A Web-Based Agent Service Platform for E-Commerce Applications

By

Chi-Wai LEE

Supervised By

Prof. Michael Rung-Tsong LYU & Prof. Irwin Kuo-Chin KING

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Philosophy

in

Computer Science and Engineering

March 2002

A Web-Based Agent Service Platform for E-Commerce Applications

submitted by

Chi-Wai LEE

for the degree of Master of Philosophy

at the Chinese University of Hong Kong

Abstract

Internet-based auction is a profitable, exciting and dynamic part of E-commerce. However, the lack of standard on negotiation protocols between buyers and auctioneers makes the full automation of E-commerce infeasible. Hence, we (1) propose a web-based agent platform for E-commerce which allows humans and software agents to perform automatic auctions over the Internet; (2) develop a set of negotiation protocols based on advanced agent technologies; and (3) design Markov models to evaluate the set of negotiation protocols.

Currently, our platform supports three types of auction including the English, Dutch, and Double auction. The platform architecture is component-based which provides a robust, reliable, scalable and easy to manage environment. Agents act as software components and run in a flexible and dynamic environment for Internet-based auction. They employ the set of negotiation protocols provided by the platform to conduct neogotiations auctomatically with minimal human efforts. People can also use the platform to conduct agent simulations to predict the price trajectory of an item. Simulation results provide a good reference for sellers to choose the best auction closing time so as to maximize their profit.

Moreover, we use Markov chains to model three types of auction including the English, Dutch and Double auction and compute the offering price using some stochastic techniques. Markov chain is a mathematical tool to analyze system behaviour. Steps involve calculation of transition probabilities, computation of the transition probability matrix, and transition of system states. The result is used to compare with the one simulated by agent simulation and to evaluate the negotiation protocols modeled by software agents.

Our experimental results show that (1) the Markov chain can model independently simulated bidder's behaviour accurately, (2) a wider range of bidder's behaviour can be simulated with agents having partial knowledge of other bidders' valuations and the auction closing time. In summary, our agent-based platform can incorporate realistic scenarios for simulation in online auction and other E-commerce applications.

Acknowledgment

First of all, I would like to thank Prof. Michael Lyu, my supervisor, for getting me interested in the area of E-commerce and agent technology, and his consistent and sound critiques of my ideas and work. I am particularly grateful for his enthusiasm, constant support, and encouragement. My research could not have been finished reasonably without insightful advice from him.

I am grateful to Prof. Irwin King for his great help throughout my study and serving on my supervisory committee. My great gratitude goes to Prof. Ho-Fung Leung and Prof. Kam-Wing Ng, who marked my term papers and gave me valuable suggestions. I also want to thank Prof. Belli Fevzi for graciously consenting to be my external marker.

Thanks to all the other I have made at the Chinese University of Hong Kong for making my stay at CUHK an enjoyable period of time.

Finally, I am always indebted to my family, especially my parents and my sister. I would like to thank them for their love. This thesis would not have been possible without all their support and confidence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Agents and E-commerce have recently been a very hot topic in the academic as well as in the commercial and legal arena. Combining these two fields offers lucrative opportunities for conducting E-business transactions. In the following context, we will study the background of E-commerce and discuss the use of software agents to automate a large number of E-business transactions.

1.1 Background and Motivations

With the advent and proliferation of the Internet, E-commerce or E-business have recently been a very hot topic in the academic as well as in the commercial arena. Nowadays, the term E-commerce refers specially to the commercial activities conducted on the Internet [1]. The term E-business is more specific, it refers to the integration of systems, processes, organizations, value chains and entire markets using Internet-based and related technologies. Bidding, auction, and bargaining are examples of E-business applications.

In recent years, companies of all sizes ranging from international corporations to small companies are migrating towards an E-commerce marketplace. Recent statistics show that the electronic market will continue to grow in the near future because the number of potential customers will grow to 90 million by the end of year 2001 [2].

For E-business transactions, customers and sellers often negotiate with each other. In the negotiation stage, two or more parties bargain with each other to determine the price or other terms of the transaction on the Internet. Traditionally, negotiations are conducted with human interactions. However, the order of E-business transaction is very large. To cite an example, eBay [3] made a monthly revenue at $70,000,000 in year 2000 [2]. Hence, it is desirable to carry out this negotiation process either automatically or at least semi-automatically with human interventions only when necessary. Another reason for the automation is the technology push of a growing standardized communication infrastructure which allow separately designed agents belonging to different organizations can interact in an open environment in real time and safely carry out transactions. The third reason is strong application pull for computer support for negotiation at the operative decision making level. Consequently, researchers and practitioners are attracted to develop automated negotiation systems [4, 5, 6]. However, the lack of standard on negotiation strategies and inflexible system design makes it difficult to develop fully automated negotiation systems.

Nevertheless, the concept of software agent provides a new way of analysis, design, and implementation of such complex systems. An agent is a software entity with some of the following characteristics: continuous execution, environmental awareness, agent communication, autonomy, adaptiveness, mobility, and reproduce. An autonomous and intelligent agent is well-suited for many areas, especially for the E-commerce.

1. **Information filtering**. Agents can monitor and retrieve useful information, and do transactions on behalf of their owners or analyze data in the global markets.

2. **Cost reduction**. Agents can try to find the best deal for the customer with less expensive cost, quicker response, and less user effort.

3. **Automatic negotiation**. Agents can make rational decisions for humans, and negotiate the price of the trade with peer agents strategically.

4. **Supply chain management**. Agents can also manage the supply chain network of a company at a low cost.

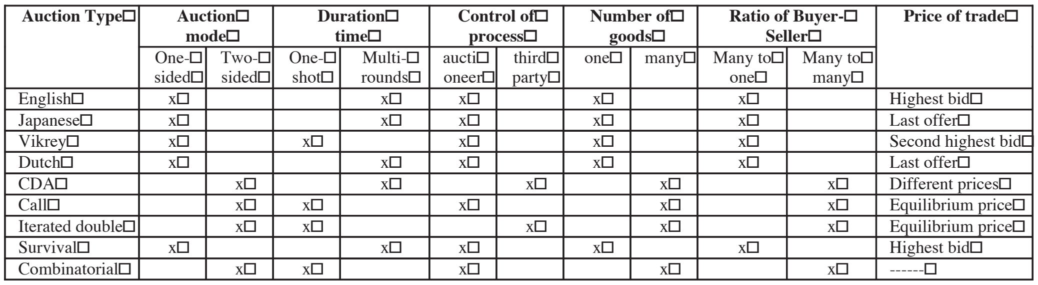
The above automations can save labor time of human negotiators, and in addition, other savings are possible because computational agents can be more effective at finding beneficial short-term contracts than humans.

Since the research potential of developing an agent-based automated negotiation system is very high and the advantages of using such system are prominent, we have designed and implemented a web-based agent platform for conducting negotiations automatically [7, 8, 9]. There are several forms of negotiation such as bidding, auction, and bargaining [10], but not all of them are supported in our system. Bidding is the simplest form of negotiation, in which a buyer specifies the product or service he wants to acquire from suppliers and asks for their bids. Auction consists of an auctioneer and potential bidders. It is usually discussed in situations where the auctioneer wants to sell an item and gets the highest possible payment for it while the bidders want to acquire the item at the lowest possible price. Bargaining is the most complex form of negotiation. It involves issuing proposal and counter-proposal between negotiation parties until a mutual agreement or disagreement is reached.

Our platform employs a set of negotiation strategies in the auction process for the following reasons: (1) an auction is useful when selling an item of undetermined quality, (2) an auction is more flexible than setting a fixed price, (3) an auction can be programmed using software agents with a negotiation strategy and the agents negotiate a solution with the seller automatically, and (4) an auction is an excellent method of distributing goods to those who value them most highly.

There are hundreds of different types of auctions [11]. Their basic types are

English auction (first-price ascending), Japanese auction (second-price ascending), First-price sealed-bid auction, Vickery auction (second-price sealed-bid), and Dutch auction (first-price descending). Some of the variants of basic types of auctions are Continuous double auction (continuous-time double auction), Call auction (discrete-time double auction), Iterated double auction (combines good features of both call auction and continuous double auction), Survival auction (combines the benefits of both sealed-bid auction and ascending-bid auction), and Combinatorial auction. The characteristics of auction types discussed above are summarized in Figure 1.1. This work is done by Leung [11].



Auction Type	Auction mode		Duration time		Control of process		Number of goods		Ratio of Buyer-Seller		Price of trade
	One-sided	Two-sided	One-shot	Multi-rounds	auctioneer	third party	one	many	Many to one	Many to many	
English	x			x	x		x		x		Highest bid
Japanese	x			x	x		x		x		Last offer
Vikrey	x		x		x		x		x		Second highest bid
Dutch	x			x	x		x		x		Last offer
CDA		x		x		x		x		x	Different prices
Call		x	x		x			x		x	Equilibrium price
Iterated double		x	x			x		x		x	Equilibrium price
Survival	x			x	x		x		x		Highest bid
Combinatorial		x	x		x			x		x	------

Figure 1.1: Comparison of different types of auctions.

Currently, the negotiation strategies supported in our auction process are English, Dutch and Double auction. In the English auction, each bidder finds his interested item and read the current price. His valuation of that item is kept private and we assume there are no interactions between bidders. The bidder may decide to leave the auction or submit bid to become the current winner. When no bidder is willing to raise the price further, the auction ends, and the bidder with the highest bid wins the item at the price of his bid. A bidder's strategy is to always bid a small amount more than the current highest bid, and stop when his valuation is reached [12]. In the Dutch auction, the seller continuously lowers the price until one of the bidders takes the item at the current price. A bidder's strategy is to bid for the highest price that he can pay for an item. Dutch auctions are efficient in terms of time because the auctioneer can decrease the price at a brisk pace [12]. The Double auction is similar to the stock market in which both of the buyer and seller can adjust the offering price until a match is found [12].

1.2 Problem Definition

There are many web sites or systems provide auction services over the Internet. The earliest one is Onsale [13], opened in May 1995, and eBay [3], opened in September 1995. They are the first few web sites taking advantage of the web technology, including the use of electronic forms, search engines, and product categories. Other famous auction service providers are Priceline's demand oriented bidding service [14], and Cathy Pacific's sealed-bid auction to sell airline tickets [15]. Nevertheless, these systems are not fully automated negotiation systems and do not allow individual bidder to choose the best negotiation strategy. Our solution is to use software agents to analyze information and respond to changing market conditions. However, many existing agent tool-kits and developing tools cannot support E-business transactions and require software developers with some backgrounds in agent theory. The challenges of developing such a complex system are concerning (1) how to represent, describe, and control systems of agents, (2) how to get them to cooperate effectively, (3) how to design a negotiation strategy for each agent, and (4) how to evaluate the negotiation strategy in terms of social welfare, such as the expected revenue and the profit of the auction.

1.3 Contributions

The main contributions of our work are:

1. **Development of a web-based agent system**

   We develop an automated negotiation system and the set of negotiation strategies [16, 17]. One of the E-commerce applications supported by the system is online auction services. It also provides a set of agent API for agent developers to develop other E-commerce applications. The system is developed using most of the leading software technologies, such as Java/Servlet which is a cross-platform server-side Java program, PHP which is a server-side script language and MySQL which is a Open Source SQL database.

2. **Agent modeling**

   We implement software agents which can analyze information and respond to changing market conditions quickly [17]. Each software agent is provided with negotiation strategies and the system allows it to choose the best strategy to maximize its profit in the auction process. Moreover, each agent can receive user requests and forward execution results to user in an interactive mode.

3. **Agent simulation**

   We simulate a wide range of negotiation behaviours, such as late bidding, jump bid, lying auctioneer, and bidder collusion [17]. Our system allows agent developers to create a pool of agents for auction simulation in a batch mode. Agent developers can monitor the growth of the expected revenue and the profit throughout the whole process which allow them to determine the best closing time of the auction in advance of the actual action.

4. **Revenue prediction using Markov model**

   We design three Markov chain models to model the English, Dutch, and Double auction [17]. Agent can use the model to predict the expected revenue, profit, and the time to reach the maximum profit under a given negotiation strategy. The result can be used as a reference to select the best strategy in the dynamic and evolving auction environment.

Our experimental results show that:

1. the Markov chain can model independently simulated bidder's behaviour accurately, predict the expected revenue of a one item English auction with small errors, allow agent developers to scale up the input parameters to match the result of an actual auction.

2. a wider range of bidder's behaviour can be simulated with agents having partial knowledge of other bidders' valuations to bid and the auction closing time. The partial knowledge is programmed in the agent logic.

   In summary, our agent-based platform can incorporate realistic scenarios for simulation in online auction and other E-commerce applications.

1.4 Thesis Organization

This thesis is organized as follows. Chapter 2 describes: (1) the roles and techniques of agents in E-commerce, (2) the auction formats and existing negotiation strategies, and (3) the principle of Markov chain model for modeling online auction and evaluating the negotiation strategies. Chapter 3 describes using Markov chain as a baseline approach to model the negotiation phase in English auction, Dutch auction, and Double auction. Chapter 4 introduces the multi-agent platform focusing on the agent API, system components, and overall architecture. Chapter 5 discusses the setup of our simulation environment for the experiments and present the experimental investigation and results. Conclusion of the thesis is provided in Chapter 6.

Chapter 2

Literature Review

This chapter introduces software agents and explains the benefits of using agents in E-commerce area. Then, we discuss online auction as one of the E-commerce applications. We will investigate auction formats and existing negotiation strategies. Finally, we will introduce the principle of Markov chain model for evaluating the negotiation strategies.

2.1 Electronic Commerce

E-commerce is a way to do real-time business transactions via Internet. The Internet environment provides a low-cost and convenient way for E-commerce. In fact, E-commerce is a broad concept that includes virtual browsing of goods on sale, selection of goods to buy, and payment method. E-commerce operates via the Internet using all or any combination of technologies designed to exchange data, to access data, and to capture data. There are two business models for E-commerce systems, they are the Business-to-Customer (B2C) model and the Business-to-Business (B2B) model. Most web-based commercial applications are in B2C E-commerce, in which users can navigate a company's web site to conduct transactions. On the other hand, the B2B model focuses on the transactions and electronic market transactions between organizations [18]. The difference between B2C and B2B model is the client category. Most of the clients of B2C E-commerce are small or medium size companies or even end-users while the clients of B2B E-commerce are large enterprises.

2.2 Agents for Electronic Commerce

Software agents are well suited for developing E-commerce systems. The relationship between agent and E-commerce can be found easily from agent's characteristics. From the end-user perspective, an agent is a program that assists people and acts on their behalf. It may possess some of the following characteristics [19, 20, 11]:

1. **Continuous execution**. Agents run continuously in achieving their design goals which is given by their owner. Their lifetime is determined by owners. Agents can remain alive in their execution environment after completing their design goals until they are terminated explicitly by owners.

