iBundle: An Efficient Ascending Price Bundle Auction

David C. Parkes
Computer and Information Science Department
University of Pennsylvania
200 South 33rd Street, Philadelphia, PA 19104
dparkes@unagi.cis.upenn.edu

ABSTRACT

Standard auction mechanisms often break down in important e-commerce applications, where agents demand bundles of complementary resources, *i.e.* "I only want B if I also get A". This paper describes iBundle, an ascending-price auction that is guaranteed to compute optimal bundle allocations with agents that follow a best-response bidding strategy. The auction prices bundles directly and allows agents to place additive or exclusive-or bids over collections of bundles. Empirical results confirm that iBundle generates efficient allocations for hard resource allocation problems. Furthermore, we show that iBundle generates solutions without complete revelation (or computation) of agent preferences.

Keywords
Iterative auction, agent-mediated electronic commerce, resource allocation, bundling problem, price discrimination.

1 INTRODUCTION

The popularity of Internet-based auctions continues to rise. Auctions are dynamic and often efficient mechanisms for selling items in complex marketplaces. Auctions are useful for sellers that are poorly informed about the value of items, for example collectibles and one-of-a-kind items (*e.g.* `www.ebay.com`, `www.zauction.com`), and in markets for time-critical items, for example electronic components (*e.g.* `www.fastparts.com`). In addition, auctions provide a foundation for mediation and brokering in automatic supply-chain and procurement systems, as the trend towards disaggregation of services and products continues [2, 4].

However, standard auction mechanisms can generate inefficient allocations of resources when agents demand *bundles* of complementary resources, *i.e.* "I only want B if I also get A". This bundling problem has been identified in many problem domains, including auctions for take-off and landing slots at an airport [19], distributed task-allocation problems [20], and factory scheduling problems [24]. Similar problems occur in procurement when products have multiple attributes, *e.g.* quality, price, delivery time, and terms of payment [5], and also in supply-chain problems where services from multiple suppliers need to be coordinated.

Auctions that price bundles as the sum price of the individual items (including sequential and simultaneous auctions) fail when there are no prices that support an efficient solution (the *existence* problem) and also when agents bid cautiously to avoid purchasing an incomplete bundle (the *exposure* problem) [8]. These problems were both identified in the recent FCC spectrum auction [12]. Although new contract types for task (or resource) re-allocation (*e.g.* swap, multilateral exchange, etc.) can support efficient allocations in a decentralized market [1], in general, costs of negotiation and search become too high without a central auctioneer [5].

Bundle auctions, that price bundles and allow agents to bid for bundles of items, can provide a solution to both the existence and exposure problems [8, 24]. Agents can use bundle bids to directly express contingent demands for items. However naive bundle auctions for G items quickly become informationally and computationally infeasible, with $2^{|G|}$ bundles to price and $O(|G|^{|G|})$ possible allocations [13, 21].

iBundle is the first ascending-price bundle auction that is guaranteed to compute optimal bundle allocations with a best-response agent bidding strategy. Moreover, we have proved [14] that the resulting prices correspond to a competitive equilibrium for the package assignment model [6]. The auction sells bundles for different prices to different agents (price discrimination), when this is necessary to support an optimal solution. Results from tests on hard resource allocation problems show that iBundle often generates solutions that are more efficient than current iterative auctions.

The auction has practical significance because it addresses the computational and informational complexity of bundle auctions. For agents, the auction can terminate before agents have revealed (or even computed) their values for every bundle. This can be important when agents have hard valuation problems [15, 16]. Furthermore, the auctioneer only needs to generate explicit prices on a subset of bundles and often solves smaller winner determination problems than in the classic sealed-bid bundle auction (the Generalized Vickrey Auction [23]). Also relevant, because the winner determination problem remains $\mathcal{NP}$-complete, is that the auction allows a tradeoff between performance and computation. Increasing the minimal bid increment can speedup termination and reduce the number of winner determination problems that the auctioneer must solve, although perhaps at the cost of suboptimal bundle allocations.

Section 2 provides a brief introduction to the bundling

In Proc. ACM Conference on Electronic Commerce (EC-99), Denver, November 1999.

Agent	A	B	C	AB	BC	AC	ABC
1	60	50	0	200*	50	60	220
2	0	60	50	60	200	50	220
3	50	0	75*	50	75	200	220

Table 1: Agent valuations in Problem 1.

problem. Section 3 describes *i*Bundle, and presents a simple best-response agent bidding strategy. In section 4 we demonstrate *i*Bundle on a simple example. Section 5 presents the empirical results. Finally we present our conclusions, and discuss opportunities for further work.

2 THE BUNDLING PROBLEM

The *bundling problem* is the problem of allocating items to agents to *maximize the sum value across the agents*, allowing agents to have non-additive values for individual items. For example, consider a factory-scheduling problem with agents that require labor and machine time, where receiving one without the other has no value. The optimal solution will allocate bundles of labor and machine time to agents to maximize the value of the complete schedule, for example minimizing the total cost of delay in completing the jobs of all agents.

Problem 1 (Table 1) introduces a problem that we will refer back to later in the paper.[1] There are $G = \{A, B, C\}$ items to allocate to $I = \{1, 2, 3\}$ agents. The table lists the value of each agent i for each bundle of items. Let $v_i(S)$ denote the value of agent i for bundle $S \subseteq G$. Agents have non-additive value for items, for example agent 1 has $v_1(A) = 60, v_1(B) = 50$ and $v_1(AB) = 200$. The optimal allocation (that maximizes the sum value) is indicated with *, and allocates bundle AB to agent 1, and item C to agent 3, for a value $V^* = 275$.

In most electronic commerce domains agents will be self-interested, and will not reveal truthful values for bundles to the auctioneer unless they have an incentive to do so. The market-based approach to the bundling problem is to prices items to provide incentives for agents to demand items that are part of the optimal allocation. In this paper, as is common in the auction literature, we assume that agents are risk-neutral with utility functions that are quasi-linear in money. Given value $v_i(S)$ for bundle S, and price $p(S)$, agent i has utility $u_i(S) = v_i(S) - p(S)$ for the bundle. Prices *support* an allocation when every agent receives a bundle that maximizes its utility, given the prices.

There are often no prices on individual items that support an optimal allocation, with prices on a bundle equal to the sum of the price of its constituent items. It is simple to prove this for Problem 1 (see Bykowsky *et al.* 1995 [8]). However, there are always *bundle prices* that support the optimal allocation [26], where the price of a bundle does not need to be equal to the sum of the price of its constituent items. For example, prices $p(AB) = 150$ and $p(C) = 60$ (with high enough prices on the other bundles) support the optimal allocation in Problem 1. Furthermore, when prices are *competitive equilibrium* (CE) prices the allocation is optimal [6].[2] Intuitively, for prices to be CE they must support an allocation that both maximizes agents' utilities and the auctioneer's revenue. In some problems the CE prices are discriminatory, with a different price for the same bundle to different agents [6].

Within this framework, one useful approach to auction design is to try to compute CE prices from bids from self-interested agents. We have proved [14] that *i*Bundle generates CE prices for agents that follow a myopic best-response bidding strategy. Agents will behave autonomously in an open system such as the Internet, with behaviors designed and implemented by different companies. However, it is often useful to show that auctions can perform well with simple and reasonable agent bidding strategies (for example best-response), because complex agent decision problems can limit the performance of electronic markets [16].

[1] The problem is derived from a problem in Bykowsky *et al.* [8].

[2] Possible competitive equilibrium bundle prices in this example are $\mathbf{p} = \{45, 60, 65, 185, 200, 190, 220\}$.

