An Electronic Marketplace: Agent-based Coordination Models for Online Auctions

Ayşe Morali
Technische Universität Darmstadt; Darmstadt, Germany

Leonardo Varela
Universidad de Los Andes; Bogotá, Colombia

Carlos A. Varela
Rensselaer Polytechnic Institute; Troy, NY 12180, USA

morali@winf.tu-darmstadt.de, l-varela@uniandes.edu.co, cvarela@cs.rpi.edu

Abstract

Different coordination models exist for managing concurrency in complex distributed systems, for example, direct interaction, tuple spaces, hierarchical structures, and publish-and-subscribe mechanisms. Online auctions are complex distributed systems, where the choice of coordination model can significantly impact the performance of multiple software agents, acting on behalf of human buyers and sellers. In this paper, we evaluate analytically and experimentally different auction types and coordination models in an *electronic marketplace*. Three important metrics are: the number of software agents, the number of messages exchanged between these agents, and the number of migrations performed by the agents. We conclude that dynamically choosing the number of agents and the coordination model can improve quality of service in electronic commerce applications. Furthermore, in online auctions where there is a lot of interaction between software agents, migration of agents to virtual stores and local bidding is more fair and efficient than using remote interaction. This observation has the potential to significantly improve and redefine existing online auction systems such as eBay.

Keywords: Coordination models, multi-agent systems, electronic commerce, online auctions

1 INTRODUCTION

The World-Wide Web [4] has fueled a plethora of electronic services that have either matched or surpassed previously existing services since their inception only a decade ago. For example, electronic newspapers today are widely read, conceivably as much as or even more than physical newspapers. They reach faster a larger worldwide audience. They introduce significant and challenging new problems, for example: on-screen formatting and navigation, advertisement methodologies, back issue indexing and search procedures, and online interaction fora such as blogs. Ultimately, electronic newspapers are an invaluable complement to their (much older) physical newspaper counterparts, in terms of bridging the gap between information provision and use.

Electronic commerce has also seen just the tip of the iceberg in terms of potential services to customers and companies alike. Only a few years ago, very few people dared enter their credit card information in an online web form. Today, it is common to use the web as an electronic purchasing medium, for example, to buy airline tickets on specialized electronic travel agencies on the web, to offer company stocks on trading sites, or to participate in online auctions, just to name a few.

As the sophistication of web services grows, as information becomes both human and machine-readable [9], and as multi-agent systems become pervasive in the electronic information age, it becomes imperative to re-think traditional processes in understanding how to tackle the significant and challenging new problems introduced by the new electronic commerce medium.

In this paper, we consider a futuristic electronic marketplace consisting of software agents representing human buyers, sellers, and auctioneers. We study and analyze different types of auctions in terms of their potential to satisfy users, and in terms of their electronic complexity, that is, number of software agents, cost and patterns of communication, and likelihood of inducing commercial transactions. We also study and analyze different multi-agent coordination models in terms of computational complexity, and electronic commerce metrics, such as user satisfaction. Finally, we envision a system that requires minimal human intervention and that dynamically adapts the electronic commerce strategy from the perspective of the sellers and auctioneers (e.g., type of auction) and from the perspective of the buyers (e.g., the inter-agent coordination model) to optimize electronic customer satisfaction. Ultimately, we aim to create an adaptive electronic marketplace that can successfully complement today's commercial activities and bring in a new perspective to the fair and efficient exchange of goods.

Structure of Paper Section 2 surveys existing auction types and coordination models. Section 3 introduces the electronic marketplace architecture and the dynamic coordination model selection approach. Section 4 presents analytical models for different auction types and coordination models in terms of metrics such as number of agents, messages and migrations. This section also shows experimental results obtained through a software prototype. Section 5 concludes with a discussion of contributions and potential future work.

2 RELATED WORK

2.1 Auction Types

Online auctions mimic auctions in the physical world and differ according to many parameters like the role of the sellers or buyers and the sort of pricing or bargaining. In this section, we introduce different types of auctions as they might be used in the cyber world.

Software agents participate in an auction on behalf of their owner, who can be a seller or a buyer. Considering the roles of the agents, auctions are classified as *seller auctions* and *buyer auctions*. Seller auctions are the classical form of auctions. Bidders place offers on one or more items. On the other hand, buyer auctions, which are also known as *reverse auctions*, work the other way around. Sellers compete with each other to satisfy the buyer. There are also *double auctions*, which are a combination of buyer and seller auctions.

Depending on the pricing and the bargaining schemas, auctions are categorized into *iterative auctions* and *sealed-bid auctions*. *English auctions* and *Dutch auctions* belong to the first group, where the price ascends iteratively. *First-price-sealed-bid auctions* and *Vickrey auctions* are the major examples of auctions in the second group. Following, we are going to describe these different types in more detail.

English auction: This is the most frequently used and common method and is known also as the *open-outcry auction* or the *ascending-price auction* [13]. The auctioneer begins here with the lowest acceptable price, which is not secret, and proceeds to solicit successively higher bids from the buyer agents, and ends with a timeout. The item is sold to the highest bidder agent at the price of her last bid [11]. In an English auction, the buyer agents may bid several times or react to the bids of others arbitrarily. The information flow of English auctions can be seen in the sequence diagram in Figure 1a.

One variation of the English auction is the *open-exit auction* [15]. Here the prices rise continuously, and bidders must publicly announce that they are dropping out when the price is too high. Once a bidder has dropped out, she may not reenter. In another variation, an auctioneer calls out each asking price and bidders lift a paddle in the online version, send an accept message, to indicate a willingness to pay that amount. Then the auctioneer calls out another price, etc. *Yankee auctions* [19] are another type of English auction, where multiple items of the same kind are negotiated in one auction. The items are allocated after the auction is finished, depending on different criteria like desired amount of items and maximum budget of the agent.

Dutch auction: This auction is also known as the *descending-price auction* [14]. It has an iterative format. Bidding starts at a relatively high price and is progressively lowered [24]. The agent that accepts the lowered price gets the item. All of the buyers have the same chance to get the item, but the ones that hesitate too long miss it. When the goods are exhausted, the bidding is over.

