans and Beckmann [16] (and many subsequently) have shown, it follows that this assignment can be supported by a set of prices such that each source entity gets at least as much surplus from the target element it is assigned as from any other. Leonard [19] further shows how to construct the lowest prices supporting this allocation.

To obtain our combinatorial price equilibrium, we start with these lowest prices supporting the optimal assignment. Note that these prices must satisfy temporal consistency (Definition 7.1), since any violation would provide a way for a source entity to improve its assignment. We can then assign prices to the remaining (not allocated) market goods to be as great as possible, while preserving temporal consistency. If $p(1,1)$ has not been defined, we set its price at some arbitrarily large value exceeding all agent valuations.[9] Starting from $(1,2)$, we price the as-yet-unpriced market good $(1,z)$ at $p(1,z-1)$. Continuing from $(2,1)$, for unpriced (y,z) we set $p(y,z) = \min(p(y,z-1), \min_{y=y'+y''} p(y',z) + p(y'',z-1))$. In addition to ensuring temporal consistency, this pricing procedure ensures that no unallocated market good can be obtained more cheaply than it could have been through purchasing target elements of the assignment problem. Since these elements are priced in an assignment equilibrium, we know that the agents would not prefer to choose an alternative. Thus, the first condition for combinatorial price equilibrium (Definition 7.2) is established.

The conditions on prices compared to reserves are ensured by including the seller agents in the assignment problem. Any market good assigned to an agent must have been priced higher than the sum of its reserve prices (condition 3(a)). And since the prices supporting the assignment of unallocated slots to the dummy sellers are the lowest possible, with the seller $\perp$ assigned zero surplus, the prices of unallocated slots must equal their reserves. Therefore, by temporal consistency, no market good achievable through unallocated slots can be priced above the sum of reserves (condition 3(b)). Finally, the second condition is also ensured by the method of pricing unallocated slots and market goods, and of maximizing prices subject to temporal consistency. ▌

7.3. Combinatorial Auction Protocols

In future work, we intend to define and analyze combinatorial protocols analogous to the ascending auction. A straightforward implementation of this protocol is not well-defined for the combinatorial case, as basic goods may be assigned to various market goods. Accordingly, we must define allocation and price quote policies as a function of combinations of bids for alternative goods, not just individual market goods. Such protocols have been proposed for the case of general combinatorial bidding by Parkes [29] and Wurman [44]. We also plan to investigate adaptations to this scheduling problem of other mechanisms proposed for combinatorial settings [2, 4, 8, 30].

Since combinatorial auctions can support suboptimal equilibrium solutions, it can be disadvantageous to open combinatorial markets when equilibria exist in basic goods. A natural approach would be to start with markets in basic goods, and open combinatorial auctions only if the protocol does not reach equilibrium.

[9] We could also cap this and other values to ensure that no good is priced at more than a good necessarily comprising a superset of its time slots, without changing the argument.

We can apply this incrementally, progressively increasing l until an equilibrium is reached. Of course, this presumes we have a way to detect equilibrium states, or at least some indication of whether opening additional combination markets will be beneficial.

8. GENERALIZED VICKREY AUCTION

The preceding analysis characterizes the performance of multiple ascending single-good auctions for the scheduling problem, and the prospects for combinatorial auctions. Neither is guaranteed to produce optimal solutions to all scheduling problems. Another mechanism, the Generalized Vickrey Auction (GVA) [36], does find efficient schedules for all of our problems. Although the main results are not new, we briefly present the GVA because it takes an important place in our spectrum of mechanisms for scheduling. We also provide a new result—stronger properties for the GVA in a particular class of scheduling problems—and an observation on the computational complexity of decentralization.