3 THE *i*BUNDLE AUCTION

*i*Bundle is an ascending-price bundle auction, that announces new bundle prices and provisional allocations at the end of each round, and allows agents to bid for bundles. The auctioneer announces *ask prices*. The *bid price* is the price that an agent bids for a bundle, and the price an agent must pay if it wins the bundle. In general the ask prices are lower bounds on acceptable bid prices (although see discount rules in section 3.1.2). Agents can place additive-or (OR) and exclusive-or (XOR) bids for bundles. For example, a bid (S_1, P_1) OR (S_2, P_2), indicates that an agent wants one or both of bundles S_1, S_2 at prices P_1 and P_2; while a bid (S_1, P_1) XOR (S_2, P_2) indicates that an agent wants at most *one* of bundles S_1 and S_2. Although every OR bid can be represented as an equivalent XOR bid (that lists every combination of bundles explicitly), OR bids allow more efficient bid representations for agents that have naturally additive values. *i*Bundle can also use a more efficient price-update rule for OR bids than XOR bids, generating less explicit bundle prices, and increasing prices more quickly (see section 3.1.3).

There are three basic variations of *i*Bundle: *i*Bundle(2) generates anonymous ask prices (same to every agent); *i*Bundle(3) generates discriminatory ask prices (possibly different for each agent); and *i*Bundle(d) switches from anonymous to discriminatory prices dynamically, agent-by-agent, to support efficient allocations.[3]

Price discrimination enables *i*Bundle to generate efficient allocations in *all* problem instances. However, discriminatory prices can be more difficult to enforce than anonymous prices (see section 3.2), and *i*Bundle(d) may be preferred to *i*Bundle(3) when efficient allocations are required but price discrimination should be avoided when possible. In fact, we show empirically that *i*Bundle(2) often performs well, and that the increase in efficiency from discriminatory prices is often only measurable at very small bid increments.

3.1 Auction Description

We now describe the auction in detail. The components of *i*Bundle that are common to all variations are *bidding*, *winner determination*, and *termination*. The *price-update* rules differ across variations. The auction proceeds in rounds.

[3] In economic terminology, 'first-order' prices are linear and anonymous; 'second-order' prices are non-linear and anonymous; and 'third-order' prices are non-linear and discriminatory [6]. This taxonomy motivates the names of each variation of *i*Bundle ('d' translates to "dynamic").

We will describe a *preprocessing* step that converts every OR bid received by the auctioneer into an equivalent set of XOR bids. With this, we describe the winner determination, termination, and price-update rules for an auction that receives only XOR bids. Let G be the set of items to be auctioned, and let I be the set of agents.

3.1.1 Announce Prices and Allocation

At the start of round t, the auctioneer announces new prices for a subset of bundles. The ask prices are initially zero for every bundle (unless the seller has a reservation price). The auctioneer only needs to announce a "minimal" set of new prices, because the price of every bundle is implicitly at least the maximum price of the bundles that it contains[4] and never decreases between rounds. For example, if the ask price of bundle AB increases to \$5, it is not necessary to announce that the price of bundle ABC is also increased to \$5. Let $p_i^t(S)$ denote the ask price for bundle S to agent i at the start of round t. In general, the ask price for bundle S might not be the same for every agent. The method to compute new prices is described in section 3.1.6.

The auctioneer also announces a provisional allocation of bundles to agents at the start of each round. No agents receive any bundles in the initial allocation. Let $\chi^t = \{[i,S] \mid i \in I, \quad S \subseteq G\}$ represent an allocation of bundles $\chi^t(i) = \{S \mid [i,S] \in \chi^t\}$ to agent i at the start of round t. For example, $\chi = \{[1, AB], [3, C]\}$, indicates that agent 1 is allocated bundle AB and agent 3 is allocated item C. The method to compute the new allocation is described in section 3.1.4.

3.1.2 Bidding

In each round an agent can submit either an XOR or an OR bid for a set of bundles. For example: $B_i^{\text{XOR}} = \{(S_1, P_1), (S_2, P_2)\}$, indicates an XOR bid for bundles S_1 and S_2 at prices P_1 and P_2 respectively; and $B_i^{\text{OR}} = \{(S_1, P_1), (S_2, P_2)\}$, indicates an OR bid for bundles S_1 and S_2. Agents must repeat bids for every bundle that they receive in the current allocation, as part of an OR bid if they are currently allocated more than one bundle. Agents cannot submit both XOR and OR bids in the same round.

In general, any bid for a bundle with a bid price *below* the ask price is invalid, and ignored by the auctioneer. There are two exceptions. First, an agent that is repeating a bid for a bundle that it receives in the current allocation may repeat the bid for the same price, even if the ask price has increased. This rule ensures that the auctioneer receives monotonically increasing revenue in each round of the auction, and motivates agents to place credible bids because decommitment from an allocation is only possible when the auctioneer receives bids that generate more revenue, and not arbitrarily. Second, an agent can take an "ϵ-discount" on the ask price of any bundle in any round. However, the auctioneer can detect this discount, and once an agent takes this discount it is forbidden from bidding for the bundle if the ask price ever increases in future rounds. This rule is designed to allow an agent to continue to bid for bundles that are priced just above its value, perhaps as the result of a previous unsuccessful bid from the same agent at a price just below its value.

[4]Bundle S_1 contains bundle S_2 if $S_2 \subset S_1$.

3.1.3 Preprocessing

At the end of every round the auctioneer first transforms every OR bid into a *set* of equivalent XOR bids. Given a bid (S_1, P_1) OR (S_2, P_2) we create two new *dummy* agents that each bid one of the original bundles as an XOR bid. From the agent's perspective the new XOR bids are equivalent to the original OR bid.[5] The winner determination, price update, and termination rules are all executed as though the dummy agents are real agents that have placed bids. The transformation is reversed before announcing new prices and allocations to the real agents.

3.1.4 Winner Determination

Then the auctioneer solves the winner determination problem, computing an allocation of bundles to agents that maximizes revenue. The auctioneer first rejects bids from agents with invalid bid prices, and checks that agents repeat bids for bundles received in the current allocation. The new provisional allocation χ^{t+1} solves:

$$\begin{aligned} \max_{\chi^{t+1}} \quad & \sum_{[i,S] \in \chi^{t+1}} P_i(S) \\ \text{such that} \quad & \text{feasible}(\chi^{t+1}) \ \wedge \ \text{bid\_consistent}(\chi^{t+1}) \end{aligned} \tag{1}$$

where $P_i(S)$ is the bid price for bundle S from agent i. Allocation χ is *feasible* if no item is allocated more than once:

$$\text{feasible}(\chi) \ \Leftrightarrow \ \chi = \{[i,S] \mid S \cap S' = \emptyset \quad \forall S, S' \in \chi\} \tag{2}$$

Allocation χ is *bid_consistent* if the auctioneer only allocates bundles to agents that bid for them, and allocates at most one bundle to each agent (respecting the XOR constraint):

$$\text{bid\_consistent}(\chi) \ \Leftrightarrow \ \chi = \{[i,S] \mid S \in B_i^{\text{XOR}}, |\chi(i)| = 1\} \tag{3}$$

where $S \in B_i^{\text{XOR}}$ iff agent i bids for bundle S, and $|\chi(i)|$ is the number of bundles allocated to agent i. Agents that placed an OR bid can receive more than one bundle because every dummy agent can receive a bundle.