2. **Environmental awareness**. Agents have the ability to interact with their execution environment and to act asynchronously and autonomously. They can sense changes in the environment and act according to those changes.

3. **Agent communication**. Agents may interact with peer agents via communication/coordination protocols in order to share their knowledges with others and achieve their design goals.

4. **Autonomy**. Agents have control over their own actions and require minimum human efforts. They are able to work independently without direct human interventions.

5. **Adaptiveness**. Agents may adapt their behaviours in accordance with their experiences to suit the preferences and interests of various users.

6. **Mobility**. Agents may travel from one host to another. The ability to travel allows agents move to a system that contains an object with which the agents want to interact.

7. **Reproduce**. Agents may be able to reproduce themselves and cooperate with each other to share the workload until they achieve their design goals.

According to a survey done by Leung [11], he categorized agent-based E-commerce applications into four categories. (1) Agents can monitor and retrieve useful information, and perform transactions on behalf of their owners or analyze data in the global markets [21]. For instance, some agent-mediated E-commerce systems aim to help users find the products they need, e.g., Jango [22], PersonaLogic [23], and Firefly [24]; (2) agents can try to find the best deal for the customer with less expensive cost, quicker response, and less user effort. For instance, some systems provide the ability to choose a supplier or a merchant based on some selection criteria, such as BargainFinder [25], Jango and Kasbah [26]; (3) agents can make rational decisions for humans, and negotiate the price of the trade with peer agents strategically. For instance, there are some E-commerce web sites for online auction, e.g. eBay [3], PriceLine [14], and Cathy Pacific [15], focusing on price negotiation for both sellers and buyers using various negotiation strategies; (4) agents can also manage the supply chain network of a company at a low cost [27].

One of the objectives of our agent system is to provide a flexible, stable, scalable and dynamic environment for online auction which are not supported by current negotiation web sites.

2.3 Online Auctions

Auctions on the Internet have become a fascinating new type of exchange mechanism. Every day hundreds of thousands of different auctions take place online, for goods ranging from Star Wars action figures to laboratory ventilation hoods. Internet technology has lowered the costs of organizing an auction. Currently, online auctions trade billions of dollars' worth of goods per year, and are growing at a rate of more than 10% per month [2].

Online auction technology provides several benefits relative to traditional auctions: (1) Internet auctions give bidders increased convenience, both geographic and temporal, relative to traditional auctions. Instead of having to come to an auction house to participate fully in the bidding, the bidder can stay at his home or office; (2) traditional auctions require all bidders to participate at the same time, tying each bidder up for the entire length of the auction, but Internet auctions typically have asynchronous bidding lasting days or weeks, giving bidders much more flexibility about when to submit bids; (3) on the Internet, one can easily obtain a relatively large group of bidders on short notice, rather than scheduling an auction a month in advance and being restricted to local bidders who could travel to the auction at the scheduled time; (4) furthermore, search engines and clickable hierarchies of categories for browsing make it more convenient for a bidder to find the goods she's looking for.

Internet auctions also have their disadvantages over traditional auctions: (1) it is difficult for bidders to inspect the goods before bidding in an online auction. Online auctioneers try to solve this problem by using the following methods: sellers can post electronic images of their items, provide large amount of text descriptions, and answer bidder's question via email; (2) another difficulty with online auctions is the potential problem of fraud. When the auction takes place in person, money can be exchanged for the good before the winning bidder leaves the room. By contrast, the winner of online auction must trust that the seller will send the good in return for the payment of the bid amount. Indeed, there have been a number of cases of fraud reported by bidders in online auctions. However, the amount of fraud is tiny compared with the number of transactions which take place.

The second disadvantage gives some difficulties to agent developers in modeling an online auction with software agents. The honesty of an agent's owner cannot be guaranteed by his agent. In other words, an buyer agent is not able to give security to another buyer agent about the time of payment. Some of the online auction providers discourage fraud by encouraging users to send their complaints to courts. Until now, we cannot find any good agent modeling methods to detect fraud and solve it by the agent itself.

Besides, frauds can also be happened in the agent's side. Some personal information such as credit card number and payment details are stored in the agent during the auction. Agent developers should guarantee his agents will not disclose its owner information to other agents. Agent should be able to refuse any communications come from untrusted peers.

2.3.1 Business Models

The two primary business models for online auctions are merchant sites and listing-agent sites. A merchant site, such as Onsale [28], offers its own merchandise for sale, acting as a retailer who happens to conduct its transactions through auction. A listing site, such as eBay [3], acts as an agent for other sellers, allowing them to register their items and running the auctions on their behalf. The listing site usually neither possesses the auctioned goods nor handles their payment and shipment; all the transaction details are worked out by the buyer and seller on their own.

2.3.2 Auction Formats

Among the merchant or listing-agent sites, we can find several different auction formats: English, Dutch, sealed-bid and Double auction [29]. The English ascending-bid auction is the most familiar type of auction in online auction. According to a survey done in August 1999 [2], there are 121 out of 142 sites used English ascending-price auctions, 21 used sealed bids, 3 used Dutch descending price rules, and 4 were continuous double auction.

English auction. Ascending-bid auctions are the most prevalent auction format in the Internet, and they make bidder participation relatively easy [30]. Once the bidder finds the item he is interested in, he can read the current high bid, and decide whether to raise it by filling out his own bid amount in a text box in his web browser. After submitting his bid, he will see an automatic update of the auction status, showing whether he can become the current winner. He can leave the site as the winner bidder, and return at any time before the close of the auction to check on its status again. Most of the large auction sites make it easy for bidders to return to their auctions of interest. Moreover, most of these sites provided automated out-bid notification email messages to let bidders know instantly when they are no longer the current winner in an auction [31].

Sealed-bid auction. In Sealed-bid auction, bidders submit a single, irrevocable sealed bid. The bids are opened simultaneously, and the winner is the highest bidder, who claims the item at the price he bid. Usually, the offering price of sealed-bid auction is lower than the English auction because bidders are less aggressive if they do not know the bidding price of other bidders. Therefore, it is not widely used compared with the English auction. The advantage of Sealed-bid auction is able to assure the privacy of bidding price.

Dutch auction. In Dutch auction, the price starts at some relatively high level and continues until the first bid determines the winner. One potential disadvantage to Dutch auction is that the closing time of the auction is too short which do not allow asynchronous bidding [32]. This problem is not solved by my agent system.

Double auction. In Double auction, it allows continuous update of offers by sellers as well as bids by buyers. Buyers and sellers post buy and sell offers, and where they are allowed to strike a deal at any time. Entry and exit of agents are allowed such that buyers and sellers come to and go out of the auction at any time. A transaction is consummated as soon as a buy offer and a sell offer cross [33]. The problem of Double auction is the creation of large market size. Usually, there are hundreds of buyer and seller in an on-going Double auction. The complexity of finding the best offering price is large and it takes time to converge the offering price to an equilibrium value.

2.3.3 Time Duration of Auctions

In online auctions, sellers often specify the closing time of the auction in advance [34]. Choosing the best closing time of the auction can maximize the profit gained by the seller. Therefore, the seller should consider this time factor seriously at the beginning of the auction. In a survey conducted in August 1999 [2], the author found that most listing sites give sellers the opportunity to choose their own auction length. To cite an example, sellers can choose a length of 3, 5, 7, or 10 days for their auctions in eBay [3]. The mean length of auctions at the different sites in that survey was 9.3 days, with a modal length of 7 days. The very short auction took place at merchant sites, most of the merchant sites offer auctions in minutes basis. Some example are Onsale's 60-minute "express auctions" and the First Auction's 3-minute "flash auctions". Time duration is an important factor to affect the profit. If a seller specify a short auction closing time, he may lost his profit because most of the interested bidders have not submitted their bids. On the other hand, a long auction closing time cannot increase the offering price too much because bidders usually become more conservative to bid towards the end of the auction.

2.3.4 Minimum Bids and Reserve Prices

Online auctions usually specify a minimum acceptable bid amount, below which no bids will be accepted. On listing-agent sites, an individual seller will chooses this as a parameter in the beginning of the auction. In addition, many sites also feature a secret reserve price, specified in advance but not revealed to the bidders until the end of the auction. If the highest bid does not exceed the amount of the reserve price, the good will not be sold.

The effect of reserve price is prominent in online auction [35]. Reserve price can be useful to the seller in the following way. The apparent model is that starting out the bidding at a falsely low minimum bid might generate interest and build bidding momentum, sending the bidding price up to the reserve price. If the setting of reserve price is too low, the bidding price might not go up.

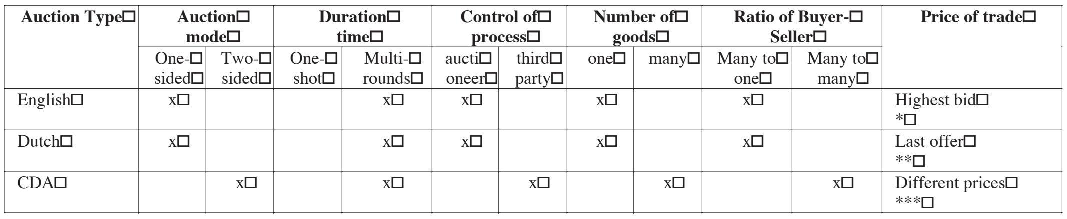
In addition, to the extent that bidding requires costly effort, reserve prices might drive away bidders. There are some discussions of the reserve price feature states, "Most buyers do not like reserve price auctions and will avoid them at all costs. It is very upsetting to win an item only to be told that your winning bid was not high enough. Overuse of high reserve prices will force people to bid on other items.". Hence, it is hard to determine a suitable reserve price for an auction.

2.3.5 Auction Properties and Bidder Behaviours

In online auctions, there are two terminologies to describe the bidding strategies. Common value auction means that when a person wants to buy an item for personal consumption, the motivation of bid submission is also determined by the valuation of prospective bidders. It assumes bidders know the valuations of the others. On the other hand, private value auction means that a bidder is motivated to pay up to his valuation, independent of valuations made by bidders. A private valuation is a subjective decision. It is private in that one bidder does not know another's value. Besides, bidder's behaviours can be classified into two categories. Risk averse means bidders likely to raise their bids so that they are more likely to win. Risk neutral means bidders will not raise their bids. If the bid is larger than their valuation, they will stop bidding.

2.3.6 Comparison of English, Dutch and Double auction

The characteristics of English, Dutch and Double auction are summarized in Figure 2.1. The objective of this comparison is to understand their features which allow us to model their negotiation phase with software agents.



Auction Type	**Auction mode**		**Duration time**		**Control of process**		**Number of goods**		**Ratio of Buyer-Seller**		**Price of trade**
	One-sided	Two-sided	One-shot	Multi-rounds	auctioneer	third party	one	many	Many to one	Many to many	
English	x			x	x		x		x		Highest bid *
Dutch	x			x	x		x		x		Last offer **
CDA		x		x		x		x		x	Different prices ***

Figure 2.1: Comparison of English, Dutch and Double auction.

The English and Dutch auction are very similar in principle. Both of them are one-sided model in which the auctioneer controls the negotiation phase. They used to sell a single item and the ratio of buyer to seller is many to one. Moreover, they have a pre-defined auction closing time and a pre-defined minimum bid value. There is only one big difference between them. It is the selling price trajectory. The selling price of an English auction grows from a lowest to a highest value. In contrast, the selling price of a Dutch auction drops from a highest to a lowest value. In summary, modeling the negotiation phase of an English or a Dutch auction by agents is straight forward.

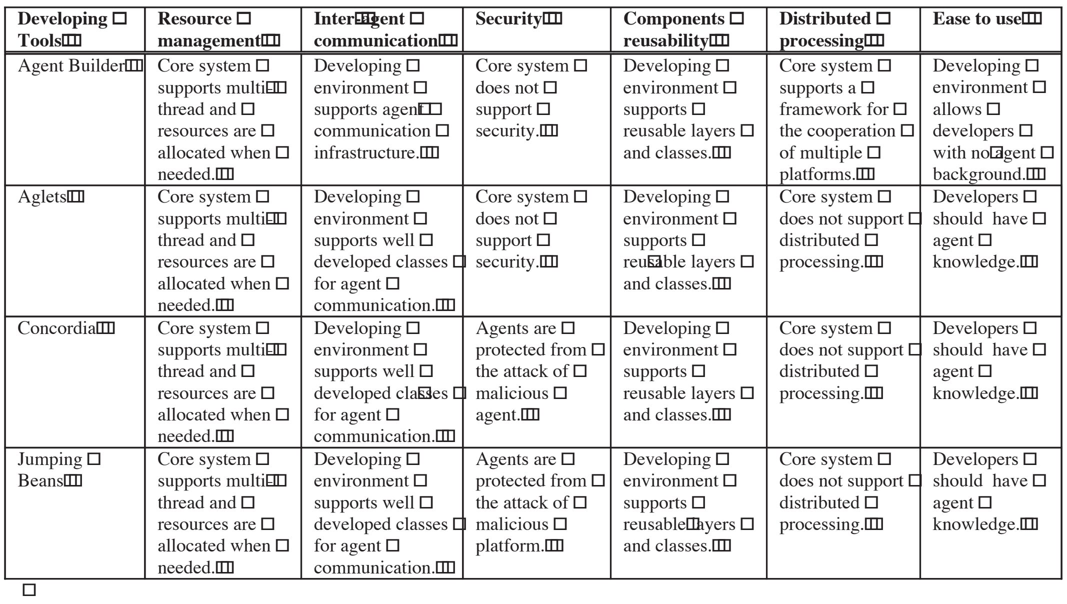
Modeling a double auction is somewhat more difficult than the previous auction types. Double auction is two-sided driven, i.e. buyer and seller can offer shout prices to each other until a match is found. The match is determined by the current value of shout prices. Usually, the highest shout price offered by a buyer will match the lowest shout price offered by a seller. Moreover, the ratio of buyer to seller is many to many. Therefore, there are more than one possibility to reach the equilibrium price. Software agent should be able to find out the best plan which complicates its logic. Hence, modeling Double auction with software agents is a complex and difficult process.

2.4 Agents Developing Tools and Tool-kits

Some agent tool-kits and developing tools have already been applied to facilitate the constructing software agents and agent systems for E-commerce applications. However, they have different strength and weakness. Their weakness makes them unable to support a high loading and secure web-based working environment. We will compare them in terms of resource management, inter-agent communication, security, components reusability, distributed processing, and ease to use.

Figure 2.2 and Figure 2.3 show the comparison of existing agent developing tools including Agent Builder, Agent Tcl [36], Tacoma [37], Aglets [38], Concordia, and Jumping Beans.

