University of Victoria

Department of Economics

Department Discussion Paper DDP0706

ISSN 1914-2838

Multiagent System Platform for Auction Simulations

Alan Mehlenbacher

Department of Economics, University of Victoria

Victoria, B.C., Canada V8W 2Y2

Abstract

I have developed a multiagent system platform that provides a valuable complement to the alternative auction research methods. The platform facilitates the development of heterogeneous agents and provides an experimental environment that is under the experimenter's complete control. Simulations with alternative learning methods results in impulse balance learning as the most promising approach for auctions.

JEL classification: C15, C72, D83.

Keywords: agent-based computational economics; agent learning;

Author Contact:
Alan Mehlenbacher, Dept. of Economics, University of Victoria, P.O. Box 1700, STN CSC, Victoria, B.C., Canada, V8W 2Y2; e-mail: amehlen@uvic.ca; Voice: (250) 721-8537; FAX: (250) 721-6214

1 Introduction

Multiagent systems have been applied to problems that are dynamic, complex, and distributed, and thus have been used to model the machines in manufacturing and process control, work orders in production scheduling, jobs and departments in business process optimization, planes in air traffic control, treatments and tests in hospital patient scheduling, messages in communication networks, and in many more areas (Weiss, 1999). In economics, agents have been used to model the behavior of, and interactions between, consumers, workers, families, firms, markets, regulatory agencies, and so on (see Tesfatsion, 2003 and 2006), and there have been a few applications of agent systems to auctions (Kim, 2007; Byde, 2002; and Hailu and Schilizzi, 2004). Section 2 discusses alternatives to multiagent systems in the analysis of auctions and why the multiagent system method was chosen for the current research.

An agent is a software entity that is autonomous, communicating, and adaptive. Autonomy means that an agent is driven by its own objectives, possesses resources (e.g., information) of its own, is capable of recording information about its environment, and can choose how to react to the environment. An agent is also a communicating software entity. Agents communicate directly with other agents by passing messages. Because each agent is autonomous, an agent must send requests to other agents for things to be done. For example, in this system, agents send messages to coordinate auctions, establish values, send bids, move to a new auctioneer, and so on. The agents are developed using object-oriented design. This means that the system consists of approximately 100 independent programs that are called "classes." Section 3 describes the design principles and, together with the Appendix, provides a guide to the classes.

An agent endeavours to improve its state (e.g., profit or revenue) in at least two ways. The first type of learning is reinforcement learning that uses feedback on results of actions to improve the results. The second type of learning is belief learning that involves updating beliefs about the environment, markets, and competitors, which may provide further improvements to the agent. Section 4 presents the results of an evaluation of different methods of agent learning.

2 Alternatives to Multiagent Systems

The major alternatives to using a multiagent system are mathematical theory, lab experiments, econometric models, and computational models.

2.1 Mathematical Theory

In a mathematical approach, mathematical machinery is developed (e.g., optimization, order statistics, supermodularity, etc.), simplifying assumptions are made, and results proven using theorems. However, applying these theoretical results to real-world auctions is problematic. For example, Milgrom (2004, p. 22) has identified the following problems: "Academic mechanism design theory relies on stark and exaggerated assumptions to reach theoretical conclusions that can sometimes be fragile. Among these are the assumptions (i) that bidders' beliefs are well formed and describable in terms of probabilities, (ii) that any differences in bidder beliefs reflect differences in their information, (iii) that bidders not only maximize, but also cling confidently to the belief that all other bidders maximize as well."

A more realistic model of industry bidders can be achieved by using artificially intelligent software agents that are designed to optimize adaptively using the information they receive from the seller. This approach directly addresses the problems identified by

Milgrom. There are fewer and more flexible simplifying assumptions[1], information to agents can be restricted to own information or expanded to information about other bids, and agents are programmed to maximize within the constraints of the abilities and information they have.

2.2 Lab Experiments

One approach to dealing with the limitations of theory has been to perform lab experiments, usually using student subjects (Kagel and Levin, 2002). These experiments have the benefit of bidders that encompass the wide range of human reasoning and feeling, but the disadvantage is the inexperience of the bidders. The subjects, whether students or adults from industry, must learn about the bidding environment from scratch, and this constrains the complexity of the mechanisms that can be studied in the lab. The subjects simply do not have the time to develop the richness of task-specific knowledge that is used again and again in a real-world industry auction (Dyer *et alia*, 1989). Lab experiments are also expensive and time consuming. Because of these constraints, the number of existing publications on human auction experiments is small, and the experiments are limited to relatively simple environments. However, the results provide useful benchmarks to assess the results of the computational models (see Section 3.3).

