Toward Flexible Trading Agents

Peter R. Wurman
North Carolina State University
Suite 110, Venture 1; Centennial Campus
Raleigh, NC 27695-7535 USA
wurman@csc.ncsu.edu

Abstract

The majority of studies of agent decision making in environments with multiple auctions assume homogeneity in the rules of the auctions. However, online markets are much more fragmented, and products are often for sale in a variety of venues with different rules. We present a conceptual approach to designing flexible trading agents that is broad enough to encompass both multiple types of auctions and a variety of user preferences. The generalizations described lead to natural representations of the agent's decision task as either a Markov decision processes or an extensive form game, depending upon the form of the market model.

1 Introduction

Electronic auctions have rapidly become one of the most vibrant and widespread applications in e-commerce. Each day, consumers list millions of items in electronic auction sites such as ebay, and it is difficult to find an industry that is not launching a vertical B2B "exchange". This profusion of auctions poses new challenges for the participants in the marketplace. Currently, if one does a search for a common product, such as a Palm Pilot, one will find there are hundreds of active listings hosted on a dozen different Web sites each with potentially different auction "types". In addition, each auction has a unique closing time, and can have other attributes which affect the buyers interest level, such as text describing the quality of the product or a measure of the seller's reputation.

For a variety of reasons, the actual Internet marketplace has evolved into a *fragmented market* in which related products are sold in various auction formats with subtle but very important differences in rules. Seemingly minor differences can induce radically different bidding behavior on the part of the bidders. Take, for example, the different strategies induced by the ascending auctions that end after a period of inactivity in which no new bids are received (eg. Amazon.com), and those that end at a fixed time (eg. ebay). The latter set of rules induces participants to "snipe", that is, to hold off bidding until the final moments, and then try to submit a high bid just before the auction closes. This strategy is ineffective in auctions with inactivity-based closing rules.

Making bidding decisions in fragmented marketplaces is complex; to our knowledge, people handle this problem by simply restricting their attention to a small subset of the auctions of potential interest. Clearly this can lead to suboptimal outcomes for both the buyers and the sellers. Software agents have the potential to greatly enhance an individual's ability to effectively participate in these environments. We call such software programs *trading agents*.

In some situations, such as consumer purchasing on the Internet, the conditions that a real agent will face will vary from episode to episode. Even when market conditions are more consistent, as might be found in a B2B exchange, we need to be able to quickly develop and deploy agents in each specialized environment. Adoption of agent technologies will be limited if an agent's bidding strategy must be hand-coded for each marketplace, and then recoded when the rules of the market change. Thus, we argue that trading agents must be designed to be flexible, that is, capable of perceiving both the context and the dynamic state of the marketplace in order to make rational decisions.

Consider the decision problem facing an agent when it desires to purchase two units of a particular object for sale. The agent searches the web and finds the following three auctions listing the desired object.[1]

- Auction A lists a single unit. The auction rules are those of a typical ascending auction with a fixed closing time, designated t_A.
- Auction B lists a single unit. The auction is also an ascending auction, however it will close after time t_B when δ_B time passes without any new bids.
- Auction C lists five units. It uses a type of multi-unit auction in which each bidder pays the price of the lowest accepted bid. Indivisible bids are accepted. This auction closes after time t_C when δ_C time passes without any new bids.

The agent could satisfy its goals by purchasing two items in auction C, one in C and one in A or B, or one in each A and B. The bidding strategy that accomplishes the selected goal is dependent upon the expected final prices in each auction, the rules of each auction, and the relative closing times

[1]All three of the auction rules described are widely used on the Internet.

of each auction. Moreover, this decision is not static—the agent's expectations of the final prices depends upon the current price quotes and the actions of the other participants.