The auctioneer breaks ties in favor of assigning bundles to more agents, and then at random – except when the same bids are received in two successive auction rounds, when the auctioneer selects the same allocation as before and the auction terminates.

3.1.5 Termination

The auction terminates when one of two simple rules hold, designed to detect quiescence. The auctioneer then imple-

[5]We could transform an OR bid into a *single* XOR bid. For example, the bid $(A, 2)$ OR $(B, 3)$ would become $(A, 2)$ XOR $(B, 3)$ XOR $(AB, 5)$. Indeed, if we allowed agents to only submit XOR bids, this is the kind of bid that an agent with an additive demand over bundles would need to submit. However, we choose to transform an OR bid into a *set* of XOR bids, bids $(A, 2)$ and $(B, 3)$ in this case, because this allows more efficient price updates. First, the auctioneer only needs to explicitly price the or-terms, such as A and B, and not all combinations of bundles, such as AB. Second, the auctioneer can increase prices more quickly. For example, if the agent receives item A but not item B, then with a single XOR bid we do not increase any prices because of the agent's bid, while with a set of XOR bids we increase the price on item B (see the price-update rules). Hence, the OR bids are somewhat more than "syntactic sugar" in *i*Bundle.

ments the final allocation and sells bundles to agents for the prices that they bid. The auction terminates when either:

[T1] All agents that bid are *happy*, or

[T2] All agents submit the same bids in successive rounds.

where an agent is *happy* at the end of round t when it receives one of the bundles for which it placed an XOR bid.[6]

Condition [T1] implies quiescence for myopic best-response agents, because no agent will increase its bids on any bundle in the future, the current allocation will continue to maximize the auctioneer's revenue, and ask prices will not increase because agents will remain happy.[7]

Condition [T2] is a little stronger, and implies quiescence for agents that compute bids deterministically, based on ask prices and the current allocation.[8] Unchanged prices between rounds are not sufficient to imply quiescence unless the allocation also remains unchanged.

3.1.6 Price Update

Finally, when the auction does not terminate, the auctioneer updates prices. The price-update rule depends on the auction variation. Prices in *i*Bundle(2) are anonymous. The ask price $p(S)$ is increased based on bids received from agents that are not happy (*i.e.* receive no bundle in the current allocation). The ask price increases when the maximum rejected bid price for the bundle from an unhappy agent is within ϵ (the minimal bid increment) of the current ask price. Let $P_i(S)$ denote the *bid price* for bundle S from agent i, with $P_i(S) = 0$ when the agent does not bid for the bundle. Then, given bids B_i^{XOR} the new ask price $p^{t+1}(S)$ is computed as:

$$p^{t+1}(S) = \max\left(p^t(S),\ \max_{i \in unhappy} P_i(S) + \epsilon\right) \qquad (4)$$

where *unhappy* is the set of agents that are not happy with the new provisional allocation.[9]

Prices in *i*Bundle(3) are discriminatory, there is a set of ask prices for every agent. Price increases to each agent are based only on bids received from that agent. Prices only increase when an agent (or one of its dummy agents when it placed an OR bid) is unhappy, and then only when the agent bid at within ϵ of the current ask price (*i.e.* did not take a discount). The new ask price $p_i^{t+1}(S)$ to agent i for bundle S is computed as:

$$p_i^{t+1}(S) = \max\left(p_i^t(S),\ P_{i' \in unhappy(i)}(S) + \epsilon\right) \qquad (5)$$

[6]By extension, an agent that placed an OR bid is "happy" only when it receives all the bundles in its bid. This is when the dummy agents for that agent are all happy.

[7]This rule is also a useful disincentive against strategic bidding, because it does not allow an agent to place a new bid if a bid is accepted and every other agent is happy.

[8]For example, suppose that agents submit the same bids in rounds 9 and 10. The prices at the start of round 10 are computed on the basis of the allocation selected, and the bids placed. Agents submit the same bids in round 10 as in round 9, and the auctioneer computes the same allocation and announces the same prices at the start of round 11. Agents with deterministic bid strategies will place the same bids in round 11 as in round 10, and we have quiescence.

[9]Notice that when an agent submits an OR bid the price is increased to ϵ above the bid price for all bundles that the agent is not allocated, because the corresponding dummy agents are not happy.

where $unhappy(i)$ is the set of dummy agents for agent i that are not happy (*i.e.* agent i when agent i placed an XOR bid and received no bundle, or a dummy agent for every bundle that agent i did not receive when agent i placed an OR bid).

Prices in *i*Bundle(d) are initially anonymous, but can become discriminatory during the auction. The auction maintains a set $anon(t)$ of agents that receive anonymous prices in round t, initially containing all agents. The price update rule for the anonymous prices is the same as for *i*Bundle(2), except that only bids from agents that continue to receive anonymous prices are considered. The price update rule for the agents that receive discriminatory prices is the same as for *i*Bundle(3).

A simple test determines whether to start charging an agent discriminatory prices. Once an agent starts to receive discriminatory prices it does in all future rounds. The auctioneer tests whether prices are increased on at least two *disjoint* bundles as a result of a single agent's bid (this rules out OR bids, because each dummy agent only bids for a single bundle, and agents that are happy). Let $incr_i(t)$ be the set of bundles that will increase in price because of the bids placed by agent i, *i.e.* $incr_i(t) = \{S \mid (\ S \in B_i^{\mathrm{XOR}}),\ (P_i(S) + \epsilon > p^t(S))\}$. Agent i starts to receive discriminatory prices if $incr_i(t)$ contains at least two disjoint bundles. The agent is removed from the set $anon(t+1)$, and its prices are initialized to the current anonymous prices, $p_i^t(S) = p^t(S)$, before they are updated.

3.2 Enforcing Non-linear and Discriminatory Prices

All variations of *i*Bundle can generate non-linear prices for items, for example $p(S_1 \cup S_2) < p(S_1) + p(S_2)$ (subadditive prices) and $p(S_1 \cup S_2) > p(S_1) + p(S_2)$ (superadditive prices) on disjoint bundles S_1 and S_2. Non-linear prices are often necessary to support efficient solutions.

An auctioneer may need to enforce non-linear pricing. Without enforcement an arbitrageur can try to profit from subadditive prices by purchasing bundles to be "disassembled" and sold to agents, or a bidding cartel of agents can form to take advantage of bundle discounts. Similarly, a single agent can try to avoid paying premiums on bundles by entering the auction under multiple pseudonyms and purchasing smaller bundles for "assembly" after purchase. The variations of *i*Bundle that use discriminatory prices require that agents cannot enter an auction under multiple pseudonyms to avoid price discrimination. Methods of enforcement include: (1) prevent the transfer of items between agents (*e.g.* the airline industry); (2) prevent agents from entering an auction under multiple pseudonyms, for example through cryptographic message authentication and signature techniques [22]; (3) reduce opportunities for after-markets.

3.3 A Best-response Agent Bidding Strategy

We present a simple and reasonable bidding strategy for an agent in *i*Bundle. The bidding strategy is a *myopic best-response* to the current state of the auction (prices and allocation). The strategy is not optimal in a game-theoretic sense, and deviations could increase an agent's utility in some auction scenarios. We do not consider agents with information about the other agents, that might place jump bids or bid for less bundles to change the outcome of the auction in their favor.

Agents with additive valuations over bundles can follow a best-response bidding strategy that places OR bids, while agents with general valuation functions, such as the agents in Problem 1 (Table 1), can follow a best-response bidding strategy that places XOR bids.