The message flow of the Dutch auctions are very similar to the message flow in English auctions. The only difference is that it does not end with time over but with price acceptance of a buyer agent. In order

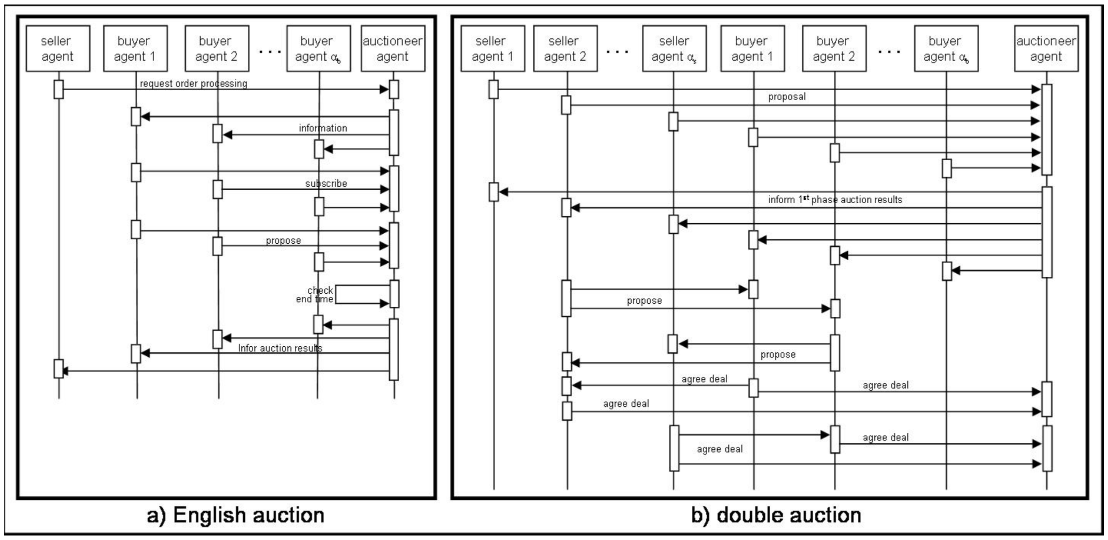

Figure 1: Sequence diagram for English and double auctions

to protect the seller from dis-profit a lower bound can be optionally predefined.

When multiple units are auctioned, normally more buyers are present. The first winner takes one of the items and pays her price and later winners pay less [24]. Once again, when the goods are exhausted, the bidding is over.

It is believed that the English system may be inferior to the Dutch in one area. From the seller's point of view, the key to any successful auction is the effect of competition on the potential buyers, and in an English auction, the under-bidder usually forces the bid up by one small step [14]. The winner may end up paying well according to her point of view but the seller does still not receive the maximum price.

Reverse auction: In reverse auctions, the successful bidder is determined by the lowest price submitted to the auctioneer at the conclusion of the auction [12]. While potential suppliers are underbidding each other, the buyer does not do anything but only observes the negotiation.

Reverse auctions have the following flow [10]: The buyer submits a list of items to the auctioneer. The auctioneer invites sellers to participate, with a specified start and closing time. All bidder's identities are kept confidential during the auction. Bidders submit prices that are ranked and disseminated to all bidders as new bids are made. An extension of the auction closing time may be triggered for a pre-determined period of time if one of the top bid rankings changes. The auction closes once no new bids are placed and the original or extended closing time expires. All bidders are immediately notified of the final bid rankings, and the auctioneer notifies the buyer of the bidding results.

Sealed-bid auction: In sealed-bid auctions, each bidder submits one bid without knowledge of all other bids [24]. That is what being sealed denotes, as opposite to open-outcry like the English or Dutch varieties. Each agent can bid only once and the other agents do not know the buying strategy or the budget of their competitors. In *first-price-sealed-bid auctions* [16] after all bids are executed they are co-instantaneously declared and the highest bidder gets the item for the price she bid. That is where the term "first-price" comes from. In the case of a buyer auction the supplier with the lowest bargain sells the item.

When multiple units are being auctioned, it is called *discriminatory auction*. In that case, the bids are sorted from high to low and the units are sold to the bidders, starting with the highest, until the supply is exhausted. Each bidder has to pay the amount she bid. This results in different prices for different winners. This is where the term "discriminatory" comes from [16].

Vickrey auction: Like the first-price sealed bid auction, there is also a *uniform second-price auction*, known with its finder's name: Vickrey [25]. This auction works also with sealed bids [24]. The difference is that in the Vickrey auction the bidder who made the highest bid gets the good at the price of the highest non-winning bid. If multiple units are being auctioned, all winners pay the price of the highest non-winning bid.

Double auction: Double auctions have been the principal trading format in U.S. financial institutions for over a hundred years [8]. In double auctions, as can be seen from the sequence diagram in Figure 1b, both sellers and buyers submit bids which are then ranked highest to lowest to generate demand and supply profiles [23]. From the profiles, the maximum quantity exchanged can be determined by matching the selling offers (starting with lowest price and moving up) with demand bids (starting with highest price and moving down) [7]. This format allows both buyers and sellers to negotiate at any particular moment.

One interesting variation of double auction is the *Double-Dutch auction* [8], where a buyer price clock starts at a very high price and goes downward. At some point the buyer stops the clock and bids on the unit at a price complimentary to her. At this point a seller clock starts upward from a very low price and continues upwards until a seller stops it and offers a unit at that price. Then the buyer clock resumes once again downwards, etc. When the two prices cross, the auction is over. The double auction has many variants and is evolving rapidly. Economists believe that the double auction will have many applications as auctions become computerized [17].

2.2 Coordination Models

Actors: The actor model of computation [1] provides a good starting point for building multi-agent systems because of its asynchronous nature of communication, distributed memory, and dynamic reconfiguration properties. Actors coordinate activities by exchanging messages sent asynchronously. In response to a message, an actor may: (1) change its otherwise completely encapsulated state, (2) send messages to other known actors, (3) create new actors, or (4) migrate to a new location.

Due to asynchronous communication and state encapsulation, actor migration is much simpler than object migration, since no execution stacks are present due to nested method invocations and no more than one logical thread is ever accessing the internal state of an actor. As long as a universal naming scheme exists to refer to mobile actors, communication between actors is location transparent [20].

While asynchronous message passing is a sufficient primitive for reasoning about open distributed systems (see e.g., [2]), higher-level coordination mechanisms are needed to effectively build complex multi-agent systems. Actor programming languages, such as SALSA [22], provide additional coordination constructs built over message passing, that facilitate building complex communication protocols, without breaking the dynamic reconfiguration aspects of the model. Coordination constructs include futures, token-passing continuations, join blocks, and first-class continuations.