There are two main categories of agent developing tools available in the market. They are Java-based and non Java-based. Java-based developing tools require agent developers use Java to implement their agents and the core system is written in Java. Usually, Java-based developing tools have



Developing Tools	**Resource management**	**Inter-agent communication**	**Security**	**Components reusability**	**Distributed processing**	**Ease to use**
Agent Builder	Core system supports multi-thread and resources are allocated when needed.	Developing environment supports agent communication infrastructure.	Core system does not support security.	Developing environment supports reusable layers and classes.	Core system supports a framework for the cooperation of multiple platforms.	Developing environment allows developers with no agent background.
Aglets	Core system supports multi-thread and resources are allocated when needed.	Developing environment supports well developed classes for agent communication.	Core system does not support security.	Developing environment supports reusable layers and classes.	Core system does not support distributed processing.	Developers should have agent knowledge.
Concordia	Core system supports multi-thread and resources are allocated when needed.	Developing environment supports well developed classes for agent communication.	Agents are protected from the attack of malicious agent.	Developing environment supports reusable layers and classes.	Core system does not support distributed processing.	Developers should have agent knowledge.
Jumping Beans	Core system supports multi-thread and resources are allocated when needed.	Developing environment supports well developed classes for agent communication.	Agents are protected from the attack of malicious platform.	Developing environment supports reusable layers and classes.	Core system does not support distributed processing.	Developers should have agent knowledge.

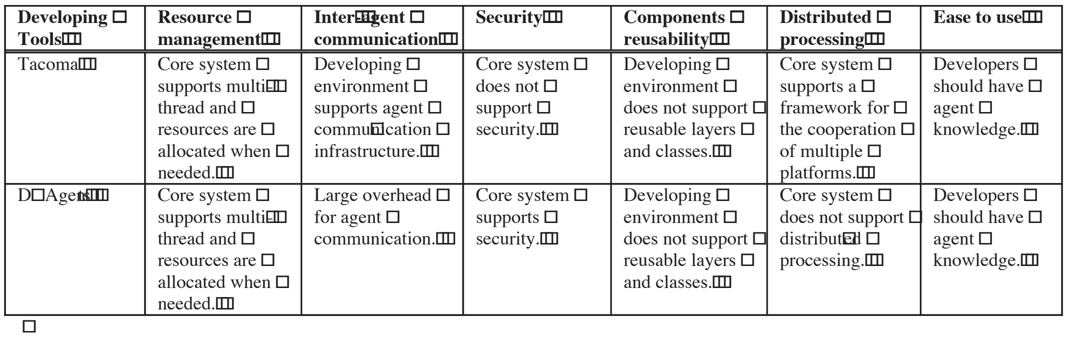
Figure 2.2: Comparison of Java-based agent developing tools.

good support in resource management and component reusability. Non Java-based developing tools allow agent developers use C, C++ or some scripting languages like Perl to implement their agents. However, non Java-based developing tools usually weak at component reusability and does not have enough security infrastructure to protect agents.

2.5 Limitations of Agent Tool-kits

Nonetheless, these agent developing tools have the following limitations.

- **Scalability**. Some tool-kits like Agent Builder takes intensive system resources. From our observations, only four to five agents can be executed concurrently. If we try to create agents over the limit, we found that the whole tool-kits will be messed and system crashes. In web-based environment, scalability is an important considerations. Usually, there are over a hundred concurrent human users. If each of them tries to create



Developing Tools	Resource management	Inter-agent communication	Security	Components reusability	Distributed processing	Ease to use
Tacoma	Core system supports multi-thread and resources are allocated when needed.	Developing environment supports agent communication infrastructure.	Core system does not support security.	Developing environment does not support reusable layers and classes.	Core system supports a framework for the cooperation of multiple platforms.	Developers should have agent knowledge.
D□Agents	Core system supports multi-thread and resources are allocated when needed.	Large overhead for agent communication.	Core system supports security.	Developing environment does not support reusable layers and classes.	Core system does not support distributed processing.	Developers should have agent knowledge.

Figure 2.3: Comparison of non Java-based agent developing tools.

one agent, the total number of agents will exceed one hundred. The above agent tool-kits cannot provide a stable and scalable environment to fulfill the high user-demand condition.

- **Feasibility**. Other than scalability in web-based environment, feasibility is also a major factor. Current agent tool-kits only allow developers to create agents inside its agent console. If the agent applications is designed for web-based, we cannot create agents simply using a HTTP request. There are some solutions to solve the above problem. One way is to install a Java Virtual Environment in the server side to forward a HTTP request to the agent tool-kits. On the other hand, the reply can be forwarded from agent tool-kits back to a HTTP reply. However, this solution is too indirect making a bad performance and developers needs to understand the flow of the system.

- **Flexibility**. Current agent tool-kits impose less flexibility on agent-interface. In web-based environment, the agent interface must be as user-friendly as possible. However, we cannot create an interface using current tool-kits. Current agent tool-kits also do not understand data in the format of HTML forms. Hence, they are not flexible enough to create a good web-based interface.

- **Security**. Security is the last but also an important factor. Current agent tool-kits do not provide any secure channel between the agent and the human user. If we employ them in the web-based environment, this will create many security problems, like stealing credit card number and password in the HTML form.

To conclude, current agent tool-kits are not suitable for web-based agent applications. The need of developing a web-based agent platform to lessen the above limitations is prominent. Hence, it motivates us to develop a complete web-based agent platform to support E-commerce applications.

2.6 Markov Processes for Stochastic Modeling

In this section, we will provide the basic principles of Markov chain for stochastic modeling. They will be used in Chapter 3 in which we try to model an online auction using Markov chain and use the Markov chain to predict the price trajectory. The result will be extremely useful for both seller and buyer to maximize his profit in the auction.

Markov chain is a mathematical tool to analyze system behaviour. It have been used in statistical physics for many years, their usefulness for general statistical modeling has been appreciated recently. The literature on Markov chain methodology and its applications is scattered and rapidly expanding. In the E-commerce area, Markov chain can be applied in supply chain management [39] and stock market simulations [40]. The steps of using Markov chain for stochastic modeling involves calculation of transition probabilities, computation of the transition probability matrix, and transition of system states.

There are other tools which can be used to predict market conditions of E-commerce applications. The most famous one is the neural network. Some previous works used neural networks to do stock market prediction [41, 42, 43]. Their goal is to do stock price pattern recognition and use the result to forecast the behaviour on the financial market. Usually, the result of using a neural network to do market prediction is good but the correctness depends heavily on training data. The major drawback of using neural network is the long converging and training time. However, the Markov model does not require any training data set and the converging time is shorter. However, we cannot conclude that Markov model is better than neural network in terms of prediction. The reason is that Markov model may simplify a real scenario and make too much assumptions. It can only be applied in modeling simple system behaviour. Figure 2.4 shows the comparison between them.

	Neural Network	**Markov Model**
Training Data Set	Need	No need
Converging Time	Long	Short
Accuracy	Good. It allows complex system behaviours and requires fewer assumptions.	Undetermined. It allows simple system behaviours and requires too many assumptions.

Figure 2.4: Comparison of Neural Network and Markov Chain.

2.6.1 Continuous Time Markov Chain

First, we will define Markov chain in continuous time with discrete state space. A Markov chain is a sequence $X_0, X_1, ..., X_n$ of discrete random variables with the property that the conditional distribution of X_{n+1} given $X_0, X_1, ..., X_{n-1}$; depends only on the value of X_n but not further on $X_0, X_1, ..., X_{n-1}$; i.e. for any set of values $h, j, ..., k$ belonging to the discrete state space,

$$prob(X_{n+1} = k | X_0 = h, ..., X_n = j) \quad = \quad prob(X_{n+1} = k | X_n = j). \qquad (2.1)$$

The continuous time Markov chain differs from the discrete time in the sense that transitions occurring between times n and $n+1$ being replaced by that of state transitions occurring in a short time interval $(t, t + \Delta t)$.

2.6.2 Poisson Arrival Model

Markov chains are often best described by diagrams, an example of continuous time Markov chain with Poisson arrival is shown in Figure 2.5. Poisson arrival is usually used in Markov chain modeling to simulate the arrival rate of events, e.g. customer arriving to a bank etc. Althrough it may not be suitable for most of the modeling examples, it provides a baseline to analyze system behaviours.

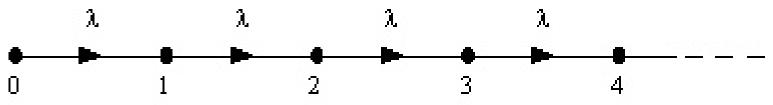


Figure 2.5: Poisson arrival model.

In figure 2.5, λ is the average number of arrivals per unit time. It is also called the arrival rate. There are four units of time in the figure. Each of them is a length of time t and we assume the arrival rates at different time intervals are constant.

The properties of Poisson arrival are:

- For a length of time t, the probability of n arrivals is $P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}$.
- For two non-overlapping intervals, the number of arrivals in each of the interval is independent.
- The inter-arrival times between successive arrivals are independent of each other.

2.6.3 Example: discrete time Markov chain with stationary probabilities

Next, let us consider some details of Markov chain with two states. This is the simplest non-trivial state space in which we will use it to model an online auction. We may regard one of the states as "success", denoted by 1 and the other as "failure", denoted by 0. Therefore, we have an example of dependent Bernoulli trials in which the probability of success or failure at each trial depends on the outcome of the previous trial.

Suppose that if the nth trial results in failure, the probability of failure at the $(n+1)$th trial is $1-\alpha$ and the probability of success at the $(n+1)$th trial is α. Similarly if the nth trial results in success, there are probabilities $1-\beta$ and β of success and failure respectively at the $(n+1)$th trial. Alternatively we may say that if the system is in state 0 at time n, there is a probability $1-\alpha$ of being in state 0 at time $n+1$ and a probability α of being in state 1 at time $n+1$. Similarly if the system is in state 1 at time n, the probabilities of being in state 1, state 0 at time $n+1$ are $1-\beta$, β respectively. These probabilities are called transition probabilities and we may write them in the matrix form as shown in Equation 2.2.

$$\mathbf{P} = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}. \tag{2.2}$$

The matrix element in position (j,k) denotes the conditional probability of a transition to state k at time $n+1$ given that the system is in state j at time n. Note that we are making the assumption that the transition probabilities are independent of time. Also, we exclude the trivial cases (i) $\alpha+\beta=0$, i.e. $\alpha=0, \beta=0$; in this case the system remains forever in its initial state; (ii) $\alpha+\beta=2$, i.e. $\alpha=1, \beta=1$; in this case the system alternates deterministically between the two states.

Now we can let a row vector $\mathbf{p^{(n)}} = (p_0^{(n)}, p_1^{(n)})$ denote the probabilities of finding the system in states 0 or 1 at time n when the initial probabilities of the two states are given by $\mathbf{p^{(0)}} = (p_0^{(0)}, p_1^{(0)})$. Consider the event of being in state 0 at time n. This event can occur in two mutually exclusive ways; either state 0 was occupied at time $n-1$ and no transition out of state 0 occurred at time n; this has probability $p_0^{(n-1)}(1-\alpha)$. Alternatively state 1 was occupied at time $n-1$ and a transition from state 1 to state 0 occurred at time n; this has probability $p_1^{(n-1)}\beta$. Considerations such as these lead to the following recurrence relations,

$$p_0^{(n)} = p_0^{(n-1)}(1-\alpha) + p_1^{(n-1)}\beta, \tag{2.3}$$

$$p_0^{(n)} = p_0^{(n-1)}\alpha + p_1^{(n-1)}(1-\beta), \tag{2.4}$$

which in matrix notation may be compactly written

$$\mathbf{p^{(n)}} = \mathbf{p^{(n-1)}P}, \tag{2.5}$$

and on iteration

$$\begin{aligned} \mathbf{p^{(n)}} &= \mathbf{p^{(n-2)}P^2} && (2.6)\\ &= \ldots && (2.7)\\ &= \mathbf{p^{(0)}P^n}. && (2.8) \end{aligned}$$

Thus, given the initial probabilities $\mathbf{p^{(0)}}$ and the matrix of transition probabilities $\mathbf{P}$, we can find the state occupation probabilities at any time n.

In our work, the row vector $\mathbf{p^{(n)}}$ is the auction state, we can use similar technique to find out the transition probabilities and compute the matrix $\mathbf{P}$. Then we can find out the auction state by Equation 2.6. From the auction state vector, we can find out the selling price of an item because we try to formulate the state as a function of selling price and number of bidders. For details, please read Chapter 3.

Chapter 3

Markov Chain Model

This chapter develops three Markov chain models for three types of auction including English, Dutch, and Double using stochastic modeling techniques. The objectives of this work are:

1. Determine the best auction closing time for the seller agent in English and Dutch auction so that its owner can maximize his/her profit.

2. Predict the selling price for English, Dutch, and Double auction.

3. Compare the predicted revenue with the actual auction data as a baseline approach to measure the correctness of the bidders's behaviour.

3.1 Markov Chain for English Auction

In this section, we will introduce the waiting queue model which is used to predict the revenue trajectory and the final selling price of English auction under some assumptions. The waiting queue model is shown in Figure 3.1. It is a system state diagram which captures the sequence of events in English auction. State transition is triggered by bidder's actions.

The sequence of events in the waiting queue model is described as follows:

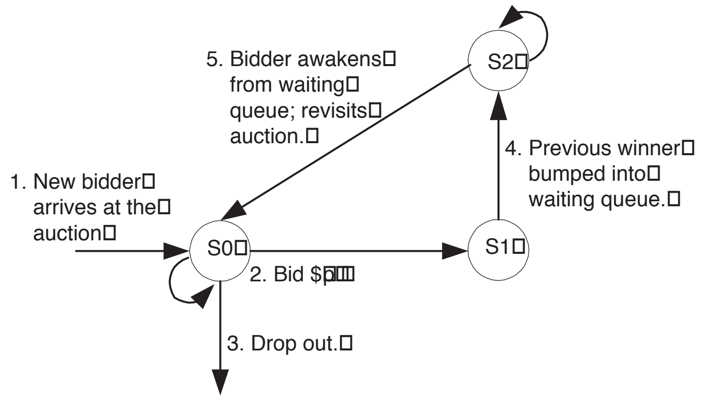

5. Bidder awakens from waiting queue; revisits auction.
S2
4. Previous winner bumped into waiting queue.
1. New bidder arrives at the auction
S0
S1
2. Bid $p
3. Drop out.

Figure 3.1: The waiting queue model.

1. The system state S0 means that new bidder arrives at the auction site. The system state remains in S0 unless a new bidder submit a bid or drop out.

