Bid Determination in Simultaneous Auctions

Amy Greenwald

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-05-16
November 2005

Brown University Technical Report (2005, Revised 2006) CS-05-16

Bid Determination in Simultaneous Auctions

Amy Greenwald *Department of Computer Science, Brown University, Providence, RI 02912*
amy@brown.edu

Abstract

In this paper, *bid determination* problems are studied. In particular, the central problems of *allocation*, *acquisition*, *completion*, and *arbitrage* are defined. The main theoretical result states that bid determination in double-sided auctions—i.e., completion—where goods can be sold as well as bought, is polynomial-time reducible to bid determination in single-sided auctions—i.e., acquisition. Such problems are faced by autonomous agents trading in simultaneous auctions.

Regarding complexity, it is noted that allocation in simultaneous auctions is formally equivalent to *winner determination* in combinatorial auctions. Since the winner determination problem is NP-hard [8], and since completion generalizes acquisition, which in turn generalizes allocation, it follows that all three of these bid determination problems are NP-hard. Consequently, we advocate the use of heuristic search methods to solve bid determination problems.

Two algorithmic approaches to bid determination are studied here, inspired by the two top-scoring TAC–2000 agents. These alternatives are compared experimentally using data from TAC–2000. Three experimental findings are reported: (i) for the dimensions of TAC–2000, optimal solutions are tractable; (ii) optimal solutions do not scale; (iii) heuristic approximation methods scale well, achieving near optimal solutions with predictable time and space requirements.

Keywords: simultaneous auctions, trading agents, e-commerce

1. Introduction

In the design of software agents that buy and sell goods in simultaneous auctions, a variety of interesting computational questions arise. Perhaps the most fundamental is to determine how to bid on goods being auctioned off in separate markets when the agent's valuations for those goods are highly interdependent. The Trading Agent Competition (TAC) travel game was designed as a testbed in which to compare and contrast various approaches to this problem.

A TAC agent is a simulated travel agent whose task is to organize itineraries for a group of eight clients who wish to travel from TACTown to some destination and back again during a five-day period. Travel goods, such as airline tickets and hotel reservations, are complementary, and tickets to entertainment events, such as the Boston Red Sox and the Boston Symphony Orchestra, are substitutable. The agent's objective is to win those goods that best satisfy its clients' preferences as inexpensively as possible, by trading in ascending call markets, continuous double auctions, and dynamic posted-pricing environments, simultaneously.

In reasoning about trading in simultaneous auctions, software agents acting on behalf of multiple clients[1] (e.g., TAC agents) may pose and solve *bid determination* problems such as the following:

- "Given only the set of goods I already hold, how can I allocate those goods to packages for my clients so as to maximize my surplus: the sum of my clients' valuations for their respective packages?" We call this the *allocation* problem.

1. TAC is an example of the more general situation where an agent is charged with compiling bundles of goods for various *independent* uses. In TAC, uses correspond to clients and bundles correspond to travel packages. Given our focus on TAC in this paper, we refer to clients specifically rather than uses generally, although the concept applies more broadly. In addition, we adopt the terminology "package" rather than "bundle."

- "Given the set of goods I already hold, as well as market prices and supply in all open auctions, what set of additional goods should I buy to augment my current holdings so that I maximize my surplus after allocating all my goods to packages, subject to the constraint that I cannot allocate more goods than the number I already hold plus the number I buy?" In this case, surplus is defined as the sum of my clients' valuations for their respective packages less procurement costs. We call this the *acquisition* problem.

- "Given the set of goods I already hold, as well as market prices, supply, and demand in all open auctions, what set of goods should I buy and sell to adjust my current holdings so that I maximize my surplus after allocating all my goods to packages, subject to the constraint that I cannot allocate more goods than the number I hold plus the number I buy less the number I sell?" In this case, surplus is defined as the sum of my clients' valuations for their respective packages less procurement costs plus sales revenue. We call this the *completion* problem.

- "Given market prices, supply, and demand in all open auctions, what set of goods should I buy and sell so as to maximize surplus (i.e., sales revenue less procurement costs), subject to the constraint that I cannot sell more goods than I buy?" We call this the *arbitrage* problem.

We refer to acquisition as a *one-sided* (or *single-sided*) problem, since it involves decision making involving only the buy side of the market. Completion and arbitrage, in contrast, are *two-sided* (or *double-sided*) problems, addressing both buy and sell decisions. Allocation could be called "zero-sided," since it does not deal with market actions at all.

In our study of bid determination, we observe that allocation in simultaneous auctions is formally equivalent to *winner determination* in combinatorial auctions. Since the winner determination problem is NP-hard [8], and since completion generalizes acquisition, which in turn generalizes allocation, it follows that all three of these bid determination problems are NP-hard.

Approximate solutions to completion, the most general among the aforementioned bid determination problems, were essential components of the architectures of the two top-scoring agents, `ATTac`–2000 [12] and `RoxyBot`–2000 [5], in the first TAC event (TAC–2000) [11]. On the theoretical side, the main contribution of this work is to demonstrate that the completion problem, which is integral to bidding in two-sided auctions, can be formally reduced to the acquisition problem, which is integral to bidding in one-sided auctions. Indeed, such a reduction can be achieved in polynomial time. We present two constructive proofs of this observation. The first is straightforward. The second, which is inspired by the implementation of `RoxyBot`–2000, holds more generally and reduces the size of the (NP-hard) optimization problem.

On the experimental side, two algorithmic approaches to bid determination are studied here. `RoxyBot`–2000 used an A^* search algorithm to optimally solve allocation and beam search to approximately solve completion [5]. `ATTac`–2000 used integer linear programming (ILP) to solve allocation and a simplified form of completion [12]. We compared the performance of these techniques on data generated during the final round of TAC–2000, and on larger datasets built by merging TAC–2000 datasets. Our key experimental findings are as follows: (i) For the dimensions of TAC–2000, optimal solutions to the allocation problem are tractable. (ii) As the problem size increases, A^* scales poorly. ILP fares better on average, but has very high variance. (iii) Beam search scales well, achieving near optimal solutions with predictable time and space requirements.

This paper is organized as follows. In Section 2, we formally define four bid determination problems, and we note the relationship between bid determination in simultaneous auctions and winner determination in combinatorial auctions and exchanges. In addition, we present two reductions from completion to acquisition. In Section 3, we describe the techniques implemented by the two top-scoring TAC–2000 agents to solve bid determination problems, and we report on the performance of these techniques on TAC–2000 data. Details of the TAC–2000 game appear in Appendix B.

2. Theoretical Findings

We start by formally defining four bid determination problems: allocation, acquisition, completion, and arbitrage. We argue that arbitrage is "easy" (i.e., it can be solved in polynomial time), but that allocation, and hence, acquisition, which generalizes allocation, and completion, which generalizes acquisition, are NP-hard. Our main observation is that completion is polynomial-time reducible to acquisition, which implies that completion (a two-sided problem) is no harder than acquisition (a one-sided problem). Inspired by the TAC Travel game, we develop our framework for multi-unit simultaneous auctions, although our hardness results hold in the single-unit setting as well.

2.1 Formal Problem Statements

Let G denote an ordered set of n distinct goods. In addition, let $N \in \mathbb{N}^n$ represent the multiset of these goods in the marketplace, with N_g denoting the number of copies of each good $g \in G$. A *package* M is any collection of goods, technically a "submultiset" of N, written $M \subseteq N$. In general, for packages $A, B \subseteq N$, we write $A \subseteq B$ whenever $A_g \leq B_g$ for all goods $g \in G$. Also, we write $g.k \in M$ whenever $k \leq M_g$ for good $g \in G$. Finally, we let $\emptyset \subseteq N$ denote a vector of $|G|$ zeros.

Given $A, B \subseteq N$, we rely on the following basic operations: for all $g \in G$,

- $(A \oplus B)_g \equiv A_g + B_g$
- $(A \setminus B)_g \equiv A_g - B_g$

For example, if $G = \{\alpha, \beta, \gamma\}$ and $N = \langle 1, 2, 3\rangle$, then $A = \langle 0, 1, 2\rangle \subseteq N$ and $B = \langle 1, 1, 1\rangle \subseteq N$, but neither $A \subseteq B$ (because $A_\gamma > B_\gamma$) nor $B \subseteq A$ (because $B_\alpha > A_\alpha$). Moreover, $(A \oplus B)_\alpha = 1$, $(A \oplus B)_\beta = 2$, and $(A \oplus B)_\gamma = 3$, whereas $(A \setminus B)_\alpha = -1$, $(A \setminus B)_\beta = 0$, and $(A \setminus B)_\gamma = 1$.

It is instructive to interpret this notation in the TAC Travel domain. In TAC Travel, the flights, hotel rooms, and entertainment events up for auction comprise an ordered set of 28 distinct goods. In principle, the multiset of goods in the marketplace is:

$$N^{\text{TAC}} = \langle \underbrace{\infty, \ldots, \infty}_{\text{8 flights}}, \underbrace{16, \ldots, 16}_{\text{8 hotels}}, \underbrace{8, \ldots, 8}_{\text{12 events}} \rangle \in \mathbb{N}^{28} \tag{1}$$

In practice, however, since each agent works to satisfy the preferences of only eight clients, it suffices to consider the multiset of goods:

$$N^{\text{TAC8}} = \langle \underbrace{8 \ldots, 8}_{\text{8 flights}}, \underbrace{8, \ldots, 8}_{\text{8 hotels}}, \underbrace{8, \ldots, 8}_{\text{12 events}} \rangle \subseteq N^{\text{TAC}} \tag{2}$$

A TAC travel package is a submultiset $M \subseteq N^{\text{TAC8}}$ that satisfies the game's feasibility constraints.

As in TAC Travel, we assume the agent is charged with compiling packages for a set of, say m, clients. Each client i is characterized by a value function $v_i : \mathcal{N} \to \mathbb{R}$, which attributes value (or cost) to packages (i.e., submultisets). (Note that the agent could encode its own preferences in an additional value function, say $v_0 : \mathcal{N} \to \mathbb{R}$.) Here $\mathcal{N}$ denotes the set of all submultisets of N.

A value function $v : \mathcal{N} \to \mathbb{R}$ exhibits *free disposal* if and only if $v(M_1) \leq v(M_2)$ whenever $M_1 \subseteq M_2$. Although we do not assume this condition everywhere, we note that in TAC Travel, free disposal holds for both the agent and its clients. This assumption also holds in most real-world travel markets (e.g., train tickets, car rentals), and applies in many other domains as well.

Because we assume the clients' interests are *independent*, the agent's surplus measures take special form in bid determination. In particular, an agent's valuation for a collection of packages is the sum of its respective clients' package-values: i.e., given a vector of packages $\vec{X} = (X_1, \ldots, X_m)$,

$$\text{Valuation}(\vec{X}) = \sum_{i=1}^{m} v_i(X_i) \tag{3}$$

Throughout this section, we assume a fixed set of goods G, a fixed multiset $N \in \mathbb{N}^{|G|}$ of goods in the marketplace, and a fixed collection of value functions $v_1, \ldots, v_m$, each of which assigns a (possibly negative) $v_i(M) \in \mathbb{R}$ to $M \subseteq N$ representing the worth of that package.