Tuple Spaces: Linda [6] introduced a powerful coordination paradigm consisting of processes communicating through a shared tuple space. Processes may: (1) write a tuple to the space, (2) read a tuple from the space leaving it in the space, or (3) read and remove a tuple from the space.

An advantage of the tuple space model is that it enables coordination to trascend space and time boundaries since: (a) processes can be in different computers when communicating with a tuple space, and (b) a tuple can outlive its writing process and therefore it can be read by a process that did not exist at the time the tuple was created.

While tuple spaces go a long way in providing a natural shared memory abstraction for inter-process coordination, their main limitation lies in the difficulty of scaling up coordination to a very large number of processes due to the bottleneck introduced at the tuple space. The logical centralization of a tuple space also imposes a significant overhead in terms of remote message passing in an open distributed system.

Publish and Subscribe: Publish and subscribe models have been proposed in the literature (e.g., [3, 5]) to enable what can be regarded as *active* tuple spaces. Rather than expecting processes to continuously poll for a given tuple in a tuple space, publish and subscribe strategies enable processes to *subscribe* to specific patterns of messages or tuples, and to *publish* tuples with the expectation that all currently subscribed processes will receive a copy of the tuple.

An advantage of publish and subscribe is that multiple subscribers which are co-located can be serviced more efficiently by creating an overlay tree structure and *multicasting* (rather than broadcasting) new information as it becomes available. For example, if web customers are watching the Soccer World Cup 2006 originating in Germany, the video stream packets need only travel once from Europe to the American continent, then be forwarded to intermediate nodes representing North and South America, and so on, which is obviously much more efficient than *broadcasting* the video streams, which would imply many one-to-one connections from America to Europe.

Several major complexities arise when considering a publish and subscribe mechanism including how to express subscription constraints, and how to dynamically maintain an efficient multicasting overlay network, in the presence of multiple information sources and stringent reliability requirements.

Hierarchical Coordination: The hierarchical model for coordination of concurrent activities [21] advocates the creation of groups of actors, or *casts*, to serve as units of coordination among related processes. An actor group is coordinated by a *director*, a special actor in charge of intra-cast coordination. Directors may themselves belong to coordination casts, thereby creating a coordination hierarchy.

The hierarchical model is based on the observation that intra-cast communication is orders of magnitude more frequent that inter-cast communication. This so-called *near-decomposability* property of hierarchic systems [18] allows for their natural evolution and scalability. Simon illustrates this argument with examples drawn from systems such as multi-cellular organisms, social organizations, and astronomical systems.

In the context of electronic commerce, multiple software agents representing a buyer form natural hierarchies with directors coordinating purchases of specific items and item categories. The highest-level director within a group of software agents representing a buyer may enforce user constraints such as the maximum budget to use on acquiring a specific shopping list.

The main advantage of hierarchical coordination over other models is its scalability. The major drawback is that in simple electronic commerce scenarios, it can induce more messaging overhead than it would be necessary with a flat coordination structure.

3 APPROACH: AN ELECTRONIC MARKETPLACE

In order to evaluate different coordination models and auction types, we have created an online marketplace, where many buyer and seller agents come together and negotiate. In this section, we describe the architecture of this marketplace, and the criteria we used to indirectly measure user satisfaction.

The electronic marketplace is a system that enables dynamically choosing the number of agents, agent migrations, and the number of messages sent, for a given buyer. The goal is to increase the possibility of the agents to get all the products desired by the buyer within the given budget, and to accomplish this task in the given time period. Furthermore, from the perspective of the sellers and auctioneers, the system enables to dynamically choose the most suitable auction type, with the goal of optimizing their satisfaction.

The marketplace consists of four main components (see Figure 2): Coordination Model Manager (CMM), Seller Manager (SM), Stores and the Transaction Manager (TM). CMM is an interface between the buyer agents and the stores. It is responsible for choosing the most suitable coordination model by analyzing the items each buyer wants to buy and the items sold in different stores. SM is the interface between the seller agents and the stores. It is responsible for associating seller agents to stores. Stores model physical stores in the real world. They are in charge of spawning auctions, where agents negotiate for selling or buying items. In each store there are many auctions going on simultaneously. The seller and buyer agents come together in auctions. Every auction is managed by an auctioneer agent, which is responsible for deciding the type of auction and reporting the transactions to the TM. TM is the storing unit of the marketplace. All of the transaction results flow to its database.

When a customer enters the marketplace, it gives the items in its item list to the CMM. CMM chooses the coordination model and according to the model creates and sends agents to stores. At the same time, sellers of the real world, represented by seller agents, enter the marketplace and give the specifications of their items to the SM. SM contacts the stores, and either sends a seller agent for each new item to the store or creates a new store for this seller.

The main criterion that auctioneers take into consideration is the number of buyer and seller agents that supplies or demands an item. Since the seller/buyer wants to sell/buy the items in a predefined time period, time is the next important criterion. CMM decides on the coordination model, creates buyer agents and their itinerary, and sends them to stores. The number of messages sent and the number of agent migrations, is strongly affected by the coordination model, and strongly affects user satisfaction.

The indicator of success for our approach is user satisfaction. Satisfaction is measured for buyers by the percentage of items bought and the average price paid for each item as compared to the market price—the average price paid for that item by all the agents in the system. For the seller, user satisfaction is measured by the percentage of the items sold and total revenue. Finally, for the auctioneers, satisfaction is measured by the number of transactions.