The research to date, both in economics and computer science, views the multi-unit or multi-object allocation problems from one of two perspectives. One body of work takes the perspective of the designer of the mechanism whose goal is to select a set of rules that maximizes some objective. The approaches studied include both auctions for sets of identical objects (Demange, Gale, & Sotomayor 1986; Gale 1990; Rodríguez *et al.* 1997) and combinatorial auctions (Lehmann, O'Callaghan, & Shoham 1999; Parkes 1999; Wellman *et al.* to appear; Wurman & Wellman 2000) The second perspective is that of the participant. This body of work focuses on decision making for scenarios in which the auctions are homogeneous (Boutilier, Goldszmidt, & Sabata 1999b; Hu & Wellman 1998; Park 1999; Rust, Miller, & Palmer 1993). We are interested in decision making for heterogeneous marketplaces.

This paper presents a conceptual approach toward building flexible trading agents. In Section 2, we describe the architectural elements of a flexible trading agent and suggest some useful formalisms. Section 3 describes some of the issues related to game theoretic and decision theoretic strategy selection.

2 Components of a Trading Agent

We use the term *marketplace* to refer to the combination of the user's agent (called the *principal agent*), the auctions, and the other bidders in the system.

Similarly, the decision making policy of a flexible trading agent can be viewed as a function with three inputs:

1. A model of the user's preferences.
2. A list of auctions that are relevant to the task and a method of determining the particular rules of each auction.
3. A model of the other bidders in the auctions.

From these inputs, the agent must derive a bidding strategy that when executed (ideally) maximizes the user's expected payoff. The *Strategy Generation Engine* (SGE) is responsible for deriving a bidding strategy. Figure 1 illustrates how these components fit together. The following discussion identifies some issues associated with each of the four components.

2.1 User Preferences

Eliciting the user's economic preferences in a manner that is not overly burdensome to the user and yet acquires enough information to be of value is a challenging problem. For example, the agent should not ask the user how much she values every make/model of computer monitor that is currently being sold online. Instead, we would like the agent to ask a few questions from which it can infer a sufficiently complete preference structure.

Although preference elicitation is an interesting area of research, for the purposes of this paper we assume that we have the users' preferences in the form of a utility function. We admit a variety of preference structures, including both

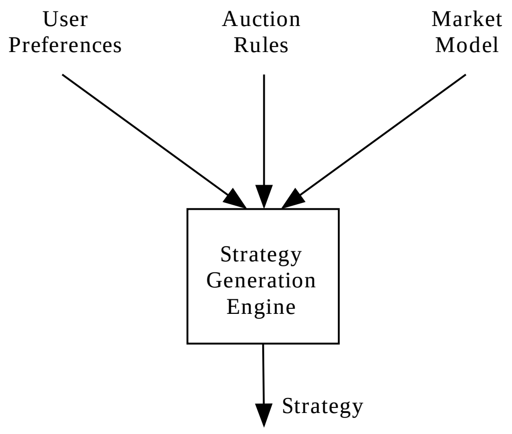

Figure 1: An architecture for trading agents.

substitutable and complementary preferences, and both independent private values and affiliated common value problems.

2.2 Auction Rules

In previous work (Wurman, Wellman, & Walsh to appear), we have described a parametrization of auction rules that encompasses the majority of common auction formats, including those used by most Internet auction sites. Moreover, the parametrized rules are instantiated in an online auction server called the Michigan Internet AuctionBot (Wurman, Wellman, & Walsh 1998).

The parameters are organized according to whether they govern the admittance of bids into the system, the revelation of information by the auction, or the policies and timing of events that clear the auction.

For example, rules associated with bid admittance determine who is allowed to bid, the format of bids, and whether and in what manner bids must "improve". Rules that control clearing specify the events that trigger a clear (e.g. at a fixed time or after no bids are received for x minutes), and the policies used to compute the prices, quantities, and trading partners involved in the resulting exchanges.

The rules are, to a large extent, orthogonal, enabling a combinatorial number of different auction types. Not only can the vast majority of auctions available on the Internet can be defined using the parametrized rules, a large number of new and potentially useful auctions can be identified.

The AuctionBot server also includes an Agent Programming Interface (API) which permits software agents to interact with the server. Through the API the agent can request the specific rules of each auction. Currently, the AuctionBot returns a text string specifying the auction rules. Work is currently underway to convert the API to an XML based interaction.