2. New bidder reads the current price. He/She must offer a going price of $\$p$ in order to bid successfully. If this is the case, system state proceeds from S0 to S1. The system state S1 means that new bidder bids successfully, the auctioneer registers he/she as a potential winner and updates the going price to $\$p + \$c$ where $\$c$ is the minimum bid increment.

3. If the bidder does not want to pay $\$p$, he/she drops out of the auction.

4. The previous potential winner is bumped to the waiting queue to join any others there. The waiting queue contains all previous potential winners who have been bumped. They wait in the queue until awaken. System state proceeds from S1 to S2. The system state S2 means that the number of bidders in the waiting queue is increased by 1.

5. A previous potential winner awakens from the waiting queue and visits the auction again, reading the new going price. The decision is the same as the arrival of a new bidder. System state proceeds from S2 to S0.

3.1.1 Variables

In this section, we will introduce variables for modeling the English auction.

Variable	Definition
A	A is the English auction.
N	number of identical items for sale.
b_i	b_i is the i^{th} bidder arrives at the auction.
r	reserve price of an item.
c	minimum bid increment of the auction.
t	index of events which is a finite positive integer.
$p(t)$	going price of the auction at time t, i.e., this is the bid required to become the potential winner.
λ	arrival rate of new bidders.
μ	departure rate at which each individual bidder awakens from the waiting queue and revisits the auction.
Q	a set of bidders in the waiting queue.
L	number of bidders in the waiting queue.
L_{max}	maximum value of L computed by the current going price.
v_i	valuation of the i^{th} bidder.
$F(p)$	cumulative distribution function(cdf), the output is the probability to submit bid at current going price.
$\overline{F}(p)$	$1 - F(p)$

Table 3.1: Variables for the English Markov model.

3.1.2 Auction Assumptions

We make the following assumptions about the auction. They are well-validated with actual auction data.

1. The auction opens at a pre-set and advertised time.

   Validation: This is how the Internet English auctions are run. The auction opening time is determined by the seller. A seller should register his item to the auctioneer and let the auctioneer post the details on the web. Hence, all interested bidders know the exact opening time. This assumption does not have any effects on the result.

2. The auction closes at a pre-set, advertised time which does not depend on the auction activity.

   Validation: This is an assumption made for the Markov model. The actual English auction has a going phase in which bids after the posted closing time are sometimes accepted. However, we are trying to model an Internet English auction in which the common practice is to let the seller to specify the auction closing time before the auction starts. To cite an example, eBay [3] allows the seller to set the maximum duration time to seven days. After the deadline, no more bids are allowed to submit.

3. The auction sells a single unit of item; the current going price is provided for making a bidding decision at the time of the bid.

   Validation: This is how the Internet English auctions are run and we provide the current going price to all bidders. We assume bidders are risk-neutral and they will not change their valuation in the auction. Bidder's behaviour will be discussed in more details in the next section. They should be able to decide whether to submit bid or not by comparing the current going price with their valuations.

4. The auction does not advertise future auctions or items, nor does it contain any information about the transaction history.

   Validation: This is an assumption made for the model. Onsale [28] tries to create a **scarcity mentality**, in which it tries to make bidders feel as if this is their last chance to purchase the item, but many bidders realize that some items will be offered again in the future. If that is the case, bidders may not show interests in the current chance and therefore decrease the probability to bid and affect the selling price. Besides, each bidder will not consider previous transaction prices before submitting his bids. Their decision is based on the current going price only. For example, if the current going price is greater than his valuation, he will not submit a new bid. This is common in Internet auction. Usually, Internet auction providers provide the current price, description of the item and seller's name to buyers only. Any buyers do not know who submit bids and their bid value before they make their bidding decision.

5. The auction sends an email to bidders immediately to (1) confirm placement of a bid, (2) notify the bidder when a bid has been bumped out of contention, and (3) notify the bidder when an auction has closed, and whether this bidder won.

   Validation: This is how the actual English auctions are run. If we cannot guarantee bidders receive reply immediately, they may lost interests in submitting new bids which reduce the final selling price of an item; thus reducing the expected revenue.

3.1.3 Bidder Assumptions

The following assumptions are made about the bidders and their behaviour. They are well-validated with actual auction data.

1. No bidders purchased the same item before.

   Validation: This is a common behaviour of most bidders. About 75% [3] of all bids comes from a bidder bidding in only that auction for only that item. Besides, bidders do not continue to bid for a particular item after they have won it. Only 7% [3] of bidders continue to bid for the same item, and fewer than half of them $\sim 3\%$ [3] win the same item again.

2. There is no collusion among bidders.

   Validation: This is an assumption made for the model. Since bidders cannot see each other's name and do not have access to email addresses or phone number, collusion would be difficult and their valuations is kept private. This kind of setting is also called private value English auction in which the value of the good depends only on the bidder's own preferences [12]. If we allow bidders interact with each other, they can take many actions to keep the value of the current transaction price which violates the motivations of English auction.

3. The i^{th} bidder has valuation v_i for the item. The v_i for all bidders is independent and $F(p)$ follows an uniform distribution. The bidders do not update their valuations in the auction.

   Validation: This is an assumption made for the model. Our model is not general enough for bidding all kinds of items. We can only apply it to bid items with a fixed price and all risk-neutral bidders know the fixed price before bidding. For example, the price of a palm handheld is $US\$449$ which is pre-set by its manufacturer. It is also shared to all buyers because buyers can know the price by reading advertisment given by the manufacturer. This means all bidders have the same v_i. Risk-neutral bidders guaratee that their probability to submit bids at different transaction price is similar and we can assume that it follows an uniform distribution.

4. If a bidder wants to become the potential winner, the bidder will place a bid in the amount of $\$p$, and bump the previous potential winner into the waiting queue. The bidder will not make several bid increments at once.

   Validation: This is a common behaviour of most bidders. Sometimes there are jump bidders who will make several bid increment, but we ignore the jump bid phenomenon.

5. If the bidder thinks the going price is too expensive ($v_i < p$), the bidder will drop out of this auction forever.

   Validation: If we assume that our model is only dedicated to bidding items with a fixed price and the fixed price is shared by all bidders, this assumption holds because bidders do not have to bid for that item when $p > v_i$. They can simply buy the same item from the manufacturer.

6. After making a successful bid, the bidder goes off to do other things. From the auctioneer's point of view: (1) He/She will awaken independently and randomly after a period of time and re-check the auction status; (2) If the bidder re-visits while the auction is still going on, he/she will behave as described in assumption 6 above; and (3) If the bidder re-visits after the auction has closed, he/she loses his chance to win the auction.
   Validation: This is an assumption made for the model, but it is quite obvious because bidders will receive email to notify him/her about the current status of the auction when an event happened.

3.1.4 Propositions of the Markov Model

The following section presents the Markov chain model for modeling English auction.

Without loss of generality, we set $\$c = \$1$, using a \$1 bid increment. Furthermore, we define an event as the arrival of a bidder at the auction, no matter whether the bidder came as a new arrival or the bidder was awakened from the waiting queue. To count as an event, the bidder needs only to arrive; he/she does not need to decide to bid.

We use the time index t to measure the number of events which have taken place, not as a measure of the exact time of an event.

Proposition 1: New bidders arrive at the auction in accordance with a Poisson arrival model with parameter λ.
Define $B(t)$ as the number of bidders who have arrived by time t. $B(t)$ is a Poisson arrival model because:

- $B(0) = 0$.
- The probability of two events happening at exactly the same time is zero. The server cannot process two arrivals simultaneously, so even two arrivals occur at the same time, they would be queued in an arbitrary order at the server and processed one after the other.
- The number of arrival at different time intervals is dependent.
- The inter-arrival time between successive arrivals are independent and it equals to $\frac{1}{\lambda}$.

Proposition 2: Bidders awaken from the waiting queue in accordance with a Poisson arrival model with parameter μL.

- The arguments are similar to the previous proposition.

Proposition 3: The state of an one-item auction can be described by $A_t(N, p, L)$**, and can be abbreviated for the model as** $A_t(p, L)$**, with** t **is the event index,** p **is the going price, and** L **is the number of bidders in the waiting queue**.

- To fully describe the state of an auction, we would use $A_t(N, p, Q)$. However, the use of set Q would make the problem become intractable. It is because there are tens or hundred bidders in real auction, and keeping track of each bidder's behaviour history would cause an explosion of the state space.
- We try to model the entrance and exit of the auction using the waiting queue. The rate of entrance to the waiting queue depends on the going price $\$p$ and the arrival rate λ of new bidders. The rate of exit from the waiting queue depends on the departure rate μ and the number of bidders L in the queue. Hence, the use of L approximates the full information contained in set Q.

Proposition 4: Assume bidders behave as assumption 6 and 7, we can compute the transition probabilities which are shown in Table 3.2 where $F(p)$ is the probability to submit bid at current going price p.

Transition	**Probability**
1. $A_t(p, L) \to A_{t+1}(p, L)$	$\frac{\lambda}{\lambda+\mu L}\overline{F}(p)$
2. $A_t(p, L) \to A_{t+1}(p + c, L + 1)$	$\frac{\lambda}{\lambda+\mu L}F(p)$
3. $A_t(p, L) \to A_{t+1}(p, L - 1)$	$\frac{\mu L}{\lambda+\mu L}\overline{F}(p\|Q)$
4. $A_t(p, L) \to A_{t+1}(p + c, L)$	$\frac{\mu L}{\lambda+\mu L}F(p\|Q)$

Table 3.2: Transition probabilities of the Markov model. λ and μ kept constant.

1. The first event was the arrival of a new bidder. He/she read the going price \$$p$ and decided to drop out. Bidder can come from the waiting queue or he can be a new comer, the total arrival rate of a bidder equals to $\lambda + \mu L$. The ratio of the number of newcomer to the total number of arrival is $\frac{\lambda}{\lambda+\mu L}$. If the bidder decides to drop out, the transition probability equals to,
$$\frac{\lambda}{\lambda + \mu L}\overline{F}(p). \tag{3.1}$$

2. The second event was the arrival of a new bidder. He/she read the going price \$$p$, and decided to bid \$$p$, bumping the previous bidder to the waiting queue.

3. The third event was the wake up of an old bidder from the waiting queue. He/she read the going price \$$p$ and decided to drop out. The ratio of number of bidders coming from waiting queue to total number of arrival is $\frac{\mu L}{\lambda+\mu L}$. If the bidder decides to drop out, the transition probability equals to,
$$\frac{\mu L}{\lambda + \mu L}\overline{F}(p). \tag{3.2}$$

4. The fourth event was the wake up of an old bidder from the waiting queue. He/she read the going price \$$p$, and decided to bid \$$p$, bumping the previous bidder to the waiting queue.

The possible transitions from $A_t(p, L)$ to other points in the state space can be mapped as shown in Figure 3.2.

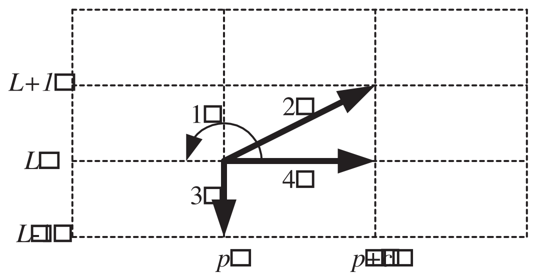


Figure 3.2: Possible state space transitions.

On a larger scale, this gives Figure 3.3 with $p = 5$ and $L = 5$. The auction will change state over the arrival of any new events.

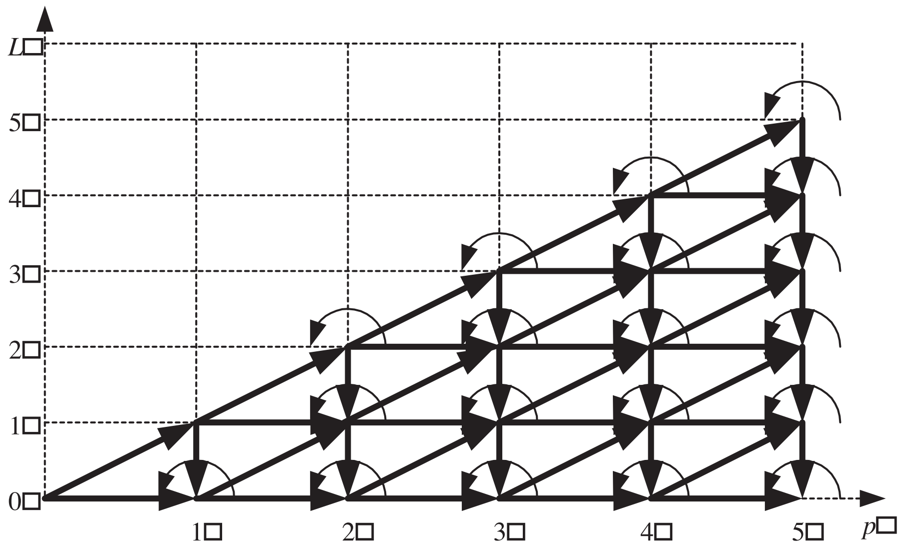


Figure 3.3: Large scale of state space of an auction.

Proposition 5: Using proposition 1 to proposition 4, we can model the auction $A_t(p, L)$ as a Markov chain.

- The conditional distribution of the future states of the auction A_{t+1} depends on the current state A_t but not on the path taken to arrive at A_t.
- Arrival of new bidders follows Poisson arrival model, hence it should be memoryless.
- Departure of bidders from the waiting queue follows the Poisson arrival model, hence it should be memoryless.
- The transition probabilities of the current auction state depends on the current value of p and L, but not the path taken to arrive at either p or L.
- Hence, it satisfies the Markov chain properties.

3.1.5 Tractability

The previous section gives some intuitions about the state space model. In order to solve the model and obtain insights into the system behaviour, the following assumptions and approximations should be made.

The biggest challenge is the treatment of the waiting queue. The model assumes it is possible to keep track of individual bidder in the waiting queue — in particular, it is possible to determine the previous bid levels of a newly awakened bidder, and from there determine the new, conditional probability that he will meet the new price of an auction, $F(p|Q)$.

This is simply too much information to keep track of, especially considering that many online English auctions have tens and possibly hundreds of bidders in them. It is impossible to predict when each individual bidder will be reawaken, and assigning probability functions to it would introduce more variability into the model without gaining much insight. Hence, we try to manage this intractability in two steps, which are outlined below.

First, we assume the waiting queue is fully stacked. A fully stacked queue will have one bidder for each level in it, i.e., it will have one bidder bids \$1, one bidder bids \$2, one bidder bids \$3, etc. We recognize that in reality the bidder at \$2 may subsequently drop out of the auction, leaving a gap at the \$2 level. However, the assumption that the queue is fully stacked assumes there are no gaps in the queue.