The GVA is a direct revelation mechanism (DRM), and thus is not a price system. Rather, it computes overall payments for agents' allocations that sometimes, but not always, translate into meaningful prices for individual goods. If agents play Bayesian-Nash or dominant strategies, any desirable choice function that can be implemented by a mechanism can be implemented by a DRM, so the appeal of this type of mechanism is quite powerful.[10] The GVA can implement optimal solutions for multi-unit scheduling problems, as well as extended versions of the problem involving, for example, multiple jobs or externalities (i.e., values for one agent that depend on the allocations obtained by other agents).

8.1. Bidding Rules

Recall that v_j is agent j's actual utility function. Each agent announces $\hat{v}_j$, its alleged utility function. The circumflexes are used to indicate that the agent is not constrained to be truthful, that is, it may be that $\hat{v}_j \neq v_j$. The auction knows the reserve values, q_i. After receiving the bids, the GVA returns an allocation, and a vector of positive or negative payments to be made to the agents.

8.2. Allocation Rules and Optimality

Recall that a solution is a mapping f, and the value of a solution is given by $v(f)$. The auction mechanism:

1. Computes a solution,

[10] Specifically, the GVA is a DRM relying on dominant strategies in the class of Groves [11] and Clarke [5] mechanisms. Green and Laffont have shown under rather general conditions that when agents have quasi-linear preferences, the only efficient social choice functions that are implementable in dominant strategies are those that are implementable by Groves-Clarke mechanisms [10].

$$f^* = \arg\max_f \sum_{i \in F_\perp} q_i + \sum_{j=1}^{m} \hat{v}_j(F_j). \tag{9}$$

2. Computes payments to agents,

$$V_j \equiv W_{-j}(f^*) - P_j(\hat{v}_{-j}), \tag{10}$$

where

$$\begin{aligned} W_{-j}(f^*) &= \sum_{i \in F_\perp^*} q_i + \sum_{s \neq j} \hat{v}_s(F_s^*), \\ P_j(\hat{v}_{-j}) &= \max_f \sum_{i \in F_\perp} q_i + \sum_{s \neq j} \hat{v}_s(F_s). \end{aligned} \tag{11}$$

The W_{-j} component represents the total reported value for agents other than j at the solution f^*. The residual payment P_j could be any function of other agents' reported valuations. However, we restrict attention here to the formula (11).

Given this allocation rule, truthful bidding of the utility function, $\hat{v}_j = v_j$, is a dominant strategy [36]. The GVA computes the optimal allocation based on the bids, and since all bids are truthful, the allocation is globally optimal.

EXAMPLE 8.1. Consider the setup of Example 7.2 (Table 6). If the agents truthfully report their value functions, the auction mechanism finds an optimal solution: $f^*(1) = 2$, $f^*(2) = 3$. It then calculates $W_{-1} = 4$, $W_{-2} = 2$, and $W_{-3} = 2$. Agent 1 receives total value $0 + [4 - P_1]$, agent 2 receives $2 + [2 - P_2]$, and agent 3 receives $2 + [2 - P_3]$. Agents are willing to participate (that is, $v_j(F_j) + V_j \geq 0$) as long as $P_j \leq 4$ for $j \in \{1, 2, 3\}$. Using (11), $P_1 = 4$ (agent 1 pays \$0), $P_2 = P_3 = 3$ (agents 2 and 3 pay \$1), and the mechanism has a net revenue of \$2.

Recall that on this same example, the single-good ascending auction protocol does not guarantee convergence, and the combinatorial scheme admits inefficient equilibria.

8.3. Limitations on the GVA

A mechanism is *individually rational* if no agent can be worse off from participating in the auction than had it declined to participate.[11] A mechanism is

[11] Individual rationality can be defined in three different ways, depending on how much information has been revealed to the agent before it must commit to its participation decision. We limit our

budget balanced if the net payment over all agents is nonnegative. Generally, these, along with optimality, are the properties we desire when agents play their equilibrium strategies in a mechanism. However, Groves-Clarke mechanisms are not always guaranteed to be budget-balanced: they may require an outside injection of resources (subsidy). Further, even when a social choice function can be implemented with a budget-balanced Groves-Clarke mechanism, we cannot guarantee that rational agents will agree ex post to participate in the allocation.