2.3 Market Models

How much and what type of knowledge of the other participants in the market does an agent need to perform effectively? This is one of the fundamental research questions in the design of trading agents. Among the choices of market models are:

1. A model of the evolution of prices that ignores the behavior of individual non-principal participants. This essentially treats the external market as a single entity we call a representative agent, with (potentially uncertain) reactions to the agent's bids.
2. A model of the individual participants in the auctions. In general there are a variety of issues that come up when modeling other participants.
   - Are the other agents' preferences known?
   - Are the other agents' strategies known?
   - What information do the other agents have about our agent, and how is that information used?

The model used by Boutilier, Goldszmidt, and Sabata (1999b) falls into the first category, while the model explored by Hu and Wellman (1998) explores multiagent learning in the second context. These examples help illustrate the spectrum of potential models. At one end are gross level models in which the behavior of the other agents is encapsulated in a single representative agent. At the other end of the spectrum we have game theoretic models which assume perfectly rational, self-interested agents with common knowledge of each other's utility functions and capabilities.

The applicability of the potential representations is also affected by features of the particular market. For instance, consider a marketplace with several sequential auctions for individual, substitutable objects and a fixed number of bidders. As the marketplace progresses and some of the objects are allocated, the number of agents competing decreases. More importantly, the past behavior of the remaining agents, and the fact that they haven't yet purchased an object, provides information to the principal agent regarding the remaining agent's preferences. A purely gross level representation may not properly capture this valuable information.

However, in practice it is quite rare to have as much information as the game theoretic model assumes. Eventually, we would like our market models to be based on the types of real data that is accessible to the agent. Many sites, such as ebay, provide identifying information that can be used to track the behavior of individual participants. We are writing software to collect this type of information from a small set of auctions in a particular product area and will use it to explore the automated construction of market models.

2.4 Strategy Generation Engine (SGE)

The primary focus of this project is to build a strategy generation engine. At an abstract level, we can separate strategy generation from strategy implementation, that is, we assume we can generate a complete policy *a priori* that specifies the agent's best action for every state it may encounter. From a practical point of view, it is often not desirable to precompute a policy for all possible states, and therefore some replanning is desirable.

The task of the strategy generation engine is to compute a strategy that maximizes the agent's utility. The methodology that is employed in this task depends upon the internal

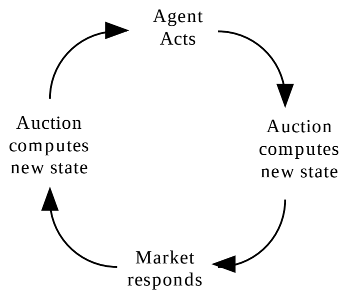

Figure 2: The cycle of actions.

model used to represent the inputs to the SGE. For example, if individual agents are modeled, then a game theoretic representation may be appropriate. On the other hand, if the market is modeled as a stochastic state machine, then a Markov Decision Process may be a useful framework for making decisions. In this section, we describe the SGE at an abstract level in the framework presented by Boutilier, Dean, and Hanks (Boutilier, Dean, & Hanks 1999).

We assume a discrete time environment in which the agent models the world as the cycle of actions depicted in Figure 2. The agent evaluates the marketplace and selects an action (i.e. bids). As a result of this action, the auctions compute new states. The other agents then respond to the new information, which in turn causes the auctions to recompute states. Each iteration of the cycle is called a *stage*, denoted t.

Formally, the environment in which a trading agent operates is a collection of auctions, $\{\alpha_1, \dots, \alpha_m\}$, each with its own set of rules. The principal agent is denoted u, and the market is inhabited by other agents, $D = \{d_1, \dots, d_n\}$. The subset of D that participates in auction i is denoted D_i. If $D_i \cap D_j = \{\}$ for all $i \neq j$ then we can safely assume that the agent's actions in i does not affect the behavior of the agents in j. However, we expect that there will be many situations in which the participation is not disjoint among the auctions of interest.

Each auction has a state, σ_t^i, which is defined by the sequence of messages (bids) that the agents have sent to the auction. The market state is the set of auction states, $S = \{\sigma_t^1 \dots \sigma_t^m\}$. The rules of each auction greatly influence the state variables that must be stored to represent the auction. In addition, the rules define the agent's permissible actions from a given state. Let $f_i(\sigma_t^i)$ be the function that returns the set of actions, including the null action, that the principal agent can take in auction i at time t.