Second, we approximate $F(p|Q)$ with $F^*(p)$. Rather than keeping track of separate posterior cdf for each bidder in the waiting queue, we average the posterior cdf of all bidders in the queue. We assign this average posterior cdf to each bidder in the queue. The calculation of $F^*(p)$ is as follows:

Define

- $\$p^*$ is the going price for the auction at the current time.
- L_{max} is the maximum number of distinct bidders who could possibly be in the waiting queue, given that the selling price has been driven from $\$r$ up to $\$p^*$. L_{max} can be calculated by using:

$$L_{max} = \frac{p^* - r}{c} - 1. \tag{3.3}$$

Given

- The current going price is $\$p^*$.
- The bidder came from the waiting queue.
- The waiting queue is fully stacked, which implies that there are L_{max} bidders in it.

Calculate

- First, calculate L_{max} which corresponds to $\$p^*$.

- Second, calculate $F^*(p)$.

  If $L_{max} = 0$,

$$F^*(p) = F(p). \tag{3.4}$$

  This is because there is nobody in the waiting queue. Hence, the posterior cdf for the empty queue is the same as the arrival of new bidders.

  If $L_{max} > 0$,

$$F^*(p) = \frac{L_{max}-1}{L_{max}} F^*(p-1) + \frac{1}{L_{max}} F(v_i | v_i \geq p - c). \tag{3.5}$$

  This is a weighted average of the cdf of all the previous occupants of the queue and the posterior cdf of the latest arrival to the queue.

Table 3.3 shows the new probability transition matrix of the auction, substituting the approximation $F^*(p)$ for the exact expression $F(p|Q)$. Moreover, a small example of calculating F^* functions can be found in Appendix A with $r = 1$ and $c = 1$.

Transition	**Probability**
$A_t(p, L) \to A_{t+1}(p, L)$	$\frac{\lambda}{\lambda+\mu L} F(p)$
$A_t(p, L) \to A_{t+1}(p + c, L + 1)$	$\frac{\lambda}{\lambda+\mu L} \overline{F}(p)$
$A_t(p, L) \to A_{t+1}(p, L - 1)$	$\frac{\mu L}{\lambda+\mu L} F^*(p)$
$A_t(p, L) \to A_{t+1}(p + c, L)$	$\frac{\mu L}{\lambda+\mu L} \overline{F}^*(p)$

Table 3.3: Approximate transition probabilities of the Markov model.

This approximation of $F^*(p)$ differs from the previous model $F(p|Q)$ in the following ways.

- The state space will be measured in terms of the going price \$$p$, and the actual number of bidders in the waiting queue L. However, the probabilistic behaviour of the waiting queue bidders will be determined by a weighted average which depends on the number of people could possibly join the queue, not by the number of people actually are in the queue.

- $F^*(p)$ is calculated based on L_{max} which is calculated from p. L_{max} is not necessarily the number of bidders currently in the waiting queue, but it is the most who could ever have been there under the current going price $\$p$.

- Using $F^*(p)$ instead of $F(p|Q)$ will give more conservative auction behaviour results, giving a lower selling price. The prediction result will converge to an equilibrium value as the number of events is increased. The reason is that the waiting queue will accumulate more and more bidders towards the end of auction. According to Equation 3.3, L_{max} will increase and give a smaller $F^*(p)$. Hence, the transition probability will decrease and keep the going price at its old value. In other words, the going price will not grow further but converge to an equilibrium value.

3.2 Markov Chain for Dutch Auction

In this section, we will introduce the Markov model which is used to predict the revenue trajectory and the final selling price of Dutch auction under some assumptions. The model is similar to the one which is used in English auction and it is shown in Figure 3.4.

The sequence of events in the model is described as follows:

1. The system state S0 means that new bidder arrives at the auction site. It remains in S0 unless a new bidder submits a bid or the new bidder decides to wait.

2. The system state proceeds from S0 to S1 if the bidder decides to revisit the auction later. The waiting queue contains all previous bidders who

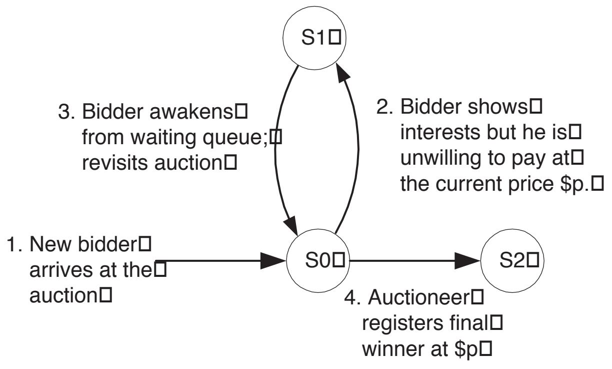

S1
3. Bidder awakens from waiting queue; revisits auction
2. Bidder shows interests but he is unwilling to pay at the current price $p.
1. New bidder arrives at the auction
S0
S2
4. Auctioneer registers final winner at $p

Figure 3.4: The Markov model for Dutch auction.

are waiting. They wait in the queue until awaken. At this time, the auctioneer drops the selling price from $\$p$ to $\$p - \$c$.

3. A previous bidder awakens from the waiting queue, and visits the auction again, reading the new selling price. The decision is the same as the arrival of a new bidder. The system state proceeds from S1 to S0 whenever a bidder awakens from the waiting queue.

4. If the new bidder bids successfully by submitting bid at the value of $\$p$, the auctioneer registers he/she as the final winner and the system state proceeds from S0 to S2.

3.2.1 Variables

In this section, we will introduce variables for modeling the Dutch auction. Table 3.4 summarizes all the variables and their definitions

3.2.2 Model Assumptions

The assumptions used in Dutch auction including auction and bidders are similar to those discussed in English auction. Some amendments are listed as follows.

Variable	**Definition**
A	A is the Dutch auction.
N	number of identical items for sale.
b_i	b_i is the i^{th} bidder arrives at the auction.
r	reserve price of an item.
c	minimum bid decrement of the auction.
t	index of events which is a finite positive integer.
$p(t)$	going price of the auction at time t, i.e., this is the bid required to become the final winner.
λ	arrival rate of new bidders.
μ	departure rate at which each individual bidder awakens from the waiting queue and revisits the auction.
Q	a set of bidders in the waiting queue.
L	number of bidders in the waiting queue.
L_{max}	maximum value of L computed by the current going price.
v_i	valuation of the i^{th} bidder
$F(p)$	cdf, the output is the probability to submit bid at current going price.
$\overline{F}(p)$	$1 - F(p)$

Table 3.4: Variables for the Dutch Markov model.

1. The auction will not be closed until the coming of the first bid from any bidders which does not depend on the auction activity.

   Validation: This is an assumption made for the Markov model and how the actual Dutch auctions are run.

2. The auction sends email to bidders immediately to notify the bidder when an auction has closed, and whether that bidder won.

   Validation: This is how the actual Dutch auctions are run.

3. If the bidder thinks the going price is too expensive ($v_i < p$), the bidder will enter the waiting queue and revisits auction again.

   Validation: This is an assumption of the model. We have no way to validate this short of sitting with unsuccessful bidders and interviewing them as they lose. However, we believe bidders will not lost their interests in several days.

4. If a bidder wants to become the final winner, the bidder will place a bid in the amount of $\$p$. The auctioneer register him as the final winner.

   Validation: This is how the actual Dutch auction runs.

5. After entering the waiting queue, the bidder goes off to do other things. From the auctioneer's point of view: (1) He will awaken independently and randomly after a period of time and re-check the auction status; (2) If the bidder re-visits while the auction is still going on, he will behave as a new bidder; and (3) If the bidder re-visits after the auction has closed, he losses his chance to win the auction.

   Validation: This is an assumption made for the model, the argument is the same as the English auction.

3.2.3 Propositions of the Markov Model

The following section presents the Markov chain model for modeling Dutch auction. Most of the propositions are similar to those discussed in English auction except for the followings.

Without loss of generality, we set $c{=}{-1}$, using a \$1 bid decrement. Furthermore, we define an event as the arrival of a new bidder or bidder awakened from the waiting queue. To count as an event, the bidder needs only to arrive; he/she does not need to make a bidding decision.

Proposition 1: Assume bidders behave as shown in Figure 3.4, we can compute the transition probabilities which are shown in Table 3.5.

Transition	**Probability**
1. $A_t(p,L) \rightarrow A_{t+1}(p+c, L+1)$	$\frac{\lambda}{\lambda+\mu L}\overline{F}(p)$
2. $A_t(p,L) \rightarrow A_{t+1}(p, L)$	$\frac{\lambda}{\lambda+\mu L}F(p)$
3. $A_t(p,L) \rightarrow A_{t+1}(p+c, L)$	$\frac{\mu L}{\lambda+\mu L}\overline{F}(p\|Q)$
4. $A_t(p,L) \rightarrow A_{t+1}(p, L-1)$	$\frac{\mu L}{\lambda+\mu L}F(p\|Q)$

Table 3.5: Transition probabilities of the Markov model.

1. The first event was the arrival of a new bidder. He/she reads the going price p and decided to enter waiting queue.

2. The second event was the arrival of a new bidder. He/she reads the going price p and decided to bid p.

3. The third event was the wake up of an old bidder from the waiting queue. He/she reads the going price p and decided to enter waiting queue.

4. The fourth event was the wake up of an old bidder from the waiting queue. He/she reads the going price p and decided to bid p.

The possible transitions from $A_t(p, L)$ to other points in the state space can be mapped in Figure 3.5.

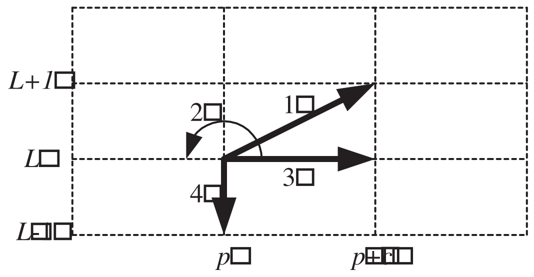


Figure 3.5: Possible state space transitions.

3.2.4 Tractability

In the previous section, we have discussed the challenge of the treatment of the waiting queue in English auction. In Dutch auction, we use similar technique to determine the conditional probability that a awakened bidder meet the new price of an auction $F(p|Q)$. The assumption is the same as before, i.e., we assume the waiting queue is fully stacked. The approximation is done by averaging the posterior cdf of all bidders in the queue.

The definition of L_{max} and p^* is still valid in this section. However, the value of c must be less than zero because the meaning of c changes to bid decrement in this case. L_{max} can be calculated by Equation 3.3 and $F^*(p)$ can be calculated by Equation 3.5. The calculation of F^* functions is similar to the English auction. Please refer to Appendix A.

3.3 Progression of the Price Vector for English and Dutch Auction

In the previous sections, we have introduced transition probabilities. We can use it to compute the transition probability matrix **P**. Each element in **P** corresponds to the transition probability from one state to another state.

Now, we let π_t be the probability vector describing the state of the auction after the t^{th} event has occurred (with an event being either a new arrival or an awakening from the queue, regardless of whether he decides to bid or not).

At $t = 0$, the auction state is defined as:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots \end{pmatrix}$$

In other words, we mean the auction will begin with probability 1 in state (0,0). We call this matrix as π_0.

The dimension of the probability vector depends on the number of states in either English or Dutch auction. Given the maximum selling price of the item, the reserved price of the item, and the minimum bid increment/decrement, the dimension of the probability vector can be calculated by Algorithm 3.1.

Algorithm 3.1 Find dimension of price vector

```
/* initializes variables */
int $count$=0;
int $dimension$=1;
float $r$=$reserved\_price$;
float $max\_price$; /* it is the buyer's valuations of the item */
/* without loss of generality, $c$ is either 1 or -1 */
float $c$=$minimum\_bid\_increment$ or $minimum\_bid\_decrement$;
for $i$=$r$ to $max\_price$ do
    if $(i - count{+}{+}{*}c + count \geq 2)$ then do
        for $j$=1 to $i-1$ do
            $dimension$++;
    end if
end for
```

Using the transition probability matrix **P** and the probability vector, other auction states starting from $t = 1$ can be calculated by Algorithm 3.2.

Algorithm 3.2 Find all price vectors

```
initialize $\pi_0$; /* See Appendix */
compute $F(p)$;
compute $F^*(p)$;
compute transition probability matrix $\mathbf{P}$ using $F(p)$ and $F^*(p)$;
for $t$=1 to $round$ do
    $\pi_t = \pi_{t-1} \cdot \mathbf{P}$;
    /* read Appendix for calculation of expected revenue */
    compute expected revenue at time $t$ from $\pi_t$;
end for
```

Mathematically, at $t = 1$, the auction state is represented by $\pi_1 = \pi_0 \cdot \mathbf{P}$ where $\mathbf{P}$ is the transition probability matrix. At $t = 2$, the auction state is represented by $\pi_2 = \pi_0 \cdot \mathbf{P^2}$. The probabilistic progression of the auction for the next several time periods can be calculated in a similar way. From that, we can obtain a set of price vector starting from the beginning of an auction to a certain time event. The bidding price will be upper bounded by the valuation of bidders. An example of price vector progression is shown in Appendix A. The set of price vector is presented in Figure A.10. Having computed the progression of price vector, we can compare the result with real auction data and draw some conclusions about the Markov chain model. The discussion will be provided in Chapter 5.

3.4 Markov Chain for Double auction

In this section, we will discuss the Markov chain for modeling Double auction. Modeling a Double auction using Markov chain is more complicated compared with the English and the Dutch auction for three reasons. (1) The state space becomes a function of number of seller agents and buyer agents, arrival rate of new buyer and new seller, and the distribution of buy and sell prices, (2) more transition probabilities are involved, and (3) more agents are participated.

We believe the Markov model is a good tool to model the Double auction. Agent's strategy plays no role here. It is because an agent meets many different agents in its lifetime, modeling individual agents tends to be incorrect and incomplete. In such a case, modeling the overall auction process is more accurate. Even when strategies about each individual agent are available, the complexity of such a model is too enormous.

In the Markov model of Double auction, we have a pool of buyer agents, a pool of seller agents, and an auction agent. The auction agent continuously matches the highest shout price given by a buyer agent to the lowest shout price given by a seller agent, given that the shout price of buyer agent is greater than the shout price of the seller agent.

If an incoming shout price of a buyer agent cannot match with the existing shout price from any one of the seller agent, the new buyer's offer becomes a new standing offer. Then seller agent will decrease its shout price and buyer agent will increase its shout price. Since buyer agents (seller agents) with shout prices higher (lower) than any standing sell (buy) offer get matched, the buyer's standing offers (if any) always have lower offer prices than the seller's standing offers (if any). Therefore, standing offers ordered by lowest to highest shout price are always in a $(bbb...bsss...s)$ sequence where b is the standing buy offer and s is the standing sell offer.