The agents representing the same sellers/buyers are responsible for buying/selling all of the items in their item lists, in a limited time, and with a limited budget. If there are communication delays or failures, agents may end up buying more items than desired or getting back with empty hands. The number of remote messages sent between the agents, which are bidding at the same time in different auctions, are more critical

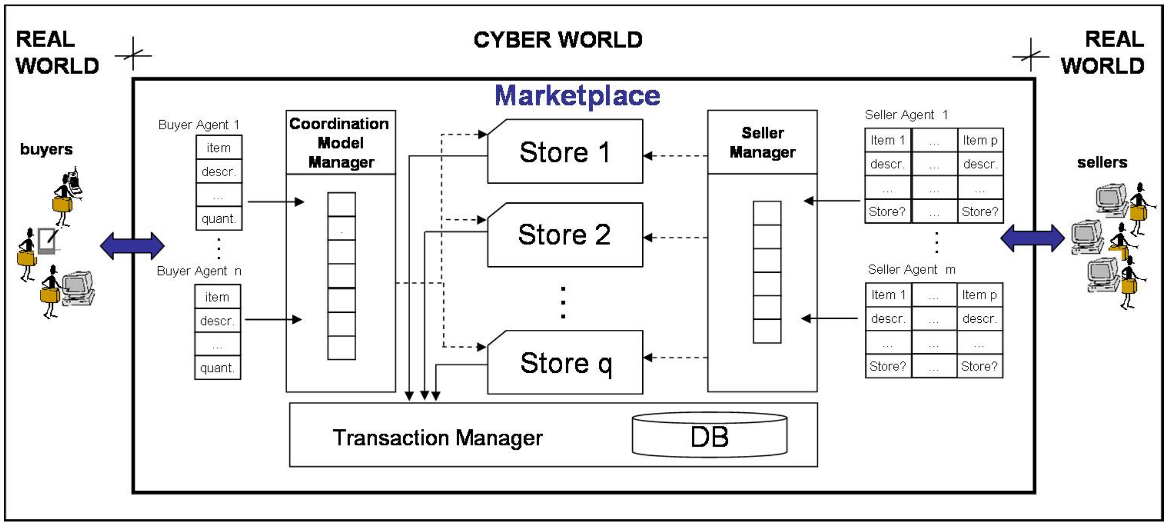

Figure 2: Architecture of the marketplace

than the internal messages, because they require more time. The number of agent migrations depends also very strongly on the chosen coordination model. Since the number of messages sent between the agents increases proportionally to the number of agents, the coordination model should help the agent owner to buy or sell the items with a minimal number of agents.

4 ANALYSIS AND SIMULATION RESULTS

In this section, we are first going to concentrate on the analytical models for the major auction types and coordination models. Later, we will move on to the simulations and results of our electronic marketplace prototype.

The analysis of the auction type and the coordination model are orthogonal to each other. Considering different types of auctions, there are two main variables affecting the success of an auction system. These are number of agents and number of messages. The coordination model affects the success of an auction environment through three variables. These are the number of agents, the number of messages and the number of migrations. The number of agents and number of messages affect each other directly. Since agents communicate with each other through messages, the number of messages increases geometrically with the increasing number of agents.

In an auction environment, time plays a very important role. Being able to migrate where the auction is taking place makes the bidding process less expensive and more fair. In an actor-based auction system the latency and bandwidth differences become less relevant than in a remote-bidding auction system like eBay. In remote-bidding auction systems, users complain about fairness of the bidding process, mostly in the last minute bid. The significance of migration versus remote communication will be detailed in the simulation section.

4.1 Comparison of the Auction Types

In this subsection we are going to compare the number of messages and the number of agents for the most significant auction types. This will help us decide under which conditions which auction type is the most efficient.

The variables of our mathematical approach are as following: α_b, symbolizes the number of buyer agents, α_s, the number of seller agents, β the average bids per agent that take place until an item is sold in an auction, σ, is the number of miniauctions in a double auction, κ, the number of messages, and α, the total number of agents.

The total number of agents is obtained in every auction type by adding the number of buyer, seller, and

	English	Reverse	Sealed-bid	Double
α_b	α_b	1	α_b	α_b
α_s	1	α_s	1	α_s
α	$\alpha_b + 2$	$\alpha_s + 2$	$\alpha_b + 2$	$\alpha_b + \alpha_s + 1$
κ	$\alpha_b(3+\beta)+2$	$\alpha_s(3+\beta)+2$	$4\alpha_b + 2$	$2(\alpha_b+\alpha_s)+\sigma(\beta+1)$

Table 1: Number of agents and messages in different auction types.

auctioneer agents. Since there is only one auctioneer in every auction type, the general formula is as follows:

$$\alpha \; = \; \alpha_b + \alpha_s + 1$$

There are five main stages in an auction process: The creation, notification, subscription, bidding process and notification of results of the auction. Referring to these stages, the computation of the number of messages needed to complete an English auction in our model is as follows: In the first stage, the seller agent requests the auctioneer to create a new auction by sending one message. In the second stage, the buyer agents request to join the auction. Such process generates as many messages as the number of buyer agents requesting. In the third stage, the auctioneer accepts or refuses the request of the buyer agents. The same number of messages as in the second stage are generated. In the fourth stage, one buyer agent places a bid by sending one message. The auctioneer notifies the other participants of the new bid. The total number of messages generated in this stage is equal to the total number of buyer agents per bid. In the final stage, the auctioneer notifies the seller and the buyers of the auction result. The total number of messages generated in this stage is equal to the number of buyer agents plus one. By adding the number of messages in these five stages, we obtain the formula in the Table 1 for the total number of messages:

$$\kappa \; = \; 1 + \alpha_b + \alpha_b + \beta\alpha_b + (\alpha_b + 1) \; = \; 1 + 3\alpha_b + \beta\alpha_b + 1 \; = \; 1 + \alpha_b(3+\beta) + 1 \; = \; \alpha_b(3+\beta) + 2$$

The reverse auction is similar to the English auction, but the sellers and the buyers change their roles. Consequently, the number of messages remains the same.

In sealed-bid auctions the total number of messages are deducted from the English auction. This is because there is only one bid per agent in the fourth stage, and the rest of the stages remain the same as the English auction.

In the first stage of a double auction, both buyers and sellers send their bids. That process generates $(\alpha_b + \alpha_s)$ messages. In the second stage, the auctioneer notifies the matching sellers and buyers that they are welcome to the second round, where miniauctions between them take place. It also notifies the not-matching buyers or sellers of their elimination from the auction. This notification stage generates $(\alpha_b + \alpha_s)$ messages. Therefore, in the first round in total $2(\alpha_b + \alpha_s)$ messages are generated. In the second and consequent round of a double auction, the number of seller and buyer agents is the same, and equals $min(\alpha_b, \alpha_s)$. In this round, σ miniauctions are situated, where $\sigma = min(\alpha_b, \alpha_s)$. In every miniauction, $(\beta + 1)$ messages on average are communicated, caused by biding done and the final notification to the auctioneer. Therefore, in the second round of a double auction, $\sigma(\beta + 1)$ messages are generated. By adding the number of messages in these two rounds, we obtain the formula for the total number of messages in Table 1.