Typically, but not always, the permissible actions depend on only the most recent bid from each agent in each auction, which enables us to make use of the Markov assumption. However, in some situations (eg. with activity rules) f can depend upon the entire history of messages. In such cases, the state may be able to be augmented with summary information which captures the current restrictions imposed by the activity rules. The set of joint actions that the agent can take at a given instant is the combination of actions in individual auctions. That is, $A_t = \odot f_i(\sigma_t^i)$.

The rules also define the auction's state transition for a given action, $a \in A$. We denote the state transition function for the auction i by g_i. Thus, a successor state, $\sigma' = g_i(a, \sigma)$, is reached when action a is taken. In most research on iterative auctions, this function is deterministic—if the agent bids more than the current highest bidder, it will be the new high bidder. However, in some recent iterative combinatorial auctions (DeMartini *et al.* 1998; Wurman & Wellman 2000) the agent cannot always predict from the information it is given the state transition that will result from an action. This occurs because the agent cannot directly observe the important aspects of the auction's state (i.e. all of the current bids). Instead, it can observe only the price information revealed by the auction. When the state is only *partially observable*, the agent may not be able to tell with certainty which state it is in, greatly complicating the decision problem.

Uncertain state transitions can also be used to model irregularities in system performance. For instance, it is a fact of Internet life that occasionally messages are lost in the network and servers crash. Also, the amount of time that it takes for a bid to reach the auction server and be admitted varies with network congestion and server load. Thus, all bids, but especially last minute bids, incur some probability of failing to reach the server in time to be counted. These random events can be accounted for in the state transition model.

Typically, the other participants will respond to the agent's action with actions of their own. It is at this point that we start to see the interplay between the market model and the decision representation. If the agent is planning on using game theoretic reasoning, it needs a method to compute the other agents' choices of actions from a given state. Typically, the other agents have the same basic choices as the principal agent, and thus the logic used in the function f can be reused. However, when each of the other agents is acting independently, the "market responds" step in Figure 2 would have to be expanded.

On the other hand, the agent may choose to use an aggregate model of the other participants that treats them as a single representative agent which takes a single (joint) action τ. To determine the representative agent's choice(s) of actions, we use the response function, μ. When the response function is deterministic, our agent's next decision point is in state $\sigma'' = g_i(\tau, g_i(a, \sigma))$ where $\tau = \mu(g_i(a, \sigma))$. When the function is not deterministic, it returns a set of possible states with associated probabilities. Note that the representative agent's response function is also constrained by the rules of the auctions.

It is natural in this model to distinguish between the effects of the principal's actions and the effects of the other agents in the system. Thus, we suggest that a flexible trading agent be based on an *explicit-event model* (Boutilier, Dean, & Hanks 1999). Doing so allows us to explore the approach of treating market models as modules with well defined programming interfaces. When necessary, we can construct an implicit model from the explicit models.

A *bidding policy*, π is a mapping between states and actions, $\pi : S \rightarrow A$. When the states are only partially observable, the policy is a mapping between observations and actions.

The purpose of an auction is to generate an allocation of the resources as a function of the messages received. The allocations that an agent receives determines its final utility. We call the auction's policy for determining allocations the *allocation function*, h. Essentially, the allocation function runs the same code as the auction server, although it can do so without the real-time constraints of the actual servers and can take advantage of certain efficiencies in the computation.

2.5 Generating Admissible Actions

The parametrization of the auction design space provides a convenient way in which to define and communicate the rules of an auction. However, the decomposition of the auction rules does not correspond to a useful parametrization of the space of bidding strategies. To illustrate the point, consider the strategies of an agent bidding in ebay's version of the English auction—which closes at a fixed time—and an agent bidding in Amazon.com's more traditional English auction. The only rule different between the two sites is the parameter that controls the timing of the clearing event. However, that difference induces very dissimilar bidding strategies. In particular, ebay's rules promote a strategy equivalent to that of the first-price, sealed-bid auction: estimate the second highest bid and bid slightly above it. Amazon's version of the auction induces the standard English type bidding behavior—keep bidding higher than the other buyers until either you win or you are unwilling to pay more.

