AGENT-BASED COMMON VALUE AUCTIONS

By

CHRISTOPHER BOYER

Bachelor of Science in Agribusiness
Texas A&M University
College Station, TX
2006

Master of Science in Agricultural Economics
Texas A&M University
College Station, TX
2008

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2011

AGENT-BASED COMMON VALUE AUCTIONS

Dissertation Approved:

Dr. B. Wade Brorsen

Dissertation Adviser

Dr. Chanjin Chung

Dr. Kellie Raper

Dr. James R. Fain

Outside Committee Member

Dr. Sheryl A. Tucker

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

IMPLICATIONS OF A RESERVE PRICE IN AN AGENT-BASED COMMON-VALUE AUCTION

Introduction

Reserve prices (or minimum bids) are widely used in various auction structures and influence bid prices when imposed. Riley and Samuelson (1981) extend Vickrey (1961) to include a reserve price in an independent private values auction, and theoretically illustrate several possible outcomes when a seller imposes a reserve price. They show that reserve prices can increase buyers' bid prices, decrease buyers' revenues, increase sellers' revenues, and decrease the probability of the item being sold. The theoretical predictions in these papers are the foundation for much of the literature on reserve prices in auctions.

Riley and Samuelson's (1981) independent private values auction model has been used to estimate the impact of reserve prices in real estate auctions (McAfee, Quan, and Vincent 2002), timber auctions (Paarsch 1997), and oil lease auctions (Hendricks et al. 1994). Today, nearly anyone with computer access can set up an eBay account and sell various items with a reserve price in an online auction (Bajari and Hortacsu 2003; Reiley 2006). Using data from online auctions such as eBay has become a popular approach to analyze the impacts of reserve prices. For example, Reiley (2006) sold *Magic: The Gathering* game cards on eBay without a reserve price and with various levels of reserve prices. As predicted, Reiley (2006) finds that increasing the reserve price decreases the number of buyers participating in the auction, which suggests that reserve prices can act as a barrier to entry for buyers. Also, buyers' bid prices increased when a reserve price was imposed by the seller. Reiley (2006) follows Riley and Samuelson's (1981) independent private values auction model to make these conclusions; even though Reiley (2006 p. 198) acknowledges that common-value auction theory could be argued as more appropriate. Reiley (2006) finds support for Riley and Samuelson's (1981) hypotheses about the impacts of a reserve price in an auction.

Little attention, however, has been given to the influence of a reserve price on a common-value auction. A general theoretical framework for a common-value auction with a reserve price is introduced by McAfee and Vincent (1992). They allow the number of buyers to be endogenous in finding the optimal reserve price, and they find the reserve price set by the government in oil lease auctions is less then optimal. Levin and Smith (1996) expand on Riley and Samuelson's (1981) theoretical model and include a common-value auction as a special case. Their theoretical model shows that the optimal reserve price is a function of the number of buyers participating in the auction and approaches the sellers' true values as the number of buyers participating in the auction increases. These theoretical papers give some insight into the effects of a reserve price in a common-value auction, but do not solve for equilibriums of the common-value auction. More recently, Goncalves (2009) established an online second-price sealed-bid common-value auction, and found that the seller can increase revenue by setting a reserve price.

With a reserve price, the seller is able to increase the bid price, but the quantity sold decreases (Goncalves 2009). Conversely, setting the reserve price above the highest bid price offered by the buyers in the auction results in a reduction in the seller's revenues (Goncalves 2009). Additionally, Goncalves (2009) finds that increasing competition by including an additional buyer does not increase revenues as much as letting the seller impose a reserve price. Reiley (2006) refers to experiments similar to his and Goncalves (2009) as "field experiments." These field experiments cannot control for all the exogenous factors that might influence the results like in a laboratory experiment (Reiley 2006). The results from these types of studies are recognized by Reiley (2006) and Goncalves (2009) as being noisy, and that reproducing them under controlled laboratory settings could be beneficial to further test the theory. However, human experiments in controlled laboratories can be expensive.

An alternative technique that economists can use to test economic theories other than using human or field experiments is agent-based computational modeling (Arifovic 1996; Bonabeua 2002; Alkemade, La Poutré, and Ammam 2006). Agent-based computational models are artificial markets where interactive agents trade (Tesfatsion 2001). These models can simulate artificial economic markets when data are not available or are expensive to obtain and when there is not an analytical solution. Equilibrium prices in various auction markets have been successfully obtained with agent-based models (Tesfatsion 2001; Guerci, Stefano, and Cincotti. 2008; Hailu and Thoyer 2010). Our review of literature shows the impact of a reserve price in a first-price common-value auction on revenues, bid prices, and the probability of a transaction occurring has not been measured with a theoretical model, or with an agent-based model.

The analytical solution of a first-price common-value auction with a reserve price is perhaps infeasible (Gordy 1998), and reproducing this auction in a laboratory experiment is likely expensive. Thus, using an agent-based auction model is an appropriate method to find the equilibrium bid price and optimal reserve price in a first-price common-value auction. Additionally, a first-price common-value auction with a reserve price has never been solved using an agent-based computational model, which makes this model a unique contribution to the agent-based model literature.

The base model we establish is an agent-based first-price common-value auction similar to Kagel and Levin's (1986) common-value auction model. The objective is to evaluate the impact of a reserve price in this artificial auction and to determine buyers' optimal bid prices and sellers' optimal reserve prices. Buyers choose their expected-revenue-maximizing bid price for the item being sold while sellers choose their expected-revenue-maximizing reserve price. Buyers and sellers use a particle swarm optimization (PSO) learning algorithm developed by Eberhart and Kennedy (1995) and extended by Zhang and Brorsen (2009) to maximize expected revenue. The PSO algorithm uses the buyers' and sellers' previous and current best bid/reserve strategies to direct their next bid/reserve strategies to the expected revenue maximizing solution. The results will also inform us on the influence of a reserve price on market power (bid-shading) and overbidding (the winner's curse) in a first-price common-value artificial auction. Kagel and Levin's (1986) common-value auction theory assumes bids are uniformly distributed, and in the base model, we follow this assumption. The agent-based model provides an opportunity to extend Kagel and Levin's (1986) theory by giving one of the buyers a normally distributed bid function and including an additional buyer.

Theory

<u>Common Value Auction</u>

Auctions are normally classified into three types: (1) common value, (2) private value, and (3) affiliated value (Milgrom and Weber 1982; Krishna 2002). In a common-value auction, the resale value of the item is equal across all buyers (Kagel and Levin 1986; Kagel and Levin 2002). However, uncertainty about the value of the item being sold results in each buyer having a different estimate of value such as in oil lease auctions (Kagel and Levin 1986; Kagel and Levin 2002). Private value auction bids are determined by a buyer’s independent private value for the item (Paarsch 1992). The value of the item for each buyer is unique to that buyer and each buyer knows his/her value with certainty, but the value of the item for other buyers is unknown. A private value auction typically includes non-durable items that are unique such as art (Paarsch 1992). The affiliated value auction is an intermediary case where the value of the item differs across buyers, but each buyer’s resale value for the item is similar (Pinkse and Tan 2005). In reality, auctions rarely strictly follow the criteria for one of these three types of auctions, and they are more likely to be a hybrid of all three auction types (Corrigan and Rousu 2011).

We establish an agent-based, first-price common-value auction model that is similar to Kagel and Levin’s (1986) model. The artificial auction is a duopsony and oligopsony auction structure, meaning the number of buyers N (N=2 and N =3) is less than the number of sellers M. Buyers act as price-setters and sellers are price-takers, which could result in buyers imposing market power on sellers by reducing their bids (bid-shading) below the market value (Varian 1992). Bid-shading is possible in various auction structures (Robinson 1985; McAfee and McMillan 1987; Klemperer 1999). McAfee and McMillan (1987) find that in infinitely repeated oligopoly auction games, a collusive outcome is sustainable, and Robinson (1985) finds bid-shading is sustainable for a first- and second-price auction.

On the other hand, a phenomenon of common-value auction theory is the winner's curse (Capen, Clapp, and Campbell 1971). This situation occurs when buyers estimate the value of an auctioned item to be greater than its actual value, and bid their estimated values for the item. This process results in the winning buyer receiving negative profits for the item (Capen, Clapp, and Campbell 1971). The winner's curse is well documented in experimental (Kagel and Levin 2002) and actual (Capen, Clapp, and Campbell 1971) common-value auctions. In human experiments, the winner's curse normally increases as the number of bidders increase, and when there are less than three buyers, the winner's curse is close to zero (Kagel and Levin 2002). The winner's curse can also decrease as buyers become more experienced (Kagel and Levin 2002). The PSO learning algorithm is similar to giving buyers an artificial experience since previous bid prices are considered in determining buyers' optimal bid prices for the current trading period (or iteration). Also, few buyers are participating in the auction. For these reasons, we anticipate the winner's curse will not occur.

Kagel and Levin (1986) developed a risk-neutral bid function for a first-price common-value auction. Their model is derived from Wilson (1977) and Milgrom and Weber (1982) and has been tested and supported several times with human experiments (Kagel and Levin 2002). Kagel and Levin (1986) assume that in the auction, the retail value of the item is *X*, which is randomly drawn from a set of values that are uniformly distributed in the interval $[\underline{x}, \bar{x}]$. Buyer *i* has an information signal x_i, which is distributed uniformly between a lower bound of max $\{x_i - \delta, \underline{x}\}$ and an upper bound of min $\{x_i + \delta, \bar{x}\}$ where δ is a deterministic value told to buyers before the auction. The bid function is

$$b(x_i) = x_i - \delta + Y(N, \delta) \tag{1.1}$$

where x_i is buyer *i*'s information signal, and $Y = [2\delta/(N+1)]\exp[-(N/2\delta)(x_i - (\underline{x} + \delta))]$, where *N* is the number of buyers in the auction. Since *Y* has a negative exponential, it approaches zero as x_i becomes greater than $\underline{x} + \delta$ and when δ increases. The bid function is assumed to be equal for all buyers so the buyer with the highest information signal always wins the item being sold. The bid function is substituted into the buyer's conditional expected revenue function

$$E[R_i^B \mid x_i > x_j \quad i \neq j] = E[X - (x_i - \delta + Y(N, \delta)) \mid x_i > x_j \quad i \neq j] \tag{1.2}$$

where R_i^B is buyer *i*'s expected revenue conditional on winning the bid. Kagel and Levin (1986) predict the expected revenue conditional on winning is

$$E[R_i^B \mid x_i > x_j \; i \neq j] = 2\delta/(N+1).$$

Reserve Price

Riley and Samuelson (1981) were the first to consider the implications of a reserve price in an independent private values auction. Their model is clearly derived and discussed, which has led to it being a highly followed theory for empirical models (Reiley 2006;

Rosenkranz and Schmitz 2007). Levin and Smith (1996) extend Riley and Samuelson's (1981) independent private values model. They provide comparative statics results to the effects of a reserve price for a variety of different auction structures as the number of buyers approaches infinity, which includes a common-value auction as a special case. They state a reserve price could: (1) be a barrier to entry for buyers in the auction, (2) result in increased revenues for the sellers, and (3) not be necessary when a certain number of buyers are participating in the auction. Levin and Smith (1996) show that the optimal reserve price in the first-price auction should converge to the seller's true value as the number of buyers increases.

When sellers choose a reserve price, buyers must bid higher than the other participants in the auction and bid greater than or equal to the sellers' reserve prices for the transactions to occur. We modify Kagel and Levin's (1986) common-value auction model by including a reserve price. Now buyer *i*'s conditional expected revenue is subject to buyer *i* bidding higher than buyer *j* and greater than or equal to the sellers' reserve price *r*. Buyer *i*'s expected revenue conditional on winning is

$$E[R_i^B \mid x_i > x_j, x_i \geq r\, i \neq j] = E[X - (x_i - \delta + Y(N,\delta)) \mid x_i > x_j, x_i \geq r\, i \neq j] \quad (1.3)$$

where *r* is the reserve price set by the sellers. Buyers with the highest signal will win the item being auctioned as long as their signal is greater than the reserve price.

The sellers choose a reserve price for the item being sold from previous transactions. Sellers only sell their item when the highest bid price is greater than or equal to their reserve price; otherwise the sellers choose not to sell. The sellers' expected revenue function is

$$E[R_h^S] = \sum_{i=1}^{N} E[b_i \mid b_i > b_j, b_i \geq r \;\; i \neq j] \tag{1.4}$$

where R_h^S is the expected revenue for seller *h*. Since this is a first-price auction, sellers accept the highest bid price that is greater than or equal to their reserve price.

Solving a common-value auction analytically is difficult (Gordy 1998), and including a reserve price perhaps makes it infeasible. An agent-based artificial auction is an appropriate method for determining the implications of a reserve price in a first-price common-value auction when buyers choose their optimal bid price and sellers choose their optimal reserve price.

Artificial Common Value Auction

The base artificial auction for this paper is a duopsony first-price common-value auction without the sellers imposing a reserve price. Our base model is similar to Kagel and Levin's (1986) model, but we make several slight modifications that are described in the next sub-section. Then we allow sellers to choose their optimal reserve price in the duopsony artificial auction while buyers choose their optimal bid price. Winning bid price for buyers and expected revenues for the seller are compared to determine the implications of a reserve price in the duopsony artificial auction. Furthermore, we extend Kagel and Levin's (1986) assumption of symmetric buyers by giving one of the buyers a bid function with a normally distributed error term, and giving the other buyer a bid function with a uniformly distributed error term. Finally, an additional buyer is introduced in the artificial auction making it an oligopsony auction. Results are presented for the oligopsony auction when no reserve price is imposed and when a reserve price is imposed. This allows us to determine the implications of a reserve price in an oligopsony auction and determine how much an additional buyer increases the price of the item being sold.

In the model, the buying and selling agents are autonomous entities that follow bounded rationality (Simon 1957). Bounded rationality is different from the full rationality assumption made in most analytical models and could explain differences between analytical solutions and agent-based solutions.

Buyer and Seller Agents

First, an artificial common-value auction model without a reserve price is established. We base our model on Kagel and Levin (1986), but make a few deviations from their model. Kagel and Levin (1986) randomly generate the true value of the item in each repeated auction because human buyers could learn the true value and base their bids on the true value instead of their signal. The true value can increase and decrease due to demand and supply factors, but will not influence the bid behavior around the true value; thus, they analyze the bid price around the random generated true value of the item each iteration. In an agent-based auction, generating a new random true value each iteration is unnecessary because buyers are forced to bid based on their signals. The first slight modification to Kagel and Levin (1986) is that the agent-based auction assumes a constant positive true value for the item being sold, which means we are evaluating bids around some constant true value that is greater than zero. Our approach is equivalent to imposing an interior solution, so we can eliminate the variable Y in the Kagel and Levin (1986) bid function from the buyers' bid function since it approaches zero as the buyers' signal moves away from the bounds on the true value. Next, Kagel and Levin (1986) restrict the buyer's bid range between a lower bound of max $\{x_i - \delta, \underline{x}\}$ and an upper bound of min $\{x_i + \delta, \overline{x}\}$. This restriction is relaxed in the agent-based auction observing all possible bid prices for buyers. Without this restriction, Kagel and Levin (1986 p. 899) state the optimal bid price defies analytical solution. The bid function we use matches Andreoni and Miller's (1995) proposed common-value bid function for an agent-based auction, and is specified as

$$b_i = x_i - \varepsilon_i \tag{1.5}$$

where b_i is the bid price for buyer *i*, $x_i \in [-1,+1]$ is the choice variable or bid strategy for buyer *i* with random variable $\varepsilon_i \sim U[0,1]$. The random variable is subtracted from the buyer's choice variable to follow Kagel and Levin's (1986) model and to match Andreoni and Miller's (1995) proposed common-value bid function. Buyers are assumed to have homogenous preferences for the item being sold in the auction.

For the agent-based model results, we report the unconditional expected revenue. The buyers expected revenues are calculated with numerical integration. Buyer *i's* expected revenue conditional on the other buyers' current bid prices is

$$E[R_i^B] = \int_{x_i-1}^{x_i} \int_{\underline{b}}^{b_i} (X - b_i) db_j db_i \quad i \neq j \tag{1.6}$$

where R_i^B is the expected revenue for the buyer *i*, $\underline{b} = \max\{x_j - 1, b_j\}$, and *X* is the retail value of the item being sold, which is assigned to be 0.5. Buyers are bid-shading when their equilibrium bid price is less than 0.5, and the winner's curse outcome is present if the equilibrium bid price is greater than 0.5.

Sellers are assumed to be homogeneous and price-takers; therefore, this agent-based auction model can be viewed as containing one representative seller or a set of sellers who share information. When the seller imposes a reserve price $r \in [-1,+1]$, the buyers expected revenue function changes. The seller's reserve price is not stochastic, and their true value for the item is zero, so the seller is expected to accept any bid greater than or equal to zero. To win the item, buyers must not only bid higher than the other buyer, but their bid price must be greater than or equal to the seller's reserve price. This restriction changes the lower bound of the expected revenue integral to the seller's reserve price. Buyer *i*'s expected revenue when a reserve price is imposed is

$$E[R_i^B] = \int_{\underline{x}}^{x_i} \int_{\underline{b}}^{b_i} (X - b_i) db_j db_i \;\; i \neq j \tag{1.7}$$

where $\underline{x} = \max\{x_i - 1, r\}$, $\underline{b} = \max\{r, b_j\}$, and *r* is the seller's reserve price.

The seller's expected revenue is

$$E[R^S] = \sum_{i=1}^{N} [X - E[R_i^B \mid b_i > b_j, b_i \geq r \;\; i \neq j]] \tag{1.8}$$

where R^S is the expected revenue for the representative seller. Then, we modify the above model by letting the error terms in buyer *i*'s bid function be distributed $\varepsilon_i \sim U[0,1]$, while buyer *j*'s bid function is distributed $\varepsilon_j \sim N(1/2, 1/12)$. Chen and Wang (2011) encourage future research to investigate artificial auction equilibrium when agents are programmed with differences in their learning abilities. Not all humans or firms participating in auctions are equal in many respects; thus, making agents non-symmetric might be more realistic to actual auctions and human experiments (Chen and Wang 2011). In addition to Chen and Wang's (2011) suggestions, Casari, Ham, and Kagel's

(2007) experiment found that intelligence, demographics, and other characteristics influence bid price in a common-value auction, which also suggests agents with non-symmetric bid functions might be interesting to investigate. Allowing the buyers to have different distributional assumptions for their bid function is possible in an agent-based auction, which illustrates a major advantage of using the agent-based technique. Finally, we return the buyers' information signals to be distributed $U[0,1]$ and include a third buyer in the auction. Results are presented for the oligopsony artificial auction model with and without the reserve price.

Numerical Integration

To solve the expected revenue functions, numerical integration techniques are used. Recently, some attention has been given in the literature to developing new methods to numerically solve simulated auction problems (Gayle and Richard 2008; Peng and Yang 2010; Fibich and Gavish 2011), but they are not easily adaptable to agent-based auctions. We combine the trapezoidal rule and Gaussian quadrature with the PSO learning algorithm developed by Zhang and Brorsen (2009). The trapezoidal rule is used to approximate the integral when both buyers have uniformly distributed[1] error terms, and Gaussian quadrature points and weights are used to estimate the integral when the errors in the buyers' bid function are normally distributed. Combining the trapezoidal rule and Gaussian quadrature with the PSO algorithm to solve an agent-based common-value auction is a unique contribution to the agent-based model literature.

[1] While more efficient integration algorithms exist in the case of all buyers having a uniform distribution (for example, we have verified the algorithm by solving the integral analytically in the duopsony case), we use the trapezoidal rule in all cases to be consistent with methods used on more complex problems.

When the buyer's error term is distributed uniformly, we use the trapezoidal rule to approximate expected revenues. When the integral is "rough" or not twice differentiable, the trapezoidal rule is better for approximating integrals than Simpson's rule (Cruz-Uribe and Neugebauer 2002). Expected revenue can be truncated at the reserve price, which makes the trapezoidal rule a more appropriate integration method than Simpson's rule. This method estimates the area of trapezoids for *n* subintervals or lengths to determine the expected revenues for each buyer. The trapezoidal rule is expressed as

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}(y_0 + 2y_1 +,...,+2y_{n-1} + y_n) \tag{1.9}$$

where $\Delta x = (b-a)/n$, *f(x)* is the probability density function of winning the item, $y_0,...,y_n$ are the grid points in the integral, *b* is the upper bound of the integral, and *a* is the lower bound of the integral. The probability weights at each point of the integral are Δx except at the tails where the probability weight is $\Delta x/2$. With the uniform distribution, *n* is assigned to be 100, resulting in a Δx of 1/100 and a distance of 1/100 between points in the integral. Pseudo code is provided in Table I-1 that outlines the programs used to approximate expected revenues.

Extending Kagel and Levin's (1986) model, we allow the error term for one of the buyers to be normally distributed, and Gaussian quadrature is used to approximate the expected revenue integral. Gaussian quadrature gives a set or probability weights and points that can approximate a continuous distribution that matches 2*k*-1 moments (Preckel and DeVuyst 1992). Preckel and DeVuyst (1992) provide a clear explanation of how to solve for the probability weights and points to approximate an integral for various distributions. This is generally expressed as

$$\int_a^b f(x)dx \approx \sum_{j=1}^{k} p^j (x^j)^l \tag{1.10}$$

where *f(x)* is the continuous probability density function, p^j is the *i*th probability weight, x^j is the *j*th point, *k* is the number of quadrature points, and $l = 0,...,2k-1$ is the number of equations. In the agent-based auction, we set *k*=9 and follow Preckel and DeVuyst's (1992) steps to find the probability weights and points for $\varepsilon \sim N(1/2, 1/12)$. The mean and variance for this distribution are chosen to match the mean and variance of the uniform distribution. Expected revenues are approximated using a similar approach to that described in Table I-1.

Particle Swarm Optimization

Multiple learning algorithms such as genetic algorithms and reinforcement learning algorithms are frequently used in agent-based models, but we use the PSO learning algorithm developed by Eberhart and Kennedy (1995) and modified by Zhang and Brorsen (2009). This algorithm solves faster than the genetic algorithm (Zhang and Brorsen 2009) and was successfully implemented by Zhang and Brorsen (2010). Zhang and Brorsen (2009) let each agent have its own "flock of birds" or clones by constructing $k = 1,...,K$ parallel auctions for each agent. Each agent has *k* clones that act in the *k* parallel auctions and use the clone's bid strategies in the *k* auctions to help the agent find its optimal strategy. The agent's best bid strategy across all parallel auctions is referred to as the best global solution, and the agent's best bid strategy in each parallel auction is referred to as the best local solution.

The *i*th agent has a choice variable (bid price or reserve price) $x_{i,k,t}$ in parallel auction *k* during iteration $t = 1,...,T$, and an adjustment velocity $v_{i,k,t} \in [-1,+1]$ that directs the agent's choice variable. The velocity change for agent *i* is a function of the local best solutions, $p^{l}_{i,k,t}$, where superscript *l* indicates the best solution in the local auction, and the agent's global best solution, $p^{g}_{i,t}$, where superscript *g* indicates the best global solution. In iteration *t*, the *i*th agent's new choice variable in parallel auction *k* is updated by $x_{i,k,t+1} = x_{i,k,t} + v_{i,k,t}$, and the velocity is modeled as

$$v_{i,k,t+1} = wv_{i,k,t} + q_1u_1(p^{l}_{i,k,t} - x_{i,k,t}) + q_2u_2(p^{g}_{i,t} - x_{i,k,t}) \qquad (1.11)$$

where *w* is the inertia weight factor, u_1 and u_2 are uniformly distributed random numbers, and q_1 and q_2 are learning parameters. The learning parameters are

$$q_{1,t} = q_{2,t} = \beta_0^{q_1} + \beta_1^{q_1}(T - t)/T, \qquad (1.12)$$

where both $\beta_0^{q_1}$ and $\beta_1^{q_1}$ are constants, which are set at one in this model, T is the maximum number of iterations, and *t* is the current iteration. When selecting a parameter value for inertia weight, an important tradeoff occurs between exploration and convergence time (Chatterjee and Siarry 2006). A large inertia weight slows convergence time, but encourages agents to explore a larger area, and a small inertia weight increases convergence time, but reduces the agents' exploration area (Zhang and Brorsen 2009). Zhang and Brorsen (2009) develop a compromise for this tradeoff by letting *w* start high and decrease as the optimization proceeds. Inertia weight as expressed in Zhang and Brorsen (2009) is

$$w_t = \beta_0^w + \beta_1^w \left(T - t\right)/T \,, \tag{1.13}$$

where both β_0^w and β_1^w are constants. We set these constants to be 0.75.

Since agents' revenues depend on the other agents' choice variables, the agent's previous best strategy possibly may not perform well in the next iteration. Therefore, the PSO learning algorithm recalculates revenues for each buyer from the previous *L* best local solutions based on the other agents' new choice variable in iteration *t*. The best local strategy is expressed as

$$p_{i,k,t}^{l} = \arg\max \left\{ R_k\left(p_{i,k,t-1}^{l}\right), \ldots, R_k\left(p_{i,k,t-L}^{l}\right), R_k\left(x_{i,k,t}\right) \middle| \, x_{j \neq i,k,t} \right\}, \tag{1.14}$$

where R_k is the revenue in parallel market *k*. Other agents' strategies in the current iteration are held constant, and the past best locals of each agent in *L* iterations are reevaluated. The bid strategy with the highest revenue is selected as the best local bid strategy. The best global choice variable is selected from the best local parameters and is expressed as

$$p_{i,t}^{g} = \arg\max \left\{ R_1\left(p_{i,1,t}^{l}\right), R_2\left(p_{i,2,t}^{l}\right), \ldots, R_K\left(p_{i,K,t}^{l}\right) \right\} \tag{1.15}$$

where *K* is the total number of parallel markets. The PSO learning algorithm is discussed in further detail in Zhang and Brorsen (2009).

Changes in the agent's strategies become smaller as the algorithm begins to find the buyers' and the seller's optimal choice variables. The equilibrium bid prices and reserve price are determined when the total change in the previous 10 bid prices for the buyers and the previous 10 reserve prices for the seller is less than 0.0000001, and the total change in the standard deviations of the reserve price and bid prices are less than 0.00000001. The PSO parameter values are in Table I-2. The model is set to have 100 evolutions (*E*=100), and each evolution has a maximum of 500 iterations (*T*=500). This configuration means the agents have up to 500 iterations in each evolution to converge to equilibrium, and the model repeats this process 100 times. The results presented in the paper are the averages and standard deviations over the 100 evolutions.

Figure I-1 summarizes the discussion about the general process of the agent-based auction without a reserve price. The figure shows the connection across the agents, the numerical integration, and the PSO algorithm. Following the arrows in the figure, agents first choose a bid strategy to enter the auction. Numerical integration is used to approximate the probability of winning the item in the auction with their current bid strategy. The expected revenues, probability of winning, and winning bid price are stored. Finally, the agents' bid strategies are tested to determine if the equilibrium criteria has been met. If it has not been met, then the PSO learning algorithm directs the agents to new bid strategies, and if it has been met, then the model progresses to the next evolution.

Results

Duopsony Artificial Auction

When no reserve price is imposed and the buyer's information signal is distributed $U[0,1]$, the average equilibrium winning bid is -0.21, which indicates that buyers are bid-shading, and avoiding the winner's curse (Table I-3). In human experiments, the winner's curse approaches zero as buyers gain experience (Kagel and Levin 2002). Evaluation of the agents' previous best global and local bid strategies is similar to increasing the buyers' experiences in a human experiment, which explains why the winner's curse is avoided in the artificial auction. The average bid price across all winning and losing bid prices for buyer one is -0.37, and buyer two's average bid price is -0.41 (Table I-3). The difference between the buyers' average bid prices is due to noise since the buyers are identical. Buyer one has a higher bid price on average than buyer two, resulting in buyer one having a higher probability of winning the item and a higher expected revenue per item. Table I-3 presents the expected revenue per item received over all the buyers' winning and losing bids. Buyer one's expected revenue per item is 0.37, and buyer two's expected revenue per item is 0.33. For the duopsony auction, Kagel and Levin's (1986) model predicts the conditional expected revenue for the winning buyer is one-third, which is different from the buyers' conditional expected revenues in the agent-based model of 0.70. As discussed above, Kagel and Levin (1986) constrained their agents' bids and they ended up with a corner solution. Their constraint would impose that buyers bid at least zero on average. The seller's expected revenue per item is -0.21, but the seller's expected revenue per item could be made positive by adding a constant to the value of the item or a positive random value as in Kagel and Levin (1986). These results do suggest that buyers have market power in this artificial auction and are able to shade bids below the item's true value.