We now describe how to build the Markov chain model using b and s. Each state in the Markov chain represents the status of the auction. For example, the $(bbss)$ state represents the case where they are two standing buy offers and two standing sell offers. If we assume offers arrive at most one at a time, the sequence of events in the model can be described as follows:

1. No new offer arrives.

2. A buy offer arrives, and it matches with the lowest seller.

3. Because of no matches, a new buy offer becomes a standing offer.

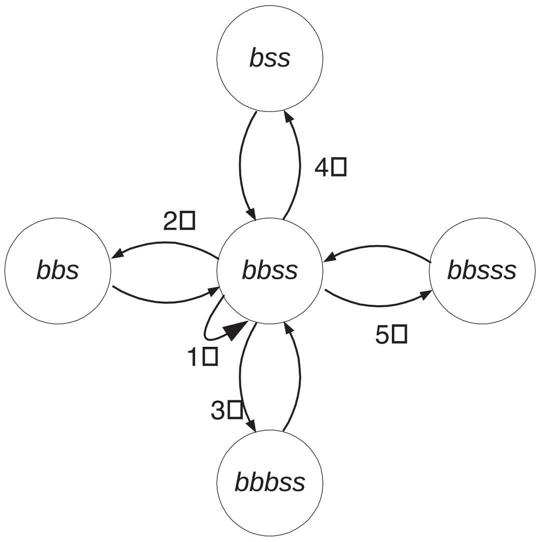


Figure 3.6: Transition from the (*bbss*) state.

4. A sell offer arrives, and it is matched with the highest buyer offer.

5. Because of no match, a new sell offer becomes the highest standing offer.

3.4.1 Variables

In this section, we will introduce variables for modeling the Double auction.

3.4.2 Model Assumptions

1. Sell offer or buy offer arrives at most one at a time.

2. Auction agent will register any one of the buyer agents as winner after it finds a match between a sell offer and a buy offer.

3. The auction sells multiple items.

4. The auction does not advertise future auctions or items.

5. The auction sends mail to buyers immediately to (1) confirm placement of shout price, and (2) notify the buyer when a deal is made.

Variable	**Definition**
A	A is the Double auction.
N	number of identical items for sale.
b_i	b_i is the i^{th} buyer arrives at the auction.
s_i	s_i is the i^{th} seller arrives at the auction.
t	index of events which is a finite positive integer.
λ	arrival rate of new buyers.
μ	arrival rate of new sellers.
N_b	number of buyers in the auction.
N_s	number of sellers in the auction.
v_i^b	valuation of the i^{th} buyer.
v_i^s	valuation of the i^{th} seller.
$F_b(p^b)$	cdf, specify the probability distribution of submitting bid at different shout price.
$F_s(p^s)$	cdf, specify the probability distribution of submitting bid at different shout price.
$\overline{F_b}(v_i^b)$	$1 - F_b(v_i^b)$
$\overline{F_s}(v_i^s)$	$1 - F_s(v_i^s)$
$P_b(p) = \int_0^p F_b(v_i^b)$	Probability a buyer accepts the shout price at \$$p$
$P_s(p) = \int_0^p F_s(v_i^s)$	Probability a seller accepts the shout price at \$$p$

Table 3.6: Variables for the Double Markov model.

6. There is no interaction between buyer agents or seller agents.

7. The i^{th} buyer has valuation v_i^b for the item. F_b is uniformly distributed. The v_i^b for all buyers is independent and identically distributed. The buyers do not update their valuations in the auction. The argument is the same as the English auction.

8. The i^{th} seller has valuation v_i^s for the item. F_s is uniformly distributed. The v_i^s for all sellers is independent and identically distributed. The sellers do not update their valuations in the auction. The argument is the same as the English auction.

3.4.3 Propositions of the Markov Model

The following section presents the Markov chain model for modeling the Double auction. We use the time index t to measure the number of events which have taken place, not as a measure of the exact time of an event.

Proposition 1: New buyers and sellers arrive at the auction in accordance with a Poisson Process with parameter λ and μ.
Define $B(t)$ as the number of buyers/sellers who have arrived by time t. $B(t)$ is a Poisson Process because:

- $B(0) = 0$.

- $B(t)$ exhibits independent increments. The number of arrivals in disjoint time increments is independent. Each new buyer/seller independently and with small probability for each point in time decides to visit the auction, without interaction with other buyers/sellers.

- The probability of two events happening at exactly the same time is zero. The server cannot process two arrivals simultaneously, so even two arrivals occur at the same time, they would be queued in an arbitrary order at the server and processed one after the other.

Proposition 2: The state of the double auction can be described by $A_t(N_b, N_s)$, with t is the event index N_b is the number of buyers and N_s is the number of sellers.

- We try to model the double auction using the state diagram as shown in Figure 3.6. To reduce the state space, we cannot keep track of each buyer/seller's behaviour. We use N_b, N_s and try to average their cdf to approximate the full information contained in the state space. This will be explained in section 3.4.4.

Proposition 3: Assume buyers and sellers behave as assumption 2, 7 and 8, we can compute the transition probabilities which are shown in Table 3.7.

Transition	**Probability**
$A_t(N_b, N_s, p) \to A_{t+1}(N_b, N_s - 1, p)$	$\frac{\lambda}{\lambda+\mu} P_b(p) P_s(p)$
$A_t(N_b, N_s, p) \to A_{t+1}(N_b + 1, N_s, p \pm c)$	$\frac{\lambda}{\lambda+\mu} P_b(p) \prod_{i=1}^{N_s}(1 - P_s(p))$
$A_t(N_b, N_s, p) \to A_{t+1}(N_b - 1, N_s, p)$	$\frac{\mu}{\lambda+\mu} P_s(p) P_b(p)$
$A_t(N_b, N_s, p) \to A_{t+1}(N_b, N_s + 1, p \pm c)$	$\frac{\mu}{\lambda+\mu} P_s(p) \prod_{i=1}^{N_b}(1 - P_b(p))$
$A_t(N_b, N_s, p) \to A_t(N_b, N_s, p)$	$1 - P_b(p) P_s(p)$

Table 3.7: Transition probabilities of the Markov model. λ and μ kept constant.

- The first event was the arrival of a new buy offer, and it is matched with a sell offer with the lowest offering price. The ratio of the number of buy offers to total number of offers is $\frac{\lambda}{\lambda+\mu}$. Hence, the transition probability equals to,

$$\frac{\lambda}{\lambda+\mu} P_b(p) P_s(p). \tag{3.6}$$

- The second event was the arrival of a new buy offer, but it cannot match any one of the sell offer. Therefore, it becomes a standing offer. The probability of rejecting all sell offers at shout price p is $\prod_{i=1}^{N_s}(1 - P_s(p))$ where N_s is the number of sell offers. Hence, the transition probability equals to,

$$\frac{\lambda}{\lambda + \mu} P_b(p) \prod_{i=1}^{N_s} (1 - P_s(p)). \tag{3.7}$$

- The third event was the arrival of a new sell offer, and it matches with a buy offer with the highest offering price. The ratio of the number of sell offers to total number of offers is $\frac{\mu}{\lambda+\mu}$. Hence, the transistion probability equals to,

$$\frac{\mu}{\lambda + \mu} P_b(p) P_s(p). \tag{3.8}$$

- The forth event was the arrival of a new sell offer, but it cannot match any one of the buy offer. Therefore, it becomes a standing offer. The probability of rejecting all buy offers at shout price p is $\prod_{i=1}^{N_b}(1 - P_b(p))$ where N_b is the number of buy offers. Hence, the transition porbability equals to,

$$\frac{\mu}{\lambda + \mu} P_s(p) \prod_{i=1}^{N_b} (1 - P_b(p)). \tag{3.9}$$

- The fifth event was that there is no new offer arrived and there is no match between existing buy offer and sell offer.

Using the state transition diagram as shown in Figure 3.6, we can represent the state space in a larger scale. If $N_b = 5$ and $N_s = 5$, the result is shown in Figure 3.7.

The transition to the "success" state happens when all seller offers can be matched. The transition to the "fail" state happens when none of the seller offers can be matched.

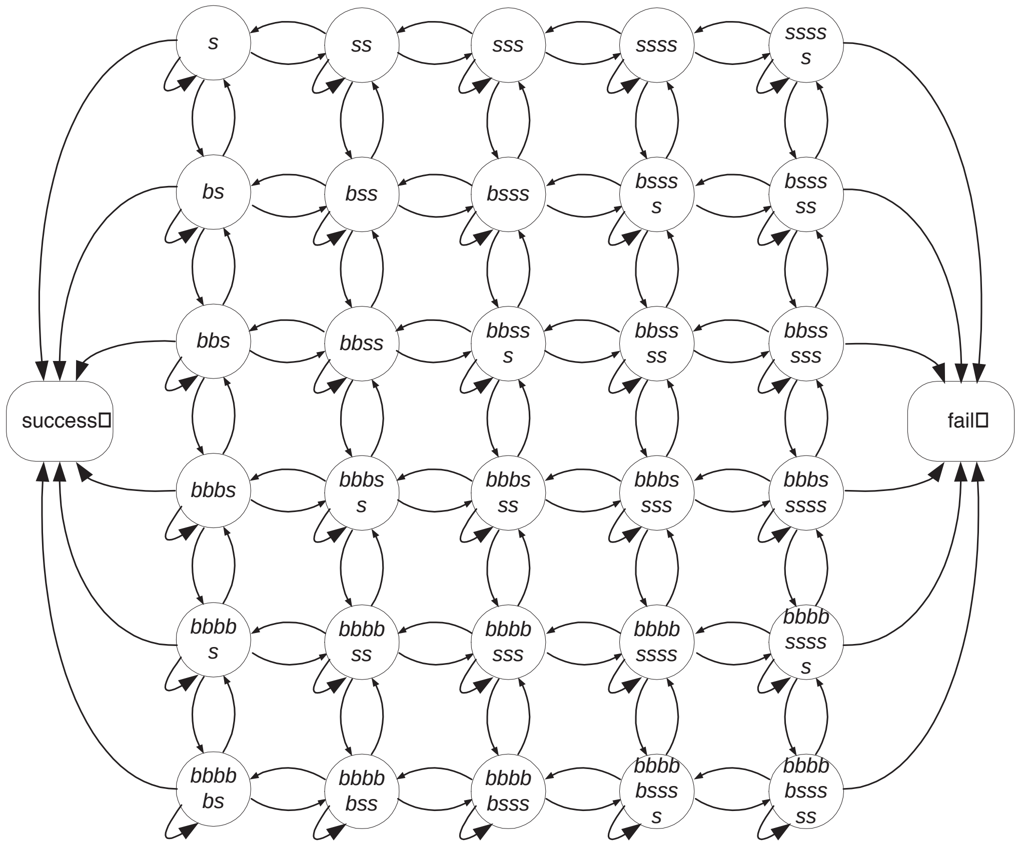


Figure 3.7: Large scale of state space of a double auction.

Proposition 4: Using proposition 1 to proposition 3, we can model the auction $A_t(N_b, N_s)$ as a Markov chain.

- The conditional distribution of the future states of the auction A_{t+1} depends on the current state A_t but not on the path taken to arrive at A_t.

- Arrival of buyers/sellers follows Poisson process, hence it should be memoryless.

- The transition probabilities of the current auction state depends on the current value of N_b and N_s, but not the path taken to arrive at either N_s or N_b.

- Hence, it satisfies the Markov chain properties.

3.4.4 Tractability

In the previous section, we give some intuitions about the state space model. The model assumes it is possible to keep track of individual bidder in the system — in particular, it is possible to determine the previous shout price of an existing standing sell/buy offer, and from there determine the new, conditional probability that the buyer/seller agent will meet the new shout price.

However, many double auctions involve hundreds of buyers and sellers in them. Hence, it is impossible to keep track of individual bidder information. Moreover, the buyer to seller ratio is many to many which expands the state model tremendously. It is impossible to assign probability functions to a new shout price. Hence, we try to manage the intractability by assuming all sellers use the same $F_s(p)$ and all buyers share the same $F_b(p)$. The distribution of either $F_s(p)$ or $F_s(p)$ will remain unchanged in the auction. We understand the approximation will deteriorate the prediction result. However, it makes the problem of state space expansion become tractable and solvable. Appendix B shows the steps of computing the transition probability matrix **P** using transition probabilities defined in Table 3.7.

3.5 Progression of the Price Vector for Double Auction

In the previous section, we have introduced the transition probabilities. Now, we can use them to compute the transition probability matrix **P**. The dimension of matrix **P** depends on the value of N_s and N_b. If $N_s = 5$ and $N_b = 5$, the dimension of matrix **P** equals to $(N_b + 1) * N_s \times 4 * (N_b + 1) * N_s$, i.e., 30×120. Each element of matrix **P** corresponds to the transition probability from one state to another state.

Again, we let π_t be the probability vector describing the state of the auction after the t^{th} event has occurred (with an event being an arrival of a new buyer or an arrival of a new seller).

Let $N_s = 5$ and $N_b = 5$; At $t = 0$, the auction state is defined as:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & ... \end{pmatrix}$$

where the dimension of the state vector equals to 1×120.

This means that the Double auction starts with probability 1 in state $(0, 0)$. Using the transition probability matrix $\mathbf{P}$ and the probability vector, other auction state can be calculated by Algorithm 3.3.