Because there are potentially millions of ways to combine the auction parameters, we cannot make much progress toward flexible agents by analyzing individual auction types. Moreover, when multiple auctions are considered, the timing of the events and relationships between the goods can interact in complex ways. Thus, we advocate automatic exploration of the space of strategies. To accomplish this, we need a way to transform the auction rules into admissible actions.

One approach which we are exploring is the use of declarative languages for reasoning about actions. The following description is meant to be suggestive of our approach. Consider a rule that defines the action of placing a bid that is ϵ more than the current price quote in auction i:

$$\begin{aligned} \texttt{bid}(i, \texttt{quote}(i, s) + \epsilon) \quad \leftarrow \quad & \texttt{winning}(i, s) < 1 \\ & \wedge \neg \texttt{closed}(i, s). \end{aligned}$$

This rule can be interpreted as: bid in auction i for a price ϵ more than the current price quote if auction i is not closed and the agent is currently winning zero units in i.

This scheme depends upon us storing quite a bit of information in the state σ. In particular, we need to store `winning`, `closed`, and `quote` for each auction. However, this information is easily derived from the auction's state transition function.

The above rule is applicable only to single unit auctions. To develop a rule set applicable to the example from Section 1, we need to introduce quantity to both the state and

Auction	closed	winning	bid	quote
1	true	1	1 @ $10	1 @ $10
2	false	0	1 @ $7	1 @ $9
3	false	0	1 @ $6	1 @ $8 2 @ $7.5 each

Table 1: A state in the thee auction example.

the action rules. Table 1 shows a state that may occur for the three auction example from Section 1. The quote for the third auction displays prices that depend upon the number of units requested.

The modified action rule is

$$\texttt{bid}(i, \texttt{quote}(i,s,q)+\epsilon, q) \quad \leftarrow \quad \texttt{winning}(i,s) < q \\ \wedge\neg\texttt{closed}(i,s).$$

This rule states that an agent can place a bid in auction i for q units for a dollar value $\texttt{quote}(i,s,q)+\epsilon$ if the auction is not closed and the agent is winning less than q units. Assuming the state in Table 1 and that the domain of i is $\{1,2,3\}$ and the domain of q is $\{1,2\}$, the following conclusions can be drawn:

$\texttt{bid}(2, \$9+\epsilon, 1)$
$\texttt{bid}(3, \$8+\epsilon, 1)$
$\texttt{bid}(3, \$15+\epsilon, 2)$

Notice that this formulation permits the action $\texttt{bid}(3, \$15+\epsilon, 2)$ even though the agent has already won an item in the first auction. In this step we are computing all possible actions. It is the task of the decision process to evaluate the utility of these actions.

The above presentation is suggestive—a complete action specification will require null actions and wait actions, and a variety of other time-related reasoning.

3 Decision Representations

We see a natural connection between decision representations and the market model. When the agent constructs a model of the other individuals in the market, it can use game theory to select a strategy. However, when the market is modeled as a reactive black box, then Markov Decision Processes are an appropriate tool.

3.1 Game-Theoretic Models

Game theory (Fudenberg & Tirole 1996), the fundamental tool for analysis in microeconomic settings, is applicable to decision making in fragmented markets. A particular fragmented market scenario can be modeled as a strategic form game using standard notation. In game theory terminology, the sequences of bids are the agent's strategies, and the sequences of bids by the other agents (or market responses) define the strategies of the opponent(s). Thus, the decision problem reduces to a problem of finding a Nash or Bayes-Nash equilibrium. However, the strategic form is a cumbersome view of the decision problem. It is more natural to view the analysis in an extensive form game.

Game Construction Let us continue with the scenario presented in Section 1. The agent is interested in buying two items for no more than $5 each, and has three auctions in which it can participate. Suppose that the current price quotes from each auction are $p_A = \$3, p_B = \$4$, and $p_C = \$2$ for one unit and $3 for two units. Suppose that the only admissible bids in each auction are exactly $1 over the current price quote.