In our formal problem statements, we list all arguments explicitly. In defining the values of solutions to bid determination problems, we elide arguments that are clear from context, particularly the set G and the multiset N referring to the fixed goods in the marketplace.

2.1.1 ALLOCATION

Given an agent's holdings, represented as a submultiset $H \subseteq N$ of the goods in the marketplace, the allocation problem is to arrange those holdings into a surplus-maximizing collection of packages for the agent's clients. In this problem (only), surplus is equated with valuation, defined as the sum of the agent's respective client's package-values.

Definition 2.1 Allocation$(G, N, \vec{v}, H)$.

Inputs: a set of goods G;
a multiset of goods N;
a vector $\vec{v}$ of value functions;
a multiset of goods $H \subseteq N$.
Output: a vector $\vec{X}$ of packages to allocate.

$$\text{ALL}(\vec{v}, H) = \max_{\bigoplus_{i=1}^{m} X_i \subseteq H} \text{Valuation}(\vec{X}) \tag{4}$$

In our definitions of bid determination, if the *free disposal* assumption holds for all client value functions, then without loss of generality, we can assume the constraints are tight: e.g., in allocation, $\bigoplus_{i=1}^{m} X_i = H$. In general, the loose constraints implicitly encode the assumption that the agent's own value function $v_0 = 0$.[2]

The *single-unit* allocation problem, in which H is a set (i.e., all entries in H are either 0 or 1), is precisely the winner determination problem in combinatorial auctions. As winner determination in combinatorial auctions is equivalent to weighted set-packing [8], and hence, NP-hard, allocation is also NP-hard.

2.1.2 ACQUISITION

Acquisition generalizes allocation. In acquisition, for each good g, in addition to an agent's current holdings of g, we are given the market supply of g, and the prices of acquiring additional copies of g. We represent this information by a single vector $\vec{p}_g \in \mathbb{R}^{N_g}$, which we call a *buyer priceline*: the kth component, p_{gk}, stores the cost an agent would incur upon acquiring the kth copy of g. We assume buyer pricelines are nondecreasing.

For example, if an agent currently holds four copies of a good $\tilde{g}$, and if four additional copies of $\tilde{g}$ are on the market at costs of \$25, \$40, \$65, and \$100, then the corresponding buyer priceline is given by $\vec{p}_{\tilde{g}} = \langle 0, 0, 0, 0, 25, 40, 65, 100, \infty, \infty, \ldots \rangle$. The leading zeroes indicate that the four goods the agent holds may be "acquired" at no cost. The tail of infinite costs indicates that the market supply consists of only eight copies of $\tilde{g}$.

In general, inventory costs need not be zero. We represent this possibility by using buyer pricelines in which even the goods that an agent holds are acquired for some strictly positive cost: e.g., $\vec{p}_{\tilde{g}} = \langle 5, 5, 5, 5, 25, 40, 65, 100, \infty, \infty, \ldots \rangle$. Goods can also be acquired at negative cost. If an agent acquires a good with a negative cost, in effect, it earns revenue from that acquisition. For example, sanitation companies are paid to collect waste.

2. The problems are defined this way because of the TAC motivation; this assumption holds true in TAC. However, the results in this paper hold for arbitrary agent value functions and $\text{Valuation}(X_0, \ldots, X_m) = \sum_{i=0}^{m} v_i(X_i)$.

Given a set of buyer pricelines $P = \{\vec{p}_g \mid g \in G\}$, we define the cost of the goods in multiset $Y \subseteq N$ as follows:

$$\forall g, \quad \text{Cost}_g(Y, P) = \sum_{n=1}^{Y_g} p_{gk} \tag{5}$$

$$\text{Cost}(Y, P) = \sum_{g \in G} \text{Cost}_g(Y, P) \tag{6}$$

We can now formally define acquisition. In acquisition, an agent's goal is to determine what set of goods it should buy so that the feasible allocation of its goods to packages maximizes its surplus: the valuation of the allocation less the cost of the goods. Here, an allocation is feasible if it allocates no more than the number of goods the agent buys.

Definition 2.2 Acquisition$(G, N, \vec{v}, P)$.

Inputs: a set of goods G;
a multiset of goods N;
a vector $\vec{v}$ of value functions;
a set of buyer pricelines P.
Output: a vector $\vec{X}$ of packages to allocate;
a multiset of goods $Y \subseteq N$ to buy.

$$\text{ACQ}(\vec{v}, P) = \max_{\bigoplus_{i=1}^{m} X_i \subseteq Y \subseteq N} \Big(\text{Valuation}(\vec{X}) - \text{Cost}(Y, P)\Big) \tag{7}$$

In the special case where each buyer priceline is constant, so that marginal costs do not increase or decrease as the agent buys additional copies of goods (i.e., there are no economies or diseconomies of scale), buy prices are said to be *linear*. With linear prices (specifically, constant buyer pricelines as far out as the agent would ever demand), acquisition is not necessarily an NP-hard problem. On the contrary, acquisition decomposes into a set of smaller subproblems. It suffices for the agent to acquire an optimal package for each client in turn.

In the above formulation of the acquisition problem, it is not immediately obvious that acquisition generalizes allocation; in particular, the agent's holdings $H \subseteq N$ are not explicitly represented. However, allocation can be seen as the special case of acquisition in which no goods can be bought and inventory costs are zero: that is, all entries in the buyer pricelines are either 0 (indicating holdings) or ∞ (indicating no further buying opportunities).

Another way to see that acquisition generalizes allocation is to observe that the (single-unit) acquisition problem in simultaneous auctions is an instance of the winner determination problem with reserve prices in combinatorial auctions, a generalization of the winner determination problem (without reserve prices), which we previously noted is equivalent to (single-unit) allocation.

2.1.3 Completion

Completion generalizes acquisition. In completion, in addition to the inputs to acquisition, we are given the market demand for g, and the revenue that could be achieved by selling each copy of g. We represent this information by a single vector $\vec{\pi}_g \in \mathbb{R}^{N_g}$, which we call a *seller priceline*. Much like a buyer priceline, the kth component of a seller priceline stores the *revenue* that an agent could earn by selling the kth copy of g. We assume seller pricelines are nonincreasing.

For example, if the market demands six copies of good $\tilde{g}$, which can be sold at prices of \$20, \$15, \$10, \$5, −\$1, −\$50, then the corresponding seller priceline is given by $\vec{\pi}_{\tilde{g}} = \langle 20, 15, 10, 5, -1, -50, -\infty, -\infty \ldots \rangle$. Analogously to buyer pricelines, the tail of infinitely negative revenues indicates

that the market demands only six copies of $\tilde{g}$. The finite, yet negative, entries indicate that the corresponding copies of $\tilde{g}$ cannot be disposed of for free (e.g., toxic waste).

Given a set of seller pricelines $\Pi = \{\vec{\pi}_g \mid g \in G\}$, we define the revenue associated with multiset $Z \subseteq N$ as follows:

$$\forall g, \quad \text{Revenue}_g(Z, \Pi) = \sum_{n=1}^{Z_g} \pi_{gk} \tag{8}$$

$$\text{Revenue}(Z, \Pi) = \sum_{g \in G} \text{Revenue}_g(Z, \Pi) \tag{9}$$

We can now formally define completion. In completion, an agent's goal is to determine what set of goods it should buy and what set of goods it should sell so that the feasible allocation of its goods to packages maximizes it surplus: the valuation of the allocation less the cost of the goods plus any revenue earned from sales. Here, an allocation is feasible if it allocates no more than the number of goods the agent buys less the number the agent sells.

Definition 2.3 Completion$(G, N, \vec{v}, P, \Pi)$.

Inputs: a set of goods G;
a multiset of goods N;
a vector $\vec{v}$ of value functions;
a set of buyer pricelines P;
a set of seller pricelines Π.
Output: a vector $\vec{X}$ of packages to allocate;
a multiset of goods $Y \subseteq N$ to buy;
a multiset of goods $Z \subseteq N$ to sell.

$$\text{COM}(\vec{v}, P, \Pi) = \max_{\bigoplus_{i=1}^{m} X_i \oplus Z \subseteq Y \subseteq N} \Big(\text{Valuation}(\vec{X}) - \text{Cost}(Y, P) + \text{Revenue}(Z, \Pi) \Big) \tag{10}$$

2.1.4 Arbitrage

Arbitrage is an opportunity for an agent to simultaneously buy and sell goods and instantaneously profit from any price discrepancies. Given the buyer priceline $\vec{p}_{\tilde{g}} = \langle 1, 50, 100, 5000, \infty, \infty, \ldots \rangle$ and the seller priceline $\vec{\pi}_{\tilde{g}} = \langle 40, 30, 20, 10, -\infty, -\infty, \ldots \rangle$, for example, an agent can both buy and sell one copy of $\tilde{g}$ and turn an immediate profit of $\$40 - \$1 = \$39$.

In the arbitrage problem, an agent's goal is to determine what set of goods to buy and sell, if any, strictly for profit, not for allocation. Specifically, an agent seeks a set of goods for which it can obtain a level of revenue, as prescribed by the seller pricelines, that exceeds the cost of buying, as prescribed by the buyer pricelines. Formally,

Definition 2.4 Arbitrage(G, N, P, Π).

Inputs: a set of goods G;
a multiset of goods N;
a set of buyer pricelines P;
a set of seller pricelines Π.
Output: a multiset of goods $Y \subseteq N$ to buy;
a multiset of goods $Z \subseteq N$ to sell.

$$\text{ARB}(P, \Pi) = \max_{Z \subseteq Y \subseteq N} \left(\text{Revenue}(Z, \Pi) - \text{Cost}(Y, P) \right) \tag{11}$$

Given sorted buyer and seller pricelines, in nondecreasing and nonincreasing orders respectively, a solution to the arbitrage problem is straightforward: for all goods $g \in G$, buy the kth unit if and only if $\pi_{gk} \geq p_{gk}$ or $0 \geq p_{gk}$, and sell the kth unit if and only if $\pi_{gk} \geq p_{gk}$ or $\pi_{gk} \geq 0$. If we assume that at least one of either buy prices or sell prices are nonnegative, then there is really only one output of the arbitrage problem, namely, the multiset A of arbitrage opportunities where $g.k \in A$ if and only if $\pi_{gk} \geq p_{gk}$ (see Algorithm 1). Indeed, by running Algorithm 1, we observe that the arbitrage problem can be solved in time linear in the size of the pricelines. The complexity of completion derives from the fact that it generalizes acquisition, which generalizes allocation (an NP-hard problem), not from the fact that it generalizes arbitrage.