When a reserve price is imposed by the seller in the artificial auction, the average equilibrium winning bid is 0.22 (Table I-3), which is an increase of 0.43 from when no reserve price is imposed. This result matches the Levin and Smith (1996) theoretical findings, and the experimental results of Goncalves (2009) and Reiley (2006). McAfee and McMillan (1987) argue that setting a reserve price is an effective counter to bid-shading, and the average equilibrium winning bid price in the artificial auction illustrates this same result. The seller's optimal reserve price in the agent-based auction is zero, which is the seller's true value and matches Levin and Smith's (1996) prediction of seller's optimal reserve prices (Table I-3). The buyers' expected revenues per item for all winning and losing bid prices decrease, and the seller's expected revenue per item increases to 0.13 when a reserve price is present (Table I-3). Theoretical (Riley and Samuelson 1981; Leven and Smith 1996) and experimental (Reiley 2006; Goncalves 2009) results have shown that a reserve price can increase the seller's revenue. The probability the item is sold decreases to 61% of the possible transactions when the optimal reserve price is imposed (Table I-3). Goncalves (2009) and Reiley (2006) experiments find a reserve price reduces the probability of a transaction, which indicates a reserve price can be a barrier to entry for buyers.

Relaxing Kagel and Levin's (1986) assumption of symmetric bid distributions, buyer one's bid function is changed to be distributed $N(1/2,1/12)$, and buyer two's bid function is distributed $U[0,1]$. The mean and variance of the two distributions are equal, but the buyers' bid functions are non-symmetric. The seller still selects the optimal reserve price of zero in this artificial auction, and the average equilibrium winning bid price is 0.21, which is slightly less than the average equilibrium winning bid price when the buyers' bid functions are symmetric (Table I-3). Buyer one has a lower probability of winning in spite of having a higher average bid price than buyer two (Table I-3). As a result, buyer two has a larger expected revenue than buyer one (Table I-3). The normally distributed information is not as valuable as the uniformly distributed information. The probability of the item being sold decreases to 56%, resulting in the seller's expected revenue decreasing to 0.11 per item (Table I-3). Results for the agent-based auction with non-symmetric buyers are slightly different than when buyers are symmetric, and this fact might be of interest to develop further in future research.

Oligopsony Artificial Auction

With three buyers and no reserve price, the average equilibrium winning bid is 0.08 (Table I-4). An additional buyer participating in the artificial auction increases the average winning bid price by 0.29 from the duopsony auction with no reserve price, which matches results in empirical studies (Meyer 1988) and field experiments (Goncalves 2009). In empirical work on rice auctions, Meyer (1988) finds prices increased as the number of bidders increased and argues that it is evidence in favor of the winners curse, yet our results suggest such a finding could be due to increased competition. Bulow and Klemperer (1996) argue that in a private value English auction, an additional buyer will increase the bid price more than a reserve price with one less buyer. Conversely, Goncalves (2009) found that including an additional buyer in the second-price common-value auction does not increase the equilibrium bid price as much as a reserve price, which matches our agent-based model results. The buyers' expected revenues per item decrease and the seller's expected revenue per item increase from the artificial duopsony auction with no reserve price (Table I-4). Since the seller does not have the choice to not sell the item under this scenario, the probability of selling the item is 100% (Table I-4).

When a reserve price is imposed by the seller in the oligopsony artificial auction, the average equilibrium winning bid price is 0.27 (Table I-4), which is an increase of 0.19 from when no reserve price is present in the oligopsony auction. This result matches both the theory (Levin and Smith 1996; Rosenkranz and Schmitz 2007) and field experiments in the literature (Reiley 2006; Goncalves 2009) and results from the duopsony artificial auction when a reserve price is imposed. The winning bid price for the oligopsony auction increases by 0.05 compared to the duopsony auction with a reserve price. The increased competition from an additional buyer increases the winning bid price even when the seller imposes a reserve price. The seller’s optimal reserve price is 0.01 (Table I-4). This price is slightly higher than the seller’s true value of the item being sold, but is likely due to rounding or randomness in the optimization algorithm. The buyers expected revenues per item decrease from when no reserve price is imposed to 0.06 (Table I-4). The seller’s expected revenue increases to 0.20 per item, and the probability of selling the item is 77%. This figure is an increase of 16% from the duopsony auction with a reserve price, and it is due to the increase in competition from the third buyer. Similar to the duopsony artificial auction and Reiley (2006), the reserve price appears to be a barrier for some buyers to participate in the auction. The reserve price has similar effects when an additional buyer participates in the auction, matching the findings in the literature (Levin and Smith 1996; Reiley 2006; Goncalves 2009).

Convergence Time for the Artificial Auctions

Table I-5 summarizes the convergence time and the number of iterations before meeting the convergence criteria in the agent-based duopsony and oligopsony auctions. This information is dependent on several factors (such as computer, compiler, and other factors) but is intended to show how the modeling changes affect convergence. The computer used has an Intel® Core^{TM} i7-2600 processor, and the models are programmed in Java using Eclipse. When no reserve price is set by the seller in the duopsony auction, the agents require on average 91.2 iterations before meeting the equilibrium criteria. Each evolution takes on average 1.67 seconds to converge, resulting in a total solving time of a little less than three minutes (Table I-5). When a reserve price is chosen by the seller in the duopsony auction, the model requires on average 101.6 iterations per evolution to converge with each evolution taking on average is 1.98 seconds per evolution or roughly three minutes (Table I-5). The seller's choice variable of their reserve price did not substantially slow convergence. When the buyers' bid functions are uniform and normal, the agent-based duopsony auction requires on average 102.8 iterations per evolution to converge, with each evolution taking an average of 0.25 seconds or a total solving time of roughly 30 seconds (Table I-5). Gaussian quadrature requires fewer points to approximate the integral than the trapezoidal rule, which causes the model to converge faster.

The figures displayed show an example of an evolution for each auction structure. Figure I-2 shows the path to convergence for the winning bid price in the duopsony auction, and Figure I-3 displays the buyers' and the seller's revenues in the duopsony auction. The figures show the agents explored several alternative strategies briefly, but roughly the last two-thirds of the iterations cause only minor refinements in the solution. Figure I-4 shows the convergence paths of the winning bid price and reserve price in the duopsony auction. The figure shows the seller's reserve price and the winning bid price move fairly closely. The expected revenue for the buyers and the seller in the duopsony auction is shown in Figure I-5. Similarly to when no reserve price is imposed, the buyers' and the seller's revenues change as the buyers and seller explore alternative bid and reserve price strategies. Figure I-6 displays the convergence path of the winning bid price and the reserve price in the duopsony auction, and Figure I-7 shows the convergence path of the seller's and buyers' expected revenues in the duopsony auction. An advantage of Gaussian quadrature over the trapezoidal rule is it requires fewer points to approximate an integral. These larger jumps in the numerical integration do not appear to cause the agents' to dramatically change their bid strategies.

For the oligopsony auction when no reserve price is imposed, the agents require 98.6 iterations on average before meeting the equilibrium criteria (Table I-5). Each evolution takes roughly 10 minutes to converge for a total solving time of approximately 16 hours (Table 5). When a reserve price is imposed in the oligopsony auction, the model requires on average 101.8 iterations per evolution to converge with each evolution taking on average a little over 10 minutes for a total solving time of roughly 17 hours (Table I-5). Similar to the duopsony auction structure, the time to converge for the model is slightly slower when seller imposes a reserve price. The oligopsony auction structure takes on average much more time to converge than the duopsony auction structure.

Conclusions

The objective of this paper is to determine the implications of a reserve price in a first-price common-value auction. Considerable theoretical and empirical work has focused on the influence of reserve prices in an independent private values auction, but little attention has been given to a first-price common-value auction. Duopsony and oligopsony agent-based auctions are established to find the optimal reserve price and the equilibrium winning bid price and the implications of a reserve price on the auction equilibrium. An agent-based auction is used to solve this problem.

Finding the analytical solution to common-value auctions is difficult, and when a reserve price is included in this auction, is perhaps infeasible. Reproducing this auction in a laboratory experiment would be expensive. Thus, using an agent-based auction model is an appropriate method to determine the implications of a reserve price in a first-price common-value auction. In addition, an agent-based first-price common-value auction with a reserve price has never been solved in the literature, making this model a unique contribution to the agent-based literature. Furthermore, we extend the agent-based model literature by introducing a combination of numerical integration techniques with the particle swarm algorithm. Finally, we address Chen and Wang's (2011) suggestion of considering non-symmetric agents by allowing the buyers' signals to have different distributions.

In the two buyer auction, buyers shade bids when a reserve price is not imposed by the seller. The equilibrium winning bid price increases when a reserve price is imposed, resulting in the seller's expected revenue increasing, and the buyers' expected revenues decreasing. Furthermore, differences in expected revenues are found in the duopsony auction when the buyers are non-symmetric, but the winning bid price remains similar to when buyers are symmetric. This finding might be interesting to explore further in future research. When a third buyer is introduced in the auction with no reserve price, the equilibrium winning bid price increases relative to the duopsony artificial auction with no reserve price. The increase in competition from an additional buyer is effective at increasing the winning bid price, but not as effective as a reserve price in a duopsony auction. When a reserve price is imposed by the seller in the oligopsony artificial auction, the equilibrium bid price increases relative to when no reserve price is present. A reserve price in the duopsony and oligopsony auctions is effective in increasing the average winning bid price.

References

Alkemade, F., J.A. La Poutré, and H.M. Amman. 2006. “Robust Evolutionary Algorithm Design for Socioeconomic Simulation.” *Computational Economics* 28(4):355–370.

Andreoni, J., and J.H. Miller. 1995. “Auctions with Artificial Adaptive Agents.” *Games and Economic Behavior* 10:39-64.

Arifovic, J. 1996. “The Behavior of the Exchange Rate in the Genetic Algorithm and Experimental Economies.” *Journal of Political Economy* 104(3):510-541.

Bonabeau, E. 2002. “Agent-Based Modeling: Methods and Techniques for Simulating Human Systems.” *Proceedings of the National Academy of Sciences of the United States of America* 99:7280-7287.

Bajari, P., and A. Hortacsu. 2003. “The Winner’s Curse, Reserve Price, and Endogenous: Empirical Insights From eBay Auctions.” *The RAND Journal of Economics* 34(2):329-355.

Bulow, J., and P. Klemperer. 1996. “Auctions Versus Negotiations.” *American Economic Review* 86(1):180–194.

Capen, E.C., R.V. Clapp, and W.M. Campbell. 1971. “Competitive Bidding in High-Risk Situations.” *Journal of Petroleum Technology* 23:641-653.

Casari, M., J. Ham, and J. Kagel. 2007. “Selection Bias, Demographic Effects, and Ability Effects in Common Value Auction Experiments.” *American Economic Review* 97(4):1278–1304.

Chatterjee, A., and P. Siarry. 2006. “Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization.” *Computers and Operations Research* 33:859–871

Chen, S-H., and S.G. Wang. 2011. “Emergent Complexity in Agent-Based Computational Economics. *Journal of Economics Surveys* 25(3):527-546.

Corrigan, J.R., and M.C. Rousu. 2011. “Are Experimental Auctions Demand Revealing When Values Are Affiliated?” *American Journal of Agricultural Economics* 93(2):514-520

Cruz-Uribe, D., and C.J. Neugebauer. 2002. “Sharp Error Bounds for the Trapezoidal Rule and Simpson’s Rule.” *Journal of Inequalities in Pure and Applied Mathematics* 3(4):Art 49.

Eberhart, R.C., and J. Kennedy. 1995. “A New Optimizer Using Particle Swarm Theory.” *Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan,* 39-43

Fibich, G., and N. Gavish. 2011. “Numerical Simulation of Asymmetric First-Price Auctions.” *Games and Economic Behavior* doi:10.1016/.geb.2011.02.010.

Gayle, W.R., and J.F. Richard. 2008. “Numerical Solutions of Asymmetric, First-Price, Independent Private Value Auctions.” *Computational Economics* 32:245-278

Goncalves, R. 2009. “Empirical Evidence on the Impact of Reserve Prices in Common Value English Auctions.” Working Paper. http://dge.uma.pt/portal/docs_para_download/pej_papers/42.pdf. Accessed June 2011.

Gordy, M.B. 1998. “Computationally Convenient Distributional Assumptions for Common-Value Auctions.” *Computational Economics* 12:61-78

Guerci, E., I. Stefano, and S. Cincotti. 2008. “Learning Agents in an Artificial Power Exchange: Tacit Collusion, Market Power and Efficiency of Two Double-Auction Mechanisms.” *Computational Economics* 32:73–98.

Hailu, A., and S. Thoyer. 2010. “What Format for Multi-Unit Multiple-Bid Auctions? Agent-Based Simulation of Auction Performance and Non-linear Bidding Behavior.” *Computational Economics* 35:189-209.

Hendricks, K., R.H. Porter, and C.A. Wilson. 1994. “Auctions for Oil and Gas Leases with an Informed Bidder and a Random Reservation Price.” *Econometrica*, 62(6):1415-1444.

Kagel, J.H., and D. Levin. 1986. “The Winner's Curse and Public Information in Common Value Auctions.” *American Economic Review* 76(5):894-920.

Kagel, J.H., and D. Levin. 2002. *Common Value Auction and the Winner’s Curse*. Princeton, New Jersey: Princeton University Press

Klemperer, P. 1999. “Auction Theory: A Guide to the Literature.” *Journal of Economic Surveys* 13:227–286.

Krishna, V. 2002. *Auction Theory*. San Diego, CA: Academic Press.

Levin, D., and J.L. Smith. 1996. “Optimal Reservation Prices in Auctions.” *The Economics Journal* 106:1271-1283.

McAfee, R.P., and J. McMillan. 1987. “Auctions and Bidding.” *Journal of Economic Literature* 25(2):699-738.

McAfee, R.P., D.C. Quan, and D.R. Vincent. 2002. “How to Set Minimum Acceptable Bids, with Application to Real Estate Auctions.” *Journal of Industrial Economics* 50(4):391-416.

McAfee, R.P., and D.R. Vincent. 1992. “Updating the Reserve Price in Common Value Auctions.” *American Economic Review* 82(2):512-518.

Meyer, D.J. 1988. “Competition and Bidding Behavior: Some Evidence from the Rice Market.” *Economic Inquiry* 26:123-131.

Milgrom, P.R., and R.J. Weber. 1982. “A Theory of Auctions and Competitive Bidding.” *Econometrica* 50:1089-1122

Paarsch, H. 1992. “Deciding Between the Common and Private Value Paradigms in Empirical Models of Auctions.” *Journal of Econometrics* 51:191-215.

Paarsch, H. 1997. “Deriving an Estimate of the Optimal Reserve Price: An Application to British Columbian Timber Sales.” *Journal of Econometrics* 78:333-357.

Peng, J., and Z. Yang. 2010. “Numerical Solution of Asymmetric, First-Price, Independent Private Value Auctions: Comment.” *Computational Economics* 36:231-235

Pinkse, J., and G. Tan. 2005. “The Affiliation Effect in First-Price Auctions.” *Econometrica* 1:263-277.

Preckel, P.V., and E. DeVuyst. 1992. “Efficient Handling of Probability Information for Decisions Analysis under Risk.” *American Journal of Agricultural Economics* 74(3):655-662.

Reiley, D.H. 2006. “Field Experiments on the Effects of Reserve Prices in Auctions: More Magic on the Internet.” *The RAND Journal of Economics* 37(1):195-211.

Riley, J.G., and W. Samuelson. 1981. “Optimal Auctions.” *American Economic Review* 71:381-392.

Robinson, M.S. 1985. “Collusion and the Choice of Auction.” *The RAND Journal of Economics* 16:141-145.

Rosenkranz, S., and P.W. Schmitz. 2007. “Reserve Prices in Auctions as Reference Points.” *The Economic Journal* 117:637-653.

Simon, H.A. 1957. *Rational Choice and the Structure of the Environment.* Cambridge, MA: MIT Press.

Tesfatsion, L. 2001. “Introduction to the Special Issue on Agent-Based Computational Economics.” *Journal of Economic Dynamics and Control* 25:281–293

Varian, H.R. 1992. *Microeconomic Analysis*. 3rd. ed. New York: W.W. Norton and Company.

Vickrey, W. 1961. “Counterspeculations, Auctions, and Competitive Sealed Tenders.” *Journal of Finance* 16(1):8-37.

Wilson, R. 1977. “A Bidding Model of Perfect Competition.” *The Review of Economic Studies* 44(3):511-518.

Zhang, T., and B.W. Brorsen. 2009. “Particle Swarm Optimization Algorithm for Agent-Based Artificial Markets.” *Computational Economics* 34:399-417.

Zhang, T., and B.W. Brorsen. 2010. “The Long Run and Short Run Impact of Captive Supplies on the Spot Market Price: An Agent-Based Artificial Market.” *American Journal of Agricultural Economics* 92:1181-1194.

Table I-1. Pseudo Code for the Duopsony with Symmetric Buyers

```
Begin outside loop where error term $\varepsilon_1$ is equal to zero and increases by 1/100 until $\varepsilon_1$ equals 1
    If $\varepsilon_1$ equals 0 or 1 then the probability weight $W_1$ is 1/200
    Else the probability weight $W_1$ is 1/100
    Set buyer 1’s random bid price equal to buyer 1’s choice variable $x_1$ minus the value for $\varepsilon_1$
    Set buyer 2’s random bid price equal to buyer 2’s choice variable $x_2$ minus the value for $\varepsilon_1$
    Set probability buyer 1 wins $P_1$ equal to zero.
    Set probability buyer 2 wins $P_2$ equal to zero.

    Begin inside loop where error term $\varepsilon_2$ is equal to zero and increases by 1/100 until $\varepsilon_2$ equals 1
        If $\varepsilon_2$ equals 0 or 1 then probability weight $W_2$ is 1/200
        Else the probability $W_2$ is 1/100
        Set buyer 1’s temporary random bid price equal to buyer 1’s choice variable $x_1$ minus the value for $\varepsilon_2$
        Set buyer 2’s temporary random bid price equal to buyer 2’s choice variable $x_2$ minus the value for $\varepsilon_2$
        If random bid price for buyer 1 is greater than temporary random bid price for buyer 2 then probability of winning for buyer 1 $P_1$ increases by $W_2$
        If random bid price for buyer 2 is greater than temporary random bid price for buyer 1 then probability of winning for buyer 2 $P_2$ increases by $W_2$
    End of the inside loop

    Buyer 1’s revenue is calculated by multiplying $W_1$ by the buyer 1’s probability of winning $P_1$ times the difference between the retail price and the chosen random bid price, and summed over all random bid prices
    Buyer 2’s revenue is calculated by multiplying $W_1$ by buyer 2’s probability of winning $P_2$ times the difference between the retail price and the chosen random bid price, and summed over all random bid prices
    Seller revenue is calculated by multiplying $W_1$ by summing buyer 1’s random bid price multiplied by probability of buyer 1 winning $P_1$ plus buyer 2’s random bid price multiplied by probability of buyer 2 winning $P_2$, and summed over all random bid prices

End of outside loop

Winning bid price is calculated by dividing the sellers profits by sum of the buyer 1 and buyer 2’s probabilities of winning
```

Table I-2. Parameter Values for the Particle Swarm Optimization Algorithm

Parameter	Symbol	Value
Intercept of inertia weight	β_0^w	0.75
Slope of inertia weight	β_1^w	0.75
Learning parameters intercept	β_0^q	0.75
Learning parameter slope	β_1^q	0.75
Parallel markets	k	10
Maximum iterations	T	500
Evolutions	E	100
Memory	L	3

Table I-3. Equilibrium Values for the Agent-Based Duopsony Auctions

Category[a]	Scenario: No Reserve Price[b]	Scenario: Reserve Price[b]	Scenario: Normal and Uniform Distributions and Reserve Price[c]
Average winning bid price	-0.21 (0.13)	0.22 (0.07)	0.21 (0.09)
Reserve price	-	0.00 (0.11)	0.00 (0.12)
Buyer 1			
Average bid price	-0.37 (0.13)	-0.12 (0.13)	-0.02 (0.02)
Transaction probability	53.0% (12.0%)	31.0% (12.0%)	22.0% (17.0%)
Expected revenue	0.37 (0.10)	0.09 (0.04)	0.07 (0.06)
Buyer 2			
Average bid price	-0.41 (0.13)	-0.12 (0.13)	-0.12 (0.14)
Transaction probability	47.0% (12.0%)	30.0% (11.0%)	34.0% (14.0%)
Expected revenue	0.33 (0.11)	0.08 (0.04)	0.10 (0.04)
Seller			
Transaction probability	100.0% (0.0%)	61.0% (18.0%)	56.0% (23.0%)
Expected revenue	-0.21 (0.11)	0.13 (0.06)	0.11 (0.06)

Note: Standard deviations are presented in parentheses.
[a] All results, except average winning bid price, are unconditional on winning.
[b] Both errors are distributed U[0,1].
[c] Buyer 1 has normally distributed errors, and buyer 2 has uniformly distributed errors.

Table I-4. Equilibrium Values for the Agent-Based Oligopsony Auctions

	Scenario	
Category[a]	No Reserve Price	Reserve Price
Average winning bid price	0.08 (0.09)	0.27 (0.07)
Reserve price	-	0.01 (0.13)
Buyer 1		
Average bid price	-0.16 (0.12)	-0.07 (0.12)
Transaction probability	36.0% (11.0%)	26.0% (9.0%)
Expected revenue	0.15 (0.06)	0.06 (0.03)
Buyer 2		
Average bid price	-0.17 (0.13)	-0.09 (0.12)
Transaction probability	35.0% (11.0%)	25.0% (9.0%)
Expected revenue	0.14 (0.05)	0.06 (0.03)
Buyer 3		
Average bid price	-0.22 (0.14)	-0.08 (0.13)
Transaction probability	29.0% (0.10%)	26.0% (0.10%)
Expected revenue	0.12 (0.05)	0.06 (0.03)
Seller		
Transaction probability	100.0% (0.0%)	77.0% (19.0%)
Expected revenue	0.08 (0.09)	0.20 (0.06)

Note: Standard deviations are presented in parentheses, and buyers' errors are distributed U[0,1].
[a] All results, except average winning bid price, are unconditional on winning.

Table I-5. Time and Number of Iterations to Converge per Evolution for the Duopsony and Oligopsony Auctions

Scenarios	Time in Seconds	Number of Iterations
Duopsony		
No reserve price	1.67	91.20
	(0.17)	(10.82)
Reserve price	1.98	101.60
	(0.27)	(16.44)
Normal and uniform distributions and reserve price	0.25	102.80
	(0.04)	(16.84)
Oligopsony		
No reserve price	588.17	98.60
	(54.65)	(11.11)
Reserve price	616.96	101.80
	(69.98)	(14.19)

Note: Standard deviations are presented in parentheses.

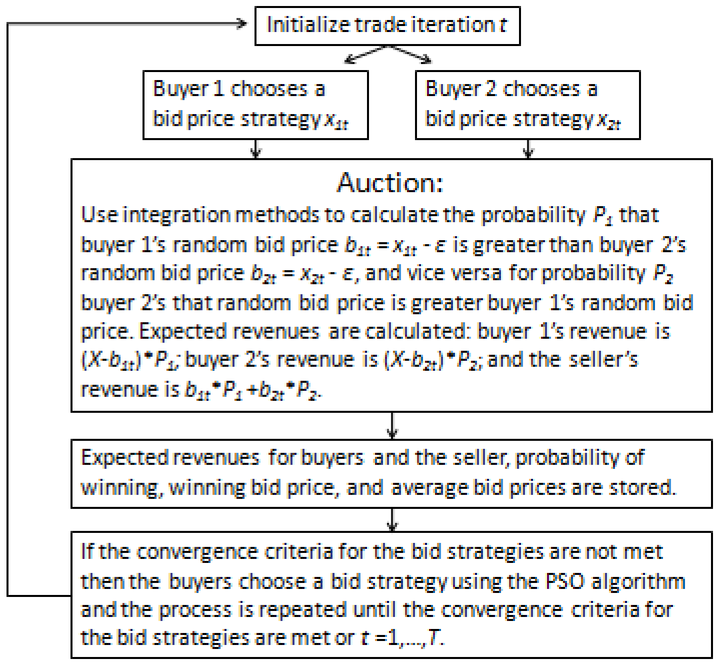
Initialize trade iteration t
Buyer 1 chooses a bid price strategy x_{1t}
Buyer 2 chooses a bid price strategy x_{2t}
Auction:
Use integration methods to calculate the probability P_1 that buyer 1's random bid price $b_{1t} = x_{1t} - \varepsilon$ is greater than buyer 2's random bid price $b_{2t} = x_{2t} - \varepsilon$, and vice versa for probability P_2 buyer 2's that random bid price is greater buyer 1's random bid price. Expected revenues are calculated: buyer 1's revenue is $(X\text{-}b_{1t})*P_1$; buyer 2's revenue is $(X\text{-}b_{2t})*P_2$; and the seller's revenue is $b_{1t}*P_1 + b_{2t}*P_2$.
Expected revenues for buyers and the seller, probability of winning, winning bid price, and average bid prices are stored.
If the convergence criteria for the bid strategies are not met then the buyers choose a bid strategy using the PSO algorithm and the process is repeated until the convergence criteria for the bid strategies are met or t =1,...,T.