[XOR-bids] A best-response bidding strategy for agent i, given values $v_i(S)$ and ask prices $p_i(S)$ is to place an XOR bid, B_i^{XOR}, for every bundle that has non-negative utility and is within ϵ of maximizing utility (we assume indifference within ϵ, the minimal bid increment):

$$B_i^{\mathrm{XOR}} = \Big\{ (S, P_i(S)) \mid v_i(S) - P_i(S) \geq 0, \quad v_i(S) - P_i(S) + \epsilon \geq \max_{S'}\{v_i(S') - P_i(S')\} \Big\} \tag{6}$$

where $P_i(S)$ is the bid price for bundle S. The agent bids as low a price as possible without limiting its bids in future rounds. Therefore, the bid price is equal to the ask price, except when the agent is repeating a bid from the current allocation (when it repeats the same bid price), or when the ask price is greater than the agent's value for the bundle (when it considers a bid price ϵ-discounted from the ask price). Both of these discounts from the ask price are consistent with the bidding rules of the auction (see section 3.1.2).[10]

The bidding strategy is a myopic best-response strategy to the current allocation and ask prices.[11] Agents only bid for bundles that maximize their utility, and maximize the probability that one of those bundles will be in the tentative allocation announced at the end of the round by providing the auctioneer with as large a choice set as possible. If a bid is unsuccessful at the lowest possible price then agents get another chance to bid, because termination rule [T1] requires that every agent is happy. The bidding strategy is also *safe* [13], agents get bundles that are nearly optimal relative to their valuations at the final prices and never risk losing utility. Agents avoid the exposure problem [8].

[OR-bids] Agents that have valuation functions that are naturally additive over bundles, *e.g.* $v(S_1 \cup S_2) = v(S_1) + v(S_2)$ for all disjoint bundles S_1 and S_2 for which the agent has positive value, can submit OR bids. A best-response strategy is to bid for all bundles with positive utility at the current prices:

$$B_i^{\mathrm{OR}} = \{ (S, P_i(S)) \mid v_i(S) - P_i(S) \geq 0\} \tag{7}$$

where $P_i(S)$ (the bid price) is discounted from the ask price in the same cases as for the XOR-bidding strategy. This is a best-response strategy for an agent with additive value for each bundle because the agent bids for as many bundles with positive utility as possible, at the lowest prices possible.[12]

[10] This strategy automatically bids for a bundle S that an agent receives in the current allocation, because the agent can bid the same price, the price of other bundles has not decreased, and the bundle was in the set of utility maximizing bundles in the previous round.

[11] The strategy also has a *no-regret* property in the sense that an agent does not want to change its bid whatever the immediate bids of other agents.

[12] This strategy may seem suboptimal in the case that the ask price for "superbundle" $S_1 \cup S_2$ is less than the total ask price for bundles S_1 and S_2. However, an agent must repeat its bid for any bundle in the current allocation, and the auctioneer will only select a bid for bundle $S_1 \cup S_2$ when the bid price is at least as great as the sum of the bid prices for S_1 and S_2.

4 WORKED EXAMPLE

In this section we present a worked example of *i*Bundle(2) and *i*Bundle(3) on Problem 1. In this example discriminatory prices are unnecessary to generate an efficient allocation, and *i*Bundle(d) is identical to *i*Bundle(2). Table 2 records the prices, bids, and allocation in each round of *i*Bundle(2) for agents that place best-response XOR bids, with a bid increment $\epsilon = 30$. For each round t the table records the ask prices at the start of the round, $p^t(S)$, the bids placed by the agents, and the allocation at the end of the round, χ^{t+1}. The bids that maximize revenue (solve the winner determination problem) in each round are indicated with *.

We will look at round 4 in more detail. Agent 1 was allocated bundle ABC at the end of round 3. In round 4 agent 1 re-bids $(ABC, 60)$ (for the same bid price as in round 3), and also bids $(AB, 30)$. The reader can check that these bids maximize the agent's utility given the ask prices (to within ϵ). Agent 2 bids $(BC, 60)$ and $(ABC, 90)$, for utility 140 and 130 respectively. Agent 3 bids $(AC, 90)$ and $(ABC, 90)$, for utility 110 and 130. The auctioneer maximizes revenue by allocating bundle ABC to agent 2 (breaking the tie between agent 2 and 3 at random). Agent 2 is happy, so the prices are updated on the basis of the bids from agents 1 and 3. The price of bundle AB increases to $p^5(AB) = P_1(AB) + \epsilon = 60$, where $P_1(AB)$ is the bid price of agent 1 for the bundle AB; similarly $p^5(AC) = P_3(AC) + \epsilon = 120$, and $p^5(ABC) = P_3(ABC) + \epsilon = 120$.

The auction terminates at the end of round 15 (by rule [T2]), and the provisional allocation $\chi^{16} = \{[1, AB], [3, C]\}$ becomes the final allocation. Bundle AB is sold to agent 1 for $p(AB) = 180$ and bundle C is sold to agent 3 for $p(C) = 60$. These are approximate CE prices for the problem (see section 2).

Table 3 records the progress of *i*Bundle(3) on Problem 1. The minimal bid increment is unchanged, $\epsilon = 30$. For each round t the table records the ask price, $p_i^t(S)$, for each agent, the bids placed by each agent (blank entries indicate that no bids are received), and the new provisional allocation, χ^{t+1}. The winning bids are indicated with *.

The prices to agents are increased in rounds where they bid but are allocated no bundle. For example, at the end of round 4 agent 1 is in the provisional allocation, and has the same prices at the start of round 5; while the prices for BC and ABC increase for agent 2, and the prices for AC and ABC increase for agent 3. The auction terminates at the end of round 14 (by rule [T2]), and bundle AB is sold to agent 1 for $p_1(AB) = 180$, bundle C to agent 3 for $p_3(C) = 60$. This is the same outcome as for *i*Bundle(2).

5 EMPIRICAL STUDY

Initial results from an empirical study of the performance of *i*Bundle on several hard bundling problems provide empirical support for the theoretical efficiency of *i*Bundle. We simulated agents that follow the best-response bidding strategies described in section 3.3. Typical measures of auction performance include allocative efficiency and revenue. We also measured the "correctness" of allocations, the average number of times that an auction finds the optimal allocation. This is often a more sensitive measure of performance than efficiency (although with less economic relevance).

In presenting the empirical results for *i*Bundle we have a dilemma, because *i*Bundle(d) and *i*Bundle(3) *provably* generate 100% efficient and correct allocations for a small enough bid increment and best-response agent bidding strategies [14]. What we show is that *i*Bundle continues to perform well with quite large bid increments, and that *i*Bundle(2) (without price discrimination) will often perform as well as *i*Bundle(d) and *i*Bundle(3).

[Efficiency] Allocative efficiency, $\mathit{eff}(\chi)$, is measured as the ratio of the value of the final allocation χ to the value V^* of the optimal allocation that maximizes total value across the agents:

$$\mathit{eff}(\chi) = \frac{\sum_{[i,S]\in\chi} v_i(S)}{V^*} \tag{8}$$

where $v_i(S)$ is agent i's value for bundle S.

[Revenue] Revenue, $rev(\chi)$, is measured as the ratio of income to the value of the optimal solution:

$$rev(\chi) = \frac{\sum_{[i,S]\in\chi} P_i(S)}{V^*} \tag{9}$$

in recognition that an auctioneer cannot expect to sell goods for more than agents value them, at least in the long run.

*i*Bundle will often terminate before agents have revealed (or computed) complete information about their values for bundles. We define *information revelation*, *inf*, designed to measure the extent to which an agent has revealed its value for each bundle to the auctioneer during the auction. This does not necessarily correspond to computational savings.