If we view the agent's problem as an extensive form game, the situation described above is one branch on the game tree. This branch, and the actions associated with it are shown in Figure 3. The rectangle represents the game state, while the rounded rectangles indicate the agent's possible actions. The actions are composed of bids placed in one or more of the three auctions (i.e. b_C denotes placing a bid in auction C for the designated value). Each action leads to a new game state, which in turn presents the agent with a new choice of actions.

In some cases, the agent may not be able to precisely predict the outcome of its actions. This can occur when the agent does not have complete information about the other agent's actions in the auction, and thus cannot precisely predict the effect that its bid will have on the auction state. These situations can be modeled using the notation for imperfect information in extensive form games (Fudenberg & Tirole 1996).

Strategy Selection The purpose of constructing a game-theoretic representation of the agent's decision problem is to enable the agent to select a strategy. Considerable work has been done on computational game theory (Koller, Megiddo, & von Stengel 1994; 1996; McKelvey & McLennan 1996), and general purpose systems, such as GAMBIT (http://masada.hss.caltech.edu/g̃ambit/Gambit.html) and GALA (http://robotics.stanford.edu/k̃oller/papers/gala.html), have been built for solving games.

However, game theoretic representations quickly become intractable as the number of strategies and number of agents increase. It may be possible to exploit regularities in the auction games to improve the performance of the general algorithms. For instance, in some situations the agent's decision problem may be effectively treated hierarchically—at a high level we may represent the agent's action in an ascending auction simply as the maximum bid it will place, and in a subplan enumerating all of the intermediate bids. Similarly, when the values of bids are not restricted (as in the example), we need methods to represent and reason with action ranges (eg. bid between $4 and $5) (cf. (Boutilier, Goldszmidt, & Sabata 1999a)).

3.2 Markov Decision Models

We also anticipate that some models of the market behavior, particularly those that model the market response as a purely reactive entity, will be amendable to decision making using Markov Decision Processes (MDPs) (Boutilier,

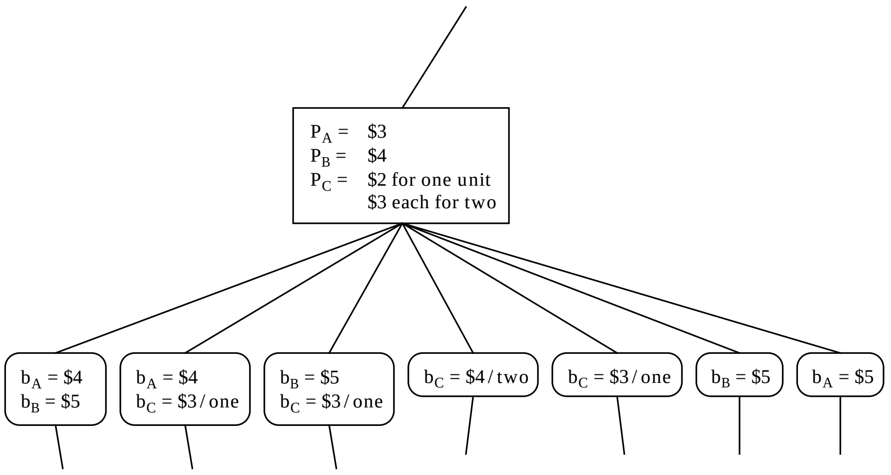

Figure 3: A portion of the game tree for an agent with a choice of bidding in three auctions with different rules.

Dean, & Hanks 1999). In the simplest case, the market function generates a single market action, τ and the state transition function is as described in Section 2.4. The general model presented in this paper admits various sources of imperfect information, such as probability distributions over the agents' preferences, affiliated common values, or simply the fact that the agent's observations of price quotes are an indirect measure of the other agent's bids. In such cases, the market response function returns a set of possible market actions which in turn result in a set of possible successor states. These situations can be modeled using Partially Observable Markov Decision Processes (POMDPs).