For our scheduling problem, we can show that it *is* possible to obtain all three desirable properties if the mechanism designer knows the reserve values, q_i, for each time slot. The payment function $V_j(p_j)$ (10) transfers to agent j the net value increment to all other agents that results from j's participation in the auction. Agent j's only effect on others is that it may get time slices that others desire, so its participation always makes other agents weakly worse off. Thus, V_j is nonpositive for all j, and the auction mechanism runs a surplus. In this situation the GVA is essentially an ideal mechanism for the scheduling problem, if the computational cost is not too high (see Section 8.4).

THEOREM 8.1. *If the GVA uses the payment function $W_{-j} - P_j$ then the individual rationality constraint is satisfied and the net monetary payments to the auction mechanism are nonnegative.*

If instead the q_i are the private information of seller agents, then the mechanism needs to elicit this information to satisfy all of the desirable properties. Myerson and Satterthwaite [27] proved that no mechanism can obtain more than two out of the three desired properties for bilateral exchange problems.

EXAMPLE 8.2. [Bilateral exchange] Suppose there is one buyer, who has a single-unit job with deadline 1 and value v. Let the seller be an agent, with reserve value q_1. Suppose $v > q_1$. The GVA would induce truthful reporting of v and q_1, give the good to the buyer, require the buyer to pay q_1, and pay v to the seller. Although the mechanism is individually rational and would produce the optimal allocation, the auction would run a deficit of $v - q_1$.

In any case, it is easy to show that for all scheduling problems in our class, it is possible to achieve any two out of three of the desirable properties.

8.4. GVA Computation

For a general problem, the heart of the GVA allocation mechanism requires the auction to solve a possibly complex (e.g., nonlinear, nonconvex, integer-constrained) optimization problem multiple times. As a baseline for computational efficiency, we note that Neapolitan and Naimipour [28] show that a simple central-

discussion to the strongest form, *ex post* rationality, which implies voluntary participation even after all agents know the proposed allocation.

ized greedy algorithm solves the single-unit, fixed-deadline scheduling problem optimally, in time $\Theta(m \lg m)$. The GVA mechanism must solve multiple optimization problems to process the bids: one to determine the optimal allocation, and one for each agent j with its bid removed to determine P_j. For a single-unit, fixed-deadline problem we can use the centralized algorithm for each optimization, with a total runtime of $\Theta(m^2 \lg m)$. Thus, inducing preference revelation (and thereby obtaining full optimality) via straightforward implementation of the GVA raises computational cost by a factor of m.

If we remove the single-unit restriction, then any centralized algorithm that can solve the scheduling problem optimally can solve the Integer Knapsack problem. Hence the multiple-unit scheduling problem is NP-Complete.[12] By the preceding argument, distributing the multiple-unit problem via the GVA contributes a factor of m to the computation.

9. DISCUSSION

We have presented two auction mechanisms—ascending single-good markets and the GVA—that can compute optimal or near-optimal solutions to the single-unit distributed scheduling problem in a computationally efficient manner. The multiple-unit problem is significantly more difficult and entails a sharper tradeoff among solution quality, computational efficiency, and the degree to which the mechanism is decentralized. The computation performed by the ascending auction is trivial, and can be distributed by goods. However, we cannot guarantee the quality of solutions produced by this mechanism for the multiple-unit problem. Combinatorial auctions support equilibria in cases where single-good markets do not, but may also admit suboptimal solutions. It remains to be seen whether we can design mechanisms for combinatorial auctions that produce desirable outcomes for plausible agent behavior. The GVA always finds the optimal solution and implements it in dominant strategies, but must in general solve multiple combinatorial problems, and may require a subsidy when seller reserves are not known.