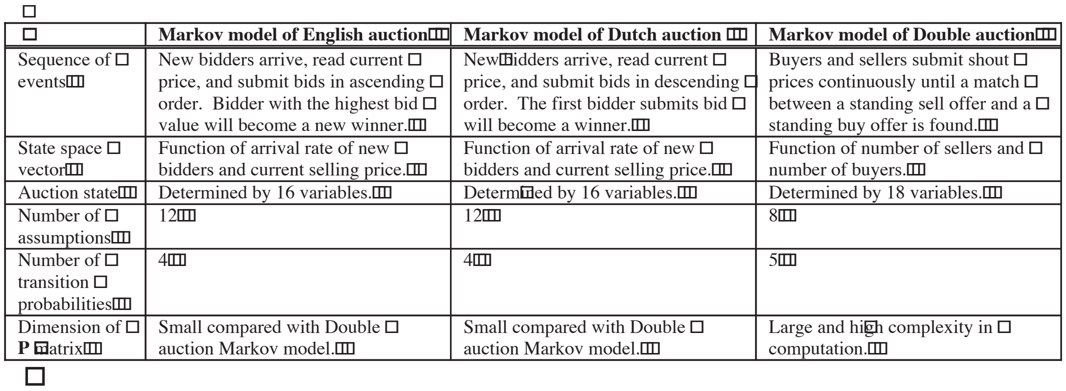
Algorithm 3.3 Find all price vectors

```
1  /* initializes variables */
2  $N_b = numberofbuyers$;
3  $N_s = numberofsellers$;
4  compute $F_b(v_i)$;
5  compute $F_s(v_i)$;
6  compute transition probability matrix $\mathbf{P}$ using $F_b(v_i)$, $F_s(v_i)$, $N_s$ and $N_b$;
7  for $t$=1 to $round$ do
8      $\pi_t = \pi_{t-1} \cdot \mathbf{P}$;
9      compute expected revenue at time $t$ from $\pi_t$;
10 end for
```

3.6 Summary

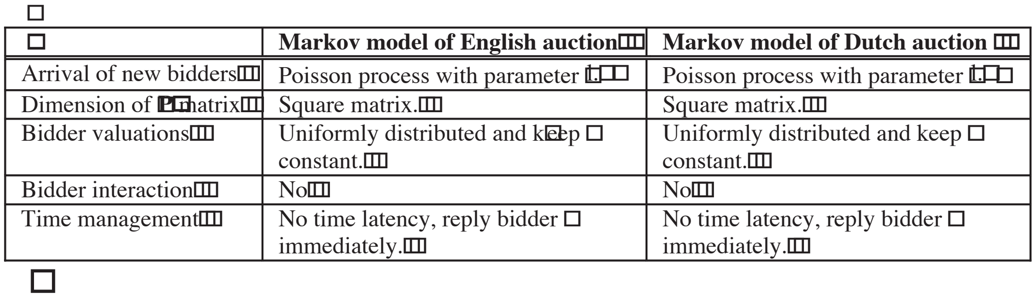
In this chapter, we have introduced three Markov chain models for predicting the expected revenue of English, Dutch, and Double auction. An online auction can be modeled as a Markov process based on some assumptions. Some assumptions are well validated using real data from eBay [3]. But some may not. Therefore, we cannot say that our model is fit for bidding all kinds of item or modeling different bidder's behaviour. The reason is that Markov model is not designed for modeling complicated system behaviour. Otherwise, the state space will become very large and the complexity to solve the mathematics will become intractable. However, it can be used as a baseline to observe simplified system behaviour and this motivates us to use other tools like agent simulation approach. We believe that more complex scenario can be modeled by agent simulation. Details will be covered in Chapter 4.

Besides, we have studied the derivation of the transition probability and the steps of calculating expected revenue. Figure 3.8 shows the differences of the three Markov chain models and Figure 3.9 shows the similarities of the Markov models between English and Dutch auction. Discussion of models and prediction results will be provided in Chapter 5.



	Markov model of English auction	**Markov model of Dutch auction**	**Markov model of Double auction**
Sequence of events	New bidders arrive, read current price, and submit bids in ascending order. Bidder with the highest bid value will become a new winner.	New bidders arrive, read current price, and submit bids in descending order. The first bidder submits bid will become a winner.	Buyers and sellers submit shout prices continuously until a match between a standing sell offer and a standing buy offer is found.
State space vector	Function of arrival rate of new bidders and current selling price.	Function of arrival rate of new bidders and current selling price.	Function of number of sellers and number of buyers.
Auction state	Determined by 16 variables.	Determined by 16 variables.	Determined by 18 variables.
Number of assumptions	12	12	8
Number of transition probabilities	4	4	5
Dimension of **P** matrix	Small compared with Double auction Markov model.	Small compared with Double auction Markov model.	Large and high complexity in computation.

Figure 3.8: Differences of Markov models.



	Markov model of English auction	**Markov model of Dutch auction**
Arrival of new bidders	Poisson process with parameter	Poisson process with parameter
Dimension of **P** matrix	Square matrix.	Square matrix.
Bidder valuations	Uniformly distributed and keep constant.	Uniformly distributed and keep constant.
Bidder interaction	No	No
Time management	No time latency, reply bidder immediately.	No time latency, reply bidder immediately.

Figure 3.9: Similarities of Markov models.

Chapter 4

System Design

Today one of the most valuable online businesses is auction. The huge success and popularity of eBay [3] online English auction service has taken the world by storm. Nonetheless, eBay [3] service is limited to bidding. It does not provide any recommendations or guidances to either bidders or sellers so as to maximize their interests, such as profit. Our system aims at providing solutions to bidders and sellers using intelligent agents. From the agent's point of views, it will

- try to maximize the profit of the selling item for the seller,
- determine the best auction closing time by simulation and
- negotiate with bidders automatically without any seller interventions.

From the system's point of view, we define the following requirements for our services.

- The system must be scalable, portable, reliable, and secure.
- The system must meet the "four As" of availability: accessible, anytime, anywhere, and on any device.
- The system must be cost effective.

We have evaluated some E-commerce web sites [4, 5, 6, 44, 45, 46, 47, 48] and agent development platforms [38, 36, 49]. Nonetheless, we found nothing that fit our needs. Current agent development platforms were either extremely expensive to maintain and develop on or they were too inflexible or unreliable for us.

4.1 System Features

The design goals of this system are to:

1. enable agent developers to design and add new agent components to the infrastructure as simply and as quickly as possible, with a minimum understanding of the architecture;

2. create a component-based architecture that is robust, reliable, and easy to manage and control in a production environment. A component is a part of a system and interact with other components to complete a task; and

3. minimize the production cost with the use of modest hardware equipments but support high user load.

Our goals can be satisfied by using an N-tier architecture, but beyond that we want one that placed no limits on possible solutions. The N-tier architecture allows developers easily build and manage web-based agent applications. Developers can separate the presentation and agent logic into two layers. Develop components for agent logic is simple by design. Agent developers write the logic using Java. Our system runs these components in a highly scalable, portable, high-performance, and secure transaction environment. With our system, agent developers should avoid the time and expense of a weeklong class just to learn the details of the system architecture followed by another week of Java training.

Moreover, our system includes many features that support high-application availability: timeout checks, automatic component restart, and recovery procedures. Agent logic components run as servlets for extremely fast response from the Apache web server. Moreover, the system will dynamically creates additional component instances to handle increased transactional demands, destroying them when they are no longer needed. This feature allows the system to allocate resources optimally to satisfy current demands. Resources are used as specified by configuration files but are only used as necessary. Moreover, we provide a web interface to manage our configuration files. Configuration files are used to specify page layouts, update rules for resource allocation, and manage user profiles.

This architecture is well-suited for many web-based agent service projects with modest hardware settings. The core system components are written in Java on Linux. We can thus easily port to any system that supports Java, from embedded platforms to an IBM 390 mainframe. We are currently running it on a PII 300 server. The total production hardware cost was under $10,000. Resources are only allocated when needed, enabling rapid responses on our modest hardware.

4.2 System Architecture

The architecture is based upon the classic N-tier model with some enhancements. Figure 4.1 shows the architectural overview. There are three layers in the system. They are the Communication Layer, Agent Logic Layer, and Database Abstraction Layer. This model separates them with each other but they work dependently [16] [1] .

[1] http://137.189.90.57/ cwlee/phpnuke/html/index.php

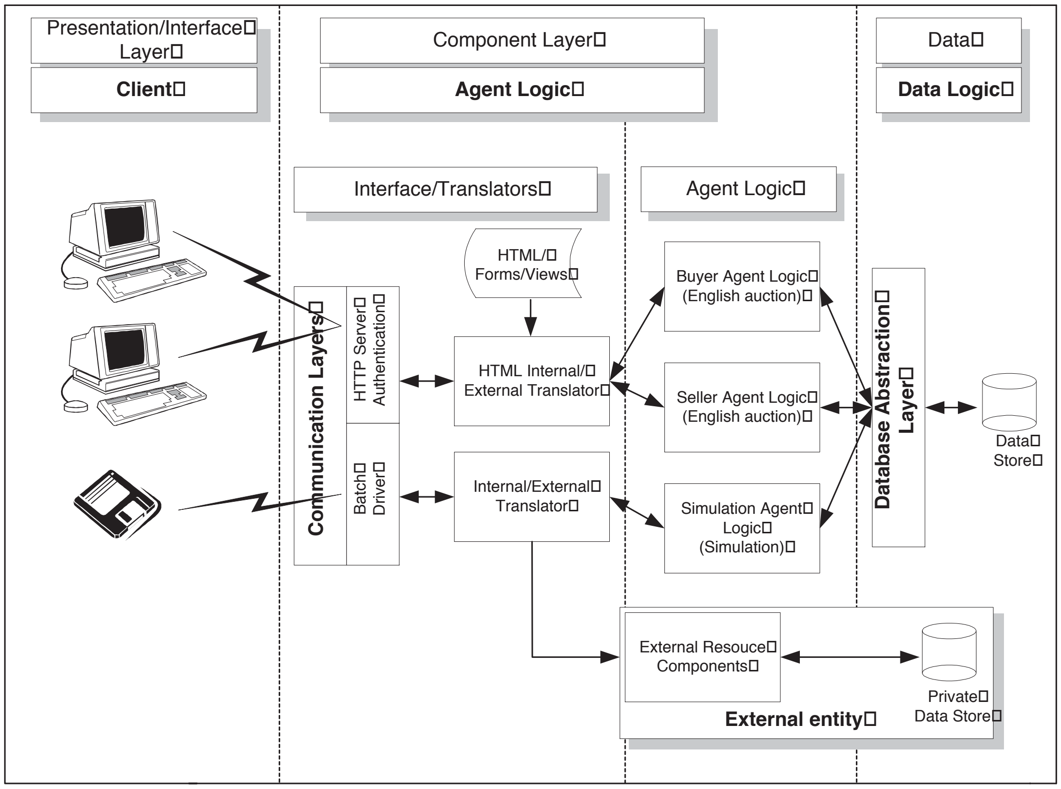

Presentation/Interface Layer
Client
Component Layer
Agent Logic
Data
Data Logic
Interface/Translators
Agent Logic
HTML/ Forms/Views
Communication Layers
HTTP Server Authentication
Batch Driver
HTML Internal/ External Translator
Internal/External Translator
Buyer Agent Logic (English auction)
Seller Agent Logic (English auction)
Simulation Agent Logic (Simulation)
Database Abstraction Layer
Data Store
External Resouce Components
Private Data Store
External entity

Figure 4.1: Architectural overview.

4.2.1 System Workflow

The system follows the N-tier model and its architecture makes it capable to deal with dynamic working environments like Internet. Currently, the Communication Layer, Agent Logic Layer, and Database Abstraction Layer co-operate with each other to provide online auction services to users. The workflow progresses as follows:

- Intelligent data clients, such as Linux workstations, Windows 2000/98/95 workstations, Mac workstations, batch inputs, virtually all commercial versions of UNIX and other devices optimized for specific data and representation functions gather information for subsequent processing and display information.

- In the Communication Layer, translator validates the transaction request and then forwards the request and data to the Agent Logic Layer.

- Agent Logic Layer's components can access databases with several methods. We can connect to any databases via their application programming interfaces, ODBC, or JDBC in the Database Abstraction Layer. The agent logic component returns the response and associated data back to translator, which sends the response back to the data client.
- The format of the response objects can be specified by giving a template to the translator in the Communication Layer. The translator merges the data with the template and returns the result. Templates can handle tabular data in the response objects.

4.2.2 Communication Layer

Translators, which is part of the Communication Layer, package data for the agent logic components. It also formats data that is sent back to the original client source. As one technique to format the data, the translator can use a template. The template is generated at run time. The dynamic content can be created by using a scripting language, the Document Object Model, or DOM– together with HTML and CSS. The script will be executed at the server, not at the client. Translators enable developers to focus on the agent logic without the distraction of how the data is formatted or presented. More importantly, this allows reuse of components if the data client changes. Having the presentation logic separated from the agent logic speeds software development and allows for faster application deployment and updates. It also enables developers, particularly web and graphics programmers, to focus on the presentation without having to worry about the agent logic or database I/O.

4.2.3 Agent Logic Layer

Our system uses a component software model. Data clients and other requesters invoke agent logic components and developers design agent logic components or modify their components by focusing on the agent processes. This development strategy enables developers to build components quickly with their existing knowledge and skills. Moreover, developers have the ability to alter the availability of individual agent logic components without affecting any other components.

4.2.4 Database Abstraction Layer

The Database Abstraction Layer provides a common interface for database interaction. This layer enables developers to interact with other databases without modifying agent logic component code.

The advantage of the Database Abstraction Layer is that an agent logic component need not be notified if the underlying data source changes, say from DB2 running on an IBM mainframe to Oracle running on Linux. Components programmers need not have their logic depend upon the specifics of a particular database. There are no SQL manipulation routines in an agent logic component, for example. All of the data source access is through a specific component that translates generic requests into a format for a particular data source.

4.3 Roles of the Agents

Our platform is a multi-agent system in which three semi-autonomous agents interact or work together to perform a user's goal in auctions. They are the *Buyer Agent(BA)*, *Seller Agent(SA)*, and *Database Agent(DA)*. Figure 4.2 shows the complete system architecture and the relationships between the platform and client computers.

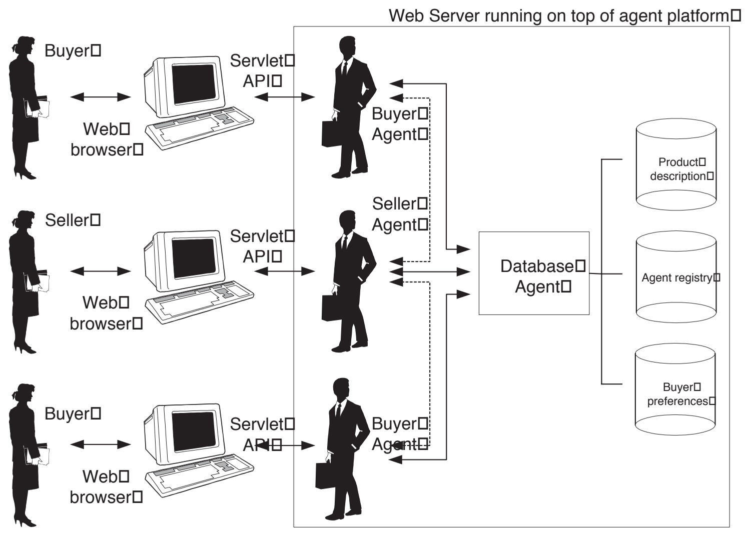


Figure 4.2: Roles of the agents.

In the proposed web-based platform, sellers and buyers should register in our registry before using our automated negotiation service. An SA will be created for the seller, and a BA will be created for the buyer. Both will be registered in the agent registry database. After that, potential sellers can advertise information about their goods and services on our web site. The product information is stored in one of the databases until the completion of the auction process. Buyers can browse advertisements and identify potential sellers through their web browser.