Figure I-1. Outline of the Duopsony Agent-Based Auction Model without a Reserve Price

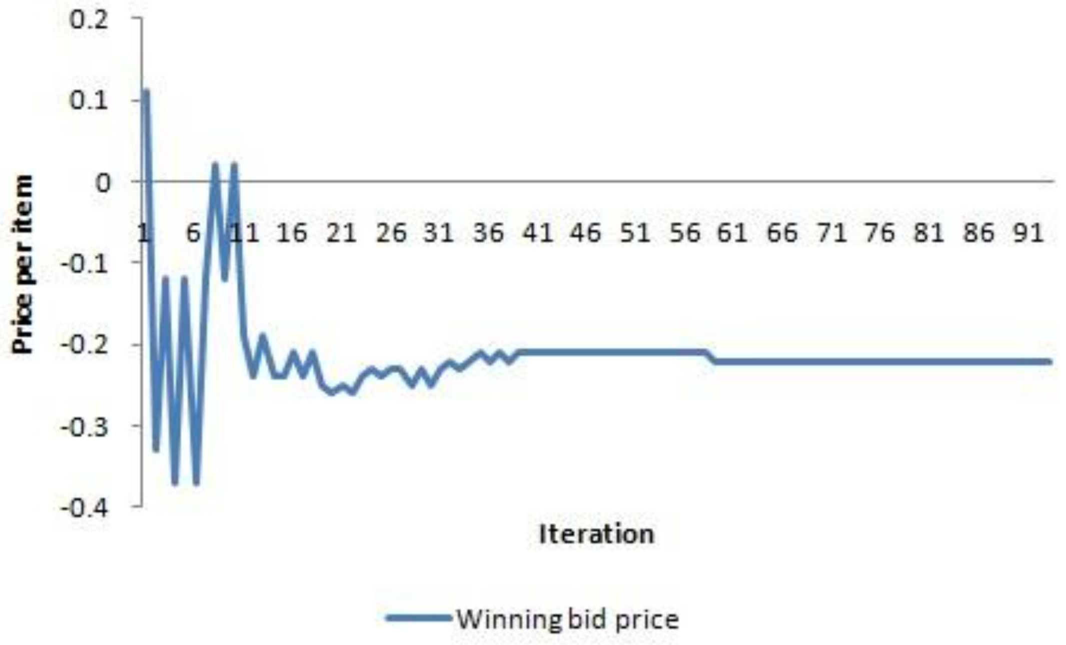

Figure I-2. Example Evolution for the Winning Bid Price with no Reserve Price in the Agent-Based Duopsony Auction

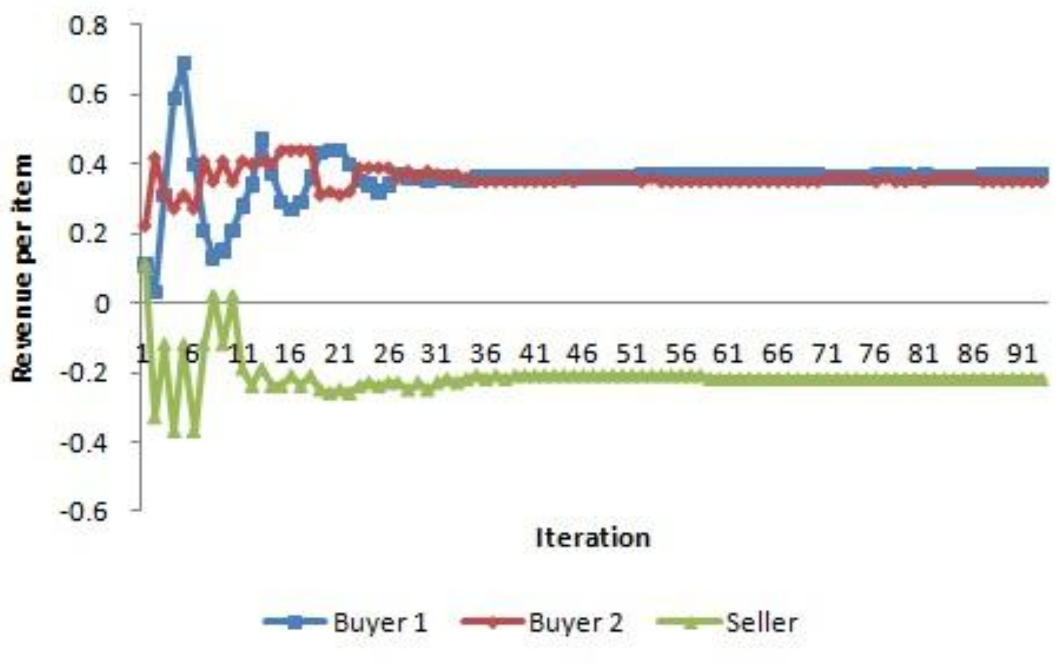

Figure I-3. **Example Evolution for Revenues in the Agent-Based Duopsony Auction with no Reserve Price**

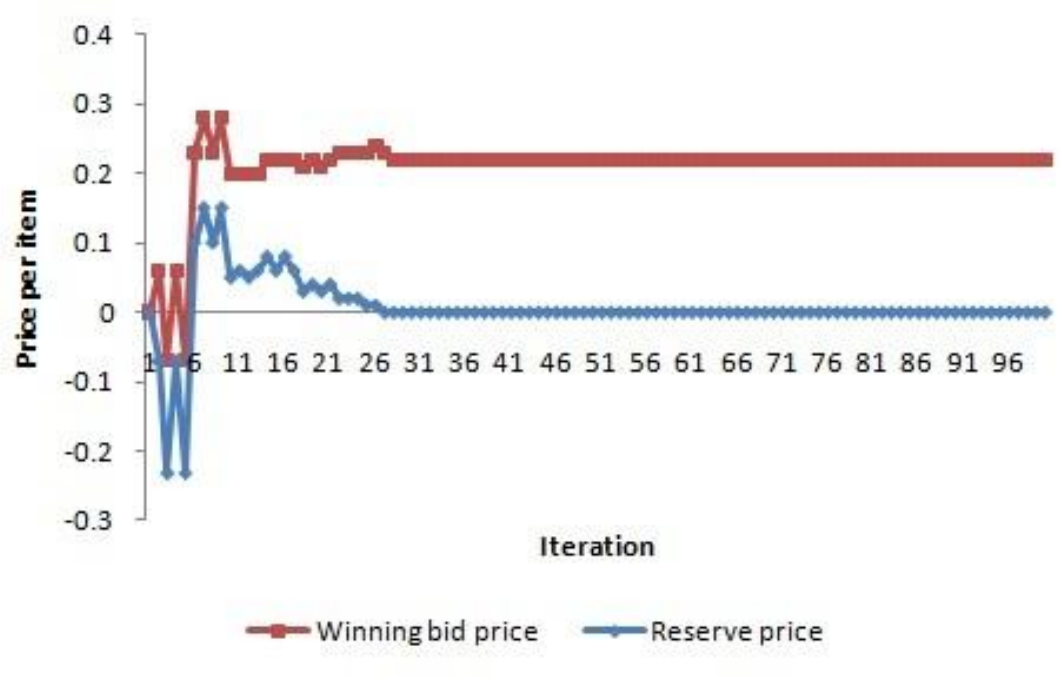

Figure I-4. Example Evolution for the Winning Bid Price and Reserve Price in the Agent-Based Duopsony Auction

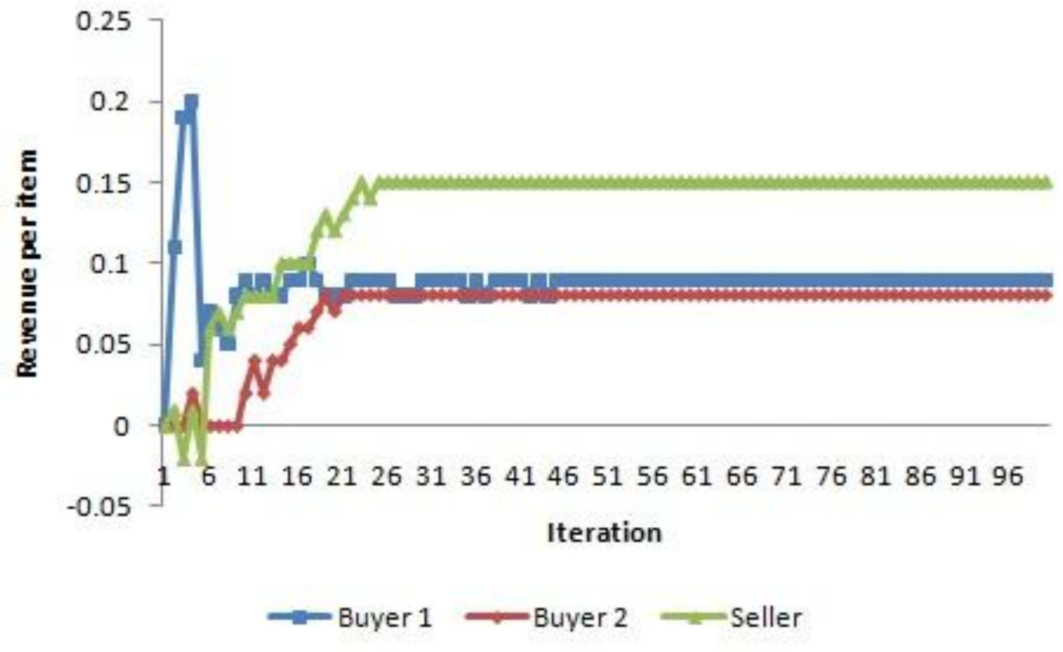

Figure I-5. Example Evolution for the Revenues in the Agent-Based Duopsony Auction with a Reserve Price

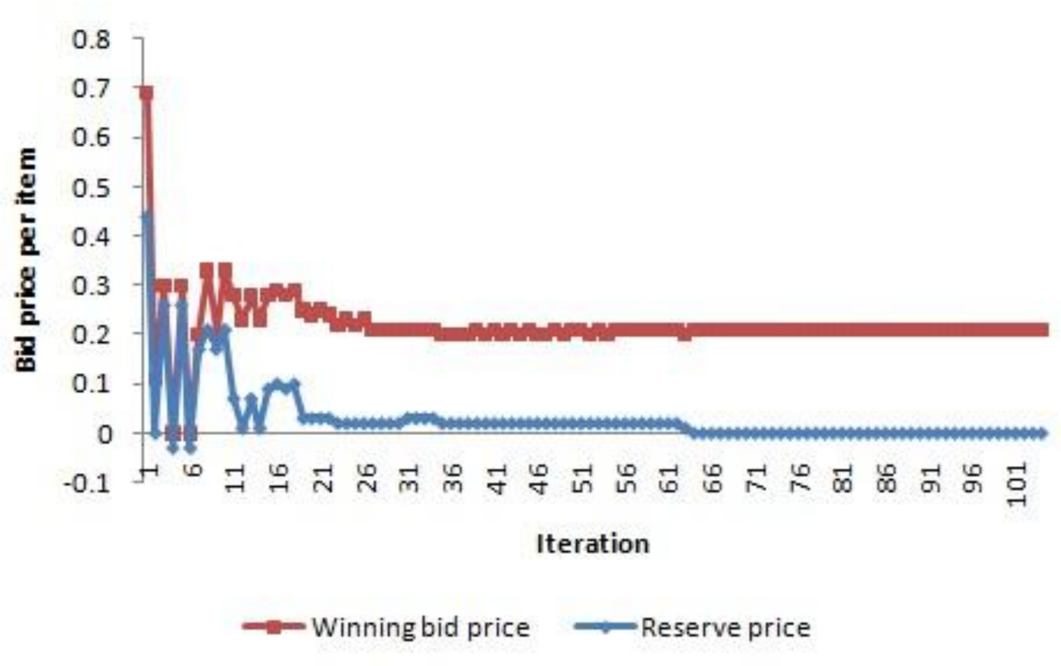

Figure I-6. **Example Evolution for the Winning Bid Price and Reserve Price with Non-Symmetric Buyers in the Agent-Based Duopsony Auction**

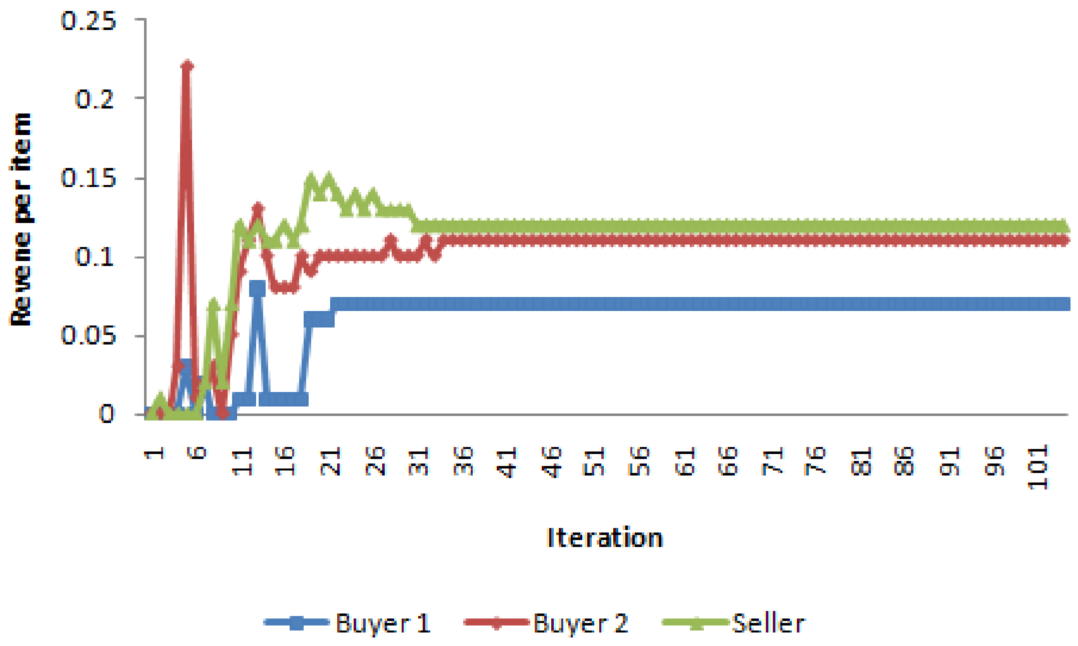

Figure I-7. Example Evolution for the Revenues in the Agent-Based Duopsony Auction with a Reserve Price and Non-Symmetric Buyers

CHAPTER II

CHANGES IN MARKET POWER FROM THE LIVESTOCK MANDATORY PRICE REPORTING ACT: AN AGENT-BASED AUCTION APPROACH

Introduction

In 1999, the U.S. Congress passed the Livestock Mandatory Price Reporting Act (MPR) to improve transparency in livestock markets by publicly reporting transaction data that includes average prices paid between meat packers and producers. This legislation was passed in response to concerns that the heavy concentration of meat packing firms gives packers market power (Koontz and Ward 2011). The act requires meat packers that annually slaughter 125,000 cattle, 100,000 swine, or 75,000 lambs to report daily transactions to the U.S. Department of Agriculture's Agricultural Marketing Service (USDA-AMS). The USDA-AMS reports summaries for the entire industry and for five regional U.S. markets. Prior to this legislation, meat packers voluntarily reported transactions to the USDA-AMS, and the agency produced publicly available reports summarizing the voluntary data. By requiring meat packers to report transactions, more information is available to producers and meat packers. A sunset provision was written into the MPR legislation with an expiration date of September 2005, and the act was renewed through 2010. Recently, this legislation was renewed again through 2015;

creating new debate and interest about the effects of MPR (see USDOJ-USDA 2010). With the legislation having another sunset provision and many questions remaining unanswered, more research is needed into the effects of MPR in cattle markets (Koontz and Ward 2011).

In an early article about MPR, Wachenheim and DeVuyst (2001) express concerns that packers could use information in USDA-AMS reports created from MPR to act strategically and exercise market power at the expense of producers. Their concerns are not based on a formal model but on empirical observation from industries that use posted prices such as the railroad industry (Fuller, Ruppel, and Bessler 1990) and long distance telephone service (MacAvoy 1995). . Azzam (2003) provides the first theoretical model to study the implications of MPR. His model follows a Cournot framework and finds the USDA-AMS reports may be of little value to producers, but might increase competition among packers. Overall, Azzam (2003) concludes MPR should increase competition among meat packers, which would cause the prices paid by packers to increase and the variance of reported prices to decrease. Njoroge (2003) modifies Azzam's (2003) model by assuming meat packers have asymmetric prior distributions of livestock prices, and finds that MPR can promote collusion among meat packers. Koontz and Ward (2011) argue that for cattle markets, the assumption of asymmetric prior distributions is less realistic than Azzam's (2003) assumption of symmetric priors. Koontz and Ward (2011) explain that large meat packers buy and process cattle in many different regional markets, which provide them with mostly symmetric information about prices. Njoroge et al. (2007) extend Njoroge (2003) by measuring welfare effects before and after MPR with a Cournot model. Depending on the value of the social benefits and costs, they find that MPR might benefit society, even though MPR can increase packers' market power. Furthermore, Azzam and Salvador (2004) use a theoretical model with risk-averse Cournot firms to measure the change in packers' market power after MPR was passed, and they find that in all five regions MPR would not increase packers' market power. In summary, the results from previous theoretical models do not agree, and questions exist about the appropriateness of the Cournot assumption for livestock markets.

The theoretical models in the MPR literature that measure market power in cattle markets use a Cournot framework, which assumes packers make production decisions based on quantity (Azzam 2003; Azzam and Salvador 2004; Njoroge 2003; Njoroge et al. 2007). However, cattle are purchased by packers in the U.S. where packers mostly choose a price for cattle, which is a Bertrand framework. Theory shows in aggregate models that the Cournot solution can find market power, while the Bertrand solution is the competitive solution (Varian 1992). Zhang and Brorsen (2010) conclude that the Cournot model predicts more market power than is observed in actual cattle markets (Crespi, Xia, and Jones 2010; Ward 2002) and argue that an alternative approach is needed. In our review of the literature, an auction model has never been used to determine the changes in packers' market power due to MPR, and this approach is closer to the way cattle are actually purchased than the Cournot models.

Theoretical auction models (Milgrom and Weber 1982) and experimental human auctions (Corrigan and Rousu 2011; Kagel and Levin 1986) have found that revealing information to buyers can increase the equilibrium price, but do not consider how providing information to sellers affects the equilibrium price. Anderson et al. (1998) use experimental data from the Packer-Feeder game, and find that an increase in public information to packers increases the market price and decreases price variability. The Anderson et al. (1998) paper is informative in understanding the effects of MPR in fed cattle auctions, but does not exactly match an auction situation and is empirical rather than theoretical. The human experiments suggest that MPR can reduce market power and increase prices paid by packers in auctions, which conflicts with the theoretical Cournot models of Njoroge (2003) and Njoroge et al. (2007). Recent econometric work with actual cattle transaction data by Cai, Stiegert, and Koontz (2011) finds that packers' market power increased after the introduction of MPR; however, factors other than MPR could be the cause of increasing market power. These inconsistencies in the theoretical models, human experiments, and econometric work warrant Koontz and Ward's (2011) call for further investigation into the changes in beef packers' market power after MPR.

Experimental auctions can be useful to study the changes in market power before and after MPR (Anderson et al. 1998; Carlberg, Hogan, and Ward 2009), but controlled human experiments can be expensive, and they usually provide little understanding of the agents' motivations. Furthermore, theoretical auction models with sufficient detail to closely match cattle markets are perhaps intractable to solve analytically. Agent-based computational modeling is becoming a complementary technique to human experiments and theoretical models (Alkemade, La Poutré, and Amman 2006; Arifovic 1996; Bonabeua 2002), and has successfully been used to model auctions (Guerci, Stefano, and Cincotti 2008; Hailu and Thoyer 2010; Tesfatsion 2001). In these models, artificial markets are created that allow buying and selling agents to trade until an equilibrium criterion is reached (Tesfatsion 2001). In agricultural economics, agent-based models have mostly been used to study land use (Balmann 1997; Berger 2001), but were used by Zhang and Brorsen (2010) with a Cournot model of the cattle procurement industry, and by Graubner, Balmann, and Sexton (2011) to address spatial price discrimination by processors of agricultural commodities.

We develop an agent-based, first-price common-value auction model to determine the effects of reducing the uncertainty of buyers and sellers. The objective is to provide a different theoretical approach to understanding the change in packers' market power before and after implementing MPR. In the auction model, buyers choose a bid price, and sellers choose a reserve price, which is a more realistic model of actual cattle procurement than the Cournot models. Common-value auction solutions are difficult to obtain analytically (Gordy 1998) and including a reserve price might make it infeasible to solve, which makes an agent-based model appropriate. Simulated buyers and sellers use Zhang and Brorsen's (2009) particle swarm optimization (PSO) learning algorithm combined with numerical integration to find their optimal bid and reserve price strategies. In our model both buyers and sellers are uncertain about cattle value. The effects of MPR are modeled by reducing this uncertainty. This research is a contribution to the MPR theoretical literature, and determining the effects of a noisy reserve price signal on equilibrium is also a contribution to the auction literature.

Auction Theory

Even though fed cattle are mostly purchased by private treaty rather than a formal auction, auction models are commonly used to model cattle sales. Auctions are normally classified as common value, private value, or affiliated value (Krishna 2002; Milgrom and Weber 1982). If the value of the item being sold is equal for all buyers or the resale value of the item is the same for all buyers, then the auction is considered a common value auction (Kagel and Levin 2002). In a private value auction, a buyer's bid price for the item is determined by the buyer's unique private value for the item such as an art auction (Paarsch 1992). The affiliated value auction is a more universal case where the resale value of the item is similar for all buyers, but value differs across buyers (Pinkse and Tan 2005). In reality, auctions rarely exactly match one of these three types of auctions, and are more likely a hybrid of these auction types (Corrigan and Rousu 2011).

Cattle sales do not exactly match any type of auction. Crespi and Sexton (2004) state cattle sales are an example of a hybrid auction type, but argue that cattle probably have affiliated values among buyers. Chung and Tostão (2009) and Tostão, Chung, and Brorsen (2011) use an independent private values auction model to estimate the equilibrium bid price for an experimental fed cattle auction. They admit the private values assumption is too restrictive for cattle, but follow it because common value and affiliated value auctions have econometric identification problems. However, wholesale beef prices received by packers are similar enough across packers to assume common value auction theory is appropriate to model cattle auctions.

We establish an agent-based first-price common value auction model similar to Kagel and Levin's (1986) analytical model, but allow sellers to impose a reserve price (or minimum bid price). We choose the first-price auction with a reserve price because it most closely resembles actual fed cattle sales (Crespi and Sexton 2004; 2005). Beef packing is highly concentrated with a few large packers (Ward 2002; 2010), which is why we choose to model the auction with two buyers. This auction structure allows buyers to exercise market power on sellers by shading bids below the market value (Klemperer 1999; Robinson 1985), and also lets us determine if market power decreases by reducing the buyers' or sellers' uncertainty of the item's value. In the last decade, packers have increased their use of alternative marketing agreements to purchase cattle, and have relied less on cash markets (Ward 2009a; 2009b; 2010). Cash market purchases of cattle by packers, however, remain a considerable portion of packers' cattle procurement (Crespi and Sexton 2004; Ward 2009a; 2009b), and formula prices are often based on cash prices.

In the agent-based auction, buyers with homogeneous preferences submit bids on an item based on their noisy information signal. The bid function used follows Kagel and Levin (1986) and matches Andreoni and Miller's (1995) proposed common value bid function for an agent-based auction, and is

$$b_i = x_i - \varepsilon_i \tag{2.1}$$

where b_i is the bid price for buyer i=1, 2, $x_i \in [-1,+1]$ is the choice variable for buyer i, $\varepsilon_i \sim U[0,1]$ $\text{cov}(\varepsilon_i, \varepsilon_j) = 0 \ \forall i \neq j$ is the noisy signal buyer i receives about the value of the item.

Sellers are assumed to be homogeneous, price-takers, and non-competitive so one representative seller is included in the model. The seller imposes a reserve price $r = m - \upsilon$, where $m \in [-1,+1]$ is the seller's choice variable and $\upsilon \sim U[0,1]$ is the seller's noisy signal, which is independent of buyers' signals. This model provides a unique contribution to the auction literature since the effect of a noisy reserve price on the equilibrium bid price in an auction has not received much attention (Reiley 2006). Buyers must bid higher than the other buyers and greater than the seller's reserve price to win the bid. In this model, expected revenue for buyer one is

$$E[R_1^B] = \int_{\underline{x}}^{x_1}\int_{\underline{b}}^{b_1}\int_{m-1}^{r}(X - b_1)\,dr\,db_2\,db_1 \tag{2.2}$$

where R_1^B is the expected revenue for buyer one, X is the buyers' retail value of the item being sold, r is the reserve price chosen by the seller, $\underline{b} = \max\{r, x_2 - 1\}$, and $\underline{x} = \max\{\underline{b}, x_1 - 1\}$. The buyers' retail value in the agent-based auction is X=0.5. Agent-based buyers do not retain the true value of the item so it is not necessary to randomly generate a new true value for the item in each repeated auction as in Kagel and Levin (1986). Buyers are bid-shading when the average winning bid price is less than 0.5, and the winner's curse outcome is present if the average winning bid price is greater than 0.5.

When all bid prices are less than the seller's reserve price then the seller does not sell the item. The seller's expected revenue is the sum of each buyer's expected winning bid price multiplied by the probability of the buyer winning and can be specified as

$$E[R^S] = \sum_{i=1}^{2}[X - E[R_i^B \mid b_i > b_j, b_i \geq r \quad \forall i \neq j] * P_i] \tag{2.3}$$

where R^S is the expected revenue for the representative seller and P_i is the probability buyer i wins the item.

The model is also set up with asymmetric buyers following Njoroge's (2003) assumption. In the asymmetric model, buyer one's noisy signal is changed to be distributed $U[0,1/2]$, which reduces buyer one's uncertainty to be less than that of the seller and buyer two. The change in mean of the uniform distribution does not matter since we report average of the bid price (b_i) rather than the average of the choice variable (x_i).

For the agent-based auction after MPR, price information provided by MPR is assumed to reduce the noisy signal for buyers and the seller. Therefore, the agent-based common value auction model is modified to represent a cattle auction after MPR by reducing the uncertainty to be $U[0,1/2]$. Results are presented for several market scenarios. We reduce the seller's noisy signal while the buyers' noisy signal remains distributed $U[0,1]$, which represents MPR reducing only the seller's uncertainty. Conversely, we reduce the buyers' noisy signal while the seller's noisy signal is not reduced, which represents MPR reducing only buyers' uncertainty. Finally, we reduce both the seller's and the buyers' noisy signal, which represents MPR reducing both buyer and seller uncertainty in cattle auctions. The winning equilibrium bid price, expected revenues, and the probability the item is sold are presented for each scenario.

This paper extends the auction literature by showing the effects of providing sellers with information or reducing their uncertainty. Milgrom and Weber (1982) show theoretically that revealing public information to buyers in an auction increases competition between buyers, and can likely increase the equilibrium bid price. Also, Kagel and Levin (1986) found releasing public information to the buyers in a common value auction increases the sellers' revenue in a human experiment. Our model differs from the models in these papers because we allow the seller to impose a reserve price, and the seller receives a noisy signal about the value of the item. Our results show how providing the seller with information changes the equilibrium, which Milgrom and Weber (1982) and Kagel and Levin (1986) do not show.

Method

Figure II-1 summarizes the general process of the agent-based auction with a reserve price. First, the buyers and the seller randomly choose a bid and reserve price strategy. The choice variables are x_1, x_2, and m. Numerical integration is used to calculate the agents' expected revenues, average bid prices, average reserve price, winning bid price, and the probability of winning for each buying and selling agent. At the end of each iteration, the convergence criterion for equilibrium is tested. If the convergence criterion is not met, then the particle swarm optimization (PSO) learning algorithm directs the agents to improved strategies for the next iteration. If the convergence criterion is met, then the model progresses to the next run. We combine numerical integration with the PSO learning algorithm to solve an agent-based auction when the agents have a noisy signal. Combining numerical integration with the PSO learning algorithm is an extension of the agent-based model literature.

Numerical Integration

Numerical integration is used to approximate the buying and selling agents' expected revenues. Recently, new methods have been developed to numerically solve simulated auction problems (Fibich and Gavish 2011; Gayle and Richard 2008; Peng and Yang 2010), but they are not easily adaptable to agent-based auctions. The trapezoidal rule is used to approximate the expected revenue integrals. Other integration algorithms such as Gaussian quadrature (Preckel and DeVuyst 1992) could be used to approximate these integrals, but we choose the trapezoidal rule because it is straightforward to program and efficient enough for the problem being considered.

The trapezoidal rule uses the area of trapezoids for *n* subintervals or lengths to approximate the integral. The trapezoidal rule is mathematically defined as

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}(y_0 + 2y_1 +,...,+2y_{n-1} + y_n) \tag{2.4}$$

where $\Delta x = (b\text{-}a)/n$, *f(x)* is the probability density function of winning the item, $y_0,...,y_n$ are the grid points in the integral, *b* is the upper bound of the integral and *a* is the lower bound of the integral. The probability weights at each point of the integral are Δx except at the tails where the probability weight is $\Delta x/2$. With the uniform distribution, *n* is assigned to be 100, resulting in a $\Delta x = 1/100$, and a probability weight of 1/100 except at the tails where the probability weight is 1/200. A large number of points are used here to reduce discreteness that can reduce accuracy in a winner-take-all problem such as an auction.

Particle Swarm Optimization

The buyers and the seller in the agent-based auction use a PSO learning algorithm developed by Eberhart and Kennedy (1995) and modified by Zhang and Brorsen (2009) to choose their bid and reserve price strategies. PSO is a stochastic global optimization method that tends to converge faster than genetic algorithms. Zhang and Brorsen's (2009) PSO algorithm has $k = 1,...,K$ parallel markets, and in each of the *k* parallel markets, the buying and selling agents have a clone participating in each one of the *k* markets. This idea of parallel markets came from watching the way flocks of birds share information to find food and avoid predators. The PSO algorithm gives each buyer and seller their own "flock of birds" to share information, but does not share information with the flocks of other buyers and sellers. Similarly, we have *k* parallel auctions with each buying and selling agent having a clone in each of the *k* parallel auctions. Therefore, the *k*th clone of an agent chooses a bid or reserve price strategy in the *k*th auction. The agents use the parallel auctions to learn what are good and bad strategies by evaluating revenues for each of their clone's strategies.

The way the PSO works is the *i*th agent has a bid or reserve price strategy where they have a choice variable $x_{i,k,t}$ in parallel auction *k* during iteration $t = 1,...T$, and an adjustment velocity $v_{i,k,t} \in [-1,+1]$ that directs the agent to a new value for the choice variable. Each velocity change is a function of the local best solution $p^{l}_{i,k,t}$, where superscript *l* indicates the best solution in the local auction and the agent's global best solution $p^{g}_{i,t}$, where superscript *g* indicates the best global solution. The *i*th agent's new choice variable in parallel auction *k* is updated by $x_{i,k,t+1} = x_{i,k,t} + v_{i,k,t}$, and the velocity is

$$v_{i,k,t+1} = wv_{i,k,t} + q_1u_1(p^{l}_{i,k,t} - x_{i,k,t}) + q_2u_2(p^{g}_{i,t} - x_{i,k,t}) \qquad (2.5)$$

where *w* is the inertia weight factor, u_1 and u_2 are random variables distributed *U*[0,1], and q_1 and q_2 are learning parameters. The learning parameters are $q_{1,t} = q_{2,t} = \beta_0^{q_1} + \beta_1^{q_1}(T-t)/T$ where both $\beta_0^{q_1}$ and $\beta_1^{q_1}$ are constants set at one and $T = 500$ is the maximum number of iterations. When selecting an inertia weight *w* there is an important tradeoff between exploration and convergence time to consider (Chatterjee and Siarry 2006). A large inertia weight slows convergence, but encourages agents to have a larger exploration area, while a small inertia weight increases convergence time, but reduces the agents' exploration area (Zhang and Brorsen 2009). Zhang and Brorsen (2009) manage this tradeoff by letting *w* start high and decrease as the model proceeds through the iterations, similar to the learning parameters. Inertia weight expressed in Zhang and Brorsen (2009) as $w_t = \beta_0^w + \beta_1^w (T - t)/T$ where both β_0^w and β_1^w are constants set at 0.75.