The three categories of mechanisms investigated here can be viewed on a spectrum,

$$\text{single-good} \leftrightarrow \text{combinatorial} \leftrightarrow \text{direct revelation}$$

where the mechanism's *scope of concern* increases as we move to the right. Toward the left, the overall mechanism decomposes into sub-mechanisms, where each sub-mechanism has a limited scope (i.e., subsets of the resources, ultimately singletons). For large-scale systems, we suspect that this decomposition is essen-

[12] Thus, solving it optimally is strongly believed to take time more than polynomial in the size of its description. However, the problem is pseudo-polynomial since dynamic programming solves it in time polynomial in the sum of all agent values (which, however, is exponential in the encoding of these values).

tial, as no single designer will even be aware of all of the resources of interest to some of the agents. Even when we imagine that all concerns are covered (as for direct mechanisms), the very use of monetary payments suggests that there exist some other concerns not included (else what use is money?), assumed separable. Thus, we suspect that mechanisms operating at all points of the spectrum will play a role in computational markets for complex allocation problems.

We view this work as a first important step in developing a broad framework for using markets to solve distributed scheduling problems. In order to move forward we must identify broader classes of scheduling problems and design associated mechanisms for which we can effectively predict agent behavior and analyze resultant protocols. We do not expect to find a single mechanism that reaches an optimal equilibrium in all situations where such equilibria exist. Instead we aim to produce a suite of mechanisms that collectively cover a broad range of problems. Ideally, we would like to be able to choose a mechanism for a given problem and know that it will reach an optimal solution when one would be supportable, or else perform acceptably in some other respect when this is not possible.

ACKNOWLEDGMENT

Thanks to Wolfram Conen, Fredrik Ygge, David Pennock, Terence Kelly, and anonymous reviewers for constructive comments. This work was supported by DARPA grant F30602-97-1-0228 from the Information Survivability program.

REFERENCES

1. Albert D. Baker. Metaphor or reality: A case study where agents bid with actual costs to schedule a factory. In Clearwater [6].

2. Jeffrey S. Banks, John O. Ledyard, and David P. Porter. Allocating uncertain and unresponsive resources: An experimental approach. *RAND Journal of Economics*, 20:1–25, 1989.

3. Sushil Bikhchandani and John W. Mamer. Competitive equilibrium in an exchange economy with indivisibilities. *Journal of Economic Theory*, 74:385–413, 1997.

4. Paul J. Brewer and Charles R. Plott. A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of right to use railroad tracks. *International Journal of Industrial Organization*, 14:857–886, 1996.

5. E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

6. Scott Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1995.

7. Gabrielle Demange, David Gale, and Marilda Sotomayor. Multi-item auctions. *Journal of Political Economy*, 94:863–872, 1986.

8. Christine DeMartini, Anthony M. Kwasnica, John O. Ledyard, and David Porter. A new and improved design for multi-object iterative auctions. Technical report, California Institute of Technology, November 1998.

9. Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions. In *Sixteenth International Joint Conference on Artificial Intelligence*, pages 548–553, Stockholm, 1999.