2.3 Econometric Models

There are two types of econometric methods that have been applied to auction data: regression analysis and structural models. For example, regression analysis is applied by De Silva *et alia* (2002, 2003) to bidding data from road construction procurement auctions, by Athey and Levin (2001) to data from U.S. timber auctions, and

[1] The single major assumption in this approach is the method the agents use to learn bidding strategies.

by Iledare *et alia* (2004) to data of oil lease auctions. The aim of the structural modelling approach is to recover from the auction data distributions of values and bids, in order to then analyze such topics as: whether the values are private, affiliated, or common; the extent of collusion; the impact of entry costs, and so on. Some researchers use parametric distribution functions (Li and Perrigne, 2003; Haile *et alia*, 2003; Li *et alia*, 2000) , but an increasing number of authors are using nonparametric methods (Campo *et alia*, 2003; Hendricks *et alia*, 2003). A thorough overview with several examples is contained in Paarsch and Hong (2006).

The major advantage of the econometric methods is that they use data from real auctions. The most serious disadvantage is that data is very difficult to obtain. In addition, econometric models are restrictive because the econometrician does not know the value estimates of the bidders, and all bidding strategies are based on these valuations. In addition, the structural models assume that bidders use a Bayesian Nash equilibrium bidding strategy, which is a very questionable assumption (Bajari and Hortacsu, 2005).

2.4 Computational Models

Another approach is to use a computational method that is not agent-based. Dynamic programming methods have been used to determine optimal bidding strategies for bidders. The use of these methods began with Friedman (1956) and is reviewed in Stark and Mayer (1971). Since a large volume of historical data on competitor bids is required to determine the optimal bidding strategy for a single bidder, the approach is useful for advising bidders in situations in which large volumes of data exists, such as online bidding (Tesauro and Bredin, 2002) and electricity markets (Attaviriyanupap *et*

alia, 2005). The main advantage of the dynamic programming approach is that it produces an optimized bidding strategy based on real-world data, but the disadvantage is that such datasets are few and far between.

In summary, there are advantages and disadvantages to each approach. The major advantages of agent computational modelling are that it does not require the simplifying assumptions of mathematical analysis, can model the experienced bidders in complex environments that are beyond the reach of lab experiments, does not require assumptions about values or Bayesian Nash equilibrium required by econometric methods, and does not require large amounts of historical data required by dynamic programming methods.

3 Design Methods

The object-oriented design methods are described in Section 3.1. Section 3.2 describes some of the major classes that have been developed for the basic agent functions, auctions, and other applications. In Section 3.3, I present the methods that are used to verify the validity of the agent models.

3.1 Object-Oriented Design

The multiagent system is designed using object-oriented principles and developed with Java, which is a platform-independent, object-oriented programming language. Two of the main advantages of an object-oriented approach are instantiation and extension. When we develop a Java program, we create a "class" that is an independent program with a specific purpose. This class can be used ("instantiated") one or more times to become an "object" that can then be executed. For example, I program a bidder agent class and then instantiate it many times to produce a large population of bidder agent

objects. Each class program consists of properties and methods. For a bidder agent, properties include name, current bid price, and value estimate; and methods include handling a message, moving to a new auction, and adjusting the bid price. All properties are for private use by the class, but these properties may sometimes be set or retrieved by other classes. Some methods are for public access but others are restricted for use only within the object. We can create base classes with common attributes and functions and extend them using more specific attributes and functions. For example, cars, trucks and busses have many common attributes and functions that we would place in a Vehicle class, which is then extended by the classes Car, Truck, and Bus. Then, we can extend the Car class to classes for SUV, Sedan, and so on. In this application, AbstractAgent class is extended by AbstractBidderAgent, which is extended by MultiUnitBidderAgent, which is extended by BankAgent. All of the extensions from the AbstractAgent class are illustrated in Figure 1.

3.2 Classes

The base multiagent platform is implemented with about 22 Java classes that are shown in Table 1. I have previously extended these classes in studies of repeated games with evolving finite automata using about 17 classes, repeated games with probabilistic finite automata using about 9 classes, and a simple trading economy using about 15 classes. The focus of this paper is auction simulations, which have been implemented using about 43 classes that are shown in Table 2. Some of the classes are described in the Appendix.

For auctions, there are auctioneer agents (sellers), bidder agents (buyers), and a coordinator agent to implement the important coordination mechanisms (Decker and

Lesser, 1995). The basic idea is that each auction format (e.g. single-unit sealed bid, single-unit English, etc.) has an associated auction class to handle the mechanics of fetching bids, choosing a winner, etc., and an associated conversation class that handles the communication between the bidders and the auctioneer. The auctioneer uses the appropriate auction class and the bidder uses the appropriate conversation class. The system supports a wide variety of options for current and future simulations. I can select the auction type (sale or procurement), payment type (first-price, second-price), bid format (sealed, English), numbers (of items being auctioned, auctions, auctioneers, and bidders), value (private value, common value, mixed value), and so on. The major design goal is to provide broad functionality so that different mechanisms can be studied for both single-unit and multi-unit auctions.

3.3 Verification of Multiagent Models

Multiagent systems, like other computational methods, have the challenge of verification. In my work, I use four approaches to verification.