If there is one buyer and several sellers then the auctioneer generates a reverse auction. In the opposite case, the auctioneer decides for the English. A third case is where many sellers and many buyers are interested in buying and respectively selling similar items. In that case the auctioneer decides for a double auction, because it minimizes the number of messages sent and allows more than one item to be purchased with different prices to different parties.

4.2 Comparison of the Coordination Models

In addition to input variables defined in Section 4.1, we are going to use ι, for the number of items in the shopping list of an agent, λ, for the number of auctions created, and α_T, for the total number of buyer agents in the system (a large value represents stiff competition). According to these variables, we will calculate κ, the number of messages and μ, the number of agent migrations.

Single-Agent: In this coordination model, the buyer is represented with one agent, migrating from one auction to the next in order to buy the products on its shopping list. Because there is no other agent in the system negotiating on behalf of the buyer, there are no messages communicated ($\kappa = 0$).

Since every buyer is represented with a unique agent in the single-agent model, there is no coordination, and this one agent migrates for each item to all of the auctions related with the item until it gets the item. The number of migrations is calculated with:

$$\mu = (\lambda - 1) \cdot \left(1 - \frac{\alpha_b}{\alpha_T}\right) + 2$$

The two at the end of the equation stands for the migration of the agent to the first auction and from the last auction back to the buyer's node. $(\lambda - 1)$ comes from the fact that, in order to buy the item, it may need to migrate to all of the auctions, except the beginning auction. Assuming that all buyer agents have equal probability of obtaining an item, we multiply it with $\left(1 - \frac{\alpha_b}{\alpha_T}\right)$. Since, the increasing number of total agents increases the concurrency, the probability of the agent to get the item in the auctions decreases. In the best case there is no competition, the agent will get the item in the first auction, it joins. Therefore, without migrating to other auctions, it returns back to the buyer's node. In the worst case scenario, it is going to migrate to all the auctions in its itinere.

Tuple Spaces: In tuple spaces, each agent continuously asks the memory space for changes, which increases the number of messages extremely.

The number of agents in the tuple spaces model changes between the number of items and the number of auctions, $(\iota \leq \alpha_b \leq \lambda)$.

Let ϕ symbolize the frequency of tuple space querying, and δ the duration of an auction. The number of queries per agent to a tuple space during an auction is $\left(\frac{\delta}{\phi}\right)$. Every time there is a bid, a tuple in the tuple space changes, therefore, β messages are needed. In order to find the number of querying messages per auction for all the agents representing the buyer in all the auctions they join, we multiply $\left(\beta + \frac{\delta}{\phi}\right)$ with $\lambda \cdot \alpha_b$. In the best case scenario, there is only one buyer agent for each item, $\alpha_b = \iota$, and it gets the item in the first auction it joins, $\lambda = 1$. Which gives us the following formula for the total number of messages in this best case scenario:

$$\kappa_{best} = \lambda \cdot \alpha_b \left(\beta + \frac{\delta}{\phi}\right) = 1 \cdot \alpha_b \left(\beta + \frac{\delta}{\phi}\right) = \iota \left(\beta + \frac{\delta}{\phi}\right)$$

In the worst case $(\alpha_b \ll \alpha_T)$, the agents have to join all the auctions offering the item they want to buy. If $\alpha_b = \lambda$, then the formula for the number of messages will be follows:

$$\kappa_{worst} = \lambda \cdot \alpha_b \left(\beta + \frac{\delta}{\phi}\right) = \lambda \cdot \lambda \left(\beta + \frac{\delta}{\phi}\right) = \lambda^2 \left(\beta + \frac{\delta}{\phi}\right)$$

We found a formula that gives the number of messages for all competition scenarios as following. When $(\alpha_b \ll \alpha_T)$, it implies that $(\alpha_b/\alpha_T \approx 0)$, and when $(\alpha_b \approx \alpha_T)$, it implies that $(\alpha_b/\alpha_T \approx 1)$. Considering these, the following formula gives the number of messages for tuple spaces:

$$\kappa = \left(\frac{\alpha_b}{\alpha_T} \cdot \iota + \left(1 - \frac{\alpha_b}{\alpha_T}\right) \cdot \ \alpha_b \cdot \lambda\right) \left(\beta + \frac{\delta}{\phi}\right)$$

The number of migrations reaches its minimum value, when $(\alpha_b = \lambda)$, because in that case every agent goes to a different auction and since all auctions are visited, there is no need for migration from one auction to another. Otherwise, $(\alpha_b < \lambda)$, and the agents have to migrate to all the auctions, offering the item they want to buy $(\lambda - \alpha_b)$. Assuming that all buyer agents have equal probability of obtaining an item, we multiply the number of migrations with $\left(1 - \frac{\alpha_b}{\alpha_T}\right)$. Furthermore, each agent migrates to the first auction in its itinerary and back from the last auction in its itinerary, which denotes $2\alpha_b$ migrations. Considering these, the formula for the total number of migrations is:

$$\mu = (\lambda - \alpha_b) \left(1 - \frac{\alpha_b}{\alpha_T}\right) + 2\alpha_b$$

Publish and Subscribe: Publish and Subscribe: The Publish and Subscribe coordination model works very similar to tuple spaces. The only difference concerning the number of messages and the number of migrations is that in publish and subscribe model, the agents do not query the tuple space regularly, but the tuple space sends a broadcast to the agents whenever there is a change.

Because of these similarities the formula for the number of migrations, which we have described in the tuple spaces, is the same for publish and subscribe and the formula for the number of messages is slightly modified. So the number of messages is calculated with:

$$\kappa = \left(\frac{\alpha_b}{\alpha_T} + \left(1 - \frac{\alpha_b}{\alpha_T} \right) \cdot \lambda \right) \cdot \beta(1 + \ \alpha_b)$$

Hierarchical Coordination: In this model, the agents associated with a buyer are organized in the form of a tree, with the root agent representing the buyer. The root has ι children, each of which is an agent for some product in the shopping list. At the second level each agent has children agents, which are responsible for one or many auctions for the item that the parent is responsible for. To be more precise, if a is a node with depth two, then it is the agent that represents the buyer in an auction relation with the item list, parent(a) is responsible for.

In this model, every leaf node sends a message to its parent and siblings after each bid of its corresponding auction. Other agents are not to be informed, since they take role in the auctions for different items.