Algorithm 1 Arbitrage(G, N, P, Π)

```
A = ∅
for all g ∈ G do
    for k = 1 ... N_g do
        if π_gk ≥ p_gk then
            increment A_g
        end if
    end for
end for
return A
```

This concludes our definitions of bid determination problems. Next, we proceed to study two reductions from completion to acquisition. For ease of exposition in the second reduction, we assume that either buy or sell prices are nonnegative, so that the arbitrage problem can be solved using Algorithm 1, and that the value of the solution depends on the algorithm's output as follows:

$$\mathrm{ARB}(P, \Pi) = \sum_{g \in G} \mathrm{Arbitrage}_g(P, \Pi) \tag{12}$$

where

$$\mathrm{Arbitrage}_g(P, \Pi) = \sum_{n=1}^{A_g} \pi_{gk} - p_{gk} \tag{13}$$

Note that the second reduction holds more generally, that is, even for real-valued buy and sell prices.

2.2 Reductions

Perhaps surprisingly, the completion problem for double-sided simultaneous auctions reduces to the acquisition problem for single-sided auctions. Indeed, we present *two* reductions from completion to acquisition in this paper: one in Section 2.2.1, which is straightforward, and another, in Section 2.2.2, which holds more generally (see Appendix A) and reduces the size of the (NP-hard) optimization problem. Via either reduction, we can solve a completion problem by instead solving an acquisition problem, and then mapping an optimal acquisition back into an optimal completion.

2.2.1 First Reduction

In our first reduction from completion to acquisition, we encode the function Revenue(Z, Π) as a dummy client's value function. Specifically, to solve the problem Completion$(G, N, \vec{v}, P, \Pi)$ with m clients, we pose the problem Acquisition$(G, N, v_1, \ldots, v_m, v_{m+1}, P)$ with $m + 1$ clients, where $v_{m+1} = \mathrm{Revenue}(Z, \Pi)$ for all $Z \subseteq N$. After solving this problem, we map the optimal acquisition $(X'_1, \ldots, X'_m, X'_{m+1}, Y')$ back into an optimal completion $(X_1, \ldots, X_m, Y, Z)$ as follows: $X_i = X'_i$, for all $i = 1, \ldots, m$, $Y = Y'$, and $Z = X'_{m+1}$. Among the set of goods that are allocated to packages

for clients in acquisition, those which are allocated to the dummy client are sold in completion. The set of goods that is bought in acquisition is precisely the set of goods that is bought in completion.

Theorem 2.5 *Given a completion problem* Completion$(G, N, v_1, \ldots, v_m, P, \Pi)$ *with m clients, we define an acquisition problem* Acquisition$(G, N, v_1, \ldots, v_m, v_{m+1}, P)$ *with $m+1$ clients as follows: for all $Z \subseteq N$, let $v_{m+1}(Z) =$* Revenue(Z, Π)*. An optimal solution $(X'_1, \ldots, X'_m, X'_{m+1}, Y')$ to the $m+1$-client acquisition problem* Acquisition$(G, N, v_1, \ldots, v_m, v_{m+1}, P)$ *maps into an optimal solution $(X'_1, \ldots, X'_m, Y', X'_{m+1})$ to the m-client completion problem* Completion$(G, N, v_1, \ldots, v_m, P, \Pi)$*.*

2.2.2 Second Reduction

In our first reduction from completion to acquisition, revenue is treated as an additional source of valuation. The same set of goods is bought in both acquisition and completion, while the goods that are allocated to a dummy client in acquisition are sold in completion. This reduction does not change the problem size. We now describe a second reduction from completion to acquisition which explicitly handles arbitrage by folding the seller pricelines into the buyer pricelines, thereby reducing the problem size. The intuition here is that in double-sided auctions, an agent cannot simply allocate the goods it buys to packages for its clients without weighing into its decisions the opportunity costs of not selling those goods on the open market. The "second" acquisition problem (i.e., that which arises via this second reduction), which consolidates the buyer and seller pricelines, is smaller than the original completion problem, and hence smaller than the "first" acquisition problem (i.e., that which arises via our first reduction).

In our second reduction, the value of an optimal completion differs from the value of an optimal acquisition by a constant value $C = \text{ARB}(G, N, P, \Pi)$. By adding this constant C to the acquisition value, an agent implicitly takes advantage of all arbitrage opportunities, even though the agent cannot explicitly sell goods in acquisition. Since it is assumed that all arbitrage opportunities are exploited, so that the kth copy of a good g is bought and "sold" whenever $\pi_{gk} \geq p_{gk}$, the only decision an agent faces in acquisition regarding such goods is to perhaps allocate them to packages for its clients, overriding the implicit "sell" decision. Accordingly, the buyer and seller pricelines are "unified" so that the agent incurs cost $\max\{\pi_{gk}, p_{gk}\}$. In effect, the agent pays an opportunity cost π_{gk} to allocate an arbitrage opportunity $g.k$ to a package for a client rather than "sell" it; otherwise, it pays p_{gk} to allocate a non-arbitrage opportunity $g.k$ as usual. Note that the prices of goods for which there exist arbitrage opportunities in completion are irrelevant to decision-making in acquisition under this reduction.

Combining the buyer priceline $\vec{p}_g$ and seller priceline $\vec{\pi}_g$ for each good g proceeds in several steps. First, an intermediate priceline $\vec{q}_g$ is formed with $q_{gk} = \max\{\pi_{gk}, p_{gk}\}$. For all arbitrage opportunities, $q_{gk} = \pi_{gk}$; otherwise, $q_{gk} = p_{gk}$. Next, $\vec{q}_g$ is sorted in nondecreasing order. Formally, the reordering of $\vec{q}_g$ is achieved by applying a *permutation* σ, that is, a bijection on $\{1, \ldots, N_g\}$. Finally, we define $\vec{p}'_g$ to be $\vec{q}_g$ permuted according to σ. We call the output of this procedure *unified pricelines*, the set of which we denote by P'. Note that these operations can be carried out in polynomial time (see Algorithm 2). Specifically, the time complexity of Unify(G, N, P, Π) (and of this reduction overall) is $O(|G|K \log K)$, where $K = \max_g N_g$.

For example, say there are four copies of some good $\tilde{g}$ on the market with buyer priceline $\vec{p}_{\tilde{g}} = \langle 0, 1, 10, 30 \rangle$ and seller priceline $\vec{\pi}_{\tilde{g}} = \langle 40, 20, 0, 0 \rangle$. First, the arbitrage opportunities are identified: $\tilde{g}.1$ can be bought for 0 and sold for 40; $\tilde{g}.2$ can be bought for 1 and sold for 20; there are no further arbitrage opportunities. Next, the intermediate priceline $\vec{q}_{\tilde{g}} = \langle 40, 20, 10, 30 \rangle$ is formed, reflecting these arbitrage opportunities. Finally, $\vec{q}_{\tilde{g}}$ is sorted so that $\vec{p}'_{\tilde{g}} = \langle 10, 20, 30, 40 \rangle$. In particular, $\sigma_{\tilde{g}} = \{1 \mapsto 4, 2 \mapsto 2, 3 \mapsto 1, 4 \mapsto 3\}$. We interpret the unified priceline as follows. The agent can allocate $\tilde{g}.3$ and $\tilde{g}.4$ at prices determined by the buyer priceline, specifically $p'_{\tilde{g}1} = q_{\tilde{g}3} = \max\{p_{\tilde{g}3}, \pi_{\tilde{g}3}\} = p_{\tilde{g}3} = 10$; similarly, $p'_{\tilde{g}3} = 30$. But if it chooses to allocate $\tilde{g}.1$ or $\tilde{g}.2$, it must "pay back" the opportunity cost that is automatically added to the objective value in acquisition within

Algorithm 2 Unify(G, N, P, Π)

for all $g \in G$ **do**
  for $k = 1 \ldots N_g$ **do**
    let $q_{gk} = \max\{\pi_{gk}, p_{gk}\}$
  end for
  find a permutation σ that sorts the entries in $\vec{q}_g$ in nondecreasing order
  {e.g., $\sigma = \{1 \mapsto 2, 2 \mapsto 1\}$ sorts $\langle 20, 10\rangle$ accordingly}
  for $k = 1 \ldots N_g$ **do**
    let $p'_{g\sigma_g(k)} = q_{gk}$
  end for
end for
return P'

the constant C. For $\tilde{g}.1$, the agent charges itself $p'_{\tilde{g}4} = q_{\tilde{g}1} = \max\{p_{\tilde{g}1}, \pi_{\tilde{g}1}\} = \pi_{\tilde{g}1} = 40$, its worth on the open market; similarly, for $\tilde{g}.2$, it charges itself $p'_{\tilde{g}2} = 20$.

After solving the acquisition problem with unified pricelines, an optimal acquisition $(\vec{X}', Y')$ maps (under h) into an optimal solution to the completion problem as follows. First, $\vec{X} = \vec{X}'$, so that the allocation of goods to packages is identical in completion and acquisition. Second, for all copies of all goods, $g.k$ is bought in completion if it is an arbitrage opportunity *or* bought in acquisition; $g.k$ is sold in completion if it is an arbitrage opportunity *and not* bought in acquisition. Moreover, an optimal completion $(\vec{X}, Y, Z)$ maps (under i) into an optimal solution to the acquisition problem with unified pricelines as follows. First, $\vec{X}' = \vec{X}$, so that the allocation of goods to packages remains identical in completion and acquisition. Second, $g.k$ is bought in acquisition if it is bought in completion *but not* sold. More precisely, $Y' = Y \setminus Z$.

It is straightforward to define the mapping i: $i(\vec{X}, Y, Z) = (\vec{X}, Y \setminus Z)$, for all completions $(\vec{X}, Y, Z)$. Algorithm 3 can be used to implement the mapping h. To explain the workings of this algorithm, we run through an example. Our example involves only a single good $\tilde{g}$ in the marketplace, of which there exist many copies.

Example 2.6 Let us assume a single client whose value function is linear: in particular, let $v_{\tilde{g}}(k) = 10k$. Also, let $\vec{p}_{\tilde{g}} = \langle 1, 2, 3, 6, 8, 10, \infty, \ldots\rangle$ and $\vec{\pi}_{\tilde{g}} = \langle 11, 9, 7, 0, \ldots\rangle$, so that $A_{\tilde{g}} = 3$, since $11 > 1$, $9 > 2$, and $7 > 3$. The Unify algorithm returns $\vec{p}'_{\tilde{g}} = \langle 6, 7, 8, 9, 10, 11, \infty, \ldots\rangle$. (The Unify algorithm also implicitly computes the permutation $\sigma_{\tilde{g}} = \{1 \mapsto 6, 2 \mapsto 4, 3 \mapsto 2, 4 \mapsto 1, 5 \mapsto 3, 6 \mapsto 5\}$.)