[Information Revelation] Information revelation, $\mathit{inf}(i)$, for agent i is measured as the sum of the final price bid by the agent for all bundles in its valuation function, as a fraction of the sum of the true value of each bundle:

$$\mathit{inf}(i) = \frac{\sum_{S\in \mathrm{Bid}_i} P_i^{\max}(S)}{\sum_{S\in \mathrm{Val\_fun}_i} v_i(S)} \tag{10}$$

where $P_i^{\max}(S)$ is the maximum bid from agent i for bundle S during the auction; Bid_i is the set of bundles that receive bids from agent i; and $\mathrm{Val\_fun}_i$ is the set of bundles with positive value in agent i's valuation function.[13] The overall auction information revelation, *inf*, is computed as the average information revelation over all agents.

5.1 Problem Sets

We tested *i*Bundle on several problem sets from the literature. The problem set characteristics are summarized in Table 4. The CalTech problem set [11] is designed to represent a hard spatial fitting problem, and has been used to test the AUSM and RAD bundle auctions [9]. Problem sets PS1 and PS2 are resource allocation problems that have been used to test the performance of a sequential auction mechanism (SEQ) with adaptive agent bidding strategies [7]. Problem sets 9-16 are generated from bid distributions used to test a new winner determination algorithm for bundle auctions [21].[14] We designed problem sets 4-8 to represent different levels of subadditivity and superadditivity over items.[15] We refer to these problem sets as *k-comp(g)*. Agents have subadditive values for combinations of items when $k < 1$, and superadditive values when $k > 1$. The parameter g indicates how many items are covered by bundles with positive value in each agent's valuation function.

#	Problem Name	$\lvert G\rvert$	$\lvert I\rvert$	Number bundles per agent	(X)or / (O)r	Num agents in sol (%)	Naive *eff* (%)	Naive *corr* (%)	Num trials
1	*CalTech*	6	5	5	X	40.0	63.2	2	50
2	*PS1*	12	4	3.97	X	89.0	82.1	20	50
3	*PS2*	12	5	4.07	X	58.4	79.3	20	50
4	*0-comp(4)*	5	5	15	X	85.6	61.2	0	50
5	*0.5-comp(4)*	5	5	15	X	80.8	63.2	0	50
6	*1-comp(4)*	5	5	15	X	71.2	63.0	0	50
7	*2-comp(4)*	5	5	15	X	49.2	65.3	4	50
8	*4-comp(4)*	5	5	15	X	43.6	63.5	6	50
9	*random*	10	5	10	X	84.8	64.9	8	25
10	*w-random*	10	5	10	X	38.4	82.8	20	25
11	*uniform*	20	5	10	X	60.0	73.0	8	25
12	*decay*	20	5	10	X	96.0	80.2	12	25
13	*random-or*	10	5	10	O	74.4	55.3	0	25
14	*w-random-or*	10	5	10	O	39.2	82.4	20	25
15	*uniform-or*	20	5	10	O	48.8	69.6	4	25
16	*decay-or*	20	5	10	O	92.8	72.5	0	25

Table 4: Problem characteristics.

Table 4 states the number of items $|G|$ in each problem set, the number of agents $|I|$, the average number of bundles with positive value for each agent, and whether the agents have OR or XOR values over bundles. Table 4 also records the average percentage of agents in the optimal allocation. All other things being equal, we would expect a greater proportion of agents to receive bundles as the number of items increases, the number of agents decreases, and the level of superadditivity decreases. For example, the number of agents in the optimal solution falls as k increases in the *k-comp* problem sets (4–8). We also computed the performance of a naive central algorithm `naive` for each problem set, to provide a baseline for the performance of the auction-based solutions. The naive algorithm repeatedly selects an agent at random (without replacement) and tries to allocate bundles to the agent until it is happy, choosing bundles in order of decreasing value.

5.2 Comparison Auction Mechanisms

We compared the performance of *i*Bundle with reported results for other auctions. AUSM and RAD are iterative auctions that allow agents to bid for bundles [11, 9], and SEQ is a sequential auction for items with agents that have adaptive bidding strategies [7]. We also implemented a simple simultaneous ascending price auction, with and without bid withdrawal (SAA-w and SAA),[16] and best-response agent bidding strategies.[17]

[13] When an agent has a naturally additive valuation function, we use the set of "seed" bundles with positive value, not all additive combinations.

[14] We generated agent valuation functions by partitioning the bid distributions across the agents. In problem sets 9-12 we give agents XOR values over the bundles, in problem sets 13-16 agents have OR values. In the "decay" bid distribution we choose parameter $\alpha = 0.55$.

[15] Thanks to Peter Wurman for providing the initial idea for these problem sets.

[16] In SAA-w agents can withdraw a bid in any round. When an agent withdraws a bid the ask price is set to the price P of the bid withdrawn. If the item remains unsold the agent must pay penalty P. This approximates the rule used in the FCC spectrum auction [17].

[17] In SAA agents bid for items that maximize utility, assuming they will win every bid [24]. Without budget constraints agents write-off incomplete bundles with this strategy, in a phenomena described by Bykowsky *et al.* as *mutually destructive bidding* [8]. The best-response strategy in SAA-w is similar, except that agents assume that they can decommit for free. Once an agent has withdrawn a bid, the penalty represents a sunk cost. There is continued debate about the

Problem		Performance	SEQ	RAD	AUSM	*i*Bundle(2)	
#	Name	measure				ϵ (%)	
						20	5
1	*CalTech*	*eff*		90.4	94	96.4	99.7
		corr		80		36	80
		rev		79	71	70.6	77.7
2	*PS1*	*eff*	87			92.4	99.4
3	*PS2*	*eff*	80			92.8	99.7

Table 5: Performance comparison with SEQ, RAD and AUSM on problems 1, 2 and 3. Bid increment ϵ (%); Efficiency *eff* (%); Correctness *corr* (%); Revenue *rev* (%).

5.3 Experimental Results

We implemented *i*Bundle, SAA and SAA-w in C++. A branch-and-bound depth-first search is used to solve the auctioneer's winner determination problem in each round [21]. Modules to generate random problem sets, and simulate agent bidding strategies were also coded in C++.

We normalized the minimal bid increment in the iterative auctions (SAA, SAA-w and *i*Bundle), to give some consistency across problem sets and between single-item auctions and bundle auctions. We adjusted the minimal bid increment ϵ according to the effect of a price increase on the utility of an agent in the final allocation.[18] We report results for bid increments of 20% and 5%.

First, we compared the performance of *i*Bundle(2) with reported results for AUSM and RAD [9, 11] on problem set 1 (Table 5). The experiments reported in DeMartini *et al.* [9] are with human participants, and it is possible that software agents could perform better (or worse). This aside, *i*Bundle(2) achieves a higher efficiency than RAD and AUSM, and is competitive in revenue. We also compared the performance of *i*Bundle with SEQ [7] on problem sets 2 and 3. *i*Bundle(2) generates almost perfect allocations, significantly outperforming SEQ (results on *corr* and *rev* are not available for SEQ). The empirical results reported for SEQ are with agents that follow sophisticated bidding strategies, learned over many repeated trials of the same problem instance. In comparison, *i*Bundle agents follow simple best-response bidding strategies.