The agent framework which we are building will be designed to be modular so that we can implement both of the above decision representations. When the agent has several potential decision representations available, it needs to perform metareasoning to select one to use for a particular problem instance. This suggests another avenue of research, namely, the automatic selection of decision representation based on dynamic assessment of problem features.

4 Conclusion

In this article we have advocated an approach to building trading agents that stresses flexibility. We have identified three inputs which are central to a trading agent: user preferences, auction rules, and a market model. These inputs feed the strategy generation and the strategy evaluation engines. Game theory and decision theory, when combined with an appropriate market model, can be used for the task of strategy selection. The architecture can handle a wide variety of auction rules and agent preferences.

We are currently building a trading agent in the spirit of the one described here.

acknowledgments

I would like to thank Stuart Feldman, Amy Greenwald, David Parkes, Mike Wellman, and Weili Zhu for valuable conversations regarding this work.

References

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

Boutilier, C.; Goldszmidt, M.; and Sabata, B. 1999a. Continuous value function approximation for sequential bidding policies. In *Fifteenth Conference on Uncertainty in Artificial Intelligence*.

Boutilier, C.; Goldszmidt, M.; and Sabata, B. 1999b. Sequential auctions for the allocation of resources with complementarities. In *Sixteenth International Joint Conference on Artificial Intelligence*.

Demange, G.; Gale, D.; and Sotomayor, M. 1986. Multi-item auctions. *Journal of Political Economy* 94:863–72.

DeMartini, C.; Kwasnica, A. M.; Ledyard, J. O.; and Porter, D. 1998. A new and improved design for multi-object iterative auctions. Technical report, California Institute of Technology.

Fudenberg, D., and Tirole, J. 1996. *Game Theory*. MIT Press.

Gale, I. 1990. A multiple-object auction with superadditive values. *Economics Letters* 34:323–28.

Hu, J., and Wellman, M. P. 1998. Online learning about other agents in a dynamic multiagent system. In *Second International Conference on Autonomous Agents (Agents 98)*, 239–246.

Koller, D.; Megiddo, N.; and von Stengel, B. 1994. Fast algorithms for finding randomized strategies in game trees. In *26th ACM Symposium on the Theory of Computing*, 750–759.

Koller, D.; Megiddo, N.; and von Stengel, B. 1996. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior* 14(2):247–259.

Lehmann, D.; O'Callaghan, L. I.; and Shoham, Y. 1999. Truth revelation in rapid, approximately efficient combinatorial auctions. In *ACM Conference on Electronic Commerce*, 96–102.

McKelvey, R. D., and McLennan, A. 1996. Computation of equilibrium in finite games. In *Handbook of Computational Economics*, volume 1. Amsterdam: Elsevier Science, B.V.

Park, S. 1999. *The P-Strategy: An Adaptive Agent Bidding Strategy Based on Stochastic Modeling for Continuous Double Auctions*. Ph.D. Dissertation, University of Michigan.

Parkes, D. C. 1999. *i*Bundle: An efficient ascending price bundle auction. *First ACM Conference on Electronic Commerce* 148–157.

Rodríguez, J.; Noriega, P.; Sierra, C.; and Padget, J. 1997. FM96.5 a java-based electronic auction house. In *Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '97)*.

Rust, J.; Miller, J. H.; and Palmer, R. 1993. *Behavior of Trading Automata in a Computerized Double Auction Market*. Reading, MA: Addison-Wesley Publishing. chapter 6, 155–98.

Wellman, M. P.; Walsh, W. E.; Wurman, P. R.; and MacKie-Mason, J. K. to appear. Auction protocols for decentralized scheduling. *Games and Economic Behavior*.

Wurman, P. R., and Wellman, M. P. 2000. AkBA: A progressive, anonymous-price combinatorial auction. In *Second ACM Conference on Electronic Commerce*, 21–29.

Wurman, P. R.; Wellman, M. P.; and Walsh, W. E. 1998. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second International Conference on Autonomous Agents*, 301–308.

Wurman, P. R.; Wellman, M. P.; and Walsh, W. E. to appear. A parametrization of the auction design space. *Games and Economic Behavior*.