In this setup, $Y'_{\tilde{g}} = 5$ in the optimal acquisition. To compute the corresponding values of $Y_{\tilde{g}}$ and $Z_{\tilde{g}}$ in the optimal completion, we invoke Algorithm 3. First, both values are initialized to the value of $A_{\tilde{g}}$. Next, running through the first $Y'_{\tilde{g}}$ entries in $\vec{p}'_{\tilde{g}}$, we find (i) three of these entries correspond to units that were not arbitrage opportunities (the first, third, and fifth, which correspond to $g.4$, $g.5$, and $g.6$), so the value of $Y_{\tilde{g}}$ is incremented by 3; and (ii) two of these entries correspond to units that were arbitrage opportunities (the second and fourth, which correspond to $g.3$ and $g.2$), so the value of $Z_{\tilde{g}}$ is decremented by 2. Ultimately, $Y_{\tilde{g}} = 6$ and $Z_{\tilde{g}} = 1$ in the corresponding optimal completion. Note that $Y'_{\tilde{g}} = (Y \setminus Z)_{\tilde{g}}$. □

Theorem 2.7 *Given a completion problem* Completion$(G, N, \vec{v}, P, \Pi)$*, consider the acquisition problem* Acquisition$(G, N, \vec{v}, P')$ *with unified pricelines P'. Algorithm 3 (the mapping h) returns an optimal solution to this completion problem, given an optimal solution to this acquisition problem. Conversely, the mapping i returns an optimal solution to this acquisition problem, given an optimal solution to this completion problem.*

Implicit in the first reduction was a bijective mapping from feasible solutions to acquisition into feasible solutions to completion. In this second reduction, however, h is not invertible because it

Algorithm 3 $h(\vec{X}', Y', P', \sigma)$

```
1: X⃗ = X⃗'
2: Y = A
3: Z = A
4: for all g ∈ G do
5:    for k = 1 ... Y'_g do
6:       if g.σ_g^{-1}(k) ∉ A then
7:          increment Y_g
8:       end if
9:       if g.σ_g^{-1}(k) ∈ A then
10:         decrement Z_g
11:      end if
12:   end for
13: end for
14: return (X⃗, Y, Z)
```

is not onto. In particular, it fails to hit those (clearly suboptimal) feasible completions in which $A \subset Z$ (non-arbitrage opportunities are sold) or $Y \subset A$ (not all arbitrage opportunities are bought). Consequently, the proof of Theorem 2.7 is not straightforward, and is relegated to Appendix A.

2.3 Winner Determination Problems

Recall that the (single-unit) allocation problem in simultaneous auctions is equivalent to the winner determination problem in combinatorial auctions. In this section, we interpret acquisition and completion as special cases of the winner determination problem in combinatorial exchanges.

In a combinatorial exchange (see, for example, Sandholm *et al.* [10]), an auctioneer receives m buy offers and n sell offers for packages of goods. Formally, buy offers are functions $v_i : \mathcal{N} \to \mathbb{R}_+$, whereas sell offers are functions $u_i : \mathcal{N} \to \mathbb{R}_-$.

The auctioneer's valuation function expresses the value to the auctioneer of the allocation $\vec{X} = (X_1, \ldots, X_m)$ of length m, which allocates package X_i to buy offer $i \in \{1, \ldots, m\}$. This function is additive:

$$\text{Valuation}(\vec{X}) = \sum_{i=1}^{m} v_i(X_i) \tag{14}$$

Analogous to valuation, we define a "de"valuation function. The devaluation function describes the cost to the auctioneer of the "de"allocation $\vec{Y} = (Y_1, \ldots, Y_n)$ of length n, which deallocates package Y_j from sell offer $j \in \{1, \ldots, n\}$. This function is also additive:

$$\text{DeValuation}(\vec{Y}) = \sum_{i=1}^{n} u_i(Y_i) \tag{15}$$

Armed with this notation, we define the winner determination problem for combinatorial exchanges, with m-buy and n-sell offers:

Definition 2.8 Exchange$(G, N, \vec{v}, \vec{u})$.

Inputs: a set of goods G;
a multiset of goods N;
a vector $\vec{v}$ of buy offers;
a vector $\vec{u}$ of sell offers.

Output: a vector $\vec{X}$ of packages to allocate;
a vector $\vec{Y}$ of packages to deallocate.

$$\mathrm{EX}(\vec{v},\vec{u}) = \max_{\bigoplus_{i=1}^{m} X_i \subseteq \bigoplus_{j=1}^{n} Y_j} \mathrm{Valuation}(\vec{X}) - \mathrm{DeValuation}(\vec{Y}) \tag{16}$$

Both acquisition and completion can be viewed as special cases of this general combinatorial exchange problem. Acquisition is the special case in which there is exactly one sell offer, which can be expressed compactly, because it is additive. Specifically, $u(Y) = \mathrm{Cost}(Y,P)$, so that $\mathrm{Acquisition}(G,N,\vec{v},P) = \mathrm{Exchange}(G,N,\vec{v},\mathrm{Cost}(\cdot,P))$.

In completion, in addition to the select sell offer in acquisition, there is one additional buy offer, which can also be expressed compactly, because it, too, is additive. Specifically, $v_{m+1}(Z) = \mathrm{Revenue}(Z,\Pi)$. Since completion reduces to acquisition, it follows that completion is also an instance of the combinatorial exchange problem. Via the first reduction, for example,

$$\begin{aligned}
& \mathrm{Completion}(G,N,v_1,\ldots,v_m,P,\Pi) && (17)\\
= \;& \mathrm{Acquisition}(G,N,v_1,\ldots,v_m,v_{m+1},P) && (18)\\
= \;& \mathrm{Exchange}(G,N,\underbrace{v_1,\ldots,v_m,\mathrm{Revenue}(\cdot,\Pi)}_{m+1 \text{ buy offers}},\underbrace{\mathrm{Cost}(\cdot,P)}_{1 \text{ sell offer}}) && (19)
\end{aligned}$$

In summary, acquisition with m clients is an m-buy, 1-sell combinatorial exchange problem, and completion is an $m+1$-buy, 1-sell combinatorial exchange problem. While the first reduction yields an $m+1$-buy, 1-sell acquisition problem (the problem size is unchanged), the second produces only an m-buy, 1-sell acquisition problem.

2.4 Summary of Theoretical Findings

In this section, we formally defined four bid determination problems for simultaneous auctions, and related them to classic winner determination problems for combinatorial auctions and exchanges. In addition, we argued that completion, which is fundamental to trading strategies in double-sided auctions, reduces to acquisition, which is fundamental to trading strategies in single-sided auctions.

In the next section, we describe two algorithmic methods that were used to solve completion in TAC–2000: one implemented in `ATTac`–2000, and another implemented in `RoxyBot`–2000. For both techniques, we observe that the second reduction, albeit more complex than the first, yields smaller acquisition problems, thus more efficient solutions to completion.

3. Experimental Results

In this section, we describe experiments that we conducted with algorithmic approaches to solving instances of the bid determination problems (specifically, allocation) that arose in the context of the TAC–2000 Travel game. We investigate two classes of techniques: heuristic search and integer linear programming. Although we exploit some of the unique structure of TAC–2000,[3] we believe that our approaches are sufficiently general to transfer to other practical problems in the realm of bid determination. Already, similar techniques have been applied in the domain of TAC Supply Chain Management (e.g., [2, 7]). We view the Trading Agent Competition as a source of interesting, large-scale datasets for evaluating bid (and winner) determination algorithms.

3.1 Heuristic Solutions

Solutions to the bid determination problems of allocation, acquisition, and completion in the realm of TAC–2000 involve assigning one (possibly null) travel package and one (possibly null) entertainment

3. See Appendix B for a detailed description of TAC–2000.

package to each of say I clients, and selling all other goods. This structure suggests formulating the problem as search through a tree of depth $2I + 1$. Search begins at the top of the tree with the set of goods that is available to be traded as specified by the buyer and seller pricelines. At each of the top $2I$ levels in the tree, a package formed from the remaining goods is allocated to a client, and the costs indicated by the buyer pricelines are incurred. (In allocation, the prices of all holdings are zero.) At the bottom level, the agent sells goods to the market, and rather than incurring costs, it earns revenues indicated by the seller pricelines.

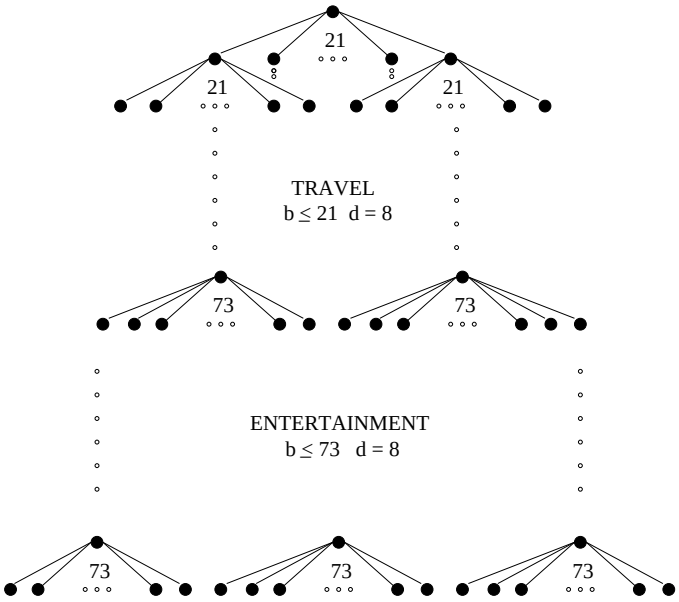

Figure 1: Heuristic Search Tree.

In recasting the completion problem as the first acquisition problem, the market is viewed as an additional client. Instead of selling any unassigned goods, these goods are assigned to the open market. In recasting the completion problem as the second acquisition problem, this additional client is folded into the buyer pricelines along with the seller pricelines. Consequently, the depth of the search tree is reduced from $2I + 1$ levels to $2I$ levels, resulting in as many as b^{2I+1} fewer search nodes, where b is the branching factor. In accordance with the TAC–2000 specification, the maximum branching factor is 21 in the upper half of our search tree and 73 in the lower half (see Figure 1). The constraints on package availability and compatibility eliminate many branches at lower levels of the tree, but the full space of legal assignments is enormous nonetheless.

To tame the search space, we applied two varieties of heuristic AI search: optimal (A^* search) and approximate (beam search). Our A^* algorithm makes use of an intricate series of admissible heuristics, which we defer to a companion paper for lack of space [5]. Our beam search, however, which employs only one simple heuristic $f(x)$ to evaluate the promise of an intermediate search node x, proved to be extremely effective and scalable in practice. Recall that in beam search, search proceeds level by level with no backtracking; at each level, only the top B nodes according to the heuristic are expanded. The parameter B is called the "beam width." Since our search tree is of fixed depth $2I$, beam search has the desirable property that it expands no more than $2IB$ nodes in total. Space and time requirements are therefore predictable.

Our heuristic $f(x)$ is inspired by the "rollout methods" that have been used in game-tree search (*e.g.*, [1, 13]). At each node x, we run a greedy algorithm to complete the allocation from x down to the bottom of the tree. Specifically, for a client that has thus far been assigned neither a travel

package nor an entertainment package, he is assigned the travel and entertainment packages that (using only goods not already assigned) jointly maximize valuation minus cost; for a client that already has a travel package, he is given the best entertainment package. The time to compute $f(x)$ is linear in the tree depth I. Therefore, the overall runtime of our beam search algorithm is quadratic in I. Since package assignments are made without replacement, this heuristic is inadmissible. It is, however, quite accurate, as we show in the next section.