First, verification is facilitated in multiagent models by explicitly modelling real world objects and relationships. For example, in the multiagent model of consumer choice in a transportation system, households, persons, and families are modeled with realistic behaviours (Salvini and Miller, 2005) based on observations and data. In my multiagent model, bidder learning is modeled using adjustment rules that are based on results from lab experiments (Ockenfels and Selten, 2005; Neugebauer and Selten, 2006).

Second, verification is strengthened by comparing simulation results to data from lab experiments for the simple cases for which there are such results. This is virtually impossible for very complex auction mechanisms, and in these cases test data itself is

generated computationally (Leyton-Brown and Shoham, 2006). For games that are less complex than auctions, there are good opportunities to test learning models against data from lab experiments (Arifovic *et alia*, 2006). The single-unit sealed bid and English auctions that I study are of moderate complexity (Mehlenbacher, 2007a, 2007b), and the results can be verified against lab experiments in the simple cases of, for example, pure private values and pure common values. Agreement with this data lends credibility to the validity of the model in the more complex cases.

Third, the model must have as few parameters as possible, and the model must produce results that are stable within a range of the parameters. For example, if a reasonable range of one parameter is [0, 1], the model must be stable within a subset of this range, e.g. [0.3, 0.8]. If there are two or more parameters, then the model must be stable for an intersection of subranges. This is admittedly a subjective process, but it provides a relative measure of confidence in the model if the results are stable over [0.3, 0.8] when the results of another model are stable over [0.5, 0.7].

Fourth, the model must converge for the variables being studied. These convergence results are important, since it is impossible to interpret auction results for bid strategies, profit, revenue, and efficiency when there is no convergence. For example, without convergence the results are different when we stop the simulation in period t+10 from the results when stopping the simulation in period t. Also, the fact that convergence occurs in less than, say, 100 periods makes it reasonable to infer that the bid strategies of human agents could converge in a realistic number of real-world auctions.

4 Learning Models

An agent can improve its profit through learning in at least two ways. The first type ("belief learning") is described in Section 4.1. Belief learning involves updating beliefs about the environment, markets, and competitors, which may provide further improvements to the agent. Several alternative methods of the second type ("action learning") are described in Section 4.2, and Section 4.3 presents an evaluation of the alternative methods. I have implemented the two types of learning with about 31 Java classes in three packages (Table 3).

4.1 Belief Learning

Belief learning is modelled by probabilistic networks (also called Bayesian networks and belief networks), and I developed the Java classes shown in Table 3 using the concepts and algorithms in Cowell *et alia* (1999) and Shafer (1996)[2]. Briefly, a probabilistic network is a directed acyclic graph in which nodes represent the random variables, an arrow from node X to node Y means that X has a direct influence on Y, and each dependent node has a conditional probability table. In constructing a probabilistic network, you choose[3] the set of relevant variables that describe the beliefs, add the nodes by adding the "root causes" first, then the variables they influence, and so on until you reach the leaves which have no direct causal influence on other variables. Finally, you define the conditional probability table for each node, which provides the probability that a given node state will occur, given the states in the preceding nodes. In order for the

---

[2] I developed a compact package of software for agents, but there are several products available that are oriented towards working with large datasets, e.g., Hugin.

[3] Given a large enough dataset, it is possible to for an agent to learn the structure of its Bayesian network (Heckerman, 1998).

agent to make inferences from observed facts, the network must be converted into a more compact form called a junction tree. First, the network is moralized, which means that all parents of a node are joined (or "married" and thus becoming "moral"!). Second, the moralized network is triangulated, which means that every polygon larger than a triangle is filled in to produce a network of connected triangles. Third, the triangles are converted into nodes of a junction tree, i.e., a junction tree is network of the triangles. During this process, the conditional probability tables are modified appropriately. Now when the agent observes some change in the environment, the change is propagated to all the nodes of the junction tree and the conditional probability tables are updated. To the agent, this means that its belief system is updated to accommodate the new information.

I performed many computational experiments with agents developing beliefs based on information they compile using the bid distribution classes listed in Table 2. These classes provide an agent with distributions of its own results for profit, winning, etc. and the results of other agents (for this, the I3 agents were provided with the identity of other bidders) in order to develop beliefs about relative strength. The bidders then used these beliefs to modify their ongoing bid strategy depending on the specific opponents in each auction. However, I found that using belief learning in this context did not significantly change the overall results for profit, revenue, and efficiency compared to agents who did not use belief learning. This result occurred because the auction-specific strategies are stationary around the ongoing bidding strategy and thus had no effect on the averages. Therefore, in the interests of parsimony, I removed belief learning from the model and have therefore not used it in the current research on auctions. However, I

believe that belief learning has potential application in other types of multiagent models, especially in macroeconomic models where expectations play a major role.

4.2 Action Learning Alternatives

There is considerable scope for choosing the action learning model for the agents. Alternatives for action learning include simple reinforcement learning, reinforcement learning methods, experience-weighted attraction, learning direction theory, genetic algorithms, and neural networks.