The strategy with the highest revenue for the agent in the *k*th auction is selected as the best local strategy. The agents' best strategy depends on the other agents' choice variables; therefore, the agent's previous best strategy may not perform well in the next iteration. The other agents' strategies in the current period are held constant and the past *L* best locals of each agent are reevaluated in auction *k*. The best local solution is expressed as

$$p_{i,k,t}^{l} = \arg\max\left\{R_k(p_{i,k,t-1}^{l}),...,R_k(p_{i,k,t-L}^{l}), R_k(x_{i,k,t}) \middle| x_{j\neq i,k,t}\right\} \tag{2.6}$$

where R_k is the revenue in parallel market *k*. The best global solution is selected from the best local parameters and is

$$p_{i,t}^{g} = \arg\max\left\{R_1(p_{i,1,t}^{l}), R_2(p_{i,2,t}^{l}), \ldots, R_K(p_{i,K,t}^{l})\right\}. \tag{2.7}$$

Table II-1 shows the values used for the PSO parameters.

Convergence Criterion

A convergence criterion is established to determine when the buyers and the seller cannot find bid and reserve price strategies that are an improvement from their previous strategies. The model converges when the total change in the previous 10 strategies for each of the agents is less than 0.0000001, and the total change in the standard deviations of the strategies are less than 0.00000001. The model has 100 runs (*E*=100), and each run has a maximum of 500 iterations (*T*=500). That is, the agents have 500 iterations (or trades) to meet the convergence criterion, and this process is repeated 100 times. The results presented are the averages and standard deviations for all 100 runs. If the convergence criterion is met by all buying and selling agents, then the next run begins. But if the convergence criterion is not met by all buying and selling agents, then the next iteration begins, and the PSO learning algorithm guides the agents to improved strategies in the next iteration.

Results

Full Noise

The agent-based auction model is initially set up with the noisy signals received by buyers and the seller to be distributed $U[0,1]$, which represents an auction prior to MPR with symmetric buyers. The average winning bid price is 0.02, and the seller's average reserve price is -0.31 (table II-2). The average winning bid price is below the buyers' retail value of 0.5, which indicates that buyers shade bids. The average bid price for buyer one across all winning and losing bids is -0.24, and buyer one won, on average, 36% of the items offered (table II-2). Buyer two's average bid price is -0.26, and won, on average, 34% of the items offered (table II-2). Expected revenue per item for all items offered is 0.17 and 0.16 for buyer one and buyer two; respectively. Since buyer one and buyer two are identical, the difference in their average bid prices is due to randomness in the simulations. Seventy percent of the items are sold, and the seller's expected revenue per item for all items sold or unsold is 0.01 (table II-2). Past research finds a reserve price can reduce the market power in an auction, and without a reserve price imposed by the seller, the seller's expected revenue would be lower[1] (Leven and Smith 1996; Reiley

2006; Riley and Samuelson 1981). But this past research has not given sellers a noisy reserve price.

Njoroge (2003) proposes buyers are asymmetric in cattle markets, so we set up an agent-based auction model with asymmetric buyers. Buyer one receives a noisy signal distributed $U[0,1/2]$ while buyer two and the seller receive a noisy signal distributed $U[0,1]$. This scenario corresponds to an auction prior to MPR when buyer one has better information about prices paid for cattle than the seller and buyer two. The average winning bid price is 0.01, and the average reserve price is -0.38 (table II-2). Expected revenue per item and the probability of winning the item for buyer one increases, and buyer two's expected revenue per item decreases slightly when buyer one has better information about prices than buyer two (table II-2). The seller's expected revenue per item does not change from the scenario with symmetric buyers prior to MPR (table II-2). Compared to the symmetric buyers' results, buyer one benefits from having better information than buyer two, which is anticipated, and the seller's expected revenue per item does not change.

Reduced Noise

Results for the agent-based auction are presented in table II-2 when the noisy signal the seller receives is reduced to be distributed $U[0,1/2]$, and the buyers' noisy signal is distributed $U[0,1]$. This scenario represents an auction after MPR, where additional public information reduces the seller's uncertainty about the value of the item, but it does not reduce the buyers' uncertainty. Compared to the results from the scenarios with full noise, the average winning bid price increases to 0.11, and the seller's average reserve price increases to -0.08 (table II-2). The probability of winning the item decreases for the buyers, and their expected revenues per item decrease (table II-2). The seller's expected revenue per item increases to 0.06, and the probably of a transaction occurring decreases (table II-2). When the seller's noisy signal about the value of the item is reduced, the buyers' market power decreases and the seller expected revenue per item increases. This finding shows that the seller benefit from reduced uncertainty, which extends the auction literature.

Table II-2 also presents results with both buyers' noisy signals reduced to $U[0,1/2]$ and the seller's noisy signal remains $U[0,1]$. This scenario symbolizes an auction after MPR, where additional public information reduces buyers' uncertainty about the value of the item, but the seller's uncertainty is not reduced. The average winning bid price is 0.05, and the seller's average reserve price is -0.46 (table II-2). The buyers' market power or bid-shading is reduced when the buyers' noisy signal decreases. Relative to the full noise results, the probability of winning the item for the buyers increases as well as their expected revenues per item (table II-2), and the seller's probability of selling the item increases and the seller's expected revenue per item increases 0.05 (table II-2). Similar to results when the seller's noise is reduced, these results reveal that sellers benefit when only the buyers' noisy signal decreases due to additional public information. This finding matches Milgrom and Weber's (1982) theoretical model and Kagel and Levin's (1986) human experiment; however, neither allows sellers to impose a reserve price. The information made available from the MPR legislation should level the playing field for buyers by making them symmetric, and results show that the seller still benefits even if only one buyer gains information. This is counter to what Njoroge (2003) and Njoroge et al. (2007) conclude.

When the noisy signals that the seller and the buyers receive are all reduced by half, the average winning bid price increases to 0.10 (table II-2). The reduction of noise in the seller's reserve price and the buyers' bid price increases competition between buyers enough to increase the average winning bid price (table II-2). Relative to the full noise scenarios, the seller's average reserve price and their expected revenue per item increases (table II-2). Results from this scenario are similar to the results from the other scenarios when the uncertainty in the auction market is reduced, the seller benefits.

The agent-based auction is not an actual cattle auction and actual markets differ from the model in numerous ways; however, the model provides a theoretical argument that reducing uncertainty through the USDA-AMS reports created by MPR decreases meat packers' market power, increases competition between meat packers, and increases the seller's expected revenue. This conclusion matches what Azzam (2003) finds theoretically and what Koontz and Ward (2011) believe is likely the result of MPR. Furthermore, if buyers with different levels of information are participating in an auction, then MPR would still benefit sellers. Even though inconsistent theoretical conclusions about MPR are still evident, the agent-based auction provides unique theoretical evidence that supports MPR as beneficial to producers.

Conclusions

We establish an agent-based, first-price common-value auction that allows the seller to impose a reserve price. This model is created to simulate fed cattle markets to determine how packers' market power changes with the implementation of MPR. Past theoretical models have disagreed about whether or not producers benefit from MPR, and recent empirical work has even shown packers' market packer in cattle markets increased after MPR (Cai, Stiegert, and Koontz 2011). Koontz and Ward (2011) express concern that there are several questions about MPR that remain unanswered.

The theoretical models in the MPR literature follow a Cournot framework. We use an agent-based common value auction model to determine the changes in market power after MPR. An auction model approach is closer to actual cattle procurement than the Cournot models, and is a unique theoretical contribution to the MPR literature. The common value auction with a reserve price is intractable to solve analytically so we develop an agent-based auction model, which makes the agent-based auction model an appropriate method to apply to this problem. Additionally, by giving the seller a noisy reserve price signal, we are able to determine how reducing the seller's uncertainty affects the buyers' market power, which is a contribution to the auction literature.

The agent-based common value auction models show that the average winning bid price increases and the buyers' market power decreases when the buyers' and/or the seller's noisy signal is reduced, and the seller's expected revenue increases. These conclusions match results from human experiments (Kagel and Levin 1986), and auction theory (Milgrom and Weber 1982). The agent-based auction indicates that if the MPR policy reduces either the buyers or sellers uncertainty, then sellers benefit from MPR.

References

Alkemade, F., J.A. La Poutré, and H.M. Amman. 2006. "Robust Evolutionary Algorithm Design for Socioeconomic Simulation." *Computational Economics* 28(4):355–370.

Anderson, J.D., C.E. Ward, S.R. Koontz, D.S. Peel, and J.N. Trapp. 1998. "Experimental Simulation of Public Information Impacts on Price Discovery and Marketing Efficiency in the Fed Cattle Markets." *Journal of Agricultural and Resource Economics* 23:262-79.

Andreoni, J., and J.H. Miller. 1995. "Auctions with Artificial Adaptive Agents." *Games and Economic Behavior* 10:39-64.

Arifovic, J. 1996. "The Behavior of the Exchange Rate in the Genetic Algorithm and Experimental Economies." *Journal of Political Economy* 104(3):510-541.

Azzam, A.M. 2003. "Market Transparency and Market Structure: The Livestock Mandatory Reporting Act of 1999." *American Journal of Agricultural Economics* 85(2):387-395.

Azzam, A.M., and S. Salvador. 2004. "Information Pooling and Collusion: An Empirical Analysis." *Information Economics and Policy* 16:275-286.

Bonabeau, E. 2002. "Agent-Based Modeling: Methods and Techniques for Simulating Human Systems." *Proceedings of the National Academy of Sciences of the United States of America* 99:7280-7287.

Balmann, A. 1997. "Farm-based Modeling of Regional Structural Change: A Cellular Automata Approach." *European Review of Agricultural Economics* 24: 85–108.

Berger, T. 2001. "Agent-based Spatial Models Applied to Agriculture: A Simulation Tool for Technology Diffusion, Resource Use Changes and Policy Analysis." *Journal of Agricultural Economics* 25: 245–260.

Cai, X., K.W. Stiegert, and S.R. Koontz. 2011. "Oligopsony Fed Cattle Pricing: Did Mandatory Price Reporting Increase Meatpacker Market Power? Proceedings of the NCCC-134 Conference on Applied Commodity Price Analysis, Forecasting, and Market Risk Management. St. Louis, MO. [http://www.farmdoc.illinois.edu/nccc134].

Carlberg, J.G., R.J. Hogan Jr., and C.E. Ward. 2009. "Game Theory Application to Fed Cattle Procurement in an Experimental Market." *Agribusiness: An International Journal* 25:56-69.

Chatterjee, A., and P. Siarry. 2006. "Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization." *Computers and Operations Research* 33:859–871.

Chung, C., and E. Tostão. 2009. "Non-parametric Estimation of Oligopsony Power in the First-Price Auction." *Journal of Agricultural Economics* 60(2):318-333.

Corrigan, J.R., and M.C. Rousu. 2011. "Are Experimental Auctions Demand Revealing When Values Are Affiliated?" *American Journal of Agricultural Economics* 93(2):514-520.

Crespi, J.M., and R.J. Sexton. 2004. "Bidding for Cattle in the Texas Panhandle." *American Journal of Agricultural Economics* 86:660-674.

Crespi, J.M., and R.J. Sexton. 2005. "A Multinomial Logit Framework to Estimate Bid Shading in Procurement Auctions: Application to Cattle Sales in the Texas Panhandle." *Review of Industrial Organization* 27:253-278.

Crespi, J.M., T. Xia, and R. Jones. 2010. "Market Power and the Cattle Cycle." *American Journal of Agricultural Economics* 92(3):685-697.

Eberhart, R.C., and J. Kennedy. 1995. "A New Optimizer Using Particle Swarm Theory." *Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan,* 39-43.

Fibich, G., and N. Gavish. 2011. "Numerical Simulation of Asymmetric First-Price Auctions." *Games and Economic Behavior* doi:10.1016/.geb.2011.02.010.

Gayle, W.R., and J.F. Richard. 2008. "Numerical Solutions of Asymmetric, First-Price, Independent Private Value Auctions." *Computational Economics* 32:245-278.

Gordy, M.B. 1998. "Computationally Convenient Distributional Assumptions for Common-Value Auctions." *Computational Economics* 12:61-78.

Graunbner, M., A. Balmann, and R.J. Sexton. 2011. "Spatial Price Discrimination in Agricultural Product Procurement Markets: A Computational Economics Approach." *American Journal of Agricultural Economics* 93(4):949-967.

Guerci, E., I. Stefano, and S. Cincotti. 2008. "Learning Agents in an Artificial Power Exchange: Tacit Collusion, Market Power and Efficiency of Two Double-Auction Mechanisms." *Computational Economics* 32:73–98.

Hailu, A., and S. Thoyer. 2010. “What Format for Multi-Unit Multiple-Bid Auctions? Agent-Based Simulation of Auction Performance and Non-linear Bidding Behavior.” *Computational Economics* 35:189-209.

Kagel, J.H., and D. Levin. 1986. “The Winner's Curse and Public Information in Common Value Auctions.” *American Economic Review* 76(5):894-920.

Kagel, J.H., and D. Levin. 2002. *Common Value Auction and the Winner’s Curse*. Princeton, New Jersey: Princeton University Press.

Koontz, S.R., and C.E. Ward. 2011.” Livestock Mandatory Price Reporting: A Literature Review and Synthesis of Related Market Information Research.” *Journal of Agricultural and Food Industrial Organizatio*n 9(1):Article 9.

Klemperer, P. 1999. “Auction Theory: A Guide to the Literature.” *Journal of Economic Surveys* 13:227–286.

Krishna, V. 2002. *Auction Theory*. San Diego, CA: Academic Press.

Levin, D., and J.L. Smith. 1996. “Optimal Reservation Prices in Auctions.” *The Economics Journal* 106:1271-1283.

Milgrom, P.R., and R.J. Weber. 1982. “A Theory of Auctions and Competitive Bidding.” *Econometrica* 50:1089-1122.

Njoroge, K. 2003. “Information Pooling and Collusion: Implications for the Livestock Mandatory Reporting Act.” *Journal of Agricultural and Food Industrial Organization* 1: Article 14.

Njoroge, K., A. Yiannaka, K. Giannakas, and A.M. Azzam. 2007. “Market and Welfare Effects of the U.S. Livestock Mandatory Reporting Act.” *Southern Economic Journal* 74:290-311.

Paarsch, H. 1992. “Deciding Between the Common and Private Value Paradigms in Empirical Models of Auctions.” *Journal of Econometrics* 51:191-215.

Peng, J., and Z. Yang. 2010. “Numerical Solution of Asymmetric, First-Price, Independent Private Value Auctions: Comment.” *Computational Economics* 36:231-235.

Pinkse, J., and G. Tan. 2005. “The Affiliation Effect in First-Price Auctions.” *Econometrica* 1:263-277.

Preckel, P.V., and E. DeVuyst. 1992. “Efficient Handling of Probability Information for Decisions Analysis under Risk.” *American Journal of Agricultural Economics* 74(3):655-662.

Reiley, D.H. 2006. “Field Experiments on the Effects of Reserve Prices in Auctions: More Magic on the Internet.” *The RAND Journal of Economics* 37(1):195-211.

Riley, J.G., and W. Samuelson. 1981. “Optimal Auctions.” *American Economic Review* 71:381-392.

Robinson, M.S. 1985. “Collusion and the Choice of Auction.” *The RAND Journal of Economics* 16:141-145.

Tesfatsion, L. 2001. “Introduction to the Special Issue on Agent-Based Computational Economics.” *Journal of Economic Dynamics and Control* 25:281–293.

Tostão, E., C. Chung, and B.W. Brorsen. 2011. “Integrating Auction Theory with Traditional Measures of Market Power.” *Agribusiness: An International Journal* 27(2)162-178.

U.S. Department of Justice and U.S. Department of Agriculture. 2010. *Public Workshop Exploring Competition Issues in Agriculture: Livestock Workshop*. Colorado, August.

Varian, H.R. 1992. *Microeconomic Analysis*. 3rd. ed. New York: W.W. Norton and Company.

Wachenheim, C., and E.A. DeVyust. 2001. "Strategic Response to Mandatory Reporting Legislation in the U.S. Livestock and Meat Industries: Are Collusive Opportunities Enhanced?" *Agribusiness: An International Journal* 17:177-97.

Ward, C.E. 2002. “A Review of Causes for and Consequences of Economic Concentration in the U.S. Meatpacking Industry.” *Current Agriculture, Food and Resource Issues* 3:1-28.

Ward, C.E. 2009a. “Extent of Alternative Marketing Arrangements for Fed Cattle and Hogs.” Oklahoma State University Oklahoma Cooperative Extension Service AGEC-615, February.

Ward, C.E. 2009b. “Price Comparison of Alternative Marketing Arrangements for Fed Cattle.” Oklahoma State University Oklahoma Cooperative Extension Service AGEC-616, February.

Ward, C. 2010. “Assessing Competition in the U.S. Beef Packing Industry.” *Choices* 25(2).

Zhang, T., and B.W. Brorsen. 2009. “Particle Swarm Optimization Algorithm for Agent-Based Artificial Markets.” *Computational Economics* 34:399-417.

Zhang, T., and B.W. Brorsen. 2010. “The Long Run and Short Run Impact of Captive Supplies on the Spot Market Price: An Agent-Based Artificial Market.” *American Journal of Agricultural Economics* 92:1181-1194.

Table II-1. Parameter Values for the Particle Swarm Optimization Algorithm

Parameter	Symbol	Value
Intercept of inertia weight	β_0^w	0.75
Slope of inertia weight	β_1^w	0.75
Learning parameters intercept	β_0^q	0.75
Learning parameter slope	β_1^q	0.75
Parallel markets	k	10
Maximum iterations	T	500
Evolutions	E	100
Memory	L	3

Table II-2. Equilibrium Values for the Agent-Based Duopsony Auctions

	Full Noise		Reduced Noise		
Category[a]	Symmetric Buyers[b]	Asymmetric Buyers[c]	Seller Reduced Noise[d]	Buyer Reduced Noise[e]	Reduced Noise[f]
Average winning bid price	0.02 (0.12)	0.01 (0.12)	0.11 (0.12)	0.05 (0.12)	0.10 (0.12)
Reserve price	-0.31 (0.26)	-0.38 (0.23)	-0.08 (0.19)	-0.46 (0.24)	-0.16 (0.20)
Buyer 1					
Average bid price	-0.24 (0.17)	-0.11 (0.14)	-0.25 (0.19)	-0.06 (0.15)	-0.05 (0.16)
Transaction probability	36.0% (17.0%)	48.0% (18.0%)	27.0% (18.0%)	45.0% (22.0%)	38.0% (24.0%)
Expected revenue	0.17 (0.09)	0.25 (0.11)	0.10 (0.08)	0.19 (0.12)	0.15 (0.11)
Buyer 2					
Average bid price	-0.26 (0.16)	-0.24 (0.18)	-0.23 (0.17)	-0.06 (0.16)	-0.04 (0.16)
Transaction probability	34.0% (16.0%)	32.0% (16.0%)	27.0% (15.0%)	45.0% (21.0%)	40.0% (23.0%)
Expected revenue	0.16 (0.08)	0.13 (0.07)	0.10 (0.06)	0.19 (0.10)	0.16 (0.10)
Seller					
Transaction probability	70.0% (22.0%)	80.0% (0.17%)	54.0% (24.0%)	90.0% (13.0%)	79.0% (24.0%)
Expected revenue	0.01 (0.08)	0.01 (0.09)	0.06 (0.07)	0.05 (0.10)	0.08 (0.10)

Note: Standard deviations are presented in parentheses. The competitive solution with no noise is a winning bid price of 0.5 with buyers having zero revenue.
[a] All results, except average winning bid price, are unconditional on winning.
[b] Errors for both the buyers and the seller are distributed U[0,1].
[c] Errors for buyer one is U[0,0.5] and for buyer two and the seller are U[0,1].
[d] Errors for the seller is distributed U[0,0.5] and the errors for the buyers are distributed U[0,1].
[e] Errors for the buyer is distributed U[0,0.5] and the errors for the seller are distributed U[0,1].
[f] Errors for both the buyers and the seller are distributed U[0,0.5].

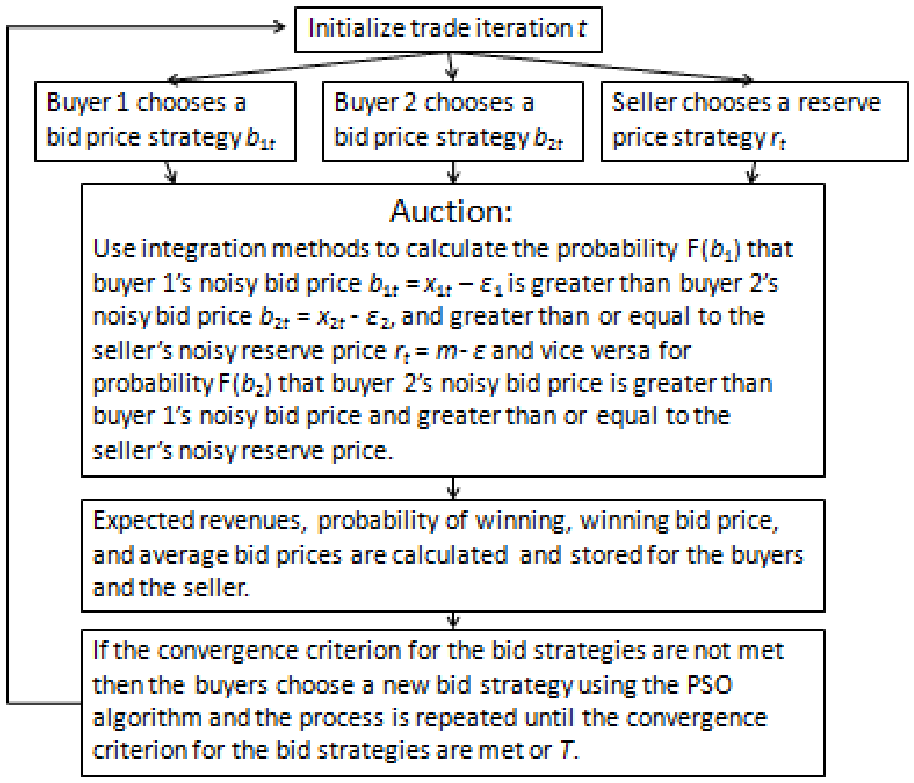
Initialize trade iteration t
Buyer 1 chooses a bid price strategy b_{1t}
Buyer 2 chooses a bid price strategy b_{2t}
Seller chooses a reserve price strategy r_t
Auction:
Use integration methods to calculate the probability F(b_1) that buyer 1's noisy bid price $b_{1t} = x_{1t} - \varepsilon_1$ is greater than buyer 2's noisy bid price $b_{2t} = x_{2t} - \varepsilon_2$, and greater than or equal to the seller's noisy reserve price $r_t = m - \varepsilon$ and vice versa for probability F(b_2) that buyer 2's noisy bid price is greater than buyer 1's noisy bid price and greater than or equal to the seller's noisy reserve price.
Expected revenues, probability of winning, winning bid price, and average bid prices are calculated and stored for the buyers and the seller.
If the convergence criterion for the bid strategies are not met then the buyers choose a new bid strategy using the PSO algorithm and the process is repeated until the convergence criterion for the bid strategies are met or T.

Figure II-1. **Overview of the Duopsony Agent-Based Auction Model**

CHAPTER III

COMMON-VALUE AUCTION VERSUS POSTED-PRICE SELLING: AN AGENT-BASED MODEL APPROACH

Introduction

With the recent development and easy access to online selling methods, sellers can choose to sell an item using either a posted-price market (e.g., Craigslist) or an auction (e.g., eBay). When the market-clearing price for the item is uncertain such as art, cattle, and government bonds, sellers are normally thought to prefer an auction over a posted-price market (Milgrom 1989). An auction allows sellers to discover the market-clearing price for the item through competition among buyers, which can be more profitable for sellers than a posted price. However, the theoretical literature reveals that auctioning an item does not always produce larger revenue for sellers than a posted-price market (Campbell and Levin 2006; Kultti 1999), and experimental data shows sellers commonly choose to sell an item with a posted price instead of using an auction (Hammond 2010).

Auctions are normally classified into three types: (1) private value, (2) affiliated value, and (3) common value (Krishna 2002; Milgrom and Weber 1982). Wang (1993) compares a seller's revenue from selling an item with a posted price to selling the same item using an auction with a reserve price when the buyers have independent private values for the item. Buyers with independent private values means buyers have their own unique value of the item (Paarsch 1992). Wang (1993) uses a theoretical model with the buyers having a noisy price signal and the seller having perfect information about the value of the item. Wang (1993) finds the auction is always optimal for sellers when the cost of auctioning the item is zero, but when there is an auctioning cost for the seller, the optimal selling method depends on the steepness of the seller's marginal revenue curve. Campbell and Levin (2006) compare sellers' revenue from selling an item with a posted price to auctioning it when the buyers' values for the item are affiliated, which means buyers' values might depend on a common unknown signal and buyers' individual values (Pinkse and Tan 2005). They use a theoretical model to demonstrate that an auction is not always optimal for a seller when buyers' values for the item are interdependent. Furthermore, Wang (1998) uses a similar theoretical model to Wang (1993) to show that auctioning an item is more profitable for the seller when buyers' private values are correlated, and Julien et al. (2002) use backwards induction to show that sellers are always better off selling the item in an auction with and without a reserve price than with a posted price.

Kultti (1999) adapts Lu and McAfee's (1996) model of an auction and a bargaining market to compare sellers' utility from selling an item with a posted-price market to selling via an auction when the value of the item being sold is common among buyers. When buyers have a common value for the item, the resale value of the item is equal across all buyers, but uncertainty about the value of the item results in each buyer potentially having a different estimate of the value (Kagel and Levin 2002). A classic example of a common-value auction in the literature is oil lease auctions (Capen et al. 1971; Kagel and Levin 2002). Kultti develops a model that compares the expected utility for buyers and sellers in a posted-price market and auction by evaluating the probability of buyers and sellers meeting in each market. Using evolutionary game dynamics, Kultti finds that the seller's expected utility in the posted-price market and the auction are always equivalent because the probability of buyers and sellers meeting in each market are equivalent. More recently, Hammond (2010) presents unique results from a field experiment of selling compact discs in an online ascending-bid second-price auction with a reserve price and an online posted-price market. Hammond (2010) cannot reject revenue equivalence for the sellers in the posted-price and auction markets, and also finds that the posted-price markets had a higher selling price and lower probability of selling the item than the auctions.

While Kultti (1999) does compare these selling methods when buyers have a common value, his model uses a different framework from Wang's (1993) private value model and Campbell and Levin's (2006) affiliated value model, which compare seller's revenue when sellers choose a reserve price and buyers choose a bid price while receiving a noisy price signal. Kultti (1999) compares the probability of buyers and sellers meeting in each market and does not consider bid shading due to buyer uncertainty. A model with buyers and sellers receiving a noisy price signal that allowed comparing a seller's revenue from auctioning an item with and without a reserve price to selling the item with a posted price when the item has a common value among buyers would be a unique contribution to the posted price and auction literature.

Solving a common-value auction analytically is difficult and allowing the seller to impose a reserve price (minimum bid) perhaps makes it intractable (Gordy 1998), and reproducing a common-value auction and a posted-price market in a laboratory or field experiment would likely require many rounds and a large sample, which would be expensive to obtain. Agent-based computational modeling is an alternative approach that can be used to compare these two selling methods. Economists use agent-based models to test economic theories when the problem is intractable and obtaining data is expensive (Alkemade et al. 2006; Arifovic 1996; Bonabeua 2002). These models are artificial markets where interactive agents simulate economic markets following trading and equilibrium rules (Tesfatsion 2001). Recently, Noe et al. (2011) successfully developed an agent-based common-value auction to examine how bid prices change under retail price uncertainty. We develop an agent-based common-value auction with a reserve price and an agent-based posted-price market to compare the seller's expected revenue. Our common-value auction model extends Noe et al.'s (2011) model by allowing the seller to impose a reserve price, and by giving buyers and the seller a noisy price signal. Additionally, using agent-based models allows us to relax the assumption of symmetric buyers found in previous theoretical work (Campbell and Levin 2006; Wang 1993), and we can evaluate these selling methods with asymmetric buyers.

We compare the seller's expected revenue from selling an item in a first-price common-value auction with a reserve price and in a posted-price market to determine the selling method that provides the seller with the larger expected revenue. We develop an agent-based posted-price market with two buyers and one seller, and an agent-based first-price common-value auction with a reserve price that includes two buyers and one seller. In the posted-price model, the seller chooses a posted price for the item being sold, and buyers choose their willingness-to-pay. Buyers purchase the item at the posted price when their willingness-to-pay is greater than or equal to the seller's posted price. In the auction model, the seller chooses a reserve price and buyers choose their bid price. Buyers purchase the item at their bid price when their bid price is greater than or equal to the seller's reserve price and greater than the other buyer's bid. In both markets, buyers and the seller use a particle swarm optimization (PSO) learning algorithm developed by Zhang and Brorsen (2009) combined with numerical integration to find their best strategy (bid price, reserve price, willingness-to-pay, and posted price).