3.2 Experiments

We now present experimental results on bid determination, comparing heuristic search and integer linear programming (ILP). (Our integer linear programs are described in Appendix C. An ILP solution to completion posed as the second acquisition problem is described. An ILP solution to the first acquisition problem would require additional variables and constraints pertaining to the selling of goods by the agent.) In our experiments, we focused on allocation, since there is an objective testbed for allocation problems: the 16 games of the TAC–2000 finals. As there are eight agents per game, these games comprise client preferences and winnings for 128 agents.

In our first series of tests we verified that the optimal utility values output by `RoxyBot`–2000's A^* algorithm were consistent with those of the ILP on our 128 datapoints. `RoxyBot`–2000 (which was designed to report optimal allocations within the regulation four minutes) achieved a median run time of 0.59 seconds on a 600 MHz linux PC. Using CPLEX 6.5.3 on a 400 MHz SPARCstation with 2Gb of RAM, the ILP's median run time was only 0.02 seconds. This series of tests confirmed what some TAC–2000 participants had already learned: optimal solutions to bid determination problems are tractable for the dimensions of TAC–2000.

Next, we experimented with scalability. We produced instances of the allocation problem with 16, 32, and 64 clients by concatenating, for each game, first two, then four, and finally all eight agents' datapoints. Our complete dataset included 240 allocation instances: 64 16-client cases, 32 32-client cases, and 16 64-client cases in addition to the original 128 8-client cases. While A^* did not scale even to the 16-client cases, the ILP fared much better, solving all but one of the 64-client cases. On the problematic case, however, the machine exhausted its 2Gb of RAM after six hours and aborted. The ILP also struggled with several other cases, spending five hours on one and over an hour on two others. Figure 2 summarizes the ILP's performance as a function of problem size. Each datapoint is graphed as a box plot reflecting the minimum, 25th percentile, median, 75th percentile, and maximum value of the running time. The ILP's running time can be characterized as fast on average, for the dimensions of TAC–2000, but with very high-variance.

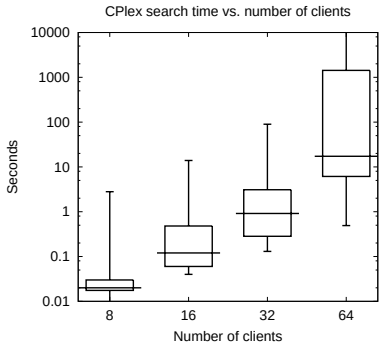

Figure 2: The ILP's running time on different problem sizes. (Note the log scale of the y-axis.)

By contrast, beam search scales predictably: its running time is quadratic in the number of clients and linear in the beam width. We therefore designed experiments to test its degree of suboptimality.

We ran a series of tests with beam widths varying from 1 to 1280 on all our instances except the one whose optimal solution was unknown (because the ILP was unable to solve it). We found that a beam width of only 1 (*i.e.*, best-first search) yielded a median accuracy of 99.4% in the 8 client case, with a median running time of less than 0.01 seconds. In the case of 64 clients, a beam width of 1 achieved a median accuracy of 97.9% in roughly 1 second. These figures are a strong testament to the power of the "rollout" heuristic used by our beam search. At the other extreme, a beam width of 1280 produced a median accuracy of 99.4% in the 64-client cases, but the median runtime was nearly 22 minutes. Figure 3 illustrates how search quality and running time, respectively, increase as a function of the beam width. Each datapoint is graphed as a box plot reflecting the minimum, 25th percentile, median, 75th percentile, and maximum value. Note that the running times have extremely low variance, while the performance is reliably above 96% of optimal for all but the smallest beam widths.

As allocation is the simplest of the bid determination problems defined, we do not expect either acquisition or completion to prove any less complex for the ILP (except in the special case of linear prices), though this remains an open question. Beam search, however, solves allocation and acquisition and hence, completion, in an identical way. We conclude that heuristic search is a practical choice for use in the inner loop of an automated trading agent.

3.3 Summary of Experimental Results

In this section, we discuss two algorithmic approaches to solving bid determination problems in the context of TAC–2000: heuristic search and integer linear programming (ILP). We observed that the second of the two reductions produces smaller-sized acquisition problems, lending itself to more efficient solution times with both ILP and beam-search based solvers.

4. Conclusion

In this paper, we have addressed some of the challenges in trading agent design:

- We defined a class of bid determination problems at the core of an agent architecture for trading in simultaneous auctions, and analyzed the relationships among these problems. Based on our theoretical findings, we conclude that completion—a special case of the winner determination problem in combinatorial exchanges—is no harder than acquisition—that is, winner determination with reserve prices in combinatorial auctions.
- We described heuristic search techniques and integer linear programming solutions to key bid determination problems that arose in the First International Trading Agent Competition. Based on our experimental results, we conclude that heuristic search is a practical choice for use in the inner loop of an agent trading in simultaneous auctions. Indeed, many successful TAC agents rely on heuristic search methods: e.g., WhiteBear [14].

Acknowledgments

The author gratefully acknowledges Justin Boyan. Justin conceived of a preliminary version of the second reduction, which was implemented in `RoxyBot`–2000. The author also acknowledges the contributions of Justin Boyan, Mike Kirby, and Jon Reiter to the experimental results. Victor Naroditskiy deserves the author's sincerest gratitude for carefully proofreading the proof of the second reduction. In addition, Mike Wellman simplified the presentation and tightened up the text in many places. Finally, the suggestions of several anonymous reviewers undoubtedly improved the presentation of this paper. This research was supported by NSF Career Grant #IIS-0133689.

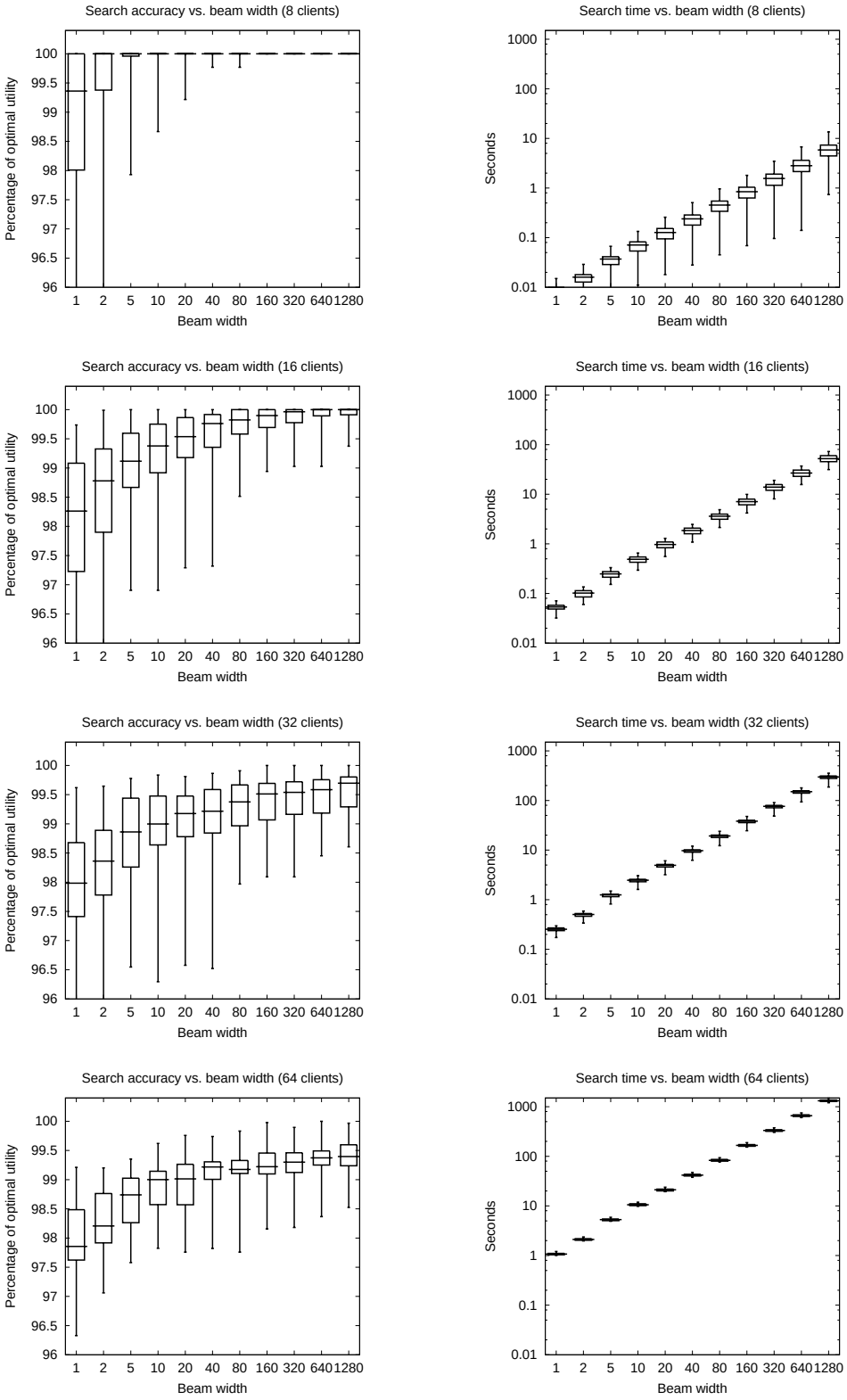

Figure 3: LEFT: Optimality of beam search, as a function of beam width and number of clients. (Note the offset on the y-axis.) RIGHT: Beam search runtimes, as a function of beam width and number of clients. (Note the log scale on the y-axis.)

References

[1] B. Abramson. Expected-outcome: A general model of static evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):182–193, Feb. 1990.

[2] M. Benisch, A. Greenwald, V. Naroditskiy, and M. Tschantz. A stochastic programming approach to TAC SCM. In *Fifth ACM Conference on Electronic Commerce*, pages 152–160, May 2004.

[3] J. Boyan and A. Greenwald. Bid determination in simultaneous auctions: An agent architecture. In *Proceedings of Third ACM Conference on Electronic Commerce*, 210–212 2001.

[4] A. Gilpin and T. Sandholm. Arbitrage in combinatorial exchanges. In *Sixth Workshop on Agent Mediated Electronic Commerce*, 2004.

[5] A. Greenwald and J. Boyan. Bidding algorithms for simultaneous auctions: A case study. *Journal of Autonomous Agents and Multiagent Systems*, 10(1):67–89, 2005.

[6] A. Greenwald, B. Guillemette, V. Naroditskiy, and M. Tschantz. Scaling up the sample average approximation method for stochastic optimization with applications to trading agents. In *International Joint Conference on Artificial Intelligence Workshop on Trading Agent Design and Analysis*, August 2005.

[7] D. Pardoe and P. Stone. TacTex-03: A supply chain management agent. *SIGecom Exchanges*, 4(3):19–28, 2004.