Figure 1 plots the efficiency (a) and correctness (b) of *i*Bundle(2) and SAA-w (with $\epsilon = 5\%$) for each problem set, together with the `naive` performance as a baseline. The SAA auction fails in many problem sets (1, 3, 8, 10–16), in the sense that agents *lose* utility through participation when the best-response bidding strategy leaves them with incomplete bundles.[19] SAA-w allows bid withdrawal, and mitigates the exposure problem in problem sets 12 and 13. For the sake of analysis we substitute the efficiency and correctness of `naive`, and the revenue from the Generalized Vickrey Auction (GVA), in problem sets where SAA and SAA-w fail.[20]

effect of bid withdrawal on auction performance [8, 18].

[18]The actual increment is set so that the effect of a price increase on every bundle in an agent's final allocation is approximately equal to ϵ% of the average value.

[19]The efficiency of an allocation is irrelevant if it is generated with bidding strategies that lead to utility loss, because we cannot expect agents to follow bad bidding strategies in the long term.

[20]The naive central algorithm provides a useful lower bound on efficiency and correctness, but revenue is undefined. The GVA provides a lower bound on revenue, for agents that follow rational bidding strategies in an auction that generates efficient solutions.

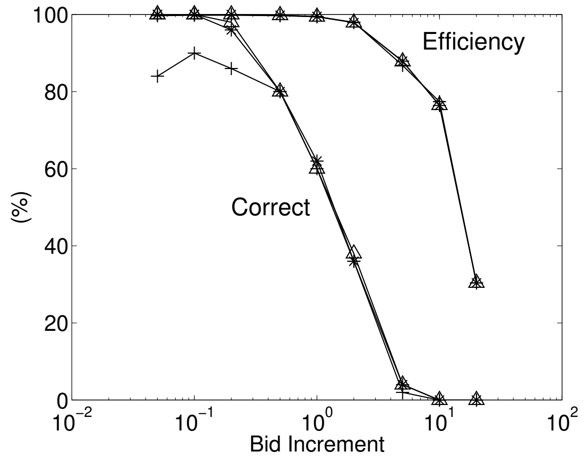

Figure 2: Performance of *i*Bundle as the bid increment ϵ decreases. '+' *i*Bundle(2); '*' *i*Bundle(d); 'Δ' *i*Bundle(3). Problem set *0.5-comp(3)*.

It is useful to consider average performance across all problem sets. *i*Bundle(2) (with $\epsilon = 5\%$) achieves an efficiency of 99%, compared to 83.3% efficiency for SAA-w, 81.6% for SAA, and 70.1% for `naive`. SAA-w performs well in problem set 2 (where there is little competition for resources), and sets 4, 5 and 6, where agents have subadditive, linear-additive or slightly superadditive values on bundles. Average correctness is 67.2% for *i*Bundle(2), 23.9% for SAA-w, and 7.8% for `naive`. Finally, *i*Bundle(2) generates 75.6% revenue, compared to 70.8% for SAA-w and 62.9% for the GVA.

The increase in efficiency with *i*Bundle(d) and *i*Bundle(3), compared to *i*Bundle(2) is marginal. With $\epsilon = 5\%$, *i*Bundle(3) achieves 99.1% efficiency, compared to 99% efficiency for *i*Bundle(2). Figure 2 compares the efficiency and correctness with each auction variation for problem set 0.5-comp(3). Although price discrimination is required for 100% correctness in this problem, the efficiency improvement is negligible. Furthermore, price discrimination only makes a difference for very small bid increments, when the communication cost begins to increase rapidly. For bid increment $\epsilon > 0.5\%$ the performance is almost identical. The relationship between average number of rounds to termination and bid increment is approximately linear, with $\epsilon = 5\%, 0.5\%, 0.05\%$ corresponding to 6, 49, and 480 rounds respectively. An auctioneer may choose not to operate below 0.5% because of high communication costs, computation costs, and indirect costs due to elapsed time.

The results show that *i*Bundle can tradeoff performance for communication, computation, and information revelation costs. *i*Bundle(2) achieves allocations that average 91.7% efficiency across all problem sets and terminates after 5.7 rounds with bid increment $\epsilon = 20\%$, down from 99% efficiency after 18 rounds with bid increment $\epsilon = 5\%$. We can see the tradeoff for problem 0.5-comp(3) in Figure 2.

Finally, *i*Bundle(2) requires an average of 57.5% information revelation at $\epsilon = 20\%$ (when the allocations are 91.7% efficient), and an average of 71% information revelation at $\epsilon = 5\%$ (when the allocations are 99% efficient). The (sealed-bid) GVA requires 100% information revelation from agents to achieve 100% efficiency. Information reve-

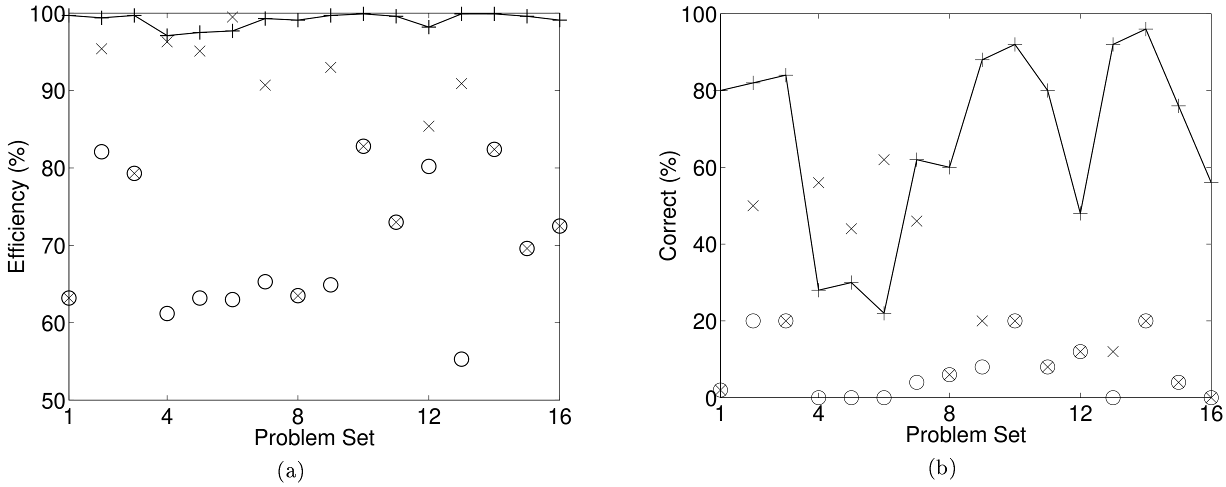

Figure 1: Performance of SAA-w 'x', *i*Bundle(2) '+', and a naive central resource allocation algorithm 'o'. Bid increment $\epsilon = 5\%$. (a) Efficiency. (b) Correctness.

lation may be smaller in real problems, because we provide agents with sparse valuation functions in the first place (this limits the size of the denominator in eq. 10).

6 DISCUSSION

The prices in *i*Bundle are ascending, and updated on the basis of bids from agents that are not accepted in the current round of the auction. Intuitively, when an agent's bid price $P(S)$ for bundle S is rejected, and the agent is allocated no bundle, then the bid price is a lower bound on the price than *any* agent must bid to receive the bundle – assuming the auctioneer receives at least the same bids for the bundles in the current allocation in the next round.

It is hard to provide further intuition for the idiosyncrasies of the price rules (*e.g.* the ϵ-discount and the rule for price discrimination) without an accompanying theoretical analysis (see [14]). The basic idea is to increase prices until every agent that bids is happy with its allocation, while supporting bids from agents that allow the auctioneer to maximize its revenue at the current prices in every round. Increasing prices on the basis of losing bids, together with the bid discount rules, and the decision to sometimes charge discriminatory prices, all work together to enable this.[21] With a small enough bid increment we have proved that *i*Bundle terminates with competitive equilibrium (CE) bundle prices that support an optimal allocation [14]. Price discrimination is sometimes necessary for optimality, but practically seems to have only a marginal effect on performance.