Theory of Agent-Based Models

We make similar assumptions in our agent-based models about the buyers and sellers cost structures to Wang's (1993) model. The cost to store the item when it is not sold is the buyers' retail value of the item, which is the same[2] for the seller in the auction and posted-price market. Also, we assume buyers' search cost is equal in the posted-price market and in the auction. This means buyers in the auction would have the same cost of searching for the same item in other auctions as buyers in the posted-price market searching for the same item in other posted-price markets. Finally, the cost of setting up and organizing an auction is equal to the cost of setting up and organizing a posted-price market.

Common Value Auction

The winner's curse phenomenon is well documented in human and field experiments (Kagel and Levin 2002) as well as in actual (Capen et al. 1971) common-value auctions. This occurs when buyers estimate the value of an item to be greater than the item's true

[2] Storage costs are assumed to be 0.5 and since the value of the item is 0.5, an equivalent assumption would be to assume that the item is perishable.

value, and bid their estimated values for the item, which results in the winning buyer receiving a negative profit for the item. If the winner’s curse is present in the common-value auction model, then an auction might produce larger expected revenue for the seller than a posted price. Conversely, bid-shading by buyers is sustainable in common-value auctions (Klemperer 1999) and in repeated game auctions (McAfee and McMillan 1987). If bid-shading occurs, the seller might be worse off auctioning the item. The winner’s curse is not expected to occur in our model due to the learning algorithm, and bid-shading by buyers is expected, which is what Noe et al. (2011) found.

Our agent-based first-price common-value auction model is based on the model used in Kagel and Levin’s (1986) human experiment. A slight modification we make to Kagel and Levin (1986) is the agent-based auction assumes a constant true value that is greater than zero instead of randomly generating a new true value in each repeated auction. They randomly generated a new true value for the item being auctioned in each repeated auction because otherwise buyers might base their bid on the previous true value of the item instead of their signal. Buyers in the agent-based auction cannot learn the true value of the item; thus, it is not necessary to randomly generate a new true value for the item in each repeated auction and bids are evaluated around some constant true value that is greater than zero. In the agent-based auction, bid prices are submitted by buyers with homogeneous preferences for an item based on their noisy information signal. The bid function follows previous bid functions in common-value auctions (Andreoni and Miller 1995; Kagel and Levin 1986), and is

$$b_i = x_i - \varepsilon_i \tag{3.1}$$

where b_i is the bid price for buyer i=1, 2, $x_i \in [-1,+1]$ is the choice variable in the auction for buyer i, and $\varepsilon_i \sim U[0,1]$ $\text{cov}(\varepsilon_i, \varepsilon_j) = 0\ \forall i \neq j$ is the noisy signal buyer i receives about the value of the item.

Another modification we make to Kagel and Levin (1986) is we allow the seller to impose a reserve price. The seller's reserve price is $r = m - \upsilon$ where $m \in [-1,+1]$ is the seller's choice variable, and $\upsilon \sim U[0,1]$ is the noisy signal and is independent of buyers' signals. By giving the seller a noisy signal about the value of the item, we can test Milgrom's (1989) theory that an uncertain value for an item usually results in the auction being preferred by sellers over a posted price. According to Milgrom (1989), the seller's noisy signal about the value of the item should cause the seller to prefer the agent-based auction over the agent-based posted-price market. Results are also included when the seller has perfect information about the reserve price. When the seller's reserve price is not stochastic, the reserve price should converge to the seller's true value of zero (Levin and Smith 1996). Furthermore, we present results for an agent-based common-value auction without a reserve price. Buyers must bid greater than the other buyer and greater than or equal to the seller's reserve price to win the item. In this model, expected revenue for buyer one is

$$E[R_1^B] = E[(X - b_1)I[b_1 > b_2]I[b_1 \geq r]] \tag{3.2}$$

where R_1^B is the expected revenue for buyer one, $I[b_1 > b_2]$ is an indicator function for when buyers one's bid price is greater than buyer two's bid price, $I[b_1 \geq r]$ is an indicator function for when buyer one's bid price is greater than the reserve price, and X is the buyers' retail value of the item being sold. The buyers' retail value is X=0.5. Buyers are bid-shading when the average winning bid price is less than 0.5, and the winner's curse outcome is present if the average winning bid price is greater than 0.5.

The seller's expected revenue is the sum of buyer one's bid price multiplied by the probability of buyer one winning the item and buyer two's bid price multiplied by the probability of buyer two winning the item. The seller's expected revenue is expressed as

$$E[R^S] = \sum_{i=1}^{2}[X - E[R_i^B \mid b_i > b_j, b_i \geq r \quad \forall i \neq j] * P_i] \tag{3.3}$$

where R^S is the expected revenue for the representative seller, and P_i is the probability buyer i wins the item.

Posted Price Market

Classical economic models derive the seller's optimal posted price by equating marginal revenue with marginal cost. Search models were developed by Stigler (1961) and McCall (1965) that consider the buyers' cost of searching for the lowest posted price available, and are commonly used in the posted-price market literature. In search models, the seller's optimal posted price is a balance of the probability of selling the item, buyers' search cost, and the price the seller is willing-to-accept for the item (Arnold and Lippman 2001). Previous theoretical work reveals a posted-price market can produce larger expected revenue than an auction for the seller in an affiliated-value auction (Campbell and Levin 2006). However, a posted-price market has not been compared to a common-value auction.

Similar to the agent-based auctions, the buyers in the agent-based posted-price market do not learn the retail value of the item; thus, it is not necessary to randomly generate a new true value for the item in each repeated auction and bids are evaluated around some constant true value that is greater than zero. Buyers with homogeneous preferences submit their willingness-to-pay for the item based on their noisy information signal. Buyers *i*'s willingness-to-pay function is

$$w_i = y_i - \varepsilon_i \tag{3.4}$$

where w_i is the willingness-to-pay for buyer *i*=1, 2, $y_i \in [-1,+1]$ is the choice variable in the posted-price market for buyer *i*, $\varepsilon_i \sim U[0,1]$ $\text{cov}(\varepsilon_i, \varepsilon_j) = 0 \ \forall i \neq j$ is the noisy signal buyer *i* receives about the value of the item.

The seller sets a posted price $p = z - \upsilon$ where $z \in [-1,+1]$ is the seller's choice variable in the posted-price market, and $\upsilon \sim U[0,1]$ is the noisy signal and is independent of buyers' signals. We also present results when the seller has perfect information. Buyers must be willing to pay a value greater than or equal to the seller's posted price to purchase the item. In this model, expected revenue for buyer one is

$$E[R_1^B] = E[(X - p)(0.5)^{I[w_2 \geq p]} I[w_1 \geq p]] \tag{3.5}$$

where R_1^B is the expected revenue for buyer one, *X* is the buyers' retail value of the item being sold, $I[w_2 \geq p]$ is an indicator function for when buyers two's willingness-to-pay is greater than or equal to the posted price, and $I[w_1 \geq p]$ is an indicator function for when buyer one's willingness-to-pay is greater than the posted price. In the agent-based posted-price model, agents arrive in the market at the same time; therefore, when both buyers have willingness-to-pay greater than or equal to the posted price, the buyers equally split the item being sold. As in the auction, the buyers' retail value in the posted-price market is *X*=0.5.

When the willingness-to-pay is less than the seller's posted price then the seller does not sell the item and the seller's revenue is zero. The seller's expected revenue is similar to the seller's expected revenue in an auction, but the seller only receives the posted price for the item. The seller's expected revenue is

$$E[R^S] = \sum_{i=1}^{2} [X - E[R_i^B \mid w_i \geq p \quad \forall i \neq j] * P_i] \tag{3.6}$$

where R^S is the expected revenue for the representative seller, and P_i is the probability buyer i purchases the item.

Method

Figure III-1 summarizes the general process of the agent-based auction with a reserve price. Buyers choose a bid price strategy and the seller chooses a reserve price strategy. Numerical integration is used to estimate the agents' expected revenues, the average winning bid price, average reserve price, average bid prices, and probability of each buyer winning. Finally, the agents' strategies are tested to determine if the convergence criterion for equilibrium is satisfied. If the convergence criterion for equilibrium is not satisfied, then the particle swarm optimization (PSO) learning algorithm directs the agents to improved strategies in the next iteration, and if the convergence criterion is met, then the model advances to the next run.

Figure III-2 outlines a similar process of the agent-based posted-price market. The seller chooses a posted price strategy for the item, and buyers choose their willingness-to-pay for the item. Numerical integration is used to estimate the agents' expected revenues, the average selling posted price, average reserve price, average willingness-to-pay, and probability of each buyer winning. The same convergence criterion is used in the posted-price model as the auction model. The posted-price model and auction model are run separately with two buyers and one seller in each market.

We run the posted-price market and auction under several different scenarios. First, we give the seller perfect information (or no noisy signal), but buyers do have a noisy signal. Then, we run the models with the seller receiving a noisy signal. This allows determining how a seller's uncertainty affects the seller's decision to sell an item with an auction or with a posted price. We also run the auction and posted-price models with asymmetric buyers to determine if unbalanced information across buyers influences the seller's selling decision. Lastly, the results are included for the common-value auction without a reserve price to compare to a posted-price market.

Numerical Integration

We extend Zhang and Brorsen (2009) by combining numerical integration with the PSO learning algorithm to solve agent-based models when the agents have a noisy signal. Buying and selling agents each choose their strategy in the separate markets and the trapezoidal rule is used to approximate the buying and selling agents' expected revenues. Other integration algorithms such as Gaussian quadrature (Miller and Rice 1983) could be used to approximate these integrals, but the trapezoidal rule is chosen because it is efficient enough for the problem being considered and is straightforward to program.

The trapezoidal rule uses the area of trapezoids for *n* subintervals or lengths to approximate the integral. The trapezoidal rule is mathematically defined as

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}(y_0 + 2y_1 +,...,+2y_{n-1} + y_n) \tag{3.7}$$

where Δx = (b-a)/n, $f(x)$ is the probability density function of winning the item, $y_0,..., y_n$ are the grid points in the integral, b is the upper bound of the integral and a is the lower bound of the integral. The probability weights at each point of the integral are Δx except at the tails where the probability weight is $\Delta x/2$. With the uniform distribution, n is assigned to be 100, resulting in a $\Delta x = 1/100$, a distance of 1/100 between the points in the integral, and a probability weight of 1/100 accepts at the tails where the probability weight is 1/200. A large number of points are used here to reduce discreteness that can reduce accuracy in a winner-take-all auction.

Particle Swarm Optimization

The buyers and the seller in the agent-based models use a PSO learning algorithm developed by Eberhart and Kennedy (1995) and modified by Zhang and Brorsen (2009) to choose their strategies. PSO is a stochastic global optimization method that tends to converge faster than genetic algorithms (Zhang and Brorsen 2009). In Zhang and Brorsen's (2009) PSO algorithm, there are $k = 1,..., K$ parallel markets, and each buying and selling agent has k clones. The idea of parallel markets with clones came from watching the way flocks of birds share information to find food and avoid predators. The PSO algorithm gives each buyer and seller their own "flock of birds" or clones to share information, but the clones do not share information with the clones of the other buyer or those of the seller. The agents use their clones' strategies in the parallel markets to determine the strategy that produces the largest revenue. The same number of parallel markets is used for the posted-price market and the auction.

In the PSO, the *i*th agent has a choice variable $x_{i,k,t}$ in parallel market *k* during iteration $t = 1,...,T$, and an adjustment velocity $v_{i,k,t} \in [-1,+1]$ that directs the agent to a new value for the choice variable. Each velocity change is a function of the local best solution $p^{l}_{i,k,t}$ where superscript *l* indicates the best solution in the local market and the agent's global best solution $p^{g}_{i,t}$ where superscript *g* indicates the best global solution. The *i*th agent's new choice variable in parallel market *k* is updated by $x_{i,k,t+1} = x_{i,k,t} + v_{i,k,t}$ and the velocity is

$$v_{i,k,t+1} = wv_{i,k,t} + q_1 u_1 (p^{l}_{i,k,t} - x_{i,k,t}) + q_2 u_2 (p^{g}_{i,t} - x_{i,k,t}) \tag{3.8}$$

where *w* is the inertia weight factor, u_1 and u_2 are random variables distributed *U*[0,1], and q_1 and q_2 are learning parameters. The learning parameters are $q_{1,t} = q_{2,t} = \beta_0^{q_1} + \beta_1^{q_1}(T-t)/T$ where both $\beta_0^{q_1}$ and $\beta_1^{q_1}$ are constants set at one and $T = 500$ is the maximum number of iterations. When selecting an inertia weight *w* there is an important tradeoff between exploration and convergence time to consider (Chatterjee and Siarry 2006). A large inertia weight slows convergence, but allows agents to have a larger exploration area, while a small inertia weight increases convergence time, but reduces the agents' exploration area (Zhang and Brorsen 2009). Zhang and Brorsen (2009) manage this tradeoff by letting *w* start high and decrease as the model proceeds through the iterations, similar to the learning parameters. Inertia weight expressed in Zhang and Brorsen (2009) as $w_t = \beta_0^{w} + \beta_1^{w}(T-t)/T$ where both β_0^{w} and β_1^{w} are constants set at 0.75.

At each iteration, the strategy with the largest revenue for the agent in the *k*th market is chosen as the best local strategy. The agents' best strategy depends on the other agents' choice variables; therefore, the agent's previous best strategy may not perform well in the next iteration. The other agents' strategies in the current period are held constant and the past *L* best locals of each agent are reevaluated in market *k*. The best local solution is expressed as

$$p_{i,k,t}^{l} = \arg\max\left\{R_k(p_{i,k,t-1}^{l}),\ldots,R_k(p_{i,k,t-L}^{l}), R_k(x_{i,k,t}) \middle| x_{j\neq i,k,t}\right\} \qquad (3.9)$$

where R_k is the revenue in parallel market *k*. The best global solution is selected from the best local parameters and is

$$p_{i,t}^{g} = \arg\max\left\{R_1(p_{i,1,t}^{l}), R_2(p_{i,2,t}^{l}),\ldots, R_K(p_{i,K,t}^{l})\right\}. \qquad (3.10)$$

Table III-1 shows the values for the PSO parameters used for the agent-based markets.

<u>Convergence Criterion</u>

A convergence criterion is established to determine when the buyers and the seller cannot find bid and reserve price strategies that improve upon their previous strategies. The model converges when the total change in the previous 10 strategies for each of the agents is less than 0.0000001 and the total change in the standard deviations of the strategies are less than 0.00000001. The model has 200 runs (*E*=200), and in each run there is a maximum of 500 iterations (*T*=500). That is, the agents have 500 iterations (or trades) to meet the convergence criterion, and this is repeated 200 times. The results presented are the agents' averages and standard deviations for all 200 runs. If the convergence criterion is met by all buying and selling agents, then the next run begins, but if the convergence criterion is not met by all buying and selling agents, then the next iteration begins and the PSO learning algorithm guides the agents to improved strategies in the next iteration.

Results

Common Value Auction

Results for the agent-based first-price common-value auction without a reserve price are displayed in the first column of Table III-2. The average winning bid price for the item is -0.17, which is interpreted as the average winning bid price is 0.17 below the constant positive true value (table III-2). Buyer one has a slightly higher average bid price, probability of winning the item, and expected revenue than buyer two. Buyers are programmed to be identical so randomness in the PSO algorithm explains the minor differences. The seller's expected revenue per item is -0.17, and since the seller has no reserve price, every item is sold (table III-2). The winner's curse does not occur in the agent-based common-value auction without a reserve price, and buyers' shade bids below their retail value.

The second column of Table III-2 shows the results for the agent-based common-value auction with a reserve price, which is an extension of Noe et al.'s (2011) model. When the seller has perfect information, the average winning bid price is 0.21, and the average reserve price is -0.01. The buyers' expected revenues per item for all winning and losing bids are 0.09 (table III-2). The item is sold 60% of the time it is offered, and the seller's expected revenue per item is 0.13 (table III-2). Allowing the seller to choose a reserve price in the agent-based common-value auction increases the average winning bid price, but buyers' still shade bids below the retail value of the item.

In the third column of Table III-2, results are presented for the agent-based common-value auction when the seller sets a reserve price based on a noisy signal. The average winning bid price in the common-value auction is 0.04 and the average reserve price is -0.31(table III-2). Compared to the auction results when the seller has perfect information, the average winning bid price and reserve price decreases, the buyers' expected revenues per item increase, and the seller's expected revenue per item decreases (table III-2). Buyers are able to increase their bid-shading in the agent-based common-value auction when the seller receives a noisy reserve price signal, explaining why buyers' expected revenues per item increases and the seller's expected revenue per item decreases.

Next, we reduce buyers' noisy signal to be distributed $U[0,1/2]$, and the seller's noisy signal remains distributed $U[0,1]$. That is, the buyers have more information about the value of the item than the seller. Results are displayed in the fourth column of Table III-2. The average winning bid price is 0.06, and the average reserve price is -0.44 (table III-2). The average winning bid price increases when the buyers receive more information, which matches Milgrom and Weber's (1982) theoretical results. Reducing both buyers' noisy signals in the agent-based common-value auction causes buyers' and the seller's expected revenue to increase (table III-2).

Finally, we present results for an agent-based common-value auction when the buyers are asymmetric (table III-2). An advantage of agent-based models is the assumption of symmetric buyers found in previous work can be relaxed, which provides an additional theoretical contribution to the auction and posted-price market literature. Buyer one has a noisy signal that is distributed $U[0,1/2]$ while buyer two and the seller receives a noisy signal distributed $U[0,1]$, representing buyer one having more information about the value of the item than the seller and buyer two. The average winning bid price is 0.03 and the average reserve price is -0.41 (table III-2). Since buyer one has less uncertainty than buyer two, buyer one has a higher average bid price, probability of winning, and expected revenue per item than buyer two. The seller's expected revenue per item did not change from when symmetric buyers are participating in the auction (table III-2). Overall, asymmetric buyers in a first-price common-value auction with a reserve price do not affect the seller's expected revenue, and appears to only benefit the buyer with less uncertainty.

Posted Price Market

Table III-3 displays the results for the agent-based posted-price market when the buyers have a common value for the item. When the seller has perfect information, the average selling posted price is 0.25 (table III-3). Buyer one and buyer two have expected revenues per item of 0.06, and the seller's expected revenue per item is 0.13 (table III-3). Compared to the common-value auction results when the seller does not impose a reserve price, the seller's expected revenue is higher in the posted-price market. However, when the seller imposes a reserve price in the auction and has perfect information, the seller is indifferent between these selling methods. Note that in all scenarios, a transaction is more likely to occur in the auction market than in the posted price market.

The seller is also given a noisy posted price signal and results are presented in the second column of Table III-3. The average selling posted price for the item is 0.10, each buyer has an expected revenue per item of 0.08, and the seller's expected revenue per item is 0.03 (table III-3). With the addition of price uncertainty by the seller in the posted-price market, average selling price decreases, expected revenues per item increase for the buyers, and the seller's expected revenue per item decreases. The agent-based models results reveal that when the buyers have a common value for the item being sold the seller's expected revenue per item in a posted-price market is equivalent to the seller's expected revenue in an auction with a reserve price.

In the third column of Table III-3, results are shown when the buyers noisy signal is reduced to be distributed *U*[0,1/2] and the seller's noisy signal is distributed *U*[0,1]. This represents a posted-price market when buyers have more information about the value of the item than the seller. Buyers increase their willingness-to-pay for the item, but the seller chooses to decrease their average selling posted price to 0.06 (table III-3). Both buyers have expected revenue per item of 0.12, and the seller's expected revenue per item is 0.02, which is a slight decrease from when they have equal uncertainty (table III-3). Compared to the auction when buyers have less uncertainty than the seller, the seller receives slightly smaller expected revenue in a posted-price market. This result indicates that when buyers' uncertainty is reduced by half the seller is slightly better off auctioning the item when buyers have a common value for the item. Seller uncertainty hurts the seller more in the posted price market than it does in the auction. This result is consistent with Milgorm and Weber's (1982) conclusion that a posted price does not work well when the seller does not know what posted price to choose.

Finally, results are presented in the fourth column of Table III-3 when asymmetric buyers participate in the posted-price market. This scenario matches the asymmetric buyers in the agent-based auction with buyer one having less uncertainty (or more information) than buyer two and the seller. The average selling posted price is 0.09, which is a slight decrease from when symmetric buyers are participating in the posted-price market (table III-3). The seller's expected revenue per item does not change from when symmetric buyers are participating in the posted-price market, and is equivalent to the seller's expected revenue per item in the auction with symmetric and asymmetric buyers. We can conclude that asymmetric buyers do not affect the seller's choice of selling an item in an auction or posted-price market when the value of the item is common between buyers. Results appear to suggest when only one buyer's noisy signal is reduced by half then the seller is indifferent between the selling methods.

Conclusions

We compare the seller's expected revenue from selling an item in an auction with a reserve price and a posted-price market when the item being sold has a common value between the buyers. An agent-based first-price common-value auction with a reserve price and an agent-based posted-price market are developed to determine the method that is optimal for the seller. Our approach to comparing these selling methods is straightforward and similar to Wang's (1993) private value models and Campbell and Levin (2006) affiliated value models. The results from the agent-based model provide a unique theoretical contribution to the literature.

Solving for the equilibrium of a common-value auction with a noisy reserve price is perhaps intractable and obtaining data through human and field experiments is expensive and cannot control for participates motivation. An agent-based model approach is an appropriate and unique way to address the objective. An agent-based common-value auction was recently presented by Noe et al. (2011), but we extend this model by allowing the seller to impose a reserve price, and give the buyers and the seller a noisy price signal. The agent-based auction model extends the agent-based modeling literature by giving the seller a reserve price.

The results from the agent-based models show that the seller receives larger expected revenue from selling the item in a posted-price market than an auction without a reserve price. When the seller imposes a reserve price, the seller is indifferent between the first-price auction and the posted-price market when (1) the seller has perfect information and the buyers have a noisy price signal, and (2) when the buyers and the seller have equal noisy signals. These results from the agent-based model match what Kultti (1999) found with his theoretical model. We also show when the buyers are asymmetric in the models, the seller's expected revenue is equivalent in the posted-price market and auction with a reserve price. However, when both buyers' noisy signals are reduced by half, the seller is slightly better off auctioning the item with a reserve price. In the agent-based model, buyers' bid prices increase when their uncertainty decreases, which matches Milgrom and Weber's (1982) theory, resulting in the seller's expected revenue to being greater than in the posted-price market.

References

Alkemade, F., J.A. La Poutré, and H.M. Amman. 2006. “Robust Evolutionary Algorithm Design for Socioeconomic Simulation.” *Computational Economics* 28(4):355–370.

Andreoni, J., and J.H. Miller. 1995. “Auctions with Artificial Adaptive Agents.” *Games and Economic Behavior* 10:39-64.

Arifovic, J. 1996. “The Behavior of the Exchange Rate in the Genetic Algorithm and Experimental Economies.” *Journal of Political Economy* 104(3):510-541.

Arnold, M.A., and S.A. Lippman. 2001. “The Analytics of Search with Posted Prices.” *Economic Theory* 17:447-466.

Bonabeau, E. 2002. “Agent-Based Modeling: Methods and Techniques for Simulating Human Systems.” *Proceedings of the National Academy of Sciences of the United States of America* 99:7280-7287.

Campbell, C.M., and D. Levin. 2006. “When and Why Not to Auction.” *Economic Theory* 27:583-596.

Capen, E.C., R.V. Clapp, and W.M. Campbell. 1971. “Competitive Bidding in High-Risk Situations.” *Journal of Petroleum Technology* 23:641-653.

Chatterjee, A., and P. Siarry. 2006. “Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization.” *Computers and Operations Research* 33:859–871.

Eberhart, R.C., and J. Kennedy. 1995. “A New Optimizer Using Particle Swarm Theory.” *Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan,* 39-43.

Gordy, M.B. 1998. “Computationally Convenient Distributional Assumptions for Common-Value Auctions.” *Computational Economics* 12:61-78.

Hammond, R.G. 2010. “Comparing Revenue From Auctions and Posted Prices.” *International Journal of Industrial Organization* 28:1-9.

Julien, B., J. Kennes, and I. King. 2002. “Auctions Beat Posted Prices in Small Market.” *Journal of Institutional and Theoretical Economics* 158(4):548-562.

Kagel, J.H., and D. Levin. 1986. “The Winner's Curse and Public Information in Common Value Auctions.” *American Economic Review* 76(5):894-920.

Kagel, J.H., and D. Levin. 2002. *Common Value Auction and the Winner’s Curse*. Princeton, New Jersey: Princeton University Press.

Klemperer, P. 1999. “Auction Theory: A Guide to the Literature.” *Journal of Economic Surveys* 13:227–286.

Kultti, K. 1999. “Equivalence of Auctions and Posted Prices.” *Games and Economic Behavior* 27:106-113.

Krishna, V. 2002. *Auction Theory*. San Diego, CA: Academic Press.

Levin, D., and J.L. Smith. 1996. “Optimal Reservation Prices in Auctions.” *The Economics Journal* 106:1271-1283.

Lu, X., and R.P. McAfee. 1996. “The Evolutionary Stability of Auctions over Bargaining.” *Games and Economic Behavior* 15:228-254.

McAfee, R.P., and J. McMillan. 1987. “Auctions and Bidding.” *Journal of Economic Literature* 25(2):699-738.

McCall, J.J. 1965. “The Economics of Information and Optimal Stopping Rules.” *Journal of Business* 300-317.

Milgrom, P.R. 1989. “Auctions and Bidding: A Primer.” *Journal of Economic Perspectives* 3:3-22.

Milgrom, P.R., and R.J. Weber. 1982. “A Theory of Auctions and Competitive Bidding.” *Econometrica* 50:1089-1122.

Miller, A.C., and T.R. Rice. 1983. “Discrete and Approximations of Probability Distributions.” *Management Science* 29(3):352-362.

Noe, T.H., M. Rebello, and J. Wang. 2011. “Learning to Bid: the Design of Auctions under Uncertainty and Adaption.” *Games and Economic Behavior* doi:10.16/j.geb.2011.08.005.

Paarsch, H. 1992. “Deciding Between the Common and Private Value Paradigms in Empirical Models of Auctions.” *Journal of Econometrics* 51:191-215.

Pinkse, J., and G. Tan. 2005. “The Affiliation Effect in First-Price Auctions.” *Econometrica* 1:263-277.

Stigler, G.J. 1961. “The Economics of Information.” *Journal of Political Economy* 69:213-225.

Tesfatsion, L. 2001. “Introduction to the Special Issue on Agent-Based Computational Economics.” *Journal of Economic Dynamics and Control* 25:281–293.

Wang, R. 1993. “Auctions versus Posted-Price Selling.” *American Economic Review* 83(4):838-851.

Wang, R. 1998. Auctions versus Posted-Price Selling: The Case of Correlated Private Valuations. *Canadian Journal of Economics* 31:395-410.

Zhang, T., and B.W. Brorsen. 2009. “Particle Swarm Optimization Algorithm for Agent-Based Artificial Markets.” *Computational Economics* 34:399-417.

Table III-1. Parameter Values for the Particle Swarm Optimization Algorithm

Parameter	Symbol	Value
Intercept of inertia weight	β_0^w	0.75
Slope of inertia weight	β_1^w	0.75
Learning parameters intercept	β_0^q	0.75
Learning parameter slope	β_1^q	0.75
Parallel markets	k	10
Maximum iterations	T	500
Evolutions	E	200
Memory	L	3

Table III-2. Equilibrium Values for the Agent-Based Common-Value Auctions

	Scenario				
Category[a]	No Reserve price	Seller Has Perfect Information[b]	Noisy Reserve Price[c]	Reduced Buyers' Noise[d]	Asymmetric Buyers and Noisy Reserve Price[e]
Average winning bid price	-0.17 (0.12)	0.21 (0.08)	0.04 (0.11)	0.06 (0.11)	0.03 (0.10)
Reserve price	-	-0.01 (0.11)	-0.31 (0.26)	-0.44 (0.22)	-0.41 (0.23)
Buyer 1					
Average bid price	-0.33 (0.13)	-0.15 (0.16)	-0.21 (0.14)	-0.04 (0.14)	-0.09 (0.11)
Transaction probability	53.0% (12.0%)	29.0% (13.0%)	37.0% (15.0%)	47.0% (20.0%)	47.0% (16.0%)
Expected revenue	0.35 (0.11)	0.09 (0.04)	0.17 (0.08)	0.20 (0.10)	0.25 (0.09)
Buyer 2					
Average bid price	-0.36 (0.15)	-0.13 (0.15)	-0.23 (0.15)	-0.05 (0.13)	-0.20 (0.15)
Transaction probability	47.0% (12.0%)	31.0% (14.0%)	34.0% (15.0%)	43.0% (20.0%)	35.0% (15.0%)
Expected revenue	0.32 (0.09)	0.09 (0.04)	0.16 (0.07)	0.19 (0.10)	0.14 (0.06)
Seller					
Transaction probability	100.0% (0.0%)	60.0% (20.0%)	72.0% (22.0%)	90.0% (15.0%)	83.0% (17.0%)
Expected revenue	-0.17 (0.12)	0.13 (0.06)	0.03 (0.08)	0.06 (0.09)	0.03 (0.08)

Note: Standard deviations are presented in parentheses.
[a] All results, except average winning bid price, are not conditional on winning.
[b] The seller's reserve price is not a function of a noisy signal but buyers noisy signal is distributed *U*[0,1].
[c] Seller and buyers noisy signals are independently distributed *U* [0,1].
[d] Errors for the buyers are distributed *U* [0,0.5] and the errors for the seller is distributed *U* [0,1].
[e] Noisy signal for buyer one is distributed *U* [0,0.5] and the noisy signal for buyer two's and the seller are distributed *U* [0,1].