Let $\lambda_{(b,i)}$ be the number of auctions buyer b joins related with item i. The worst case performance in terms of the number of messages transmitted arises, when $\alpha_b \ll \alpha_T$ and the buyer agent joins all the auctions. In this special case $\sum_{i=1}^{\iota} \lambda_{(b,i)} = \lambda_b$. Since, for each bid every agent sends a message to its parent and siblings, and there exists $\lambda_{(b,i)}$ siblings; number of messages transmitted is

$$\kappa = \sum_{i=1}^{\iota} \left(\lambda_{(b,i)} \right)^2 \cdot \beta$$

From these, we can express worst case performance as the solution of the following quadratic convex program:

$$sup \ \sum_{i=1}^{\iota} \left(\lambda_{(b,i)} \right)^2 \cdot \beta$$

$$st \ \sum_{i=1}^{\iota} \left(\lambda_{(b,i)} \right) = \lambda_b \ and \ \lambda_{(b,i)} \geq 0 \ \ \forall i \in \{1, 2, ..., \iota\}$$

Supremum of the above problem is $(\lambda_b)^2 \cdot \beta$, and it is achieved, when $\lambda_{(b,i)} = \lambda_b$ for some $i \in (1, 2, ..., \iota)$ and $\lambda_{(b,i')} = 0, \ \forall i' \in (1, 2, ..., \iota) \setminus \{i\}$. Then:

$$\kappa_{worst} = (\lambda_b)^2 \cdot \beta$$

In the best case, the agents get the items in the auction they first join, therefore the total number of auctions joined is equal to ι (one auction per item in the shopping list). The number of total messages is obtained, when $\lambda_{(b,i)} = 1, \forall i \in \{1, 2, ...\iota\}$.

$$\kappa_{best} = \Sigma_{i=1}^{\iota} (\lambda_{(b,i)})^2 \cdot \beta = \Sigma_{i=1}^{\iota} \beta = \iota \cdot \beta$$

For the average case; let α_i be the number of buyers desiring to buy item i; let λ_i be the number of auctions related with item i. Expected number of messages between the agents of the buyer is given by

$$E\left[\kappa\right] \ = \ E\left[\sum_{i=1}^{\iota} (\lambda_{(b,i)})^2 \cdot \beta \right] \ = \ \beta \cdot E\left[\sum_{i=1}^{\iota} (\lambda_{(b,i)})^2 \right] = \beta \cdot \sum_{i=1}^{\iota} E\left[(\lambda_{(b,i)})^2 \right]$$

Assuming $\lambda_i \leq \alpha_i$, every agent participating in the auction has an equal probability of winning, and the winner agent does not join the remaining auctions related with i^{th} item,

$$E\left[(\lambda_{(b,i)})^2 \right] = \sum_{j=1}^{\lambda_i} \frac{j^2}{\alpha_i} = \frac{1}{\alpha_i} \cdot \sum_{j=1}^{\lambda_i} j^2 = \frac{\lambda_i(\lambda_i + 1)(2\lambda_i + 1)}{6\alpha_i}$$

Therefore;

$$E[\kappa] = \beta \cdot \sum_{i=1}^{\iota} \frac{\lambda_i(\lambda_i + 1)(2\lambda_i + 1)}{6\alpha_i}$$

In the hierarchical coordination model the number of migrations is equal to the number of edges in the tree, which is one less than number of nodes in the tree. Therefore;

$$\mu = \left(1 + \iota + \sum_{i=1}^{\iota} \lambda_{(b,i)}\right) - 1 = \iota + \sum_{i=1}^{\iota} \lambda_{(b,i)}$$

Under the assumptions in the analysis of the expectation of κ

$$E[\mu] = E\left[\iota + \sum_{i=1}^{\iota} \lambda_{(b,i)}\right] = \iota + \sum_{i=1}^{\iota} E[\lambda_{(b,i)}] = \iota + \sum_{i=1}^{\iota} \sum_{j=1}^{\lambda_i} \frac{j}{\alpha_i} = \iota + \sum_{i=1}^{\iota} \frac{\lambda_i(\lambda_i + 1)}{2\alpha_i}$$

Comparing these four coordination models, we reach the conclusion that the hierarchical coordination model gives the best results for large-scale systems. Because of its scalability property, it minimizes the number of remote messaging by creating groups of agents. Therefore it allows minimal number of messages and migrations for large number of agents.

4.3 Software Simulation and Experimental Results

The software simulation presented shows something that an analysis cannot: time measurements. Thus, the goal of the prototype is to validate the analytical model and to provide insights into the significance of remote communication versus migration in terms of time. Furthermore, relationships between number of agents and different variables can be measured. The software prototype of our electronic marketplace and the experimental results are described in the following paragraphs.

Software Prototype

Our electronic marketplace prototype implements the open-exit English auction (see Section 2.1) with a single-agent approach coordination model (see Section 2.2). The SALSA[22] programming language was used to implement it. The language and system take advantage of the concurrency and migration characteristics of the actor model. Given enough bidding interaction, migration reduces network usage by avoiding remote messaging.

The system is a simplified version of the proposed model. Stores can join the system dynamically. Stores spawn auctions regularly. Buyer agents are created dynamically in the system. An itinerary of stores to visit is created as the buyer agents join the system. Each agent has a time to return. This time is divided evenly among stores in the itinerary. Therefore, buyer agents migrate from one store to the next in their itinerary. Once an agent migrates to a store, it joins relevant auctions. The buyer agents try to fulfill the shopping list of their buyer.

Experimental results

In this section, we will try to obtain the amount of time, the number of messages needed, the number of bids and the price to complete a transaction. We will also evaluate how these variables change when the number of buyer agents increase in different contexts. Three scenarios are as follows:

- **A demand driven world:** The supply of items is not enough to match the demand.
- **A supply driven world:** The supply of items is more than enough to match the demand.
- **A balanced supply and demand world:** The supply and demand of items are balanced.

All of these scenarios are evaluated with fixed characteristics such as budget and time in the system for buyers. A characteristic that is not fixed is the number of stores, which grows with the number of agents to make the bidding process more fair.