[8] M. Rothkopf, A. Pekeč, and R. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8), 1998.

[9] T. Sandholm and S. Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of AAAI*, pages 90–97, 2000.

[10] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 69–76, July 2002.

[11] P. Stone and A. Greenwald. The first international Trading Agent Competition: Autonomous bidding agents. *Electronic Commerce Research: Special Issue on Dynamic Pricing*, 5:229–265, 2005.

[12] P. Stone, M. Littman, S. Singh, and M. Kearns. Attac-2000: An adaptive autonomous bidding agent. *Journal of Artificial Intelligence Research*, 15:189–206, 2001.

[13] G. Tesauro and G. R. Galperin. On-line policy improvement using Monte-Carlo search. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in NIPS*, volume 9. MIT Press, 1997.

[14] I. Vetsikas and B. Selman. WhiteBear: An empirical study of design tradeoffs for autonomous trading agents. In *Game Theoretic Decision Theoretic Agents Workshop*, August 2002.

[15] M. Wellman, P. Wurman, K. O'Malley, R. Bangera, S. Lin, D. Reeves, and W. Walsh. A Trading Agent Competition. *IEEE Internet Computing*, April 2001.

Appendix A. Second Reduction

In this appendix, we formally prove that our second reduction is indeed valid. We start by describing the proof technique (Theorem A.1) that we use to establish this fact. This proof technique is necessary, because, unlike the first reduction in which the (implicit) mapping from feasible solutions to acquisition into feasible solutions to completion is a bjiection, in the second reduction, h is not onto and i is not injective. In particular, the mappings h and i are not inverses of one another.

Theorem A.1 *Given a first constrained optimization problem, call it F,*

$$\max_{feasible\ x\ \in\ X} f(x) \tag{20}$$

a second constrained optimization problem, call it G,

$$\max_{feasible\ y\ \in\ Y} g(y) \tag{21}$$

and a function $h : Y \to X$ that maps solutions of G into solutions of F, an optimal solution y^* of G maps to an optimal solution $x^* \equiv h(y^*)$ of F, if

1. *if y is feasible, then $h(y)$ is feasible, for all $y \in Y$*
2. *there exists a function $i : X \to Y$ that maps solutions of F into solutions of G such that*
   (a) *if x is feasible, then $i(x)$ is feasible, for all $x \in X$*
   (b) *there exists a constant $C \in \mathbb{R}$ such that*
      i. *$f(h(y)) \geq g(y) + C$, for all feasible $y \in Y$*
      ii. *$g(i(x)) + C \geq f(x)$, for all feasible $x \in X$*

Proof A.1 The solution $x^* \equiv h(y^*)$ of F is feasible, by Condition 1, since y^* is feasible. It remains to show that x^* is optimal. First, $f(x^*) = f(h(y^*)) \geq g(y^*) + C$, by Condition 2(b)i. Second, for all feasible $x \in X$, $g(i(x)) + C \geq f(x)$, by Condition 2(b)ii. Therefore, noting that $i(x)$ is feasible by Condition 2a, $f(x^*) \geq g(y^*) + C \geq g(i(x)) + C \geq f(x)$, for all feasible $x \in X$, since y^* is optimal. □

Remark A.2 Any functions h and i that satisfy Conditions 2(b)i and 2(b)ii in Theorem A.1 with equality are "pseudo-inverses," in the following sense:

- $f(x) = f(h(i(x)))$, for all $x \in X$
- $g(y) = g(i(h(y)))$, for all $y \in Y$

Proof A.2 For all $x \in X$,

$$\begin{aligned} f(x) &= g(i(x)) + C && (22)\\ &= f(h(i(x))) - C + C && (23)\\ &= f(h(i(x))) && (24) \end{aligned}$$

Equation 22 follows from Condition 2(b)ii; Equation 23 follows from Condition 2(b)i.

For all $y \in Y$,

$$\begin{aligned} g(y) &= f(h(y)) - C && (25)\\ &= g(i(h(y))) + C - C && (26)\\ &= g(i(h(y))) && (27) \end{aligned}$$

Equation 25 follows from Condition 2(b)i; Equation 26 follows from Condition 2(b)ii. □

We are now ready to establish the validity of the second reduction. Before doing so, however, we note one further distinction between the first and second reductions, namely: the second reduction does not rely on the assumption that the agent's surplus incorporates an additive valuation function. Hence, we restate and prove a slightly more general version of Theorem 2.7 in which there is only one value function $v : \mathcal{N} \to \mathbb{R}$, so that the function Valuation is arbitrary.

Theorem A.3 *Given a completion problem* Completion(G, N, v, P, Π)*, consider the acquisition problem* Acquisition(G, N, v, P') *with unified pricelines P'. Algorithm 3 (the mapping h) returns an optimal solution to this completion problem, given an optimal solution to this acquisition problem. Conversely the mapping i returns an optimal solution to this acquisition problem, given an optimal solution to this completion problem.*

To establish this theorem, we prove a series of lemmas corresponding to the sufficient conditions given by Theorem A.1. We then apply our lemmas and Theorem A.1 *twice*, once to prove that h maps optimal acquisitions to optimal completions, and once to prove that i achieves the reverse.

Lemma A.4 *If an acquisition (X', Y') is feasible, then so is the completion $h(X', Y')$. Similarly, if a completion (X, Y, Z) is feasible, then so is the acquisition $i(X, Y, Z)$.*

Proof A.4 By the definition of h (see Algorithm 3), $Y' = (Y \setminus A) \oplus (A \setminus Z)$ and $Z \subseteq A \subseteq Y$. These two observations imply that $Y' = Y \setminus Z$. Using the fact that $Y' = Y \setminus Z$ under both mappings (the mapping i is defined as such), we establish Conditions 1 and 2a.

For all acquisitions $X', Y' \subseteq N$,

$$\begin{aligned}
(X', Y') \text{ is feasible} \quad &\text{iff} \quad X' \subseteq Y' && (28)\\
&\text{iff} \quad X \subseteq Y \setminus Z && (29)\\
&\Rightarrow \quad X \oplus Z \subseteq Y && (30)\\
&\text{iff} \quad (X, Y, Z) \text{ is feasible} && (31)\\
&\text{iff} \quad h(X', Y') \text{ is feasible} && (32)
\end{aligned}$$

Equation 29 follows from the definition of h. Equation 30 follows because, for all $A, B, C \subseteq N$ such that $C \subseteq B$, if $A \subseteq B \setminus C$, then $A \oplus C \subseteq B$. Indeed, $Z \subseteq Y$.

For all completions $X, Y, Z \subseteq N$,

$$\begin{aligned}
(X, Y, Z) \text{ is feasible} \quad &\text{iff} \quad X \oplus Z \subseteq Y && (33)\\
&\Rightarrow \quad X \subseteq Y \setminus Z && (34)\\
&\text{iff} \quad X' \subseteq Y' && (35)\\
&\text{iff} \quad (X', Y') \text{ is feasible} && (36)\\
&\text{iff} \quad i(X, Y, Z) \text{ is feasible} && (37)
\end{aligned}$$

Equation 34 follows because, for all $A, B, C \subseteq N$, if $A \oplus C \subseteq B$, then $A \subseteq B \setminus C$. Equation 35 follows from the definition of i. □

Lemma A.5 *If $h(X', Y') = (X, Y, Z)$, then*

$$\text{Valuation}(X) - \text{Cost}(Y, P) + \text{Revenue}(Z, \Pi) = \text{Valuation}(X') - \text{Cost}(Y', P') + \text{ARB}(P, \Pi) \quad (38)$$

Proof A.5 Under the mapping h, $X = X'$. Hence, it suffices to show:

$$\text{Arbitrage}_g(P, \Pi) - \text{Cost}_g(Y', P') = \text{Revenue}_g(Z, \Pi) - \text{Cost}_g(Y, P) \quad (39)$$

But since $Z \subseteq A \subseteq Y$ under the mapping h, Equation 39 is equivalent to Equation 44 as follows:

$$\text{Arbitrage}_g(P,\Pi) - \text{Cost}_g(Y',P') = \text{Revenue}_g(Z,\Pi) - \text{Cost}_g(Y,P) \quad (40)$$

$$\sum_{k=1}^{A_g}(\pi_{gk} - p_{gk}) - \sum_{k=1}^{Y'_g} p'_{gk} = \sum_{k=1}^{Z_g}(\pi_{gk} - p_{gk}) - \sum_{k=Z_g+1}^{Y_g} p_{gk} \quad (41)$$

$$\sum_{k=1}^{A_g}(\pi_{gk} - p_{gk}) - \sum_{k=1}^{Z_g}(\pi_{gk} - p_{gk}) + \sum_{k=Z_g+1}^{Y_g} p_{gk} = \sum_{k=1}^{Y'_g} p'_{gk} \quad (42)$$

$$\sum_{k=Z_g+1}^{A_g}(\pi_{gk} - p_{gk}) + \sum_{k=Z_g+1}^{Y_g} p_{gk} = \sum_{k=1}^{Y'_g} p'_{gk} \quad (43)$$

$$\sum_{k=Z_g+1}^{A_g}\pi_{gk} + \sum_{k=A_g+1}^{Y_g} p_{gk} = \sum_{k=1}^{Y'_g} p'_{gk} \quad (44)$$

Finally, Equation 44 follows from the fact that the first $Y'_g = (Y \setminus Z)_g$ values in the unified priceline $\vec{p}'_g$ are precisely the smallest $A_g - Z_g$ values in $\vec{\pi}_g$ corresponding to arbitrage opportunities and the smallest $Y_g - A_g$ values in $\vec{p}_g$ corresponding to non-arbitrage opportunities. □

Lemma A.6 *If $i(X,Y,Z) = (X',Y')$, then*

$$\text{Valuation}(X') - \text{Cost}(Y',P') + \text{ARB}(P,\Pi) \geq \text{Valuation}(X) - \text{Cost}(Y,P) + \text{Revenue}(Z,\Pi) \quad (45)$$

Proof A.6 Under the mapping i, $X = X'$. Hence, it suffices to show:

$$\text{Arbitrage}_g(P,\Pi) - \text{Cost}_g(Y',P') \geq \text{Revenue}_g(Z,\Pi) - \text{Cost}_g(Y,P) \quad (46)$$

But since $Y' = Y \setminus Z$ under the mapping i, Equation 46 is equivalent to Equation 47:

$$\sum_{k=1}^{A_g}(\pi_{gk} - p_{gk}) - \sum_{k=1}^{(Y\setminus Z)_g} p'_{gk} \geq \sum_{k=1}^{Z_g}\pi_{gk} - \sum_{k=1}^{Y_g} p_{gk} \quad (47)$$

Since, by feasibility $Z \subseteq Y$, to infer Equation 47, it suffices to show:

$$\sum_{k=1}^{A_g}(\pi_{gk} - p_{gk}) - \sum_{k=1}^{(Y\setminus Z)_g} p'_{gk} \geq \sum_{k=1}^{Z_g}(\pi_{gk} - p_{gk}) - \sum_{k=Z_g+1}^{Y_g} p_{gk} \quad (48)$$

Three cases arise.
Case $Z \subseteq A \subseteq Y$: Since $Z \subseteq A$, it in fact suffices to show:

$$\sum_{k=Z_g+1}^{A_g}(\pi_{gk} - p_{gk}) + \sum_{k=Z_g+1}^{Y_g} p_{gk} \geq \sum_{k=1}^{(Y\setminus Z)_g} p'_{gk} \quad (49)$$

Moreover, since $Z \subseteq A \subseteq Y$, the following inequality actually suffices:

$$\sum_{k=Z_g+1}^{A_g}\pi_{gk} + \sum_{k=A_g+1}^{Y_g} p_{gk} \geq \sum_{k=1}^{(Y\setminus Z)_g} p'_{gk} \quad (50)$$

But this final inequality holds immediately because $\sum_{k=Z_g+1}^{A_g} \pi_{gk} + \sum_{k=A_g+1}^{Y_g} p_{gk} = \sum_{k=Z_g+1}^{Y_g} q_{gk}$. Note that this inequality is sometimes strict. In the following example where $A = 5$, if $Z = 3$ and $Y = 5$, then the LHS of Equation 50 evaluates to $7+6$ whereas the RHS evaluates to $6+6$. Moreover, the completion profit (revenue less cost) is $(10+9+8)-(1+2+3+4+5) = 12$, while the acquisition profit is (the arbitrage constant value less cost) $(10+9+8+7+6)-(1+2+3+4+5)-(6+6) = 13$.