Table III-3. Equilibrium Values for the Agent-Based Posted-Price Market with a Common Value

	Scenario			
Category[a]	Seller Has Perfect Information[b]	Noisy Posted Price[c]	Reduced Buyers' Noise[d]	Asymmetric Buyers and Noisy Posted Price[e]
Average selling posted price	0.25 (0.14)	0.10 (0.20)	0.06 (0.19)	0.09 (0.18)
Buyer 1				
Average willingness-to-pay	0.03 (0.31)	0.01 (0.30)	0.18 (0.23)	0.20 (0.23)
Transaction probability	26.0% (19.0%)	21.0% (17.0%)	27.0% (17.0%)	30.0% (20.0%)
Expected revenue	0.06 (0.06)	0.08 (0.09)	0.12 (0.10)	0.12 (0.06)
Buyer 2				
Average willingness-to-pay	0.03 (0.27)	0.02 (0.29)	0.18 (0.27)	-0.03 (0.28)
Transaction probability	25.0% (18.0%)	22.0% (17.0%)	27.0% (17.0%)	17.0% (15.0%)
Expected revenue	0.06 (0.06)	0.08 (0.08)	0.12 (0.10)	0.07 (0.07)
Seller				
Transaction probability	51.0% (22.0%)	42.0% (24.0%)	54.0% (21.0%)	47.0% (24.0%)
Expected revenue	0.13 (0.09)	0.03 (0.12)	0.02 (0.14)	0.03 (0.13)

Note: Standard deviations are presented in parentheses.
[a] All results, except average selling posted price, are not conditional on winning.
[b] The seller's posted price is not a function of a noisy signal but buyers noisy signal is distributed *U* [0,1].
[c] Seller and buyers noisy signals are independently distributed *U* [0,1].
[d] Errors for the buyers are distributed *U* [0,0.5] and the errors for the seller is distributed *U* [0,1].
[e] Noisy signal for buyer one is distributed *U* [0,0.5] and the noisy signal for buyer two's and the seller are distributed *U* [0,1].

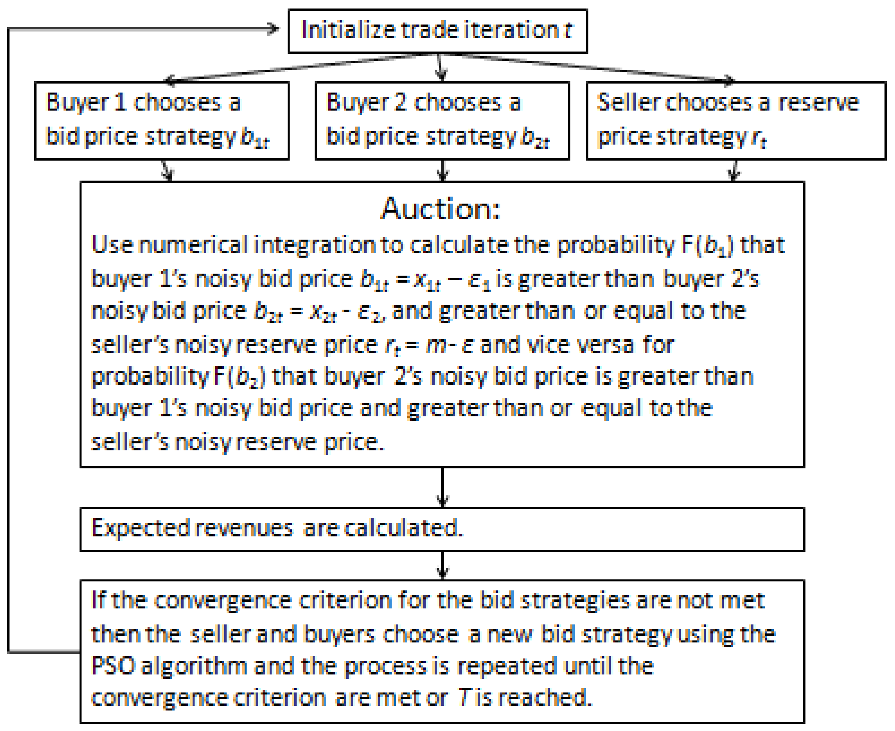
Initialize trade iteration t
Buyer 1 chooses a bid price strategy b_{1t}
Buyer 2 chooses a bid price strategy b_{2t}
Seller chooses a reserve price strategy r_t
Auction:
Use numerical integration to calculate the probability F(b_1) that buyer 1's noisy bid price $b_{1t} = x_{1t} - \varepsilon_1$ is greater than buyer 2's noisy bid price $b_{2t} = x_{2t} - \varepsilon_2$, and greater than or equal to the seller's noisy reserve price $r_t = m - \varepsilon$ and vice versa for probability F(b_2) that buyer 2's noisy bid price is greater than buyer 1's noisy bid price and greater than or equal to the seller's noisy reserve price.
Expected revenues are calculated.
If the convergence criterion for the bid strategies are not met then the seller and buyers choose a new bid strategy using the PSO algorithm and the process is repeated until the convergence criterion are met or T is reached.

Figure III-1. Overview of the Agent-Based Auction Model

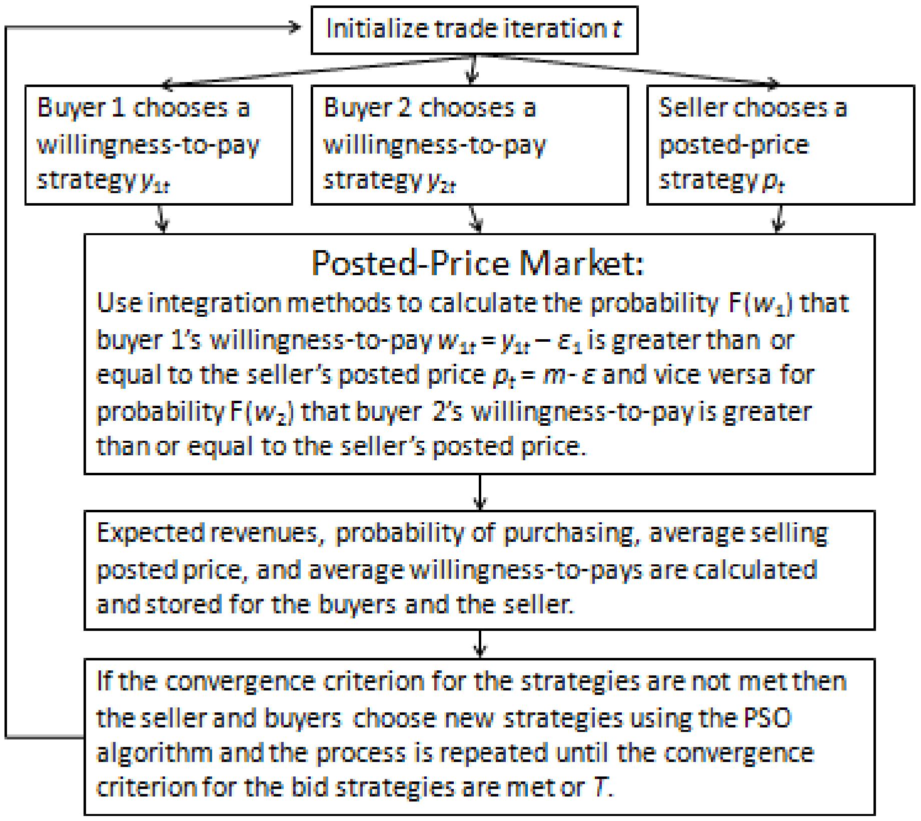
Initialize trade iteration t
Buyer 1 chooses a willingness-to-pay strategy y_{1t}
Buyer 2 chooses a willingness-to-pay strategy y_{2t}
Seller chooses a posted-price strategy p_t
Posted-Price Market:
Use integration methods to calculate the probability $F(w_1)$ that buyer 1's willingness-to-pay $w_{1t} = y_{1t} - \varepsilon_1$ is greater than or equal to the seller's posted price $p_t = m - \varepsilon$ and vice versa for probability $F(w_2)$ that buyer 2's willingness-to-pay is greater than or equal to the seller's posted price.
Expected revenues, probability of purchasing, average selling posted price, and average willingness-to-pays are calculated and stored for the buyers and the seller.
If the convergence criterion for the strategies are not met then the seller and buyers choose new strategies using the PSO algorithm and the process is repeated until the convergence criterion for the bid strategies are met or T.

Figure III-2. Overview of the Agent-Based Posted-Price Model

APPENDIXES

APPENDIX A--Program Loops for Calculating Probabilities

Layers of three loops are programmed to approximate the expected revenues in the agent-based model: (1) an outside loop, (2) a middle loop, and (3) an inside loop. The seller's noisy reserve price is $r = m - \omega$ where ω the noisy signal is in the outside loop and *m* is the seller's choice variable or strategy. The outside loop starts with $\omega = 0$ and ω increases by $\Delta x = 1/100$ until $\omega = 1$. The probability weight for the outside loop is w_{ω}. Within the outside loop, the middle loop begins, where buyers are given a noisy signal. For buyer one, its noisy bid price is $b_1 = x_1 - \gamma$ where γ is the noisy signal for the buyers and x_1 is buyer one's choice variable or strategy. The middle loop starts with $\gamma = 0$ and γ increases by Δx until $\gamma = 1$. The probability weight for this loop is w_{γ}. Within the middle loop, the inside loop starts and calculates the probability of winning for the buyers. Following the example for buyer one, in the inside loop buyer two's noisy bid price is $b_2 = x_2 - \upsilon$ where υ is the noisy signal in the inside loop and x_2 is buyer two's choice variable or strategy. The inside loop starts with $\upsilon = 0$ and υ increases by Δx until $\upsilon = 1$, and a probability weight of w_{υ}. Buyer one's probability of winning the item is

$$F(b_1) = \left[\sum_{\upsilon=0}^{100} w_{\upsilon} \times w_{\gamma} \mid x_1 - (\gamma/100) > x_2 - (\upsilon/100), x_1 - \gamma \geq r - (\omega/100) \right] \quad \text{(A.1)}$$

where $F(b_1)$ is the probability of buyer one winning the item being offered. The same process is followed to solve for buyer two's probability of winning and is expressed as

$$F(b_2) = \left[\sum_{\upsilon=0}^{100} w_{\upsilon} \times w_{\gamma} \mid x_2 - (\gamma/100) > x_1 - (\upsilon/100), x_2 - (\gamma/100) \geq r - \omega \right] \quad \text{(A.2)}$$

where $F(b_2)$ is the probability of buyer one winning the item being offered. When this loop is complete, expected revenue for the buyers and the seller are approximated in the middle loop. Buyer one's expected revenue is

$$E[R_1^B] = \sum_{\gamma=0}^{100}(0.5-(x_1-\gamma)\times F(b_1))\times w_\omega, \tag{A.3}$$

buyer two's expected revenue is

$$E[R_2^B] = \sum_{\gamma=0}^{100}(0.5-(x_2-\gamma)\times F(b_2))\times w_\omega, \tag{A.4}$$

and the seller's expected revenue is

$$E[R^S] = \sum_{\varepsilon_m=0}^{1}[((x_1-\varepsilon_m)\times F(b_1))+((x_2-\varepsilon_m)\times F(b_2))]\times w_o. \tag{A.5}$$

The average winning bid price for the item is $WB = [R^S/(F(b_1)+F(b_2))]$.