Simple reinforcement learning uses profit to reinforce action weights. Thus, the actions are usually modelled as discrete states that can be weighted, and only one type of information is used (profit). This method has been applied with some success to normal form games (Erev and Roth , 1998) and to auctions by Armantier (2004), Daniel *et alia* (1998), Seale *et alia* (2001), Bower and Bunn (2001), and Nicolaisen *et alia* (2001). I experimented with this simple reinforcement method, but I also extended it using two types of states: the average profit of the bidder and the average profit of the bidder's opponents. In the first, the state is 0 if the bidder's own average profit is negative, and 1 if it is positive. In the second, the state is 0 if the opponents on average are losers (negative average profits), and 1 if the opponents are on average profitable. The action weights occur in pairs, one for each state, that are updated as in the simple reinforcement learning but now depending on the state.

More sophisticated methods of reinforcement learning have not previously been used in auctions, so I performed simulations for common-value first-price sealed-bid auctions using dynamic programming, temporal difference, and Q-Sarsa methods (Sutton and Barto, 1998). The dynamic programming method reinforces actions by both actual

profit and expected future profits (based on past profits) as in Sutton and Barto (1998, Chapter 4). I use the states as described above for the extended simple reinforcement learning methods, along with a state transition table containing the probabilities of transition from one state to another. The weights are then updated by combining the profit for the current state with the weights for the states indicated by the state transition table. In the temporal difference method, the agent uses profit to reinforce the current state-action pair as well as the state-action pair that preceded the current action. This approach closely follows Sutton and Barto (1998, Chapter 6). Q-Sarsa learning involves reinforcing the current state-action pair as well as all of the state-action pairs that preceded this action. This involves the use of eligibility traces as described in Sutton and Barto (1998, Chapter 7).

Experience-weighted attraction uses profit for winners and foregone profit for losers to reinforce discrete action states. Camerer has used this method extensively in games (Camerer, 2003; Camerer *et alia*, 2002), and it has been applied to auctions by Rapoport and Amaldoss (2004).

Learning direction theory (Selten, 1998) has been applied as impulse balance learning to auctions by Selten and Buchta (1998), Selten *et alia* (2005), Ockenfels and Selten (2005), and Neugebauer and Selten (2006). The method has also been used to interpret experimental data by Garvin and Kagel (1994) and Kagel and Levin (1999). Impulse balance learning uses foregone profit upon losing as an upward impulse on a continuous bidding strategy and money on the table upon winning as a downward impulse. The downward impulse is weighted by a balance factor that is the ratio of the expected value of the upward impulse to the downward impulse. I augmented this

method to include adjustment using actual loss by the winner and foregone loss (the amount the agent would have lost if it had won) by the losers. The agent adjusts the bid strategy for a loser to bid higher depending on the level of foregone profit and bid lower depending upon the level of foregone loss. A winner reduces its bid in proportion to the money on the table if it made a profit and in proportion to the actual loss if it made a loss.

Genetic algorithms have been applied to auctions by Dawid (1999) and Andreoni and Miller (1995), and neural networks have been used by Bengio *et alia* (1999). Genetic algorithms require discrete states, and genetic algorithms and neural networks use only profit to guide the optimization.

4.3 Action Learning Evaluation

To guide selection of an appropriate learning method, we need to establish the level of intelligence required. Since the research is motivated by an interest in real-world asset-sale auctions such as those for timber sales, drilling licences, and treasury securities, the agents must simulate experienced real-world auction bidders. A credible learning method for simulating these sophisticated bidders must satisfy four criteria: (1) be a realistic representation of how humans can potentially maximize profit in the auction environment, (2) potentially utilize all available information feedback, (3) handle continuous bids, and (4) be extendable.

Human reasoning cannot be captured with a single computational paradigm but is situational and adaptable and involves a combination of heuristics and rules-of-thumb, together with logic and optimization when required (Dyer *et alia*, 1989; Hutchinson and Gigerenzer, 2005; Ohtsubo and Rapoport, 2006). In a study using experienced construction executives, Dyer *et alia* (1989, p. 115) concluded that “success in the field

thus derives not from conformity to a narrow notion of rationality, but from acquiring and utilizing detailed knowledge of a particular market environment." Genetic algorithms and neural networks are general-purpose methods that require the researcher to fit the reasoning to the algorithm and do not accommodate the specific economic reasoning that goes into developing the various auction strategies. The more straightforward methods like simple reinforcement, experience-weighted attraction, and impulse balance are superior in this regard.

Research in auctions (Dyer and Kagel, 1996; Dyer *et alia*, 1989) demonstrates that bidders acquire and use detailed knowledge in their specific auction environments. Thus, a realistic learning method must accommodate different levels of information and utilize more than just profit. Except for experience weighted attraction and impulse balance, the methods use only profit and are thus too informationally restrictive. Experience weighted attraction uses profit and foregone profit, but impulse balance uses money on the table and foregone profit and can be extended to use profit, loss, and foregone loss.

A further limitation of most of the learning methods is that they are implemented using discrete states. If the discretization is too fine, the implementation is too inefficient; if it is too coarse, the bidding is not realistic enough for meaningful economic conclusions. The impulse balance model is the exception since it deals efficiently with continuous[4] increases or decreases in the bidding strategy.