π	10	9	8	7	6	5	4	3	$\cdots$
p	1	2	3	4	5	6	7	8	$\cdots$
q	10	9	8	7	6	6	7	8	$\cdots$
p'	6	6	7	7	8	8	9	9	$\cdots$

Case $Z \subseteq Y \subseteq A$: Since $Z \subseteq A$, as above it suffices to show:

$$\sum_{k=Z_g+1}^{A_g} (\pi_{gk} - p_{gk}) + \sum_{k=Z_g+1}^{Y_g} p_{gk} \geq \sum_{k=1}^{(Y \setminus Z)_g} p'_{gk} \tag{51}$$

Moreover, since $Z \subseteq Y \subseteq A$, it in fact suffices to show:

$$\sum_{k=Z_g+1}^{Y_g} \pi_{gk} + \sum_{k=Y_g+1}^{A_g} (\pi_{gk} - p_{gk}) \geq \sum_{k=1}^{(Y \setminus Z)_g} p'_{gk} \tag{52}$$

But this middle term is nonnegative, so that

$$\sum_{k=Z_g+1}^{Y_g} \pi_{gk} \geq \sum_{k=1}^{(Y \setminus Z)_g} p'_{gk} \tag{53}$$

suffices. But this final inequality holds immediately because $q_{gk} = \pi_{gk}$ for all $k \in \{Z_g + 1, \ldots, Y\}$. Note that this inequality is sometimes strict. In the following example where $A = 6$, if $Z = 2$ and $Y = 4$, then the LHS of Equation 53 evaluates to $8+7$ whereas the RHS evaluates to $5+6$. Moreover, the completion profit (revenue less cost) is $(10+9)-(-1+0+1+2) = 17$, while the acquisition profit is (the arbitrage constant value less cost) $(10+9+8+7+6+5)-(-1+0+1+2+3+4)-(5+6) = 20$.

π	10	9	8	7	6	5	4	3	$\cdots$
p	-1	0	1	2	3	4	50	60	$\cdots$
q	10	9	8	7	6	5	50	60	$\cdots$
p'	5	6	7	8	9	10	50	60	$\cdots$

Case $A \subseteq Z \subseteq Y$: Since $A \subseteq Z$, it suffices to show:

$$\sum_{k=Z_g+1}^{Y_g} p_{gk} + \sum_{k=A_g+1}^{Z_g} (p_{gk} - \pi_{gk}) \geq \sum_{k=1}^{(Y \setminus Z)_g} p'_{gk} \tag{54}$$

But this middle term is nonnegative, so that

$$\sum_{k=Z_g+1}^{Y_g} p_{gk} \geq \sum_{k=1}^{(Y \setminus Z)_g} p'_{gk} \tag{55}$$

suffices. But this final inequality holds immediately because $q_{gk} = p_{gk}$ for all $k \in \{Z_g + 1, \ldots, Y\}$. Note that this inequality is sometimes strict. In the following example where $A = 2$, if $Z = 4$ and $Y = 6$, then the LHS of Equation 55 evaluates to $6+5$ whereas the RHS evaluates to $3+4$. Moreover, the completion profit (revenue less cost) is $(10+9+0+0)-(1+2+3+4+5+6) = -1$, while the acquisition profit is (the arbitrage constant value less cost) $(10+9)-(1+2)-(3+4) = 9$.

π	10	9	0	0	0	0	$\cdots$
p	1	2	3	4	5	6	$\cdots$
q	10	9	3	4	5	6	$\cdots$
p'	3	4	5	6	7	8	$\cdots$

□

In summary, while the mapping h from acquisitions to completions preserves the value of the objective function (up to the arbitrage constant), the mapping i from completions to acquisitions does not. Lemma A.5 is proved with equality, while Lemma A.6 is an inequality in general.

We now establish our main theorem.

Proof A.3 To prove that h maps optimal acquisitions to optimal completions, apply Theorem A.1 as follows: choose F as the completion problem with $f(X, Y, Z) = \text{Valuation}(X) - \text{Cost}(Y, P) + \text{Revenue}(Z, \Pi)$, G as the acquisition problem with $g(X', Y') = \text{Valuation}(X') - \text{Cost}(Y', P')$, $h = h$, $i = i$, and $C = \text{ARB}(P, \Pi)$. Lemmas A.4, A.5, and A.6 establish the sufficient conditions. In particular, Lemma A.5 establishes Condition 2(b)i and Lemma A.6 establishes Condition 2(b)ii.

Conversely, to prove that i maps optimal completions to optimal acquisitions apply Theorem A.1 as follows: choose F as the acquisition problem with $f(X', Y') = \text{Valuation}(X') - \text{Cost}(Y', P')$, G as the completion problem with $g(X, Y, Z) = \text{Valuation}(X) - \text{Cost}(Y, P) + \text{Revenue}(Z, \Pi)$, $h = i$, $i = h$, and $C = -\text{ARB}(P, \Pi)$. Again the sufficient conditions follow from the above series of lemmas, but this time Lemma A.5 establishes Condition 2(b)ii and Lemma A.6 establishes Condition 2(b)i. □

Appendix B. TAC–2000 Travel Game

A TAC–2000 Travel agent [15] is a simulated travel agent whose task is to organize itineraries for a group of clients who wish to travel from TACTown to Boston and back during a five day period. Travel and entertainment goods are traded at 28 online auctions that run simultaneously for fifteen minutes. An agent's objective is to secure the goods necessary to satisfy the particular desires of its clients, but to do so as inexpensively as possible. An agent's score is the difference between the valuations it earns for its clients and the agent's expenditures.

B.1 Supply

The market supply consists of three types of travel goods: (i) flights in and out of Boston (I and O, respectively), (ii) hotel room reservations at two competing hotels, namely, the Grand Hotel (G) and Le Fleabag Inn (F), and (iii) entertainment tickets for the Boston Red Sox (R), the Boston Symphony (S), and the theater (T). There is a separate auction for each combination of good and day, yielding twenty-eight auctions in total: eight flight auctions (there are no inbound flights on the fifth day, and there are no outbound flights on the first day), eight hotel auctions (two hotel types and four nights), and twelve entertainment ticket auctions (three entertainment event types and four nights). All auctions are *simultaneous*. The rules of the various auctions are as follows:

(i) An infinite supply of flights is sold by the "TAC seller", a specially designated supplier, in a continuously-clearing, dynamic, posted-pricing environment in which prices follow a random walk. No resale of flights is permitted.

(ii) The TAC seller also makes available sixteen hotel rooms per hotel per night, which are sold at ascending, multi-unit, sixteenth-price auctions. In other words, the winning bidders are those who bid among the top sixteen, and these bidders uniformly pay the sixteenth-highest price. Transactions clear when the auctions close, which typically occurs at the end of a game instance, although these auctions are subject to early closing after random periods of inactivity. No bid withdrawal or resale is permitted in hotel auctions.

(iii) Entertainment tickets are traded among TAC–2000 agents in continuous double auctions. Agents can act as either buyers or sellers, and transactions clear continuously. Each agent receives an initial endowment of tickets for each event on each night—zero with probability 1/4, one with probability 1/2, and two with probability 1/4. Ticket resale is permitted.

B.2 Demand

A TAC–2000 game instance pits eight trading agents against one another, each agent representing eight clients. The market demand is determined by the sixty-four clients' preferences. Each client is characterized by a random set of preferences for ideal arrival and departure dates (IAD and IDD, respectively, which range over days 1 through 4),[4] a grand hotel room reservation value (HV, which takes integer values between 50 and 150), and reservation values for each of the three types of entertainment events (RV, SV, and TV—integers between 0 and 200—for Red Sox, symphony, and theater, respectively). A sample set of preferences appears in Table 1; these preferences were those of the clients assigned to `RoxyBot`–2000 during game 3065 of TAC–2000.

Client	IAD	IDD	HV	RV	SV	TV
1	1	2	99	134	118	65
2	1	3	131	170	47	49
3	1	1	147	13	55	49
4	3	3	145	130	60	85
5	1	3	82	136	68	87
6	2	3	53	94	51	105
7	1	2	54	156	126	71
8	1	4	113	119	187	143

Table 1: `RoxyBot`–2000's client preferences in game 3065. These values jointly establish the bidding agent's valuation function over various combinations of goods.

The job of each TAC–2000 agent is to assemble a feasible package of goods for each of its clients. A *package* is characterized by arrival and departure dates (AD and DD, respectively, ranging over days 1 through 5), a hotel type (H, which takes on value G for Grand Hotel or F for Le Fleabag Inn), and entertainment tickets ($\mathrm{I}(j,k)$ is an indicator variable that represents whether or not the package includes a ticket on night j to event $k \in \{r,s,t\}$; we also write R1, for example, to indicate that the package includes a Boston Red Sox ticket on night 1). In order to obtain positive valuation for a client, an agent must construct a *feasible* package for that client; otherwise, the client's valuation is zero. A feasible package is one in which (i) the arrival date is strictly less than the departure date, (ii) the same hotel is reserved during all intermediate nights, (iii) at most one entertainment event per night is included, and (iv) at most one of each type of entertainment ticket is included. Given a feasible package, a client's valuation for that package is calculated as follows:

$$\text{valuation} = 1000 - \text{travelPenalty} + \text{hotelBonus} + \text{funBonus} \tag{56}$$

where

$$\begin{aligned}
\text{travelPenalty} &= 100(|\text{IAD} - \text{AD}| + |\text{IDD} - \text{DD}|) \\
\text{hotelBonus} &= \begin{cases} \text{HV} & \text{if H} = \text{G} \\ 0 & \text{otherwise} \end{cases} \\
\text{funBonus} &= \textstyle\sum_j \left[\mathrm{I}(j,r)\text{RV} + \mathrm{I}(j,s)\text{SV} + \mathrm{I}(j,t)\text{TV}\right]
\end{aligned}$$

4. For notational convenience, we remap outbound flight j to $j-1$, for $j \in \{2,3,4,5\}$.

B.3 Allocation

A TAC–2000 agent faces the allocation problem at the end of a game instance when it must find the assignment of goods to clients that maximizes total client valuation. At the end of game 3065, `RoxyBot`–2000 held the set of goods listed in Table 2. The optimal allocation is depicted in Table 3, given the client preferences presented in Table 1. Packages were allocated to clients as shown, and the total valuation obtained was 9999.