APPENDIX B--Mandatory Price Reporting Program

```
package PSOmpr;
/* Program Name: Mandatory Price Reporting Act
* Author: Chris Boyer
* Date:   9/10/11
*/

/* ******************Program Files
description:*************************************************
 * 1. Buyer: models the buyers' behavior; all buyers have the same behavioral rule but have different
initialized starting values;
 * 2. PSO: models the PSO algorithm, have two functions:
 *       a). retest and selected best locals for each parallel market;
 *       b). select the best global for all markets;  *
 * 3. ModelPSO:  Main program;
 * 4. PSOEquilibrium: program to determine the equilibrium;
 * 5. RetailMarket: retail price is perfect elastic, and is $100 all the times;
 * 4. Criterion: stores the parameter set of fixed and changing categories;
 * 5. Seller: Seller behavior
 * 6. Spot Market: Buyers quantity competitive in the spot market;
  * */

import java.io.FileNotFoundException;import java.io.FileOutputStream;import java.io.PrintStream;import
java.text.DecimalFormat;
import java.util.*;import tools.P;

/* ******************Model Class**************************************/
public class ModelPSO {
        static int parameterNo=1;
        static boolean FixedRetailPrice=true;
        static double Retailstd=0;
        static int TotalbuyerNumber=1;
        static int TotalsellerNumber=1;
        static double Constraint=1;
        static int EquilibriumNo=100;// run 100 times for each setup and calculate the mean and std for
the results of all runs;
        static int Evolution=500;//400 iterations of each run. with iterations, agents repeat the game and
learn to use the optimal strategies under each setup;
        static int ParallelMarket=10;
        static int PSOTestNumber=3;
        static int WeekPerCycle=1;
        double variationloop=100;
        static Random r=new Random(System.currentTimeMillis()*5986587); // identify random variable

        public static void main(String args[]) throws FileNotFoundException{
        double testprint=1;
        FileOutputStream out,eout,aout,Pout, a2out, P2out, Sout;
        PrintStream printOut,eprintOut,aprintOut,PprintOut, a2printOut,P2printOut, SprintOut;

        // titles the output files
        //Acutal Output ;
```

```
            String
name="PSO"+TotalsellerNumber+"Seller"+BuyerCPCostParameter1+"CapacityCost.txt";out=new
FileOutputStream(name);printOut=new PrintStream(out);
        //Parameters
             String Sname="PSO Parameter Sellers.txt"; Sout=new FileOutputStream(Sname);
SprintOut=new PrintStream(Sout);
             //4 buyer Actual Output
             String aname="PSO for Buyer1.txt";aout=new FileOutputStream(aname); aprintOut=new
PrintStream(aout);
            //4 buyer Auction Parameters
            String Pname="PSO parameters Buyer1.txt"; Pout=new FileOutputStream(Pname);
PprintOut=new PrintStream(Pout);
            String a2name="PSO for Buyer2.txt";a2out=new FileOutputStream(a2name); a2printOut=new
PrintStream(a2out);
            //4 buyer Auction Parameters
            String P2name="PSO parameters Buyer2.txt"; P2out=new FileOutputStream(P2name);
P2printOut=new PrintStream(P2out);

/*******Initialization*********/
                        int    i,j,k,pi,fj;double tmpDouble;
                        // identify other classes except buyers and seller
                        ModelPSO          m=new ModelPSO();
                        RetailMarket      bbm=new RetailMarket(m);
                        SpotMarket           dm=new SpotMarket(m);
                        PSOEquilibrium em=new PSOEquilibrium(m);
                        PSO PSO=new PSO(m);
                        Criterion cn=new Criterion(m);
                        // end of class identification
                        //set up template for one of the print outs
                        aprintOut.print("ChoiceV\tMarketP\tProbSold");
                        a2printOut.print("ChoiceV\tMarketP\tProbSold");
                        printOut.print("ResP\tMarketP\tProbSold");
                        SprintOut.print("w\tc1\tc2\tReserveP\t");
                        PprintOut.print("w\tc1\tc2\tMarketP\t");
                        P2printOut.print("w\tc1\tc2\tMarketP\t");
                        for(i=0;i<m.TotalbuyerNumber;i++){aprintOut.print("\t"+"
Bid"+i+"\t\t"+"Profit"+i+"\t");        PprintOut.print(i+" BR\t");}
                        for(i=0;i<m.TotalbuyerNumber;i++){a2printOut.print("\t"+"
Bid"+i+"\t\t"+"Profit"+i+"\t");P2printOut.print(i+" BR\t");}
                        for(i=0;i<m.TotalsellerNumber;i++){printOut.print("\tChoiceV"+i+"\t"+"Profit"+i+ "\t");
            SprintOut.print(i+" RR\t");}

            PprintOut.print("time\tEquilibrium");P2printOut.print("time\tEquilibrium");SprintOut.print("time\
tEquilibrium");

                        //set up java output that is printed in the console
                        for(i=0;i<TotalbuyerNumber;i++){
                                    if(i==0)
                                                P.t("\t=========buyer "+i+"============"); else
P.t("\t===========buyer "+i+"===============");
                                    }
                        P.ln();
                        for(i=0;i<TotalbuyerNumber;i++){
                                    if(i==0){P.t("\t  CR");P.t("Cbg\tClg "+"\tCgp-pr"+"\tClp-pr"+"\t Q"+"\t
Cost"+"\tprofit");}
                                    else {P.t("\t\t  CR");
```

```
                P.t(" Cbg\tClg "+"\tCp-pr"+"\t Cl-pr"+"\t Q"+"\t Cost"+"\tprofit");}
            }// end of print commands
            //CriterionNo is 4, this is looping the PSO criteria through four times.
            // this makes the criteria the same for all buyers and sellers
            for(int CN=0;CN<cn.CriterionNo;CN++){
                // this is identifying which parameter estimates to use for the model
                // gets info from PSO class.
                PSO.PSOParameterSetting(m,cn,
CN,cn.MarketStructureFixed,cn.AlgorithmParameterFixed);
            // loop that runs model 100 evolutions
        for(int ec=0;ec<EquilibriumNo;ec++){
            if(ec==EquilibriumNo-1)P.t("\n Setting No. "+(CN+1)+" : Finish "+EquilibriumNo+"
runs");
            //class variables
            bbm=new RetailMarket(m);
            dm=new SpotMarket(m);
            em=new PSOEquilibrium(m);
            //establishes an array with number of buyers and sellers
            AuctionBuyer1[] ab1 = new AuctionBuyer1[TotalbuyerNumber];AuctionBuyer2[] ab2 =
new AuctionBuyer2[TotalbuyerNumber];
            Seller[] s=new Seller[TotalsellerNumber];
            // constructs a list with number of parallelMarkets
            LinkedList[] sellerList=new LinkedList[m.ParallelMarket];
            LinkedList[] buyer1List=new LinkedList[m.ParallelMarket];
            LinkedList[] seller2List=new LinkedList[m.ParallelMarket];
            // constructs a list of parallel markets
            Iterator[] sellerit=new Iterator[m.ParallelMarket];
            Iterator[] buyerit=new Iterator[m.ParallelMarket];
            Iterator[] seller2=new Iterator[m.ParallelMarket];
            Seller seller;AuctionBuyer1 buyer1;AuctionBuyer2 buyer2;
            // loops that fill the list of buyers and sellers
            for(i=0;i<TotalbuyerNumber;i++)ab1[i]=new AuctionBuyer1(m,i);
            for(i=0;i<TotalbuyerNumber;i++)ab2[i]=new AuctionBuyer2(m,i);
            for(i=0;i<TotalsellerNumber;i++)s[i]=new Seller(m, dm, i);

        /******************Begin Evolution      ****************************/
        /****************Begin Main Program****************************/
        /**************Initialization      *********************************/
            //start measuring solving time
               long time;
             em.EquilibriumInitial(); time=System.currentTimeMillis();
             r=new Random(System.currentTimeMillis()*528789+ec*534757);
            // starts the iterations or evolutions of trades
            for(int e=-m.PSOTestNumber;e<m.Evolution;e++){
            //updates parms for each evolution
             PSO.UpdatePSOParameter(m, e, cn.AlgorithmParameterFixed);

      /******For every parallel market, buyers and their clones trade simultaneously ************/
             // loop for each parallel markets are looped - during each evolution
            for(int pm=0;pm<ParallelMarket;pm++){
            //creates a blank list of available buyers
              LinkedList AvailableBuyerList=new LinkedList();  AvailableBuyerList.clear();

                    //sets all the buyers to the initial values - calls from buyer
            for(i=0;i<TotalbuyerNumber;i++){
```

```
                        ab1[i].EvolutionInitialization(m, bbm, em,PSO, e,
pm);ab2[i].EvolutionInitialization(m, bbm, em,PSO, e, pm);//initialize}

                for(i=0;i<TotalsellerNumber;i++){s[i].EvolutionInitialization(m, bbm, em,PSO, e, pm);}
                //loop for week which is 1.
                //Change weekpercycle to simulate multiple trades
                for(int week=0;week<WeekPerCycle;week++){
                for(i=0;i<TotalbuyerNumber;i++){
                ab1[i].WeekInitialization(m,bbm,em,dm, e,pm);ab2[i].WeekInitialization(m,bbm,em,dm,
e,pm);
                        //sets week values - called from buyer class              }
                for(i=0;i<TotalsellerNumber;i++){  s[i].WeekInitialization(m,bbm,em,dm, e,pm);
                                //sets week values - called from buyer class   }

                        // Auction Deal in Spot market
                dm.AuctionDeal(ab1, ab2, s, m, week, pm);
                dm.PreviousAverageBidPrice[pm]=dm.AverageBidPrice[pm]; //store market price;
                dm.PreviousWinBidPrice[pm]=dm.WinBidPrice[pm];
                dm.PreviousAverageReservePrice[pm]=dm.AvgRandomRevserve[pm];
                for(i=0;i<ab1.length;i++)ab1[i].WeeklyProfit(m, dm, bbm, week, pm); // buyers' profit;
                for(i=0;i<ab2.length;i++)   ab2[i].WeeklyProfit(m, dm, bbm, week, pm);
                for(i=0;i<s.length;i++)       s[i].WeeklyProfit(m, dm, bbm, week, pm);}//end weeks;

for(i=0;i<ab1.length;i++){ ab1[i].StorageParameter(m,pm);ab1[i].StoragePrevious(m, pm);//auction}
for(i=0;i<ab2.length;i++){ ab2[i].StorageParameter(m,pm);ab2[i].StoragePrevious(m, pm);//auction}
for(i=0;i<s.length;i++){s[i].StorageParameter(m,pm);s[i].StoragePrevious(m, pm);//auction}
        }//end parallel market

        //For buyers in auction
        for(i=0;i<ab1.length;i++){
                //for the first 3 iterations, bests global are generated with uniform distribution;

        if(e<0){ab1[i].BestGlobalParameter[1]=0.4+0.6*Math.random();ab1[i].BestGlobalParameter[2]=0
.2+0.8*Math.random();}
                //for the rest iterations, use PSO to select from best locals;
                else PSO.ChooseLearningParameterbuyer1(ab1[i], ab1, ab2, s, m, dm, bbm,em, PSO, e);
        }

        for(i=0;i<ab2.length;i++){
                //for the first 3 iterations, bests global are generated with uniform distribution;

        if(e<0){ab2[i].BestGlobalParameter[1]=0.4+0.6*Math.random();ab2[i].BestGlobalParameter[2]=0
.2+0.8*Math.random();}
                //for the rest iterations, use PSO to select from best locals;
                else PSO.ChooseLearningParameterbuyer2(ab1, ab2[i], ab2, s, m, dm, bbm,em, PSO, e);
        }

        //For buyers in auction
        for(i=0;i<s.length;i++){
                //for the first 3 iterations, bests global are generated with uniform distribution;

        if(e<0){s[i].BestGlobalParameter[1]=0.4+0.6*Math.random();s[i].BestGlobalParameter[2]=0.2+0.
8*Math.random();}
                //for the rest iterations, use PSO to select from best locals;
                else PSO.ChooseLearningParameterseller(ab1, ab2, s[i], s, m, dm, bbm,em, PSO, e);}
```

```
		/******print results 0f every iteration******/
if(e>=0){
aprintOut.println();aprintOut.print(tools.P.df1.format(dm.AverageBidPrice1[00]));aprintOut.print("\t");
aprintOut.print(tools.P.df1.format(dm.WinBidPrice[00]));aprintOut.print("\t");
aprintOut.print(tools.P.df1.format(dm.P1[00]));
for(i=0;i<m.TotalbuyerNumber;i++) {
aprintOut.print("\t\t"+tools.P.df1.format(ab1[i].Bid[00])+"\t");
aprintOut.print("\t"+tools.P.df1.format(ab1[i].WeeklyProfit[00][0])+"\t");}
a2printOut.println();a2printOut.print(tools.P.df1.format(dm.AverageBidPrice2[0]));a2printOut.print("\t");
a2printOut.print(tools.P.df1.format(dm.WinBidPrice[0]));a2printOut.print("\t");
a2printOut.print(tools.P.df1.format(dm.P2[00]));
		for(i=0;i<m.TotalbuyerNumber;i++) {
		a2printOut.print("\t\t"+tools.P.df1.format(ab2[i].Bid[00])+"\t");a2printOut.print("\t"+tools.P.df1.f
ormat(ab2[i].WeeklyProfit[00][0])+"\t");}

printOut.println();printOut.print(tools.P.df1.format(dm.AvgRandomRevserve[0]));printOut.print("\t");
printOut.print(tools.P.df1.format(dm.WinBidPrice[0]));printOut.print("\t");
printOut.print(tools.P.df1.format(dm.TotalSold[0]));
		for(i=0;i<m.TotalsellerNumber;i++) {
printOut.print("\t\t"+tools.P.df1.format(s[i].r[0])+"\t");
printOut.print("\t"+tools.P.df1.format(s[i].WeeklyProfit[0][0])+"\t");}
		}//end print

		/******determine the market equilibrium******/
em.EquilibriumStorageParameter(ab1, ab2,s, m, dm,e, parameterNo);
if(e%20==0&&e>20)em.TestEquilibrium(ab1, ab2,s, m,dm, e, parameterNo);//last one: parameter Number
if(em.JudgeEquilibrium==true||e>=m.Evolution-1){em.StorageBuyer1(ab1, e);em.StorageBuyer2(ab2,
e);em.StorageSellers(s, e);
		if(e<m.Evolution-1)em.EquilibriumEnd=e-em.CriterionPerSet+1;
		else em.EquilibriumEnd=m.Evolution; break;}
		else continue;
		}//end evolution;

		/******calculate the running time of each round********/
		cn.EquilibriumIteration[ec]=em.EquilibriumEnd;
		cn.EquilibriumTime[ec]=System.currentTimeMillis()-time;
			cn.EquilibriumMarketBid1[ec]=em.MarketBid1[0];
for(i=0;i<m.TotalbuyerNumber;i++)cn.EquilibriumBidRatio1[i][ec]=em.CommonValueBid1[i][0];
		cn.EquilibriumMarketBid2[ec]=em.MarketBid2[0];
for(i=0;i<m.TotalbuyerNumber;i++)cn.EquilibriumBidRatio2[i][ec]=em.CommonValueBid2[i][0];
		cn.EquilibriumReserve[ec]=em.rTemp[0];

		for(i=0;i<m.TotalsellerNumber;i++)cn.EquilibriumReserveRatio[i][ec]=em.ReserveTemp[i][0];
		}//end equilibrium; ec reached maximum

		em.MeanDeviation(cn.EquilibriumIteration, cn.Iteration, 100, -
1);em.MeanDeviation(cn.EquilibriumTime, cn.Time, 100, -1);
		em.MeanDeviation(cn.EquilibriumMarketBid1, cn.MarketBid1, 100, -1);
		for(i=0;i<m.TotalbuyerNumber;i++)em.MeanDeviation(cn.EquilibriumBidRatio1[i],
cn.CommonValueBid1[i], 200, -1);
		em.MeanDeviation(cn.EquilibriumMarketBid2, cn.MarketBid2, 100, -1);

		for(i=0;i<m.TotalbuyerNumber;i++)em.MeanDeviation(cn.EquilibriumBidRatio2[i],cn.Common
ValueBid2[i],200,-1);
		em.MeanDeviation(cn.EquilibriumReserve, cn.ReservePrice, 100, -1);
```

```
            for(i=0;i<m.TotalsellerNumber;i++)em.MeanDeviation(cn.EquilibriumReserveRatio[i],cn.Reserv
eTemp[i],200,-1);

                        /******print the results when market reaches equilibrium******/
                        PprintOut.println("");P2printOut.println(""); SprintOut.println("");
                        if(cn.MarketStructureFixed==false){

            PprintOut.print(tools.P.df1.format(m.ParallelMarket)+"\t");PprintOut.print(tools.P.df1.format(m.P
SOTestNumber)+"\t");
                                    PprintOut.print("\t");         }
                        else
if(cn.AlgorithmParameterFixed==true){PprintOut.print(tools.P.df1.format(PSO.w)+"\t");PprintOut.print(to
ols.P.df1.format(PSO.c1)+"\t");
                        PprintOut.print(tools.P.df1.format(PSO.c2)+"\t");}

            else{PprintOut.print(tools.P.df1.format(PSO.wup)+"\t");PprintOut.print(tools.P.df1.format(PSO.c
up)+"\t");
                        PprintOut.print(tools.P.df1.format(PSO.cup)+"\t");}
                        if(cn.MarketStructureFixed==false){
                        P2printOut.print(tools.P.df1.format(m.ParallelMarket)+"\t");
                        P2printOut.print(tools.P.df1.format(m.PSOTestNumber)+"\t");
                                    P2printOut.print("\t");}
                        else if(cn.AlgorithmParameterFixed==true){

            P2printOut.print(tools.P.df1.format(PSO.w)+"\t");P2printOut.print(tools.P.df1.format(PSO.c1)+"\t
");
                        P2printOut.print(tools.P.df1.format(PSO.c2)+"\t");}
                        else{
                        P2printOut.print(tools.P.df1.format(PSO.wup)+"\t");
                        P2printOut.print(tools.P.df1.format(PSO.cup)+"\t");
                        P2printOut.print(tools.P.df1.format(PSO.cup)+"\t");   }

                        if(cn.MarketStructureFixed==false){

            SprintOut.print(tools.P.df1.format(m.ParallelMarket)+"\t");SprintOut.print(tools.P.df1.format(m.P
SOTestNumber)+"\t");                                    SprintOut.print("\t");}
                        else if(cn.AlgorithmParameterFixed==true){

            SprintOut.print(tools.P.df1.format(PSO.w)+"\t");SprintOut.print(tools.P.df1.format(PSO.c1)+"\t");
                        SprintOut.print(tools.P.df1.format(PSO.c2)+"\t");}
                        else{

            SprintOut.print(tools.P.df1.format(PSO.wup)+"\t");SprintOut.print(tools.P.df1.format(PSO.cup)+"
\t");
                        SprintOut.print(tools.P.df1.format(PSO.cup)+"\t");}

                        //*print out mean of market price, buyers' strategies, run time and iterations the program
used reaching equilibrium ;

            PprintOut.print(tools.P.df1.format(cn.MarketBid1[0])+"\t");P2printOut.print(tools.P.df1.format(cn
.MarketBid2[0])+"\t");
                        SprintOut.print(tools.P.df1.format(cn.ReservePrice[0])+"\t");
            for(i=0;i<m.TotalbuyerNumber;i++){PprintOut.print(tools.P.df0.format(cn.CommonValueBid1[i]
[0])+"\t");}
```

```
            for(i=0;i<m.TotalbuyerNumber;i++){P2printOut.print(tools.P.df0.format(cn.CommonValueBid2[i
][0])+"\t");}

            for(i=0;i<m.TotalsellerNumber;i++){SprintOut.print(tools.P.df0.format(cn.ReserveTemp[i][0])+"\
t");}
            PprintOut.print(tools.P.df0.format(cn.Time[0])+"\t");
            P2printOut.print(tools.P.df0.format(cn.Time[0])+"\t");
            SprintOut.print(tools.P.df0.format(cn.Time[0])+"\t");
            if(cn.Iteration[0]>=m.Evolution-1) PprintOut.print("N/A\t");else
            PprintOut.print(tools.P.df0.format(cn.Iteration[0])+"\t");
            if(cn.Iteration[0]>=m.Evolution-1) P2printOut.print("N/A\t");else
            P2printOut.print(tools.P.df0.format(cn.Iteration[0])+"\t");
            if(cn.Iteration[0]>=m.Evolution-1) SprintOut.print("N/A\t");else
            SprintOut.print(tools.P.df0.format(cn.Iteration[0])+"\t");
            //print out standard deviation;
            PprintOut.println("");PprintOut.print("\t\t\t");P2printOut.println("");P2printOut.print("\t\t\t");
            SprintOut.println("");SprintOut.print("\t\t\t");

            PprintOut.print(tools.P.df1.format(cn.MarketBid1[1])+"\t");P2printOut.print(tools.P.df1.format(cn
.MarketBid2[1])+"\t");
                        SprintOut.print(tools.P.df1.format(cn.ReservePrice[1])+"\t");

            for(i=0;i<m.TotalbuyerNumber;i++){PprintOut.print(tools.P.df0.format(cn.CommonValueBid1[i]
[1])+"\t");}

            for(i=0;i<m.TotalbuyerNumber;i++){P2printOut.print(tools.P.df0.format(cn.CommonValueBid2[i
][1])+"\t");}

            for(i=0;i<m.TotalsellerNumber;i++){SprintOut.print(tools.P.df0.format(cn.ReserveTemp[i][1])+"\
t");}
            PprintOut.print(tools.P.df0.format(cn.Time[1])+"\t");
            P2printOut.print(tools.P.df0.format(cn.Time[1])+"\t");
            SprintOut.print(tools.P.df0.format(cn.Time[1])+"\t");
            if(cn.Iteration[0]>=m.Evolution-1) PprintOut.print("");else
            PprintOut.print(tools.P.df0.format(cn.Iteration[1])+"\t");
            if(cn.Iteration[0]>=m.Evolution-1) P2printOut.print("");else
            P2printOut.print(tools.P.df0.format(cn.Iteration[1])+"\t");
            if(cn.Iteration[0]>=m.Evolution-1) SprintOut.print("");else
            SprintOut.print(tools.P.df0.format(cn.Iteration[1])+"\t");
            }//end criterion
            P.t("\n\n End of the simulation");    PprintOut.close();P2printOut.close();SprintOut.close();
}


/* *****************Buyer 1 Class************************************************/

package PSOmpr;import java.util.*;import tools.*;import java.lang.*;
public class AuctionBuyer1 {
            Random r= new Random();Random s= new Random();Random rp = new Random();

            AuctionBuyer1(ModelPSO m,int id){
                        ID=id;
            cost0=m.BuyerCostParameter0;cost1=m.BuyerCostParameter1;cost2=m.BuyerCostParameter2;
                        PreviousBidRatio=new double[m.ParallelMarket];
                        Bid =new double[m.ParallelMarket];
```

```
			PreviousBid1=new double[m.ParallelMarket];
			CommonValueBid=new double[m.ParallelMarket];
			Quantity=new double[m.ParallelMarket];
			PreviousQuantity=new double[m.ParallelMarket];
			vector=new double[m.ParallelMarket][2];BestLocalParameter=new 
double[m.ParallelMarket][3];BestGlobalParameter=new double[3];
			LocalParameter=new double[m.PSOTestNumber+1][m.ParallelMarket][3];
			Cost=new double[m.ParallelMarket];PreviousCost=new double[m.ParallelMarket];
			Profit=new double[m.ParallelMarket];
			WeeklyProfit=new double[m.ParallelMarket][m.WeekPerCycle];
			PreviousProfit=new double[m.ParallelMarket];
			CommonValue=new double[m.ParallelMarket];
			PercentQuantity=new double[m.ParallelMarket];}

			int ID;double[] Cost; double[] PreviousCost;double cost0, cost1, cost2;
			double [][]BestLocalParameter;double[] BestGlobalParameter;double[][][] 
LocalParameter;double[][] vector;
			double[] PreviousBidRatio;
			double[] Bid;
			double[] Quantity;
			double[] PreviousQuantity;
			double[][] WeeklyProfit;   double[] Profit;double[] PreviousProfit;
			double[] PercentQuantity;
			boolean[] PutBid;

			void EvolutionInitialization(ModelPSO m, RetailMarket bbm, PSOEquilibrium em, PSO 
PSO, int e, int pm) {
			for(int i=0;i<m.WeekPerCycle;i++) WeeklyProfit[pm][i]=0;
			CommonValueBid(m, PSO, e, pm); }

			void WeekInitialization(ModelPSO m, RetailMarket bbm, PSOEquilibrium em, 
SpotMarket dm,int e, int pm){
			Bid(m, em, dm, e, pm);     Quantity[pm]=0;  Profit[pm]=0;
			} // end of weekinitialization

			void CommonValueBid(ModelPSO m, PSO PSO, int e, int pm){
			// this is for the first 20 trades when values are initialized
			if(e<0){ CommonValueBid[pm]=0.2+0.3*Math.random();

		LocalParameter[e+m.PSOTestNumber][pm][2]=0.2+0.3*Math.random();
					BestGlobalParameter[2]=0.2+0.3*Math.random();

		BestLocalParameter[pm][2]=LocalParameter[e+m.PSOTestNumber][pm][2];
					return;   }// after that e > 0
			// velocity
vector[pm][1]=PSO.w*vector[pm][1]+PSO.c1*m.r.nextDouble()*(BestLocalParameter[pm][2]-
PreviousBidRatio[pm])+PSO.c2*m.r.nextDouble()*(BestGlobalParameter[2]-PreviousBidRatio[pm]);
			// bid stragety
			CommonValueBid[pm]=vector[pm][1]+PreviousBidRatio[pm];
			 if(CommonValueBid[pm]> 1){   CommonValueBid[pm]=1;    }
			 else if(CommonValueBid[pm]<-1){   CommonValueBid[pm]=-1;   }
			 vector[pm][1]=CommonValueBid[pm]-PreviousBidRatio[pm];
			} // end of BidStrategy

			void Bid(ModelPSO m, PSOEquilibrium em, SpotMarket dm, int e, int pm){
				Bid[pm]=CommonValueBid[pm];
```

```
                    } // end of Bid

                    void WeeklyProfit(ModelPSO m, SpotMarket dm, RetailMarket bbm, int week, int pm){
                            WeeklyProfit[pm][week]=dm.Eprofit1[pm];
                    } // end of Profit

                    void StorageParameter(ModelPSO m, int pm){
                            double profitC=0;int week;double count=0;
                       if(m.WeekPerCycle>1)week=1;else week=0;
                            for(int i=week;i<m.WeekPerCycle;i++){
                            profitC+=WeeklyProfit[pm][i];count++;}
                            profitC=profitC/count;Profit[pm]=WeeklyProfit[pm][week];
                    } // end of Storage Parameter

                    void StoragePrevious(ModelPSO m, int pm){
                             PreviousBidRatio[pm]= CommonValueBid[pm]; PreviousBid1[pm]=Bid[pm];
PreviousQuantity[pm]= Quantity[pm];
                             PreviousProfit[pm]=Profit[pm]; PreviousCost[pm]= Cost[pm];

                    } // end of StoragePrevious

} // end of BuyerAuction1


/* ******************Buyer 2 Class**************************************************/
package PSOmpr;import java.util.*;import tools.*;import java.lang.*;
public class AuctionBuyer2 {
          Random r= new Random();Random s= new Random();Random rp = new Random();

          AuctionBuyer2(ModelPSO m,int id){
                    ID=id;cost0=m.BuyerCostParameter0;cost1=m.BuyerCostParameter1;
                    PreviousBidRatio=new double[m.ParallelMarket];
                    Bid =new double[m.ParallelMarket];PreviousBid=new double[m.ParallelMarket];
                    TempBid= new double[m.ParallelMarket];
                    CommonValueBid=new double[m.ParallelMarket];
                    Quantity=new double[m.ParallelMarket];
                    PreviousQuantity=new double[m.ParallelMarket];
                    vector=new double[m.ParallelMarket][2];
                    BestLocalParameter=new double[m.ParallelMarket][3];BestGlobalParameter=new
double[3];
                    LocalParameter=new double[m.PSOTestNumber+1][m.ParallelMarket][3];
                    Cost=new double[m.ParallelMarket];PreviousCost=new double[m.ParallelMarket];
                    Profit=new double[m.ParallelMarket];WeeklyProfit=new
double[m.ParallelMarket][m.WeekPerCycle];
                    PreviousProfit=new double[m.ParallelMarket];
                    PercentQuantity=new double[m.ParallelMarket];
          }
                    int ID;double[] Cost; double[] PreviousCost;double cost0, cost1, cost2;double
[][]BestLocalParameter;
                    double[] BestGlobalParameter;double[][][] LocalParameter;     double[][] vector;
                    double[] PreviousBidRatio;
                    double[] Bid;double[] PreviousBid;
                    double[] Quantity;double[] PreviousQuantity;
                    double[][] WeeklyProfit;    double[] Profit;double[] PreviousProfit;
                    double[] PercentQuantity;
                    boolean[] PutBid;
```

```
        void EvolutionInitialization(ModelPSO m, RetailMarket bbm, PSOEquilibrium em, PSO PSO, int
e, int pm) {
                        for(int i=0;i<m.WeekPerCycle;i++)           WeeklyProfit[pm][i]=0;
                        CommonValueBid(m, PSO, e, pm);
        }

        void WeekInitialization(ModelPSO m, RetailMarket bbm, PSOEquilibrium em, SpotMarket
dm,int e, int pm){
                        Bid(m, em, dm, e, pm);    Quantity[pm]=0;  Profit[pm]=0;
        } // end of weekinitialization

        void CommonValueBid(ModelPSO m, PSO PSO, int e, int pm){
                // this is for the first 20 trades when values are initialized
                        if(e<0){ CommonValueBid[pm]=0.2+0.3*Math.random();

        LocalParameter[e+m.PSOTestNumber][pm][2]=0.2+0.3*Math.random();
                                BestGlobalParameter[2]=0.2+0.3*Math.random();

        BestLocalParameter[pm][2]=LocalParameter[e+m.PSOTestNumber][pm][2];
                                return;   }// after that e > 0
                // velocity
vector[pm][1]=PSO.w*vector[pm][1]+PSO.c1*m.r.nextDouble()*(BestLocalParameter[pm][2]-
PreviousBidRatio[pm])+PSO.c2*m.r.nextDouble()*(BestGlobalParameter[2]-PreviousBidRatio[pm]);
                // bid stragety
                  CommonValueBid[pm]=vector[pm][1]+PreviousBidRatio[pm];
                   if(CommonValueBid[pm]>1){   CommonValueBid[pm]=1;    }
                  else if(CommonValueBid[pm]<-1){  CommonValueBid[pm]=-1;   }
                  vector[pm][1]=CommonValueBid[pm]-PreviousBidRatio[pm];
        } // end of BidStrategy

        void Bid(ModelPSO m, PSOEquilibrium em, SpotMarket dm, int e, int pm){
                        Bid[pm]=CommonValueBid[pm];//Math.max(CommonValueBid[pm],
dm.information);
        } // end of Bid

        void WeeklyProfit(ModelPSO m, SpotMarket dm, RetailMarket bbm, int week, int pm){
                        WeeklyProfit[pm][week]=dm.Eprofit2[pm];
        } // end of Profit

        void StorageParameter(ModelPSO m, int pm){
                        double profitC=0;int week;double count=0;
                   if(m.WeekPerCycle>1)week=1;else week=0;
                        for(int i=week;i<m.WeekPerCycle;i++){
                        profitC+=WeeklyProfit[pm][i];count++;}
                        profitC=profitC/count;Profit[pm]=WeeklyProfit[pm][week];
        } // end of Storage Parameter

        void StoragePrevious(ModelPSO m, int pm){
                         PreviousBidRatio[pm]= CommonValueBid[pm]; PreviousBid[pm]=Bid[pm];
PreviousQuantity[pm]= Quantity[pm];
                         PreviousProfit[pm]=Profit[pm]; PreviousCost[pm]= Cost[pm];

        } // end of StoragePrevious

} // end of BuyerAuction2
```

```
/* ******************Seller Class***************************************************/
package PSOmpr;import java.util.*;import tools.P;
public class Seller {
            Random s= new Random();
            Seller(ModelPSO m,SpotMarket dm,int id){
                        ID=id;
                        PreviousBidRatio=new double[m.ParallelMarket];
                        r =new double[m.ParallelMarket];   PreviousBid=new double[m.ParallelMarket];
                        ReservePrice=new double[m.ParallelMarket];
                        Quantity=new double[m.ParallelMarket];
                        PreviousQuantity=new double[m.ParallelMarket];
                        vector=new double[m.ParallelMarket][2];
                        BestLocalParameter=new double[m.ParallelMarket][3];BestGlobalParameter=new
double[3];
                        LocalParameter=new double[m.PSOTestNumber+1][m.ParallelMarket][3];
                        Cost=new double[m.ParallelMarket];PreviousCost=new double[m.ParallelMarket];
                        Profit=new double[m.ParallelMarket];WeeklyProfit=new
double[m.ParallelMarket][m.WeekPerCycle];
                        PreviousProfit=new double[m.ParallelMarket];
            }

                        int ID;double[] Cost; double[] PreviousCost;double cost0, cost1, cost2;
                        double [][]BestLocalParameter;double[] BestGlobalParameter;double[][][]
LocalParameter;double[][] vector;
                        double[] PreviousBidRatio;
                        double[] r;double[] PreviousBid;
                        double[] ReservePrice;
                        double[] Quantity;double[] PreviousQuantity;
                        double[][] WeeklyProfit;   double[] Profit;double[] PreviousProfit;
                        double[] PercentQuantity;
                        boolean[] PutBid;

                        void EvolutionInitialization(ModelPSO m, RetailMarket bbm, PSOEquilibrium em, PSO
PSO, int e, int pm) {
                                    for(int i=0;i<m.WeekPerCycle;i++) WeeklyProfit[pm][i]=0;
                                    ReservePrice(m, PSO, e, pm);
                        }

                        void WeekInitialization(ModelPSO m, RetailMarket bbm, PSOEquilibrium em,
SpotMarket dm,int e, int pm){
                                    R(m, em, dm, e, pm);Quantity[pm]=0;Profit[pm]=0;

                        } // end of weekinitialization

                        void ReservePrice(ModelPSO m, PSO PSO, int e, int pm){
                                    // this is for the first 20 trades when values are initialized
                                    if(e<0){ReservePrice[pm]=0.2+0.3*Math.random();

            LocalParameter[e+m.PSOTestNumber][pm][2]=0.2+0.3*Math.random();
                                                BestGlobalParameter[2]=0.2+0.3*Math.random();

            BestLocalParameter[pm][2]=LocalParameter[e+m.PSOTestNumber][pm][2];
                                                return;}// after that e > 0
                        // velocity
vector[pm][1]=PSO.w*vector[pm][1]+PSO.c1*m.r.nextDouble()*(BestLocalParameter[pm][2]-
PreviousBidRatio[pm])+PSO.c2*m.r.nextDouble()*(BestGlobalParameter[2]-PreviousBidRatio[pm]);
                        // bid stragety
                        ReservePrice[pm]=vector[pm][1]+PreviousBidRatio[pm];
                        if(ReservePrice[pm]>1){   ReservePrice[pm]=1;  }
                        else if(ReservePrice[pm]<-1){  ReservePrice[pm]=-1;   }
                        vector[pm][1]=ReservePrice[pm]-PreviousBidRatio[pm];
                        } // end of BidStrategy

                        void R(ModelPSO m, PSOEquilibrium em, SpotMarket dm, int e, int pm){
                                   r[pm]=ReservePrice[pm];
                        } // end of Bid

                        void WeeklyProfit(ModelPSO m, SpotMarket dm, RetailMarket bbm, int week, int pm){
                                    WeeklyProfit[pm][week]=dm.SellerEprofit[pm];
                        } // end of Profit

                        void StorageParameter(ModelPSO m, int pm){
                                    double profitC=0;int week;double count=0;
                             if(m.WeekPerCycle>1)week=1;else week=0;
                                    for(int i=week;i<m.WeekPerCycle;i++){
                                    profitC+=WeeklyProfit[pm][i];count++;}
                                    profitC=profitC/count;      Profit[pm]=profitC;
                        } // end of Storage Parameter

                        void StoragePrevious(ModelPSO m, int pm){
                         PreviousBidRatio[pm]= ReservePrice[pm];   PreviousBid[pm]=r[pm];
PreviousQuantity[pm]= Quantity[pm];
                         PreviousProfit[pm]=Profit[pm]; PreviousCost[pm]= Cost[pm];
            } // end of StoragePrevious
} // end of Seller


/* ******************Auction Market
Class**************************************************/
package PSOmpr;import java.util.*;import tools.*;

public class AuctionMarket {
            double Distribution;
            double PreviousAveragePrice[];
            double AveragePrice[];
            double Prob[];double Prob1[];double Prob2[];          double P1[];double P2[];
            double Eprofit1[];double Eprofit2[];
            double TempEprofit1[];double TempEprofit2[];
            double AverageBidPrice[];double AverageBidPrice1[];double AverageBidPrice2[];
            double PreviousWinBidPrice[];
            double TotalAverageBidPrice[];double TempAverageBidPrice[];
            double PreviousAverageBidPrice[];
            double PreviousAverageReservePrice[];
            double TempBid1[];double TempBid2[];
            double RandomBid1[];double RandomBid2[];
            double TempRandomBid1[];double TempRandomBid2[];
            double SumRandomBid1[];double SumRandomBid2[];double SumRandomReserve[];
            double TotalProfit[];double AverageProfit[];
            double TotalReserve[];double AverageReserve[];double RandomAverageReserve[];
            double TempRandomAverageReserve[];
```

```
	double SellerEprofit[];double SellerEprofit1[];double SellerEprofit2[];
	double TotalBid1[];double TotalBid2[];
	double AvgRandomRevserve[];
	double TotalSold[];double TempTotalSold[];
	double Factor1[];double Factor2[];
	double WinBidPrice[];

	SpotMarket(ModelPSO m){
		AveragePrice=new double[m.ParallelMarket] ;
		WinBidPrice=new double[m.ParallelMarket];
		PreviousAveragePrice=new double[m.ParallelMarket];
		TotalQuantity1=new double[m.ParallelMarket];TotalQuantity2=new
double[m.ParallelMarket];
		Prob=new double[m.ParallelMarket];Prob1=new double[m.ParallelMarket];Prob2=new
double[m.ParallelMarket];
		Eprofit1=new double[m.ParallelMarket];Eprofit2=new double[m.ParallelMarket];
		TempEprofit1=new double[m.ParallelMarket];TempEprofit2=new
double[m.ParallelMarket];
		AverageBidPrice=new double[m.ParallelMarket];
		AverageBidPrice1=new double[m.ParallelMarket];AverageBidPrice2=new
double[m.ParallelMarket];
		SumRandomReserve=new double[m.ParallelMarket];
		PreviousWinBidPrice=new double[m.ParallelMarket];
		PreviousAverageBidPrice=new double[m.ParallelMarket];
		PreviousAverageReservePrice=new double[m.ParallelMarket];
		TempBid1=new double[m.ParallelMarket];	TempBid2=new double[m.ParallelMarket];
		RandomBid1=new double[m.ParallelMarket];RandomBid2=new
double[m.ParallelMarket];
		TempRandomBid1=new double[m.ParallelMarket];	TempRandomBid2=new
double[m.ParallelMarket];
		SumRandomBid1=new double[m.ParallelMarket];SumRandomBid2=new
double[m.ParallelMarket];
		TotalProfit=new double[m.ParallelMarket];
		AverageProfit=new double[m.ParallelMarket];
		TotalReserve=new double[m.ParallelMarket];AverageReserve=new
double[m.ParallelMarket];
		RandomAverageReserve=new double[m.ParallelMarket];
		SellerEprofit= new double [m.ParallelMarket];SellerEprofit1= new double
[m.ParallelMarket];SellerEprofit2= new double [m.ParallelMarket];
		TempRandomAverageReserve=new double[m.ParallelMarket];
		TempSellerEprofit=new double[m.ParallelMarket];
		TempAverageBidPrice=new double[m.ParallelMarket];TotalAverageBidPrice=new
double[m.ParallelMarket];
		TotalBid1=new double[m.ParallelMarket];	TotalBid2=new double[m.ParallelMarket];
		AvgRandomRevserve=new double[m.ParallelMarket];
		TotalSold=new double[m.ParallelMarket];	TempTotalSold=new
double[m.ParallelMarket];
		Factor1=new double[m.ParallelMarket];		Factor2=new double[m.ParallelMarket];
		P1=new double[m.ParallelMarket]; P2=new double[m.ParallelMarket];
	}

	void DirectMarketInitialzation(ModelPSO m,AuctionBuyer1[] p, Seller[] f){
		Arrays.fill(AveragePrice,0);Arrays.fill(PreviousAveragePrice, 0);
	}
```

```
public void AuctionDeal(AuctionBuyer1[] ab1, AuctionBuyer2[] ab2, Seller[] s, ModelPSO m, int week,
int pm){
        TotalReserve[pm]=0;TotalBid1[pm]=0;TotalBid2[pm]=0;
        for(int t=0;t<ab1.length;t++){TotalBid1[pm]=ab1[0].Bid[pm];// choice variable}
        for(int t=0;t<ab2.length;t++){TotalBid2[pm]=ab2[0].Bid[pm];//choice variable [0,1]}
        for(int t=0;t<s.length;t++){TotalReserve[pm]=s[0].r[pm];}
        AverageReserve[pm]=TotalReserve[pm]/m.TotalsellerNumber;

Distribution = 100;
AverageBidPrice[pm]=0;WinBidPrice[pm]=0;TempEprofit1[pm]=0;TempEprofit2[pm]=0;SellerEprofit[p
m]=0;;TotalSold[pm]=0;SumRandomReserve[pm]=0;RandomAverageReserve[pm]=0;Eprofit1[pm]=0;Epr
ofit2[pm]=0;SellerEprofit[pm]=0;P1[pm]=0;P2[pm]=0;SumRandomBid1[pm]=0;SumRandomBid2[pm]=0
;

//Loop with the Stochastic Reserve Price
for(double c3=0; c3<=Distribution;c3++){
if(c3!=0 && c3!=Distribution)Factor1[pm]=1/Distribution;
else Factor1[pm]=(1/Distribution)/2;
RandomAverageReserve[pm]=AverageReserve[pm]-c3/m.variationloop;
        for(double c1=0;c1<=m.variationloop;c1++){
                if(c1!=0 && c1!=m.variationloop)Factor2[pm]=1/m.variationloop;
                else Factor2[pm]=.5/m.variationloop;
                Prob2[pm]=0;Prob1[pm]=0;
                TempRandomBid1[pm]=0;TempRandomBid2[pm]=0;
                RandomBid1[pm]=TotalBid1[pm]-c1/m.variationloop ;
                RandomBid2[pm]=TotalBid2[pm]-c1/m.variationloop ;

                // Inside loop bidder 2
                for(double c2=0;c2<=m.variationloop;c2++){
                        if(c2!=0 && c2!=m.variationloop)Prob[pm]=1/m.variationloop;
                        else Prob[pm]=(1/m.variationloop)/2;

                TempRandomBid2[pm]=TotalBid2[pm]-c2/m.variationloop;
                if(RandomBid1[pm]>TempRandomBid2[pm] &&
RandomBid1[pm]>=RandomAverageReserve[pm])Prob1[pm]=Prob1[pm]+(Prob[pm])*Factor2[pm];
                else if(RandomBid1[pm]==TempRandomBid2[pm] &&
RandomBid1[pm]>=RandomAverageReserve[pm])Prob1[pm]=Prob1[pm]+(Prob[pm]/2)*Factor2[pm];

                TempRandomBid1[pm]=TotalBid1[pm]-c2/m.variationloop;
                if(RandomBid2[pm]>TempRandomBid1[pm] &&
RandomBid2[pm]>=RandomAverageReserve[pm])Prob2[pm]=Prob2[pm]+(Prob[pm])*Factor2[pm];
                else if(RandomBid2[pm]==TempRandomBid1[pm] &&
RandomBid2[pm]>=RandomAverageReserve[pm])Prob2[pm]=Prob2[pm]+(Prob[pm]/2)*Factor2[pm];
                }
        P1[pm]+=Prob1[pm]*Factor1[pm];
        P2[pm]+=Prob2[pm]*Factor1[pm];
        SumRandomBid1[pm]+=RandomBid1[pm]*Factor1[pm];
        SumRandomBid2[pm]+=RandomBid2[pm]*Factor1[pm];
        TotalSold[pm]=TotalSold[pm]+(Prob1[pm]+Prob2[pm])*Factor1[pm];
        Eprofit1[pm]=Eprofit1[pm]+((0.5-RandomBid1[pm])*Prob1[pm])*Factor1[pm];
        Eprofit2[pm]=Eprofit2[pm]+((0.5-RandomBid2[pm])*Prob2[pm])*Factor1[pm];
        SellerEprofit[pm]=SellerEprofit[pm]+(RandomBid1[pm]*Prob1[pm]+RandomBid2[pm]*Prob2[p
m])*Factor1[pm];
        }
        SumRandomReserve[pm]+=RandomAverageReserve[pm];
}//end of third loop*/
```

```
AvgRandomRevserve[pm]=SumRandomReserve[pm]/Distribution;
AverageBidPrice1[pm]=(SumRandomBid1[pm]/m.variationloop);
AverageBidPrice2[pm]=(SumRandomBid2[pm]/m.variationloop);
AverageBidPrice[pm]=(AverageBidPrice1[pm]+AverageBidPrice2[pm])/2;
WinBidPrice[pm]=(SellerEprofit[pm]/TotalSold[pm]);
PreviousAverageBidPrice[pm]=AverageBidPrice[pm];
PreviousWinBidPrice[pm]=WinBidPrice[pm];
PreviousAverageReservePrice[pm]=AvgRandomRevserve[pm];//AverageReserve[pm];
}//end of AuctionDeal
}// end of auction market


/* ******************PSO algorithm ***************************************************/
package PSOmpr;import java.util.*;
public class PSO {
        double w=.4,c1=1,c2=1;//w=0.98,c1=2,c2=2;double wup,cup;int Converge=0;

    PSO(ModelPSO m){Converge=m.Evolution;}

        void UpdatePSOParameter(ModelPSO m, int e,boolean Fixed){
                if(Fixed==true)return;
                else{
                if(e<0)e+=m.PSOTestNumber;
                w=wup*(double)Math.max(0,(this.Converge-e))/(double)this.Converge;
                c1=cup*(double)Math.max(0,(this.Converge-e))/(double)this.Converge;            //local
                c2=c1;//global}
        }

        void PSOParameterSetting(ModelPSO m,Criterion cn,int CN,boolean
MarketStructureFixed,boolean AlgorithmParameterFixed){
                if(MarketStructureFixed==false){
                        m.ParallelMarket=cn.pm[CN];m.PSOTestNumber=cn.localTest[CN];
                        if(AlgorithmParameterFixed==true){
                                w=.4;c1=1;c2=1; }
                                else{wup=.5;      cup=1;Converge=400;}
                }
                else if(AlgorithmParameterFixed==true){w=cn.w[CN];   c1=cn.c1[CN];
c2=cn.c2[CN];}
                else{wup=cn.wup[CN];cup=cn.cup[CN];Converge=cn.Converge[CN];}
        }

        //Choose PSO learning parameter: best local & best global;
        //Choose PSO learning parameter: best local & best global for Auction ;
        void ChooseLearningParameterbuyer1(AuctionBuyer1 buyer1, AuctionBuyer1[] ab1,
AuctionBuyer2[] ab2,Seller[] s,ModelPSO m,SpotMarket dm,RetailMarket bbm,PSOEquilibrium em,PSO
PSO,int e){
                        int count=0;
                        double LocalListAuction[][][]=new
double[m.PSOTestNumber+1][m.ParallelMarket][3];//[0:parameter][1:capacity parameter]
                        //initial
                        for(int pm=0;pm<m.ParallelMarket;pm++){
                                buyer1.BestLocalParameter[pm][0]=-2000000000;

                                for(count=0;count<m.PSOTestNumber+1;count++){
                                        LocalListAuction[count][pm][1]=-
                                1;LocalListAuction[count][pm][2]=-1;}
```

```
                                        }

                                        for(int pm=0;pm<m.ParallelMarket;pm++){
                                                count=0;
                                                for(int ev=0;ev<m.PSOTestNumber;ev++){
                                                boolean same=false;
                                                for(int kk=0;kk<count;kk++){

        if(buyer1.LocalParameter[ev][pm][2]==LocalListAuction[kk][pm][2]){same=true;break;}
                                                }
                                                if (same==false)
        {LocalListAuction[count][pm][2]=buyer1.LocalParameter[ev][pm][2];  count++;}
                                                }

        LocalListAuction[count][pm][2]=buyer1.PreviousBidRatio[pm];count++;
                                        }

                                        for(int pm=0;pm<m.ParallelMarket;pm++){
                                          LinkedList AvailablebuyersList=new LinkedList();
AvailablebuyersList.clear();count=0;

                                        while(LocalListAuction[count][pm][2]>=-1) {
                                                        LocalListAuction[count][pm][0]=0;

        buyer1.CommonValueBid[pm]=LocalListAuction[count][pm][2];buyer1.Bid(m, em,dm, e,pm);
                                                        for(int i=0;i<ab1.length;i++){


        if(ab1[i].ID!=buyer1.ID)ab1[i].CommonValueBid[pm]=ab1[i].PreviousBidRatio[pm]; }
                                                        for(int week=0;week<m.WeekPerCycle;week++){
                                                                for(int
i=0;i<ab1.length;i++){ab1[i].WeekInitialization(m,bbm,em,dm, e,pm);}
                                                                dm.AuctionDeal(ab1, ab2, s, m, week, pm);
                                                                buyer1.WeeklyProfit(m, dm, bbm, week,
pm);
                                                }//end weeks;

        buyer1.StorageParameter(m,pm);LocalListAuction[count][pm][0]+=buyer1.Profit[pm];
                                                if((LocalListAuction[count][pm][0]-
buyer1.BestLocalParameter[pm][0])>=-1){

        buyer1.BestLocalParameter[pm][2]=buyer1.CommonValueBid[pm];

        buyer1.BestLocalParameter[pm][0]=LocalListAuction[count][pm][0];   }
                                                        count++;
                                                        if(count>=m.PSOTestNumber)break;
                                                        }
                                                for(int
i=0;i<m.PSOTestNumber;i++){buyer1.LocalParameter[i][pm][2]=buyer1.LocalParameter[i+1][pm][2];}

        buyer1.LocalParameter[m.PSOTestNumber][pm][2]=buyer1.BestLocalParameter[pm][2];
                                        }//end pm;
                                        TestGlobalParameterTradeAuction1(buyer1, ab1,
ab2,s,m,dm,bbm,PSO,e);
                                }
```

```
            //Choose PSO learning parameter: best local & best global;
            //Choose PSO learning parameter: best local & best global for Auction ;
            void ChooseLearningParameterbuyer2( AuctionBuyer1[] ab1, AuctionBuyer2 buyer2,
AuctionBuyer2[] ab2,Seller[] s,ModelPSO m,SpotMarket dm,RetailMarket bbm,PSOEquilibrium em,PSO
PSO,int e){
                        int count=0;
                        double LocalListAuction2[][][]=new
double[m.PSOTestNumber+1][m.ParallelMarket][3];//[0:parameter][1:capacity parameter]
                              //initial
                        for(int pm=0;pm<m.ParallelMarket;pm++){
                                    buyer2.BestLocalParameter[pm][0]=-2000000000;
                                    for(count=0;count<m.PSOTestNumber+1;count++){
                                          LocalListAuction2[count][pm][1]=-
                                    1;LocalListAuction2[count][pm][2]=-1;}
                              }

                              for(int pm=0;pm<m.ParallelMarket;pm++){
                                    count=0;
                                    for(int ev=0;ev<m.PSOTestNumber;ev++){
                                    boolean same=false;
                                    for(int kk=0;kk<count;kk++){

            if(buyer2.LocalParameter[ev][pm][2]==LocalListAuction2[kk][pm][2]){same=true;break;}
                                    }
                                    if (same==false){
            LocalListAuction2[count][pm][2]=buyer2.LocalParameter[ev][pm][2];
                                          count++;}
                                    }

            LocalListAuction2[count][pm][2]=buyer2.PreviousBidRatio[pm];count++;}

                              for(int pm=0;pm<m.ParallelMarket;pm++){
                                     LinkedList AvailablebuyersList=new LinkedList();
AvailablebuyersList.clear();count=0;
                                    while(LocalListAuction2[count][pm][2]>=-1) {
                                          LocalListAuction2[count][pm][0]=0;

            buyer2.CommonValueBid[pm]=LocalListAuction2[count][pm][2];
                                          buyer2.Bid(m, em,dm, e,pm);
                                          for(int i=0;i<ab2.length;i++){

            if(ab2[i].ID!=buyer2.ID)ab2[i].CommonValueBid[pm]=ab2[i].PreviousBidRatio[pm]; }

                                    for(int week=0;week<m.WeekPerCycle;week++){
                                    for(int
i=0;i<ab2.length;i++)ab2[i].WeekInitialization(m,bbm,em,dm, e,pm);
                                                dm.AuctionDeal(ab1, ab2, s, m, week, pm);
                                                buyer2.WeeklyProfit(m, dm, bbm, week,
pm);
                                          }//end weeks;
                                          buyer2.StorageParameter(m,pm);

            LocalListAuction2[count][pm][0]+=buyer2.Profit[pm];
                                    if((LocalListAuction2[count][pm][0]-
buyer2.BestLocalParameter[pm][0])>=-1){
```

```
            buyer2.BestLocalParameter[pm][2]=buyer2.CommonValueBid[pm];

            buyer2.BestLocalParameter[pm][0]=LocalListAuction2[count][pm][0]; }
                                                            count++;if(count>=m.PSOTestNumber)break;}

                                                    for(int
i=0;i<m.PSOTestNumber;i++){buyer2.LocalParameter[i][pm][2]=buyer2.LocalParameter[i+1][pm][2];}

            buyer2.LocalParameter[m.PSOTestNumber][pm][2]=buyer2.BestLocalParameter[pm][2];
                                    }//end pm;
                        TestGlobalParameterTradeAuction2(ab1, buyer2, ab2,s,m,dm,bbm,PSO,e);

                        }

            void ChooseLearningParameterseller(AuctionBuyer1[] ab1, AuctionBuyer2[] ab2,Seller seller,
Seller[] s,ModelPSO m,SpotMarket dm,RetailMarket bbm,PSOEquilibrium em,PSO PSO,int e){
                  int count=0;double LocalListSeller[][][]=new
double[m.PSOTestNumber+1][m.ParallelMarket][3];
                  //initial
                  for(int pm=0;pm<m.ParallelMarket;pm++){
                        seller.BestLocalParameter[pm][0]=-2000000000;

            for(count=0;count<m.PSOTestNumber+1;count++){LocalListSeller[count][pm][1]=-1;
LocalListSeller[count][pm][2]=-1; }
                  }

                  for(int pm=0;pm<m.ParallelMarket;pm++){
                        count=0;
                        for(int ev=0;ev<m.PSOTestNumber;ev++){
                        boolean same=false;
                        for(int kk=0;kk<count;kk++){

            if(seller.LocalParameter[ev][pm][2]==LocalListSeller[kk][pm][2]){same=true;break;}}
                        if (same==false){
            LocalListSeller[count][pm][2]=seller.LocalParameter[ev][pm][2];count++;}}
                        LocalListSeller[count][pm][2]=seller.PreviousBidRatio[pm];   count++;}


                  for(int pm=0;pm<m.ParallelMarket;pm++){
                         LinkedList AvailablesellerList=new LinkedList();  AvailablesellerList.clear();
            count=0;
                        while(LocalListSeller[count][pm][2]>=-1) {

            LocalListSeller[count][pm][0]=0;seller.ReservePrice[pm]=LocalListSeller[count][pm][2];seller.R(
m, em,dm, e,pm);                                                  for(int
i=0;i<s.length;i++){if(s[i].ID!=seller.ID)s[i].ReservePrice[pm]=s[i].PreviousBidRatio[pm]; }

                                    for(int week=0;week<m.WeekPerCycle;week++){
                                                    for(int
i=0;i<s.length;i++)s[i].WeekInitialization(m,bbm,em,dm, e,pm);
                                                    dm.AuctionDeal(ab1, ab2, s, m, week, pm);
            seller.WeeklyProfit(m, dm, bbm, week, pm);
                                    }//end weeks;

            seller.StorageParameter(m,pm);LocalListSeller[count][pm][0]+=seller.Profit[pm];
```

```
                                    if((LocalListSeller[count][pm][0]-
seller.BestLocalParameter[pm][0])>=-1){
                                            seller.BestLocalParameter[pm][2]=seller.ReservePrice[pm];
                                            seller.BestLocalParameter[pm][0]=LocalListSeller[count][pm]
                                    [0];}
                                    count++;
                                    if(count>=m.PSOTestNumber)break;
                                    }
                            for(int
i=0;i<m.PSOTestNumber;i++){seller.LocalParameter[i][pm][2]=seller.LocalParameter[i+1][pm][2];
        }

        seller.LocalParameter[m.PSOTestNumber][pm][2]=seller.BestLocalParameter[pm][2];
                        }//end pm;
                TestGlobalParameterTradeSeller(ab1, ab2, seller, s,m,dm,bbm,PSO,e);
                }

//Choose PSO learning parameter: best local & best global;
void TestGlobalParameterTradeAuction1(AuctionBuyer1 buyer1, AuctionBuyer1[] ab1, AuctionBuyer2[]
ab2,Seller[] s,ModelPSO m, SpotMarket dm,RetailMarket bbm,PSO PSO,int e){
        //use the best instead of average of local as global
        buyer1.BestGlobalParameter[0]=-2000000;
   for(int pm=0;pm<m.ParallelMarket;pm++){
        if(buyer1.BestLocalParameter[pm][0]>buyer1.BestGlobalParameter[0]){
                buyer1.BestGlobalParameter[0]=buyer1.BestLocalParameter[pm][0];
                buyer1.BestGlobalParameter[2]=buyer1.BestLocalParameter[pm][2];    }
   }
}

void TestGlobalParameterTradeAuction2(AuctionBuyer1[] ab1, AuctionBuyer2 buyer2, AuctionBuyer2[]
ab2,Seller[] s,ModelPSO m, SpotMarket dm,RetailMarket bbm,PSO PSO,int e){
        //use the best instead of average of local as global
                buyer2.BestGlobalParameter[0]=-2000000;
        for(int pm=0;pm<m.ParallelMarket;pm++){
        if(buyer2.BestLocalParameter[pm][0]>buyer2.BestGlobalParameter[0]){
                buyer2.BestGlobalParameter[0]=buyer2.BestLocalParameter[pm][0];
                buyer2.BestGlobalParameter[2]=buyer2.BestLocalParameter[pm][2];    }
     }
}

void TestGlobalParameterTradeSeller(AuctionBuyer1[] ab1, AuctionBuyer2[] ab2,Seller seller, Seller[]
s,ModelPSO m, SpotMarket dm,RetailMarket bbm,PSO PSO,int e){
        //use the best instead of average of local as global
        seller.BestGlobalParameter[0]=-2000000;

   for(int pm=0;pm<m.ParallelMarket;pm++){
        if(seller.BestLocalParameter[pm][0]>seller.BestGlobalParameter[0]){
                seller.BestGlobalParameter[0]=seller.BestLocalParameter[pm][0];
                seller.BestGlobalParameter[2]=seller.BestLocalParameter[pm][2];        }
   }
}

}// end of PSO algorithm
```

```
/* ******************Equilibrium
Criterion************************************************/
package PSOmpr;import java.util.*;import tools.P;
public class PSOEquilibrium {
            int   CriterionPerSet=5;
            double JudgeDevCR[]={0.0000001,0.000000001};
            int EquilibriumEnd=0;
            boolean  JudgeEquilibrium;
            double[]    rTemp;
            double[]    MarketBidTemp1;double[]          MarketBid1;
            double[]    MarketBidTemp2;double[]          MarketBid2;
            double[] ReserveRandom;
            double[][]  CapacityRatio;
            double[][]  CommonValueBidTemp1;double[][] CommonValueBidTemp2;
            double[][] ReserveTemp;double[][] Reserve;
            double[][] BidPriceTemp;
            double     convergeTime;
            double[][] CommonValueBid1;double[][] CommonValueBid2;

            PSOEquilibrium(ModelPSO m){
                        rTemp=new double[CriterionPerSet];
                        MarketBidTemp1=new double[CriterionPerSet];
                        MarketBid1=new double[CriterionPerSet];
                        MarketBid2=new double[CriterionPerSet];
                        ReserveRandom=new double[CriterionPerSet];
                        MarketBidTemp2=new double[CriterionPerSet];
                        CapacityRatio=new double[m.TotalbuyerNumber][2];
                        CommonValueBid1=new double [m.TotalbuyerNumber][2];
                        CommonValueBid2=new double [m.TotalbuyerNumber][2];
                        CommonValueBidTemp1=new double[m.TotalbuyerNumber][CriterionPerSet];
                        CommonValueBidTemp2=new double[m.TotalbuyerNumber][CriterionPerSet];
                        ReserveTemp=new double[m.TotalsellerNumber][CriterionPerSet];
                        Reserve=new double[m.TotalsellerNumber][CriterionPerSet];
                        //use mean value to determine the criteria
            }

            void EquilibriumInitial(){  convergeTime=0;JudgeEquilibrium=false;}
//Buyer 1
void EquilibriumStorageParameter(AuctionBuyer1[] ab1,AuctionBuyer2[] ab2, Seller[] s, ModelPSO
m,SpotMarket dm,int e,int parameterNo){
                        if(e<CriterionPerSet&&e>0){

            MarketBidTemp1[e]=dm.PreviousWinBidPrice[0];MarketBidTemp2[e]=dm.PreviousWinBidPric
e[0];
                                    rTemp[e]=dm.PreviousAverageReservePrice[0];
                                    for(int
i=0;i<ab1.length;i++)CommonValueBidTemp1[i][e]=ab1[i].PreviousBidRatio[0];
                                    for(int
i=0;i<ab2.length;i++)CommonValueBidTemp2[i][e]=ab2[i].PreviousBidRatio[0];
                                    for(int i=0;i<s.length;i++)ReserveTemp[i][e]=s[i].PreviousBidRatio[0];
                                    }//;if evolutionary iteration No. is less than CriterionNo, only store the market
price and buyers' strategies of one selected market;

                        else{
                        for(int k=0;k<CriterionPerSet-1;k++){
```

```
				MarketBidTemp1[k]=MarketBidTemp1[k+1];MarketBidTemp1[CriterionPerSet-
1]=dm.WinBidPrice[0];
				MarketBidTemp2[k]=MarketBidTemp2[k+1];MarketBidTemp2[CriterionPerSet-
1]=dm.WinBidPrice[0];
				rTemp[k]=rTemp[k+1];rTemp[CriterionPerSet-1]=dm.AvgRandomRevserve[0];}
				for(int i=0;i<ab1.length;i++){
				for(int k=0;k<CriterionPerSet-
1;k++)CommonValueBidTemp1[i][k]=CommonValueBidTemp1[i][k+1];
					CommonValueBidTemp1[i][CriterionPerSet-1]=ab1[i].PreviousBidRatio[0];
				}//;storage buyersacution' strategy parameters of selected market;*/;

				for(int i=0;i<ab2.length;i++){
				for(int k=0;k<CriterionPerSet-
1;k++)CommonValueBidTemp2[i][k]=CommonValueBidTemp2[i][k+1];
					CommonValueBidTemp2[i][CriterionPerSet-1]=ab2[i].PreviousBidRatio[0];}
					for(int i=0;i<s.length;i++){
					for(int k=0;k<CriterionPerSet-1;k++)ReserveTemp[i][k]=ReserveTemp[i][k+1];
					ReserveTemp[i][CriterionPerSet-1]=s[i].PreviousBidRatio[0]; }
				}//;else continuously update the temporal storage set;
		}

		void TestEquilibrium(AuctionBuyer1[] ab1, AuctionBuyer2[] ab2, Seller[] s, ModelPSO
m,SpotMarket dm,int e,int parameterNo){
				if(JudgeEquilibrium==true)return;
				double mean[]=new double[2];double std[]=new double[2];
				double cr[]=new double[CriterionPerSet];double mp[]=new double[CriterionPerSet];
				double h[]=new double[2];
				boolean EorNot=false;
				/*1st, test if market price reach equilibrium for one selected market*/
				System.arraycopy(MarketBidTemp1, 0, mp, 0, mp.length);MeanDeviation(mp,h,e,-1);
				if(h[1]>JudgeDevCR[0])return;
				System.arraycopy(MarketBidTemp2, 0, mp, 0, mp.length);MeanDeviation(mp,h,e,-1);
				if(h[1]>JudgeDevCR[0])return;
				System.arraycopy(rTemp, 0, mp, 0, mp.length);MeanDeviation(mp,h,e,-1);
				if(h[1]>JudgeDevCR[0])return;

				/*2nd, test if buyers' strategy parameters reach equilibrium for one selected market;*/
				for(int i=0;i<ab1.length;i++){
					System.arraycopy(CommonValueBidTemp1[i], 0,cr, 0, cr.length);
					MeanDeviation(cr,h,e,-1);
					if(h[1]>JudgeDevCR[0])return;}
				for(int i=0;i<ab2.length;i++){
					System.arraycopy(CommonValueBidTemp2[i], 0,cr, 0, cr.length);
					MeanDeviation(cr,h,e,-1);
					if(h[1]>JudgeDevCR[0])return;}
				for(int i=0;i<s.length;i++){
					System.arraycopy(ReserveTemp[i], 0,cr, 0, cr.length);
					MeanDeviation(cr,h,e,-1);
					if(h[1]>JudgeDevCR[0])return;}
				/*If the above two tests return true value, test if each buyer has same strategy parameters
in every parallel market;*/
				for(int i=0;i<ab1.length;i++){
					for(int
k=0;k<m.ParallelMarket;k++)CommonValueBidTemp1[i][k]=ab1[i].PreviousBidRatio[k];
					System.arraycopy(CommonValueBidTemp1[i], 0,cr, 0, cr.length);
					MeanDeviation(cr,h,e,-1);
					if(h[1]<=JudgeDevCR[0])continue;
					else return;}
				for(int i=0;i<ab2.length;i++){
					for(int
k=0;k<m.ParallelMarket;k++)CommonValueBidTemp2[i][k]=ab2[i].PreviousBidRatio[k];
					System.arraycopy(CommonValueBidTemp2[i], 0,cr, 0, cr.length);
					MeanDeviation(cr,h,e,-1);
					if(h[1]<=JudgeDevCR[0])continue;
					else return;}
				for(int i=0;i<s.length;i++){
					for(int
k=0;k<m.ParallelMarket;k++)ReserveTemp[i][k]=s[i].PreviousBidRatio[k];
					System.arraycopy(ReserveTemp[i], 0,cr, 0, cr.length);
					MeanDeviation(cr,h,e,-1);
					if(h[1]<=JudgeDevCR[0])continue;
					else return;}
				JudgeEquilibrium=true;
	}

	void StorageBuyer1(AuctionBuyer1[] ab1, int e){
				int l=CriterionPerSet;
				MeanDeviation(MarketBidTemp1,MarketBid1 , e,l);
				for(int i=0;i<ab1.length;i++)
MeanDeviation(CommonValueBidTemp1[i],CommonValueBid1[i],e,l);}
	void StorageBuyer2(AuctionBuyer2[] ab2, int e){
				int l=CriterionPerSet;
				MeanDeviation(MarketBidTemp2,MarketBid2 , e,l);
				for(int i=0;i<ab2.length;i++)
MeanDeviation(CommonValueBidTemp2[i],CommonValueBid2[i],e,l);}
	void StorageSellers(Seller[] s, int e){
				int l=CriterionPerSet;
				MeanDeviation(rTemp,ReserveRandom , e,l);
				for(int i=0;i<s.length;i++) MeanDeviation(ReserveTemp[i],Reserve[i],e,l);//}

	void MeanDeviation(double[] d,double[] h,int e,int l){
				if(e<d.length)return;
				double temp=0,temp1=0;
				int begin=0;
				if(l<0){l=d.length;begin=0;}
				else begin=d.length-l;
				for(int k=begin;k<d.length;k++){
				temp+=d[k];}
				temp=temp/(double)l;
				h[0]=temp;//storage parameter mean
				for(int k=begin;k<d.length;k++)temp1+=(temp-d[k])*(temp-d[k]);
				h[1]=Math.abs(Math.sqrt(temp1/(l-1)));//storage parameter deviation;
std=Math.sqrt(temp1/(d.length-1));
	}
}// End of Equilibrium Criterion

/* ******************Parameters **************************************************/
package PSOmpr;
import java.util.Arrays;
public class Criterion {
	static int CriterionNo=1;
	static boolean AlgorithmParameterFixed=false;//true;
```

```
        static boolean MarketStructureFixed=true;
        //Fixed parameter
        double[] w   ={.4,  .4,  .4,  .1 };//inertia weight
        double[] c1  ={1.5,  1,  .5,   1};//self  confidence factor
        double[] c2  ={1.5,  1,  .5,   1};//swarm confidence factor
        static double[] wup  ={ .75, .75,   .98,    .98   };
        static double[] cup ={ .75,   .5,   .4,    1.5,  };
        static int[] Converge ={ 400, 400,   400,    400  };
        static int[] pm ={ 5, 10,  3};
        static int[] localTest={ 3,  3,  3};
        double[]   MarketPrice;
        double[] MarketBid1;double[] MarketBid2;
        double[][] CommonValueBid1;double[][] CommonValueBid2;
        double[]   Time;
        double[]   Iteration;
        double[]   EquilibriumIteration;double[]   EquilibriumTime;
        double[]   EquilibriumMarketBid1;double[][] EquilibriumBidRatio1;double[][]
EquilibriumBidRatio2;
        double[] EquilibriumMarketBid2;
        double[][] EquilibriumReserveRatio;
        double[] EquilibriumReserve;
        double[] ReservePrice;
        double[][] ReserveTemp;

        Criterion(ModelPSO m){
                Iteration=new double[2];
                Time=new double[2];
                MarketPrice=new double[2];MarketBid1= new double[2];MarketBid2= new double[2];
                CommonValueBid1=new double[m.TotalbuyerNumber][2];CommonValueBid2=new
double[m.TotalbuyerNumber][2];
                EquilibriumIteration=new double[m.EquilibriumNo];
                EquilibriumTime=new double[m.EquilibriumNo];
                EquilibriumMarketBid1=new double[m.EquilibriumNo];   EquilibriumBidRatio1=new
double[m.TotalbuyerNumber][m.EquilibriumNo];
                EquilibriumBidRatio2=new double[m.TotalbuyerNumber][m.EquilibriumNo];
EquilibriumMarketBid2=new double[m.EquilibriumNo];
                EquilibriumReserve=new double[m.EquilibriumNo];
                EquilibriumReserveRatio =new double[m.TotalsellerNumber][m.EquilibriumNo];
                ReservePrice=new double[2];
                ReserveTemp=new double[m.TotalsellerNumber][2]; }
}// End of Parameters
```