Finally, the method should be extendable so that other auction mechanisms and context variables can be accommodated in future studies, but only a method like impulse

---

[4] Continuous in this context means real numbers that are not restricted to integers and that are represented by 32 bits.

balance can be practically extended in this way. The basic method uses money on the table and foregone profit, and I have extended it to use actual profit, actual loss, foregone loss, and estimates of these impulses when information is restricted.

In summary, the method that comes closest to satisfying the criteria is the augmented impulse balance method. Thus, this method is developed and expanded in subsequent chapters.

5 Conclusion

The multiagent system approach with agents using modified impulse balance learning has the advantages of not requiring the simplifying assumptions of mathematical theory and of not being constrained in complexity by the limited experience of experimental subjects. Impulse balance learning provides the best foundation for learning in auctions since it is a realistic representation of experienced human bidders, utilizes several types of information feedback, handles continuous bids, and is extendable. Therefore, I modify and extend the impulse balance method in multiagent system simulations of sealed-bid auctions (Mehlenbacher, 2007a), English auctions (Mehlenbacher, 2007b), and treasury auctions (Mehlenbacher, 2007c).

6 Acknowledgements

I am indebted to David Scoones, Linda Welling, Don Ferguson, and Tony Marley for their valuable suggestions. I am also grateful to the Social Sciences and Humanities Research Council of Canada for scholarship support during the course of this research.

7 References

Andreoni, J., Miller, J.H., 1995. Auctions with artificial adaptive agents. *Games and Economic Behavior*. 19: 39-64.

Arifovic, J., McKelvey, R. Pevnitskaya, S., 2006. An initial implementation of the turing tournament to learning in repeated two person games. *Games and Economic Behavior,* 57: 93-122.

Armantier, O., 2004. Does observation influence learning? *Games and Economic Behavior*. 46: 221-239.

Athey, S., Levin, J., 2001. Information and competition in U.S. forest service timber auctions. *Journal of Political Economy*, 109, No. 2, 375–417.

Attaviriyanupap, P., Kita, H., Tanaka, E., Hasegawa, J., 2005. New bidding strategy formulation for day-ahead energy and reserve markets based on evolutionary programming. *Electrical Power and Energy Systems*, 27: 157–167.

Bajari, P. and Hortacsu, A., 2005. Are structural estimates of auction models reasonable? Evidence from experimental data. *Journal of Political Economy*, 113(4): 703–741.

Bengio, S., Bengio, Y., Robert, J., Belanger, G., 1999. Stochastic learning of strategic equilibria for auctions. *Neural Computation*. 11: 1199-1209.

Bower, J., Bunn, D., 2001. Experimental analysis of the efficiency of uniform-price versus discriminatory auctions in the england and wales electricity market. *Games and Economic Behavior*. 25: 561-592.

Byde, A., 2002. Applying evolutionary game theory to auction mechanism design. HPL-2002-321, Hewlet-Packard Company.

Camerer, C., 2003, *Behavioral game theory: experiments in strategic interaction*. Princeton University Press.

Camerer, C., Ho, T.-H., Chong, J. K., 2002. Sophisticated experience-weighted attraction learning and strategic teaching in repeated games. *Journal of Economic Theory*. 104: 137-188.

Campo, S., Perrigne, I. , Vuong, Q., 2003. Asymmetry in first-price auctions with affiliated private values. *Journal of Applied Econometrics*, 18: 179-207.

Cowell, R. G., A. P. Dawid, S.L. Lauritzen, Spiegelhalter, D.J. 1999, *Probabilistic Networks and Expert Systems*, Springer.

Daniel, T. E., Seale, D.A., Rapoport, A., 1998. Strategic play and adaptive learning in the sealed-bid bargaining mechanism. *Journal of Mathematical Psychology*. 42: 133-166.

Dawid, H., 1999. On the convergence of genetic learning in a double auction market. *Journal of Economic Dynamics and Control*. 23: 1545-1567.

De Silva, D. G., Dunne, T., Kosmopoulou, G., 2002. Sequential bidding in auctions of construction contracts. *Economics Letters*, 76: 239–244.

De Silva, D. G., Dunne, T., Kosmopoulou, G., 2003. An empirical analysis of entrant and incumbent bidding in road construction contracts. *The Journal of Industrial Economics*, 51(3): 295–316.

Dyer, D., Kagel, J. H., and Levin, D., 1989. A comparison of naïve and experienced bidders in common value offer auctions: laboratory analysis. *The Economic Journal,*. 99, 108-115.

Dyer, D., Kagel, J.H., 1996. Bidding in common value auctions: how the commercial construction industry corrects for the winner's curse. *Management Science*. 42, 1463-1475.

Erev, I., Roth, A.E., 1998. Predicting how people play games: reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*. 88: 848-881.

FIPA, 2002. *FIPA Communicative Act Library Specification*. Foundation for Intelligent Physical Agents, Document SC00037J.