The termination rules could be relaxed for bounded-rational agents that can make *mistakes* in the bids that they place, because of auction time constraints and the complexity of local valuation problems. For example, we could run the auction without [T1] and with [T2] applied to several consecutive rounds, to allow agents to change an allocation by placing new bids. However, both of these changes may increase agents' ability to bid strategically and deviate from best-response strategies.

Related bundle auctions have been proposed in recent years, including AUSM [3] and Ascending k-Bundle Auction (AkBA) [25, chapter5]. AUSM does not generate ask prices for bundles and requires complex agent bidding strategies, for which theoretical analysis is very difficult [8, 13]. AkBA is a family of auctions that are similar to *i*Bundle, but use a linear program to update prices, and never charge discriminatory prices. A1BA, thought to be the most promising of the family, is not believed to support optimal allocations in all problems for any simple agent behavior. RAD [9] is an iterative auction that allows bids for bundles of items, but does not support non-linear bundle prices.

7 CONCLUSIONS

We have presented results on the performance of *i*Bundle, a new iterative bundle auction. *i*Bundle generates efficient solutions in a reasonable number of rounds with a best-response agent bidding strategy. Initial analysis suggests that *i*Bundle continues to perform well without price discrimination, on the basis of bundle prices alone. We believe that its ability to tradeoff allocative efficiency for computation and communication cost will be important in practical applications.

Iterative auctions can allow more efficient computation than sealed-bid auctions when agents have hard valuation problems, and this can lead to higher allocative efficiency [15]. *i*Bundle can generate solutions without information from agents on their value for every bundle. Agents only need to determine the bundles that *maximize utility* given prices to be able follow a best-response bidding strategy, and this can be done with *approximate values* for bundles.

Although *i*Bundle can leverage new algorithms for the winner-determination problem [10, 21], the fundamental complexity of the bundling problem remains (the auctioneer's problem is $\mathcal{NP}$-complete). One future direction is to introduce an approximation algorithm for this problem, and study the effect on the bidding strategies of agents and the performance of the auction. Another approach is to look

[21] *i*Bundle reduces to the English auction (for agents that do not make jump bids, sometimes called the Japanese auction) for a single item. The price in the auction is increased whenever more than one agent bids for the item at the current ask price, and the auction terminates when one agent remains in the auction.

for useful decompositions of the bundling problem, and apply reduced-scope bundle (or other) auctions to parts of the resource space.

There are several other possible extensions to this work. These include investigating the effect of jump bids and strategic bidding behavior on performance. A particular goal is to understand the extent to which *i*Bundle addresses the free-riding (or threshold) problem [8, 13]. We plan to apply *i*Bundle to a real e-commerce problem domain, and extend the auction to environments with multiple sellers.

8 ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant SBR 96-02053. Thanks to Pete Wurman, Fredrik Ygge, Lyle Ungar and the anonymous reviewers for their helpful comments and suggestions.

9 REFERENCES

[1] Andersson, M., and Sandholm, T. Leveled commitment contracts with myopic and strategic agents. In *Proc. 15th National Conf. on Artificial Intelligence (AAAI-98)* (1998), pp. 38–45.

[2] Bakos, Y. Towards friction-free markets: The emerging role of electronic marketplaces on the Internet. *Comm. ACM 41* (1998), 35–42.

[3] Banks, J. S., Ledyard, J. O., and Porter, D. Allocating uncertain and unresponsive resources: An experimental approach. *The Rand Journal of Econ. 20* (1989), 1–25.

[4] Bichler, M., Beam, C., and Segev, A. An electronic broker for business-to-business electronic commerce on the Internet. *Int. Journal of Cooperative Information Systems 7* (1998), 315–331.

[5] Bichler, M., Kaukal, M., and Segev, A. Multi-attribute auctions for electronic procurement. In *Proc. 1st IBM IAC Workshop on Internet Based Negotiation Technologies* (1999). Yorktown Heights.

[6] Bikchandani, S., and Ostroy, J. M. The package assignment model. Tech. rep., Anderson School of Management and Department of Economics, UCLA, 1998.

[7] Boutilier, C., Goldszmidt, M., and Sabata, B. Sequential auctions for the allocation of resources with complementarities. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)* (1999), pp. 527–534.

[8] Bykowsky, M. M., Cull, R. J., and Ledyard, J. O. Mutually destructive bidding: The FCC auction design problem. Tech. Rep. SSWP 916, CalTech, 1995. Revised June 1998; November 1998.

[9] DeMartini, C., Kwasnica, A. M., Ledyard, J. O., and Porter, D. A new and improved design for multi-object iterative auctions. Tech. rep. SSWP 1054, CalTech, 1998. Revised March 1999.

[10] Fujishima, Y., Leyton-Brown, K., and Shoham, Y. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)* (1999), pp. 548–553.

[11] Ledyard, J. O., Porter, D., and Rangel, A. Experiments testing multiobject allocation mechanisms. *Journal of Economic and Management Strategy 6* (1997), 639–675.

[12] McMillan, J. Selling spectrum rights. *Journal of Economic Perspectives 8* (1994), 145–62.

[13] Milgrom, P. Putting auction theory to work: The simultaneous ascending auction. *Journal of Political Economy 108* (1999). To appear.

[14] Parkes, D. C. *i*Bundle: A provably optimal ascending price bundle auction. Tech. rep., University of Pennsylvania, 1999. In preparation.

[15] Parkes, D. C. Optimal auction design for agents with hard valuation problems. In *Proc. 2nd Workshop on Agent Mediated Electronic Commerce (AmEC-99)* (July 1999). Stockholm.

[16] Parkes, D. C., Ungar, L. H., and Foster, D. P. Accounting for cognitive costs in on-line auction design. In *Agent Mediated Electronic Commerce (LNAI 1571)*, P. Noriega and C. Sierra, Eds. Springer-Verlag, 1999, pp. 25–40.

[17] Plott, C. R. Laboratory experimental testbeds: Application to the PCS auction. *Journal of Economics and Management Strategy 6* (1997), 605–638.

[18] Porter, D. The effect of bid withdrawal in a multi-object auction. *The Review of Economic Design* (1998). To appear.

[19] Rassenti, S. J., Smith, V. L., and Bulfin, R. L. A combinatorial mechanism for airport time slot allocation. *Bell Journal of Economics 13* (1982), 402–417.

[20] Sandholm, T. An implementation of the Contract Net Protocol based on marginal-cost calculations. In *Proc. 11th National Conf. on Artificial Intelligence (AAAI-93)* (1993).

[21] Sandholm, T. An algorithm for optimal winner determination in combinatorial auctions. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)* (1999), pp. 542–547.

[22] Stinson, D. R. *Cryptography: Theory and Practice.* CRC Press, Inc., 1995.

[23] Varian, H., and MacKie-Mason, J. K. Generalized Vickrey auctions. Tech. rep., University of Michigan, 1995.

[24] Wellman, M. P., Walsh, W. E., Wurman, P. R., and MacKie-Mason, J. K. Some economics of market-based distributed scheduling. In *Proc. 18th Int. Conf. on Distributed Computing Systems* (1998).

[25] Wurman, P. R. *Market Structure and Multidimensional Auction Design for Computational Economies.* PhD thesis, University of Michigan, 1999.