.

VITA

Christopher N. Boyer

Candidate for the Degree of

Doctor of Philosophy

Thesis: AGENT-BASED COMMON VALUE AUCTIONS

Major Field: Agricultural Economics

Education: Completed the requirements for the Doctor of Philosophy in Agricultural Economics at Oklahoma State University, Stillwater, Oklahoma in December of 2011. Completed the requirements for the Master of Science in Agricultural Economics from Texas A&M University, College Station, Texas in August 2008. Completed the requirements for the Bachelor of Science in Agribusiness from Texas A&M University, College Station, Texas in May 2006.

Experience: Research Assistant for Department of Agricultural Economics at Oklahoma State University from August 2008 to December 2011; research and Teaching Assistant for the Department of Agricultural Economics at Texas A&M University from August 2006-August 2008.

Professional Memberships: Agricultural and Applied Economics Association, Southern Agricultural Economics Association, and Western Agricultural Economics Association

Name: Christopher N. Boyer Date of Degree: December, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: AGENT-BASED COMMON VALUE AUCTIONS

Pages in Study: 122 Candidate for the Degree of Doctor of Philosophy

Major Field: Agricultural Economics

Scope and Method of Study: This research is composed of three essays about agent-based common value auctions. The objective of the first essay is to establish an agent-based first-price common-value auction to determine the impact of a reserve price with two buyers and with three buyers. In the second essay, the agent-based common-value auction model is used to provide theoretical insight into the likely change in beef packers' market power before and after the Livestock Mandatory Price Reporting Act. The objective in the third essay is to determine if a first-price common-value auction with a reserve price or a posted-price market provides a seller with the larger expected revenue using agent-based models. In these three essays several theoretical contributions are made to the auction literature, and developing an agent-based common-value auction extends the agent-based modeling literature.

Findings and Conclusions: Results from these essays provide unique insight into auction theory, agent-based modeling, and federal agricultural policy. From the first essay, a reserve price increases the equilibrium winning bid price and decreases the probability that the item is sold in the two and three buyer auctions. Additionally, a reserve price increases the winning bid price more than an additional buyer and no reserve price. In the second essay, results provide a unique theoretical argument that the Livestock Mandatory Price Reporting Act benefits producers by reducing beef packers' market power. Results from the third essay show the seller is indifferent between a posted price and auctioning an item when the seller and the buyers have similar noisy signals. However, when the seller has perfect information or buyers have less uncertainty than the seller, the seller prefers the posted-price market.

ADVISER'S APPROVAL: B. Wade Brorsen