Friedman, L., 1956. A competitive-bidding strategy. *Operations Research*, 4(1):104-112.

Garvin, S., Kagel, J.H., 1994. Learning in common value auctions: some initial observations. *Journal of Economic Behavior and Organization*. 25: 351-372.

Haile, P. A., Hong, H., Shum, M., 2003. Nonparametric tests for common values in first-price sealed-bid auctions. *Yale University Working Paper*.

Hailu, A., Schilizzi, S., 2004. Are auctions more efficient than fixed price schemes when bidders learn?" *Australian Journal of Management*. 29: 147-168.

Heckerman, D., 1998. A tutorial on learning with Bayesian networks. In: Jordon, M.I. (Ed), *Learning in Graphical Models,*. Kluwer Academic Publishers, pp. 301-354.

Hendricks, K., Pinkse, J., Porter, R. H., 2003. Empirical implications of equilibrium bidding in first-price, symmetric, common value auctions. *The Review of Economics Studies*, 70, 115-145.

Hendricks, K., Porter, R. H., and Wilson, C. A.,1994. Auctions for oil and gas leases with an informed bidder and a random reservation price. *Econometrica*, 62, No. 6, 1415-1444.

Hutchinson, J. C., Gigerenzer, G., 2005. Simple heuristics and rules of thumb: Where psychologists and behavioral biologists might meet. *Behavioral Processes*. 69: 97-124.

Iledare, O., Pulsipher, A., Olatubi, W., Mesyanzhinov, D. 2004. an empirical analysis of the determinants and value of high bonus bids for petroleum leases in the U.S. outer continental shelf (OCS). *Energy Economics*, 26: 239-259.

Kagel, J.H., Levin, D., 1999. Common value auctions with insider information. *Econometrica*. 67: 1219-1238.

Kagel, J.H., Levin, D., 2002. *Common Value Auctions and the Winner's Curse*. Princeton University Press.

Kim, Y. S., 2007. Maximizing sellers' welfare in online auction by simulating bidders' proxy bidding agents. *Expert Systems with Applications* 32(2): 289-298.

Milgrom, P., 2004. *Putting Auction Theory to Work*. Cambridge University Press.

Leyton-Brown, K. and Y. Shoham, 2006. A test suite for combinatorial auctions. In: Cramton, P., Shoham, Y., and Steinberg, R. (Eds), *Combinatoria Auctions.* The MIT Press, pp. 451-478.

Li, T., Perrigne, I, 2003. Timber sale auctions with random reserve prices. *Review of Economics and Statistics*, 85. 15, No. 1, 189-200.

Li, T., I. Perrigne, Vuong, Q., 2000. Conditional independent private information in OCS wildcat auctions. *Journal of Econometrics*, 98, 129-161.

Mehlenbacher, A., 2007a. Multiagent system simulations of sealed-bid auctions with two-dimensional value signals. University of Victoria, Economics Department Discussion Paper, DDP0707.

Mehlenbacher, A., 2007b. Multiagent system simulations of signal averaging in English auctions with two-dimensional value signals. University of Victoria, Economics Department Discussion Paper, DDP0708.

Mehlenbacher, A., 2007c. Multiagent system simulations of treasury auctions. University of Victoria, Economics Department Discussion Paper, DDP0709.

Neugebauer, T., Selten, R., 2006. Individual behavior of first-price auctions: the importance of information feedback in computerized experimental markets. *Games and Economic Behavior* , 54, 183-204.

Nicolaisen, J., Petrov, V., Tesfatsion, L., 2001. Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. *ISU Economic Report No. 52*, Iowa State University.

Ockenfels, A., Selten, R., 2005. Impulse balance equilibrium and feedback in first price auctions. *Games and Economic Behavior.* 51: 155-170.

Ohtsubo, Y. , Rapoport, A., 2006. Depth of reasoning in strategic form games. *Journal of Socio-Economics*. 35: 31-47.

Paarsch, H. J., Hong H., 2006, *An Introduction to the Structural Econometrics of Auction Data*. The MIT Press.

Rapoport, A., Amaldoss, W., 2004. Mixed-strategy play in single-stage first-price all-pay auctions with symmetric bidders. Journal of Economic Behavior and Organization. 54: 585–607.

Salvini, P.A. and E.J. Miller, 2005. ILUTE: An operational prototype of a comprehensive microsimulation model of urban systems", *Networks and Spatial Economics*, 5: 217-234.

Seale, D. A., Daniel, T.E., Rapoport, A., 2001. The information advantage in two-person bargaining with incomplete information. *Journal of Economic Behavior and Organization*. 44: 177-200.

Selten, R., 1998. Features of experimentally observed bounded rationality. *European Economic Review*, 42: 413-436.

Selten, R., Buchta, J., 1998. Experimental sealed bid first price auctions with directly observed bid functions. In: Budescu, D. V., Erev, I., and Zwick, R. (Eds), *Games and Human Behavior: Essays*. Lawrence Erlbaum Associates, pp. 53-78.