10. J. R. Green and J.-J. Laffont. *Incentives in Public Decision Making*. North-Holland, 1979.
11. Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.
12. Faruk Gul and Ennio Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87:95–124, 1999.
13. Patrick T. Harker and Lyle H. Ungar. A market-based approach to workflow automation. Presented at the NSF Workshop on Workflow and Process Automation in Information Systems (Athens, GA, May 8-10). Available at `http://www.cis.upenn.edu/~ungar/NSF.html`, 1996.
14. J. S. Jordan. The competitive allocation process is informationally efficient uniquely. *Journal of Economic Theory*, 28:1–18, 1982.
15. Alexander S. Kelso and Vincent P. Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica*, 50:1483–1504, 1982.
16. Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.
17. James F. Kurose and Rahul Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38:705–717, 1989.
18. Erhan Kutanoglu and S. David Wu. On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. *IIE Transactions*, 31:813–826, 1999.
19. Herman B. Leonard. Elicitation of honest preferences for the assignment of individuals to positions. *Journal of Political Economy*, 91:461–479, 1983.
20. T. W. Malone, R. E. Fikes, K. R. Grant, and M. T. Howard. Enterprise: A market-like task scheduler for distributed computing environments. In B. A. Huberman, editor, *The Ecology of Computation*, pages 177–205. North-Holland, 1988.
21. Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, New York, 1995.
22. R. Preston McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.
23. R. Preston McAfee and John McMillan. Analyzing the airwaves auction. *Journal of Economic Perspectives*, 10(1):159–175, 1996.
24. Paul Milgrom. Putting auction theory to work: The simultaneous ascending auction. *Journal of Political Economy*, 108, 2000.
25. Tracy Mullen and Michael P. Wellman. A simple computational market for network information services. In *First International Conference on Multiagent Systems*, pages 283–289, San Francisco, CA, 1995.
26. Tracy Mullen and Michael P. Wellman. Market-based negotiation for digital library services. In *Second USENIX Workshop on Electronic Commerce*, pages 259–269, Oakland, CA, 1996.
27. Roger B. Myerson and Mark A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29:265–281, 1983.
28. Richard E. Neapolitan and Kumarss Naimipour. *Foundations of Algorithms*. D. C. Heath and Company, Lexington, MA, 1996.
29. David C. Parkes. *i*bundle: An efficient ascending price bundle auction. In *ACM Conference on Electronic Commerce*, pages 148–157, Denver, 1999.
30. S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.
31. Juan A. Rodríguez-Aguilar, Francisco J. Martín, Pablo Noriega, Pere Garcia, and Carles Sierra. Competitive scenarios for heterogeneous trading agents. In *Second International Conference on Autonomous Agents*, pages 293–300, Minneapolis, 1998.

32. Michael H. Rothkopf, Aleksander Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44:1131–1147, 1998.

33. Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Sixteenth International Joint Conference on Artificial Intelligence*, pages 542–547, Stockholm, 1999.

34. Lloyd S. Shapley and Martin Shubik. The assignment game I: The core. *International Journal of Game Theory*, 1:111–130, 1972.

35. Michael Stonebraker, Paul M. Aoki, Witold Litwin, Avi Pfeffer, Adam Sah, Jeff Sidell, Carl Staelin, and Andrew Yu. Mariposa: A wide-area distributed database system. *VLDB Journal*, 5:48–63, 1996.

36. Hal R. Varian and Jeffrey K. MacKie-Mason. Generalized Vickrey auctions. Technical report, Department of Economics, University of Michigan, June 1994.

37. Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18:103–117, 1992.

38. Carl A. Waldspurger and William E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the First Symposium on Operating System Design and Implementation (OSDI)*, pages 1–11, 1994.

39. William E. Walsh and Michael P. Wellman. Efficiency and equilibrium in task allocation economies with hierarchical dependencies. In *Sixteenth International Joint Conference on Artificial Intelligence*, pages 520–526, Stockholm, 1999.

40. Michael P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

41. Michael P. Wellman. A computational market model for distributed configuration design. *AI EDAM*, 9:125–133, 1995.

42. Michael P. Wellman. Market-oriented programming: Some early lessons. In Clearwater [6].

43. Michael P. Wellman and Peter R. Wurman. A trading agent competition for the research community. In *IJCAI-99 Workshop on Agent-Mediated Electronic Trading*, Stockholm, August 1999.

44. Peter R. Wurman. *Market Structure and Multidimensional Auction Design for Computational Economies*. PhD thesis, University of Michigan, 1999.

45. Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second International Conference on Autonomous Agents*, pages 301–308, Minneapolis, 1998.

46. Peter R. Wurman, Michael P. Wellman, and William E. Walsh. A parametrization of the auction design space. *Games and Economic Behavior*, this volume, 2000.

47. Fredrik Ygge and Hans Akkermans. Decentralized markets versus central control: A comparative study. *Journal of Artificial Intelligence Research*, 11:301–333, 1999.