The results of testing the prototype are summarized in Figure 3. The upper-left graph plots the behavior of number of messages per transaction as the number of agents increases. The upper-right graph plots the

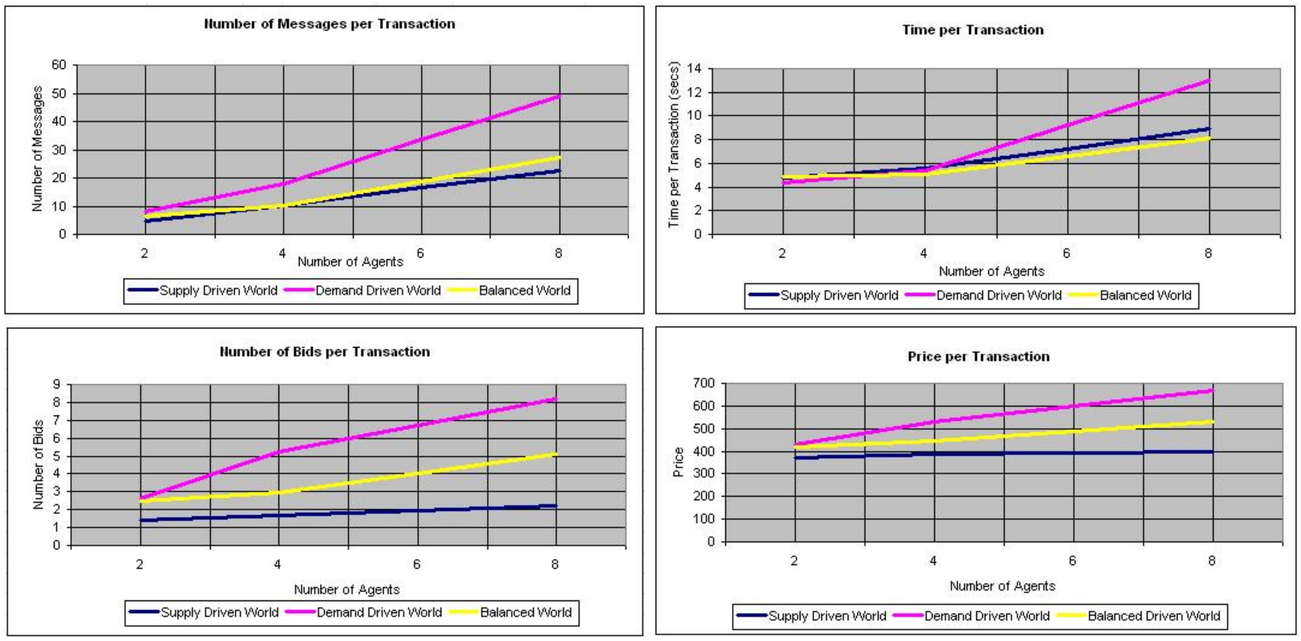

Figure 3: Messages, bids, time and price needed to complete a transaction as the number of agents grow.

behavior of time per transaction as the number of agents increases. The lower-left graph plots the behavior of the number of bids per transaction as the number of agents increases. The lower-right graph plots the behavior of the price as the number of agents increases.

Number of messages and bids per transaction: In a supply driven world, the number of messages slightly increases as the number of agents increases. This is because the probability of competition grows as the system becomes more complex. As probability of competition grows, so does the number of bids and therefore the number of messages. The most critical case is the demand driven world: lots of competition among agents leads to exponential increase in messages. This was predicted in the analytical model of Section 4.1. In a balanced supply-demand world, the number of bids to get an item tends to increase in a linear fashion, and so does the number of messages.

Time spent per transaction: In a supply driven world as more agents join the system, more stores offer the item desired. Therefore time per transaction increases since there are more stores to migrate to in order to fullfill the itinerary of the agent. In a demand driven world the time per transaction increases due to competition. For buyer agent to complete a transaction, the number of bids increases, and so does the time. Likewise in a balanced supply-demand world, the amount of time needed to complete a transaction increases.

Price per transaction: Since the number of stores for this simulation is a function of the number of agents, the price per transaction tends to increase slightly. In a demand driven world, in the case where the number of stores were fixed, the prices per transaction would be a lot higher than the prices obtained.

Migration vs. Remote Messaging

Two approaches to implement auction systems are migration and remote messaging. In the former, the parties of the auction are in the same location, buyer agents migrate to the seller agents location where the auction takes place. In the latter, the bidding process is held through remote messaging. The prototype proposed fits in the first approach while eBay-like systems fit in the second. To be able to come up with time measurements we introduce Table 2.

Table 2 (taken from [22]) shows ranges of values measured with minimal actors and empty messages, as well as larger actors. This values are used to show different aspects of remote comunication and migration. These numbers were taken with a testbed containing three LANs and one WAN. The WAN configuration was

Local actor creation	386μs
Local message sending	148μs
LAN message sending	30-60ms
WAN message sending	2-3 secs
LAN minimal actor migration	150-160ms
LAN 100Kb actor migration	240-250ms
WAN minimal actor migration	3-7secs
WAN 100Kb actor migration	25-30secs

Table 2: Local, LAN and WAN Time Ranges [22]

tested with three machines: one in Europe, one in USA and one in Japan. Local times are in microseconds, LAN times are in milliseconds, and WAN times are in seconds.

In eBay-like systems, the cost of comunication is very expensive. All the messaging is remote. Consider a scenario in which a buyer agent wants to buy two items. The agent is in a supply driven world context. An english auction is being held and the buyer agent is the only one participating. From the simulation results, an average of four and a half messages are sent to complete one transaction. The bidding process will be very simple, since there are no messages generated after outbids. To complete two transactions, there is a need of nine messages.

In an eBay-like system, nine remote messages are all that is needed. In a LAN environment, the time to complete nine messages is 270-540ms. In our prototype, nine local messages are needed and two migrations. In a local environment, the time to complete nine messages is 1332μs, or 1,3ms. The two LAN migrations are completed in 300-320ms. The time to buy two items would be 301,3 to 321,3ms. Therefore in LAN environments the two systems would perform similar as far as time.

The eBay-like systems are widely used in WAN environments. In a WAN environment nine messages will be completed after 18-27 seconds. In our prototype, the messaging would cost the same, but the migration would change. Two migrations in a WAN environment would cost 6-14 seconds. In the worst case of our prototype performance, the time would be less than in the best case of the eBay-like system.

Considering that eBay-like systems are more complex than our example scenario, the migration tool would improve the fairness of the bidding process. The bandwith and latency would no longer be a problem. It is also true that the bidding decision is taken in microseconds. Therefore the system must become autonomous. Human interaction in terms of user preferences will only be needed once at the beginning and once at the end of the process.