Selten, R., Abbink, K., Cox, R., 2005. Learning direction theory and the winner's curse. *Experimental Economics* 8: 5-20.

Shafer, G., 1996, *Probabilistic Expert Systems*, SIAM.

Sutton, R. S., Barto, A. G., 1998, *Reinforcement learning: an introduction*, The MIT Press.

Tesauro, G., Bredin, J. L. Strategic sequential bidding in auctions using dynamic programming. in: *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, ACM Press.

Tesfatsion, L., 2003. Agent-based computational economics: modeling economies as complex adaptive systems. Information Sciences 149: 263-269.

Tesfatsion, L. , Judd, K., 2006. *Handbook of Computational Economics: Volume 2 Agent-Based Computational Economics*. Elsevier.

Weiss, G., 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press.

8 Tables

Table 1. Base Packages and Classes		
Package	Class	Extends
1. agent	1. AbstractAgent 2. AgentInfo 3. Registry	
2. distributions	4. RandomNumber 5. Beta 6. Normal 7. Uniform	 RandomNumber RandomNumber RandomNumber
3. grid	8. Cell 9. Coordinates 10. Grid 11. Options	
4. gui	12. BasicMenu 13. GuiFrame 14. HelpFrame 15. InfoPanel	JMenuBar JFrame JFrame JPanel
5. statistics	16. Moments 17. Regression 18. TimeSeries	
6. support.filesupport	19. Tracing	
7. support.guisupport	20. Console 21. MenuCreator 22. RadioButtonPanel	 JPanel

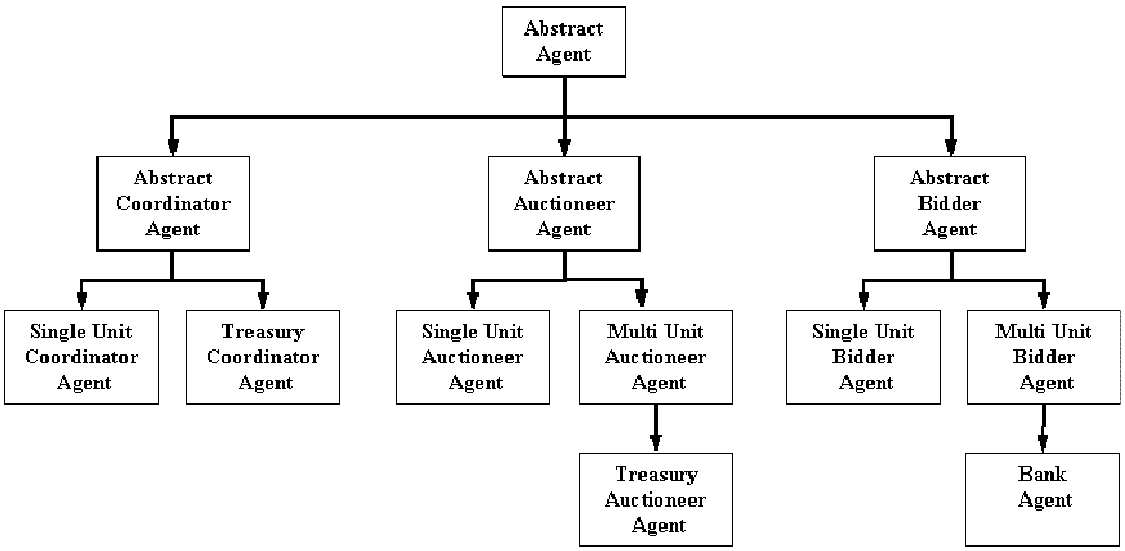
Table 2. Auction Packages and Classes

Package	Class	Extends
1. auction.agent	1. AbstractAuctioneerAgent	AbstractAgent
	2. AbstractBidderAgent	AbstractAgent
	3. AbstractCoordinatorAgent	AbstractAgent
	4. BankAgent	MultiUnitBidderAgent
	5. BidDistributions	
	6. CentralBankAgent	MultiUnitAuctionerAgent
	7. MultiUnitAuctionerAgent	AbstractAuctioneerAgent
	8. MultiUnitBidderAgent	AbstractBidderAgent
	9. MultiUnitBidDistributions	BidDistributions
	10. SingleUnitAuctioneerAgent	AbstractAuctioneerAgent
	11. SingleUnitBidderAgent	AbstractBidderAgent
	12. SingleUnitBidDistributions	BidDistributions
	13. SingleUnitCoordinatorAgent	AbstractCoordinatorAgent
	14. TreasuryCoordinatorAgent	AbstractCoordinatorAgent
2. auction.bidding	15. Auction	
	16. AuctionResult	
	17. Bid	
	18. MultiUnit	Auction
	19. MultiUnitEnglish	MultiUnit
	20. MultiUnitSealed	MultiUnit
	21. SecondaryTreasuryMarket	
	22. SingleUnit	Auction
	23. SingleUnitEnglish	SingleUnit
	24. SingleUnitSealed	SingleUnit
3. auction.conversation	25. MultiUnitConversation	
	26. MultiUnitSealedConversation	MultiUnitConversation
	27. MultiUnitEnglishConversation	MultiUnitConversation
	28. SingleUnitConversation	
	29. SingleUnitSealedConversation	SingleUnitConversation
	30. SingleUnitEnglishConversation	SingleUnitConversation
4. auction.grid	31. AuctionAgentCell	Cell
	32. AuctionAgentGrid	Grid
	33. AuctionAgentOptions	Options
5. auction.gui	34. AuctionAgentGuiFrame	GuiFrame
	35. AuctionAgentOptionDialogSingleUnit	JDialog
	36. AuctionAgentOptionDialogTreasury	JDialog
	37. SliderHandlerSingleUnit	
	38. SliderHandlerTreasury	
6. auction.simulation	39. AveragingImpulseOutput	
	40. BidImpulseOutput	
	41. EfficiencyOutput	
	42. ProfitOutput	
	43. RevenueOutput	