Good	Day 1	Day 2	Day 3	Day 4
R	2	2	1	2
S	2	0	2	0
T	1	1	1	0
G	4	3	3	1
F	2	2	0	1
I	6	0	2	0
O	1	4	2	1

Table 2: `RoxyBot`–2000's final set of goods in game 3065.

Client	AD	DD	H	Tickets	Valuation
1	1	2	G	S1, R2	1351
2	1	2	G	R1	1201
3	1	1	G	—	1147
4	3	3	G	R3	1275
5	1	2	F	R1, T2	1123
6	3	3	G	T3	1058
7	1	2	F	S1, R2	1282
8	1	4	G	T1, S3, R4	1562

Table 3: `RoxyBot`–2000's final allocation in game 3065. The total valuation is 9999.

Appendix C. ILP Solutions

In this section, we present integer linear programming (ILP) solutions to the TAC–2000 allocation and acquisition (and hence, completion) problems. These solutions are related to the ILP formulation of acquisition reported by `ATTac`–2000 [12]. But our formulation is more general: it accommodates (i) the buying and (implicit) selling of entertainment tickets and (ii) the buying and (implicit) selling of each copy of each good at different prices, via pricelines.

C.1 Allocation

We now present our ILP formulation of the TAC–2000 allocation problem. The indices, constants, and variables are listed in Tables 4, 5, and 6, respectively. For the dimensions of TAC–2000, there are 256 boolean variables, and 92 constraints. We seek to maximize the following objective function:

$$\max \left(\sum_{i,t} V_{it} v_{it} + \sum_{i,s} V_{is} w_{is} \right) \tag{57}$$

subject to the following sets of constraints. The numbers in parentheses indicate TAC–2000 values.

Index	Description	Number	TAC–2000 Values
i	clients	I	8
j	days	J	4
k	event types	K	3
l	hotel types	L	2
m	flight types	M	2
t	travel packages	T	20
s	entertainment tickets	S	12

Table 4: ILP Indices and Ranges.

Constant	Description
V_{it}	valuation to client i of travel package t
V_{is}	valuation to client i of entertainment ticket s
X_{jk}	number of entertainment tickets of type k for day j
Y_{jl}	number of hotel reservations of type l for day j
Z_{jm}	number of flights of type m for day j

Table 5: ILP Constants.

0. all variables are boolean: $\forall i, s, t, v_{it}, w_{is} \in \{0, 1\}$
1. I (8) constraints: cannot assign more than 1 travel package per client $\forall i, \sum_t v_{it} \leq 1$
2. $J \times (K + L + M)$ (28) constraints: cannot assign more tickets than the number available
   - $J \times K$ (12) constraints: can assign at most X_{jk} entertainment tickets of type k for day j

     $$\forall j, k, \quad \sum_i \sum_{\{s|\text{DAY}(s)=j,\text{TYPE}(s)=k\}} w_{is} \leq X_{jk}$$

   - $J \times (L + M)$ (16) constraints: analogous, for hotels and flights
3. $I \times K$ (24) constraints: can assign at most 1 entertainment ticket of type k per client

   $$\forall i, k, \quad \sum_j \sum_{\{s|\text{DAY}(s)=j,\text{TYPE}(s)=k\}} w_{is} \leq 1$$

4. $I \times J$ (32) constraints: if a client is in town on day j s/he can be assigned at most 1 ticket; if a client is not in town on day j, s/he cannot be assigned any tickets;

   $$\forall i, j, \quad \sum_k \sum_{\{s|\text{DAY}(s)=j,\text{TYPE}(s)=k\}} w_{is} \leq \text{INTOWN}_{ij}$$

   where

   $$\text{INTOWN}_{ij} = \sum_{\{t|\text{IN}(t)\leq j\leq\text{OUT}(t)\}} v_{it}$$

This concludes the list of constraints for the TAC–2000 allocation problem. We draw your attention to constraint 4, which is where our formulation differs from that of `ATTac`–2000's. This set of constraints replaces the following two sets of `ATTac`–2000 constraints:

Variable	Description	Number	TAC–2000 Values
v_{it}	is client i assigned package t?	$I \times T$	160
w_{is}	is client i entertainment ticket s?	$I \times S$	96

Table 6: ILP Variables.

- $I \times J$ (32) constraints: can assign at most 1 ticket on day j per client
- $I \times J \times K$ (96) constraints: cannot assign entertainment tickets if not in town

The key observation that allows this simplification is that constraint 1 forces $\text{INTOWN}_{ij} \in \{0, 1\}$. Hence, constraint 4 satisfies both the requirements enforced by `ATTac`–2000. This observation reduces the number of constraints from 188 to 92.

C.2 Completion as Acquisition

Before stating our ILP formulation of the completion problem as it pertains to the particular choice of auction mechanisms in TAC–2000, we present an ILP for TAC–2000 completion in general double markets. This initial formulation allows for the trading not only of entertainment tickets, but of flights and hotel rooms as well. Moreover, we do not require that different copies of the same good be bought and sold at the same price. We implement pricelines, which associate a possibly unique price with each distinct copy of each good. As our solution to completion, we present a solution to the second acquisition problem, which does not consider the market to be an additional client.

In addition to the constants and variables specified in our formulation of the TAC–2000 allocation problem, the acquisition ILP makes use of additional constants and variables as follows: (i) let C_{jg} denote the supply of good g on day j, where g ranges over k, l, m; (ii) let P_{jgn} denote the price of buying the nth copy of good g on day j, where $1 \leq n \leq C_{jg}$; and (iii) let q_{jgn} be a boolean variable indicating "the quantity of good g bought on day j is greater than or equal to n," where $1 \leq n \leq C_{jg}$. For notational convenience, we also define the following abbreviation, which denotes the number of copies bought of good g for day j:

$$\text{BUY}_{jg} = \sum_{n=1}^{C_{jg}} q_{jgn}$$

We seek to maximize valuations minus costs, as follows:

$$\max \sum_{i,t} V_{it} v_{it} + \sum_{i,s} V_{is} w_{is} - \sum_{j,k,n} P_{jkn} q_{jkn} \tag{58}$$

subject to constraints 0, 1, 3, and 4, of the the allocation problem, extending constraints 0 and 2 into constraints $0'$ and $2'$ below, and adding one further consideration, number 5.

$0'$. all variables are boolean: $\forall i, s, t, v_{it}, w_{is} \in \{0, 1\}$ and $\forall j, k, n, q_{jkn} \in \{0, 1\}$

$2'$. $J \times (K + L + M)$ (28) cannot assign more goods than the number owned plus what is bought

- $J \times K$ (12) constraints: can assign at most X_{jl} entertainment tickets of type k on day j plus the number bought

$$\forall j, k, \quad \sum_{i} \sum_{\{s | \text{DAY}(s)=j, \text{TYPE}(s)=k\}} w_{is} \leq X_{jk} + \text{BUY}_{jk}$$

- $J \times (L + M)$ (16) constraints: analogous, for hotels and flights

5. $J \times (K + L + M)$ (28) constraints: cannot buy more goods than market supply

$$\forall j, g, \quad \text{BUY}_{jg} \leq C_{jg}$$

This general formulation of TAC–2000 acquisition relies on the $(I \times T) + (I \times S)$ variables in TAC–2000 allocation plus $J \times (K + L + M) \times C_{jg}$ additional variables indicating quantities bought. An ILP solution to the first acquisition problem would require an additional $J \times (K + L + M) \times D_{jg}$ additional variables indicating quantities "sold" (i.e., assigned to the market). (D_{jg} denotes the demand for good g on day j.) Similarly, this formulation of the acquisition problem incorporates $J \times (K + L + M)$ constraints stating that an agent cannot buy more goods than the number the market supplies. An ILP solution to the first acquisition problem would require an additional $J \times (K + L + M)$ constraints stating that an agent cannot "sell" (i.e., assign to the market) more goods than the number the market demands. Finally, in the first acquisition problem, constraint $2'$ would be modified to state that an agent cannot assign more goods than the number owned plus what is bought minus what is "sold," but this would not affect the total number of constraints.

Specializing this general formulation of the TAC–2000 acquisition problem to the particular choices of market mechanisms, supply, and demand that were assumed in TAC–2000 straightforward. Since the supply of flights is infinite, $C_{j0} = C_{j1} = \infty$, and since there are 16 hotel rooms available of each type l on each day j, the supply of hotels $C_{jl} = 16$. In addition, we assume that the supply and demand of entertainment tickets $C_{jk} = 1$. Our justification for this final assumption is that entertainment tickets are traded continuously, so that a TAC–2000 agent that repeatedly solves the acquisition problem has repeated opportunities to buy and sell entertainment tickets. Thus, this seeming limitation is not so limiting in practice.

Given these values of C_{jg} and D_{jg}, we can now count the number of variables and constraints in our formulation. First, since pricelines for flights are constant, $q_{jmn} = q_{jmn'}$ for all n, n'; thus, we require only $J \times M$ (8) integer variables q_{jm} denoting the quantity of flights m purchased for day j. Next, since the supply of type l hotel rooms $C_{jl} = 16$, we require $J \times L \times 16$ (128) variables q_{jln} indicating "the quantity of hotel rooms of type l bought on day j is greater than or equal to n." Lastly, since we assume $C_{jk} = 1$, we require only $J \times K \times 1$ (12) boolean variables q_{jk} indicating the purchase of a ticket of type k on day j. Finally, the objective function can be restated in terms of these variables (and the corresponding constants) as follows:

$$\max \left(\sum_{i,t} V_{it} v_{it} + \sum_{i,s} V_{is} w_{is} \right) - \left(\sum_{j,m} P_{jm} q_{jm} + \sum_{j,l,n=1}^{C_{jl}} P_{jln} q_{jln} + \sum_{j,k} P_{jk} q_{jk} \right) \tag{59}$$

In total, our formulation of the TAC–2000 completion problem (as the second acquisition problem) requires 404 variables (the original 256 allocation variables and 148 additional acquisition variables) and 112 constraints—the latter figure arises because the supply of flights is infinite, which simplifies constraint 5 to only 20 additional constraints beyond the basic allocation constraints. As already mentioned, formulating completion as the first acquisition problem would require additional variables and constraints pertaining to the selling of goods by the agent.