5 DISCUSSION AND FUTURE WORK

In this paper, we have considered a futuristic electronic marketplace consisting of software agents representing human buyers, sellers, and auctioneers.

We studied and analyzed different types of auctions in terms of their potential to satisfy buyers, sellers, and/or auctioneers, and in terms of their electronic complexity, that is, number of software agents, cost and patterns of communication, and likelihood of inducing transactions. English auctions favor sellers in terms of profit, particularly on demand-driven scenarios. Reverse auctions favor buyers in terms of budget, particularly on supply-driven scenarios. Sealed auctions have minimum requirements on messaging which favors environments with expensive network communication–latency and bandwidth. Double auctions provide the most fairness to buyers and sellers in terms of exchange prices, and maximizes transactions–a key auctioneer metric, yet incurs in the most communication overhead and can therefore significantly benefit from agent co-locality.

We also studied and analyzed different multi-agent coordination models in terms of computational complexity, and electronic commerce metrics, such as user satisfaction. While a buyer could have a single agent to accomplish its shopping task, thereby requiring no coordination, this strategy diminishes the probability of success getting all the items desired, or getting the best prices. This is particularly true upon fierce competition from other buyers. Therefore, we analyzed the number of agents and messages required by three specific coordination models: tuple spaces, publish and subscribe model, and hierarchical coordination, with the goal of maximizing the probability of user satisfaction.

Finally, we prototyped an actor system to simulate an electronic marketplace, and we provided interesting empirical results that validate the approach of dynamically choosing auction types and buyer agent coordination models.

Further work involves considering empirically obtained actor creation, actor migration, and remote communication times to feed back into the analytical models to help determine the best strategy not just in terms of message passing and number of agents, but also in terms of communication and coordination times on a given network infrastructure. The most informed and best coordinated buyer agent teams should outperform other buyer agents in terms of meaningful electronic commerce metrics such as number of successful transactions and purchase prices as compared to average purchase prices by other (human and otherwise) buyers.

Ultimately, the goal is to create an adaptive electronic marketplace that can successfully complement today's commercial activities and bring in a new perspective to the fair and efficient exchange of goods.

References

[1] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.

[2] G. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. A foundation for actor computation. *Journal of Functional Programming*, 7:1–72, 1997.

[3] Marcos Kawazoe Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar Deepak Chandra. Matching events in a content-based subscription system. In *PODC*, pages 53–61, 1999.

[4] T. Berners-Lee, R. Cailliau, A. Luotenen, H. F. Nielsen, and A. Secret. The World-Wide Web. *Communications of the ACM.*, 37(8), Aug 1994.

[5] C. Callsen and G. Agha. Open Heterogeneous Computing in ActorSpace. *Journal of Parallel and Distributed Computing*, pages 289–300, 1994.

[6] N. Carriero and D. Gelernter. *How to Write Parallel Programs.* MIT Press, 1990.

[7] R.A. Feldman and R. Mehra. Auctions theory and applications. *IMF Staff Papers*, pages 485–511, 1993.

[8] D. Friedman. The double auction market: Institutions, theories, and evidence. In D. Friedman and J. Rust, editors, *Proceedings of the Workshop on Double Auction Markets*, Santa Fe Institute Studies in the Sciences of Complexity, pages 3–25, Cambridge, 1993. Perseus Publishing.

[9] J. Hendler, T. Berners-Lee, and E. Miller. Integrating applications on the semantic web. *Journal of the Institute of Electrical Engineers of Japan*, 122(10):676–680, October 2002.

[10] S.D. Jap. An exploratory study of the introduction of online reverse auctions. *Journal of Marketing*, 67(4):118–122, October 2003.

[11] P. Milgrom. Auctions and bidding: A primer. *Journal of Economic Perspectives*, 3:3–22, 1989.

[12] Associated General Contractors of America. Associated General Contractors of America White Paper on Reverse Auctions for Procurement of Construction. January 2001. www.agc.org/content/public/pdf/Member_Resources/ReverseAuctionWhitePaper.pdf.

[13] David C. Parkes and Lyle H. Ungar. Iterative combinatorial auctions: Theory and practice. In *AAAI/IAAI*, pages 74–81, 2000.

[14] K. Reynolds. Auction types: Dutch, 1996. http://www.agorics.com/Library/Auctions/auction3.html.

[15] K. Reynolds. Auction types: English, 1996. http://www.agorics.com/Library/Auctions/auction2.html.

[16] K. Reynolds. Auctions: First-price, sealed bid; discriminatory, 1996. http://www.agorics.com/Library/Auctions/auction4.html.

[17] K. Reynolds. The double auction, 1996. http://www.agorics.com/Library/Auctions/auction6.html.

[18] H. A. Simon. *The Sciences of the Artificial*, chapter The Architecture of Complexity: Hierarchic Systems. MIT Press, 3rd edition, 1996.

[19] R. Tenorio and R.F. Easley. Bidding strategies in internet yankee auctions: Theory and evidence. *IEEE Parallel and Distributed Technology*, March 1991. http://ideas.repec.org/p/sce/scecf9/1021.html.

[20] C. Varela. *Worldwide Computing with Universal Actors: Linguistic Abstractions for Naming, Migration, and Coordination.* PhD thesis, U. of Illinois at Urbana-Champaign, 2001. http://osl.cs.uiuc.edu/Theses/varela-phd.pdf.

[21] C. Varela and G. Agha. A Hierarchical Model for Coordination of Concurrent Activities. In P. Ciancarini and A. Wolf, editors, *Third International Conference on Coordination Languages and Models (COORDINATION '99)*, LNCS 1594, pages 166–182, Berlin, April 1999. Springer-Verlag. http://osl.cs.uiuc.edu/Papers/Coordination99.ps.

[22] Carlos Varela and Gul Agha. Programming dynamically reconfigurable open systems with SALSA. *ACM SIGPLAN Notices. OOPSLA'2001 Intriguing Technology Track Proceedings*, 36(12):20–34, December 2001. http://www.cs.rpi.edu/~cvarela/oopsla2001.pdf.

[23] M. Vetter and S. Pitsch. An agent-based market supporting multiple auction protocols, 1999.

[24] M. Vetter and S. Pitsch. Towards a flexible trading process over the internet. In Frank Dignum and Carles Sierra, editors, *AgentLink*, volume 1991 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2001.

[25] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, March 1961.