Table 3. Learning Packages and Classes		
Package	Class	Extends
1. learning	1. Action	
	2. SingleUnitLearning	
	3. Rla	SingleUnitLearning
	4. RLas	SingleUnitLearning
	5. EWA	SingleUnitLearning
	6. DP	SingleUnitLearning
	7. TD	SingleUnitLearning
	8. Q	SingleUnitLearning
	9. IB	SingleUnitLearning
	10. IBA	SingleUnitLearning
	11. SingleUnitImpulse	SingleUnitLearning
	12. MultiUnitAdjustment	
	13. MultiUnitRules	MultiUnitAdjustment
2. probnet.algorithm	14. CreateJunctionTree	
	15. FindCliques	
	16. InitializePotentials	
	17. Moralize	
	18. PerfectOrder	
	19. Triangulate	
3. probnet.bayesnetwork	20. ActiveBN	
	21. BayesNetwork	Network
	22. BayesNode	Node
	23. ChainComponent	Node
	24. JunctionTree	Network
	25. JunctionTreeNode	Node
	26. Key	
	27. Network	
	28. Node	
	29. PotentialTable	Table
	30. Separator	Node
	31. Table	

9 Figures

Figure 1. Simple Class Diagram for Auction Bidder Classes



10 Appendix

This Appendix describes some of the design concepts used in implementing the functionality for Agents, Conversations, and Auctions.

Agent Classes

There is a base AbstractAgent class that provides functions common to all agents. AbstractCoordinatorAgent, AbstractAuctioneerAgent, and AbstractBidderAgent classes extend AbstractAgent and then these in turn are extended for single-unit, multi-unit, and treasury auctions.

A coordinator agent has two major tasks: to create the other agents and coordinate the auctions. For each auction, the coordinator broadcasts a message to every auctioneer to hold an auction and directs the agents to move if there is more than one auctioneer. The coordinator can randomly distribute the bidders equally or unequally to the auctioneers.

An auctioneer agent has three major tasks: execute the auction, notify the bidders, and print results. An auctioneer creates an auction object of the appropriate type (e.g., SingleUnitSealed, MultiUnitSealed, etc.) based on the type of auction that has been set by the experimenter. The auctioneer then uses the auction object to execute the auction, fetch bids, pick winners, and send results to the bidders. For the benefit of the experimenter, the auctioneer agent also prints results for the experimenter using classes in the auction.simulation package.

A bidder agent has three major tasks: learn how to improve bidding, calculate a bid and send it to the Auctioneer using the Bid class (The Bid class holds attributes for a bid: the bidder, value signal, action that led to the bid, and the bid amount plus the resulting ranking, profit, foregone profit, and so on.), and move to a new auctioneer (if the *Bidders* option is "random"). Each bidder agent has a learnBidFactor method that is called when the auction object requests

the bidder's participation in an auction. The learnBidFactor method in turn calls one of the learning algorithms (see Section 3) to calculate the bid factor. For the benefit of the experimenter, the bidder agent also prints results for the experimenter using the classes in the auction.simulation package.

Conversation Classes

The bidder communicates with the auctioneer using protocols encapsulated in conversation classes. The message types are consistent with FIPA Agent Communication Language (FIPA, 2002).

The SingleUnitConversation and MultiUnitCoversation classes tell the bidders to learn and inform them of auction results. They are extended by the classes for sealed-bid and English auctions that retrieve the bids from the bidders. The process involves a single message for sealed-bid auctions, but involves many messages for the English auctions. Starting with a low price, SingleUnitEnglish iterates through a loop: send to active bidders the price and the latest dropout price; remove bidders who reject this price level from the auction; increment the price.

Auction Classes

Each auction involves the following four major functions: manage the auction, fetch bids, pick the winner(s), and calculate payment(s). The processes of auction management, picking the winner, and calculating the payment are handled by the SingleUnit and MultiUnit classes. Since the process of fetching bids differs for sealed-bid and English auctions, this function is handled by extensions of these classes.