[26] Wurman, P. R., and Wellman, M. P. Equilibrium prices in bundle auctions. In *Proc. AAAI-99 Workshop on Artificial Intelligence for Electronic Commerce* (1999), pp. 56–61.

Round	Prices							Bids			Allocation	Revenue
	A	B	C	AB	BC	AC	ABC	Agent 1	Agent 2	Agent 3		
1	0	0	0	0	0	0	0	(AB,0) (ABC,0)*	(BC,0) (ABC,0)	(AC,0) (ABC,0)	[1, ABC]	0
2	0	0	0	0	30	30	30	(AB,0) (ABC,0)	(BC,30) (ABC,30)*	(AC,30) (ABC,30)	[2, ABC]	30
3	0	0	0	30	30	60	60	(AB,30) (ABC,60)*	(BC,30) (ABC,30)	(AC,60) (ABC,60)	[1, ABC]	60
4	0	0	0	30	60	90	90	(AB,30) (ABC,60)	(BC,60) (ABC,90)*	(AC,90) (ABC,90)	[2, ABC]	90
5	0	0	0	60	60	120	120	(AB,60)	(BC,60) (ABC,90)	(C,0) (AC,120) (ABC,120)*	[3, ABC]	120
6	0	0	0	90	90	120	120	(AB,90) (ABC,120)*	(BC,90) (ABC,120)	(C,0) (AC 120) (ABC,120)	[1, ABC]	120
7	0	0	30	90	120	150	150	(AB,90) (ABC,120)	(B,0)* (BC,120) (ABC,150)	(A,0) (C,30) (AC,150)* (ABC,150)	[2, B], [3, AC]	150
8	0	0	30	120	120	150	150	(A,0) (B,0) (AB,120)* (ABC,150)	(B,0) (BC,120) (ABC,150)	(A,0) (C,30)* (AC,150) (ABC,150)	[1, AB], [3, C]	150
9	0	30	30	120	150	150	180	(A,0) (AB,120)	(B,30)* (BC,150) (ABC,180)	(A,0) (C,30) (AC,150)* (ABC,180)	[2, B], [3, AC]	180
10	30	30	30	150	150	150	180	(A,30) (AB,150)* (ABC,180)	(B,30) (BC,150) (ABC,180)	(C,30)* (AC,150) (A,30) (ABC,180)	[1, AB], [3, C]	180
11	30	60	30	150	180	150	210	(A,30) (AB,150)	(B,60)* (C,30) (BC,180) ABC(210)	(A,30) (C,30) (AC,150)*	[2, B], [3, AC]	210
12	60	60	30	180	180	150	210	(A,60)* (B,30) (C,0) (AB,180) (ABC,210)	(B,60) (C,30) (BC,180)* (ABC,210)	(A,30) (C,30) (AC,150)	[1, A], [2, BC]	240
13	60	60	60	180	180	180	210	(A,60) (B,30) (AB,180)* (ABC,210)	(B,60) (C,30) (BC,180) (ABC,210)	(A,30) (C,60)* (AC,180) (ABC,210)	[1, AB], [3, C]	240
14	60	90	60	180	210	180	240	(A,60) (AB,180)* (ABC,210)	(B,60) (C,30) (BC,180) (ABC,210)	(A,30) (C,60)* (AC,180) (ABC,210)	[1, AB], [3, C]	240
15	60	90	60	180	210	180	240	(A,60) (AB,180)* (ABC,210)	(B,60) (C,30) (BC,180) (ABC,210)	(A,30) (C,60)* (AC,180) (ABC,210)	[1, AB], [3, C]	240

Table 2: Worked example, *i*Bundle(2) on Problem 1, Bid incr. $\epsilon = 30$. Provisional allocations indicated *.

Round		Prices							Bids							Allocation	Revenue
		A	B	C	AB	BC	AC	ABC	A	B	C	AB	BC	AC	ABC		
1	Agent 1	0	0	0	0	0	0	0				0			0*	[1, ABC]	0
	Agent 2	0	0	0	0	0	0	0					0		0		
	Agent 3	0	0	0	0	0	0	0						0	0		
2	Agent 1	0	0	0	0	0	0	0				0			0	[2, ABC]	30
	Agent 2	0	0	0	0	30	0	30					30		30*		
	Agent 3	0	0	0	0	0	30	30						30	30		
3	Agent 1	0	0	0	30	0	0	30				30			30	[3, ABC]	60
	Agent 2	0	0	0	0	30	0	30					30		30		
	Agent 3	0	0	0	0	0	60	60						60	60*		
4	Agent 1	0	0	0	60	0	0	60				60			60*	[1, ABC]	60
	Agent 2	0	0	0	0	60	0	60					60		60		
	Agent 3	0	0	0	0	0	60	60						60	60		
5	Agent 1	0	0	0	60	0	0	60				60			60	[2, ABC]	90
	Agent 2	0	0	0	0	90	0	90					90		90*		
	Agent 3	0	0	0	0	0	90	90						90	90		
6	Agent 1	0	0	0	90	0	0	90				90			90	[3, ABC]	120
	Agent 2	0	0	0	0	90	0	90					90		90		
	Agent 3	0	0	0	0	0	120	120			0		0	120	120*		
7	Agent 1	0	0	0	120	0	0	120				120*			120	[1, AB], [3, C]	120
	Agent 2	0	0	0	0	120	0	120					120		120		
	Agent 3	0	0	0	0	0	120	120			0*		0	120	120		
8	Agent 1	0	0	0	120	0	0	120				120			120	[2, ABC]	150
	Agent 2	0	0	0	0	150	0	150		0	0	0	150	0	150*		
	Agent 3	0	0	0	0	0	120	120			0		0	120	120		
9	Agent 1	0	0	0	150	0	0	150	0	0		150*	0	0	150	[1, AB], [3, C]	180
	Agent 2	0	0	0	0	150	0	150		0	0	0	150	0	150		
	Agent 3	0	0	30	0	30	150	150	0		30*	0	30	150	150		
10	Agent 1	0	0	0	150	0	0	150	0	0		150*	0	0	150	[1, AB], [2, C]	180
	Agent 2	0	30	30	30	180	30	180		30	30*	30	180	30	180		
	Agent 3	0	0	30	0	30	150	150	0		30	0	30	150	150		
11	Agent 1	0	0	0	150	0	0	150	0	0		150*	0	0	150	[1, AB], [3, C]	210
	Agent 2	0	30	30	30	180	30	180		30	30	30	180	30	180		
	Agent 3	30	0	60	30	60	180	180	30		60*	30	60	180	180		
12	Agent 1	0	0	0	150	0	0	150	0	0		150	0	0	150	[2, B], [3, AC]	240
	Agent 2	0	60	60	60	210	60	210	0	60*	30	60	180	30	210		
	Agent 3	30	0	60	30	60	180	180	30		60	30	60	180*	180		
13	Agent 1	30	30	0	180	30	30	180	30	30		180*	30	30	180	[1, AB], [3, C]	240
	Agent 2	0	60	60	60	210	60	210	0	60	30	60	180	30	210		
	Agent 3	30	0	60	30	60	180	180	30		60*	30	60	180	180		
14	Agent 1	30	30	0	180	30	30	180	30	30		180*	30	30	180	[1, AB], [3, C]	240
	Agent 2	30	90	60	90	210	60	240	0	60	30	60	180	30	210		
	Agent 3	30	0	60	30	60	180	180	30		60*	30	60	180	180		

Table 3: Worked example, *i*Bundle(3) on Problem 1, Bid incr. $\epsilon = 30$. Provisional allocations indicated *.