A buyer initiates the negotiation phase by issuing an initial negotiation request to the BA. The BA requests for product or services specifications from the buyer. The BA queries the product description database and the agent registry for the list of SAs that may satisfy the buyer's interest. The buyer chooses one of the potential SAs from the list and specifies the maximum and minimum bid. The BA stores the buyer's preferences into the database and sends a start-negotiation request to the SA. After that, the BA will negotiate with the SA with the English ascending-bid auction which will be described in Section 2.3. During the negotiation phase, the BA is responsible for confirming the placement of a bid to the buyer and notifying the buyer when he is no longer the current winner.

The SA receives seller's request for the product or service he wants to sell and advertises the request to the product description database. After that, it will wait for the initial negotiation request from any BAs. The SA plays as an auctioneer and negotiates with the BA until the completion of the auction or the auction is terminated by the buyer. During the auction process, the SA assumes (1) the seller sells a single item, (2) the seller does not advertise future auctions or items, and (3) the auction process closes at a preset time which does not depend on the auction activity. The SA is also responsible for notifying the buyer after the auction has closed.

The DA manages the set of databases including the Product Description, the Agent Registry, and the Buyer Preferences. The Product Description database stores the specification of the seller's product or services. If the item is selling in an English Auction, an example is shown in Table 4.1. The Agent

Attributes	**Value**
Seller Name	T&T computer shop
Product Name	Microsoft Windows 98 2nd Edition
Reserve Price	$600
Minimum Bid Increment	$20
Closing Time	23:00 03/02/01
Comments	Operating System

Table 4.1: Specification example.

Registry is a yellow page for the DA. The content of Agent Registry is agent's name, agent's type, and agent's data. The agent name composes of the IP address of the buyer or seller and the creation time of that agent. The agent type specifies the agent's role and the agent data stores temporary values.

The Buyer Preference database is designed for the BA. It stores the negotiation strategy of the buyer such as the minimum bid, the minimum bid increment, and the maximum bid. The BA will follow these negotiation constraints until an agreement is reached or the auction process is terminated by the seller or the buyer.

4.4 Implementation Details

Our system is build on top of the Linux. Linux is a kernel, the core part of an operating system that handles networking, hardware management, and basically makes the whole thing run. Most people, however, refer to Linux as the entire operating system and applications together, an alternative to Microsoft Windows or Apple's MacOS. Linux can replace Windows on your desktop, or windows NT on your server.

4.4.1 Choosing a Scripting Language

There are many scripting languages available for the Linux platform. Several are appropriate for the Communication Layer. Scripting languages are used in the server-side to generate dynamic web documents automatically. They are also used to tie together the user interface presented to the user, which will be written in HTML and accessed via a web browser, and the back end of the system and database used by the auctions. In the system, the translator component is the one which widely uses the scripting language.

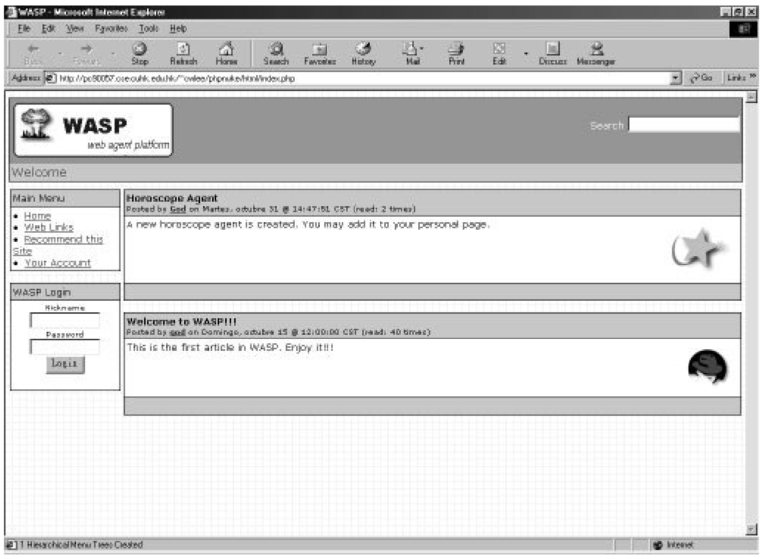
PHP

We choose PHP(ver. 4.0.4) as our server-side scripting language used in the translator component. The details are as follows. PHP [50] is an open-source, HTML-embedded scripting language. Unlike Perl, which was born as a tool to assist in system administration, it was designed from the ground up to work with web pages. Hence, the biggest advantage of PHP is that it was optimized for scripting for the Web only, with no more other things to do, and because of this, can generate web pages faster and easily integrate into HTML document. Moreover, PHP has a less confusing and stricter format without losing flexibility. Hence, it is easy to learn and study.

In the translator component, there are two types of PHP modules. They are the system modules and the language modules. The system modules are related to the user interface of the platform. Our system can be divided into two types of interfaces. They are the administration page and the user page. Some system modules are used in both interfaces while some of them are used in a particular interface. For example, header, footer, and config are used in both interfaces. For details, please read the following description.

- admin.php : This module is for generate the administrator's page and change settings of the web page
- auth.inc.php : This module is for administrators authentication
- config.php : This module is to configure the main options for your site
- counter.php : This module is to make some stats like browser, OS, and hits
- footer.php : This module is to be included in the foot of each page
- header.php : This module is to be included in the head for each page
- index.php : This modules is the main index file
- mainfile.php : This module is a collection of some useful global functions
- user.php : This module is to manage all registered user's options

Figure 4.3 and Figure 4.4 show the main page and the administrator page of the system which is created by the system modules. All contents are generated at run time after receiving a HTTP request from the client browser. Nonetheless, the dynamic content generation task does not consume too much system resources which allows our system to support more than hundreds of users at a time.


WASP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Messenger
Address http://pc90057.cse.cuhk.edu.hk/~cwlee/phpnuke/html/index.php
WASP
web agent platform
Search
Welcome
Main Menu
Home
Web Links
Recommend this Site
Your Account
WASP Login
Nickname
Password
Login
1 Hierarchical Menu Trees Created
Internet

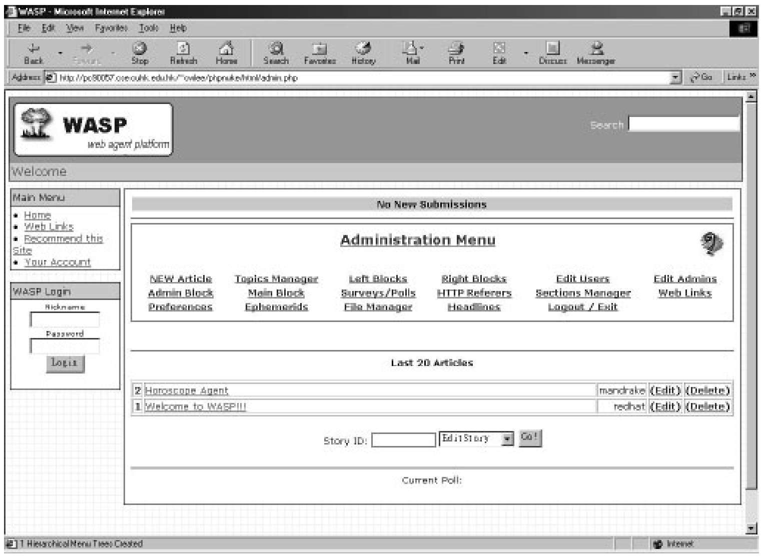
Horoscope Agent
Posted by God on Martes, octubre 31 @ 14:47:51 CST (read: 2 times)

A new horoscope agent is created. You may add it to your personal page.

Welcome to WASP!!!
Posted by god on Domingo, octubre 15 @ 12:00:00 CST (read: 40 times)

This is the first article in WASP. Enjoy it!!!

Figure 4.3: Main page created by system modules.



2	Horoscope Agent	mandrake	(Edit)	(Delete)
1	Welcome to WASP!!!	redhat	(Edit)	(Delete)

Figure 4.4: Administrator page created by system modules.

Our system is designed for multi-lingual purpose so that different user can set different default language. Hence, we need to translate sentences all the time. This task is implemented by the language modules. The idea is very simple; we do a translation lookup and give a sentence to a language module to translate. The return value will be a string which consists of the translated sentence. Currently, we can support 21 languages including English, French, German, Chinese, ..., etc.

4.4.2 Choosing a Database

This system requires a RDBMS, or Relational Database Management System. An RDBMS is a software package that stores data in rows and columns as tables. Various tables can be related to one another in order to answer questions posed by the end user. Their questions are known as queries.

MySQL

We choose MySQL [51](ver. 3.23.38) as database. The details are as follows. It was first released to the public in November 1996 and has always been available with source code. It has proven to be a very fast, multithreaded, multiuser, and robust SQL database server for a growing number of companies such as SGI, ValueClick, Nortel/Insight, Tucows.com, Cisco, and many more.

Other features of MySQL is that it is considered very fast with large record sets. Further, there seems to be a growing relationship between the team that develops MySQL and the team that develops PHP. This resulted in the MySQL library being packaged with the PHP 4.0 distribution. However, there are two main shortcomings with MySQL in relation to our system. The first one is in the area of transactions. MySQL has only limited support for foreign keys. The foreign key is an important concept of the relational model. It is the way relationships are represented and can be thought of as the glue that holds a set of tables together to form a relational database. The second one is that MySQL does not support subqueries. Luckily, our application is small and likely would not run into too much difficulty with MySQL's support of foreign keys, subqueries, and transactions, it was still a good choice for us.

We have set up six tables in MySQL to store permanent or temporary data. The schema of each table and the attribute type/value are provided in Appendix C.

4.4.3 Server-Side Java with Jakarta-tomcat

In our system, the basic unit of an agent logic component is the servlet, a small program that is executed in response to an HTTP request and then generates a legal HTTP response. Since servlets are written in Java, they are written as object classes, inherited from a servlet ancestor and can take advantage of Java's threading and execption handling. Moreover, servlets run inside of a Java virtual machine, an abstraction layer that can run on any operating platform. This means that the same servlets can run on nearly any operating system, providing greater portability for our system.

Now that we must installed the JDK and the Jakarta Tomcat to support servlets. Jakarta is the overall name for Java-related projects sponsored by the Apache Software Foundation, and Tomcat is the ASF's project for servlets and JSPs.

In our system, Tomcat runs as an integration to the Apache web server. The advantages of integration are:

- Tomcat is not as fast as Apache when it comes to static pages.
- Tomcat is not as configurable as Apache.
- Tomcat is not as robust as Apache.

The integration steps are quite simple, we need to rewrite one of the apache configuration file. A sample of the configuration file is listed in Appendix C.

4.4.4 Agent API

We provide a large set of classes, interfaces and exceptions that can be used by agents for accessing the functionality of the platform. For detailed descriptions of these members and their methods, please see the following section. They are the essestial elements for creating agent, classes related to internal system operation are not listed.

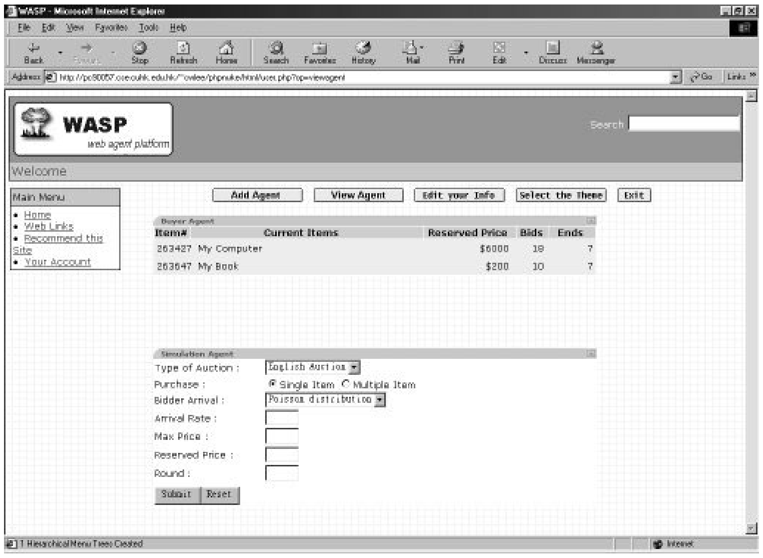
Package java.agent

Provide classes necessary to create, initialize and remove agent. In our system, an agent can be considered as a servlet. The servlet is running inside the Tomcat container. Hence, the agent programmer should the servlet packages as well as our agent package in his servlet code which looks like the following:

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.net.*;
import java.io.*;
import java.util.*;
import java.agent.*;   // this is our agent package
```

Our agent package contains a number of useful methods for agent creation, agent communication and interaction with human users. In agent creation, we will register the newly created agent in the database. A new agent ID and the name of its owner will be recorded. The agent ID is a unique string to identify an agent. In agent communication, the agent can use our system API to store its messages, semi-data and the agent ID of the receiver in the database. The database acts as a blackboard for communications. Any messages posted on the broad will be read by agents. Agent with the same ID as the receiver ID will take the messages and erase it from the blackboard. In interaction with human users, our package provides methods for getting web page parameters in the format of HTML forms. These parameters can be used by the agent throughout its lifetime. The response to the human user can by generated by creating web page from the PHP modules to format the agent data.

Figure 4.5 shows the agent's interface of buyer agent and simulation agent. The interface is prepared by the Interface/Translator module inside the Communication Layer. The creation of the interface is very flexible, it can be HTML forms, tables, etc. Agent developers are advised to design the interface in a simple but friendly style in order to reduce interface setup time but still keep the ease of use for its owner.



Item#	Current Items	Reserved Price	Bids	Ends
263427	My Computer	$6000	18	7
263647	My Book	$200	10	7

Type of Auction : English Auction
Purchase : [✓ Single Item] Multiple Item
Bidder Arrival : Poisson distribution
Arrival Rate :
Max Price :
Reserved Price :
Round :

Figure 4.5: User interface of buyer agent and simulation agent.

Method Summary

Table 4.2 summarizes some of the essential methods in class java.agent.Agent. They are used for agent creation, agent communication and interaction with human users.

Method	**Input Argument**	**Return value**
createAgent()	Object[] obj	String id
remove()	String id	void
getInfo()	String id	String agentData
flush()	void	void
setSaveInterval()	int anInterval	void
setFlushInterval()	int anInterval	void
doGet()	HttpServletRequest request , HttpServletResponse	void

Table 4.2: Method summary.

4.4.5 Class java.agent.Agent

- An agent is create by calling the createAgent() method. Our system will perform several internal tasks during the creation procedure, (1) registers the agent inside the system's internal database, and (2) enables the agent to access the system's functionality by delivering fundamental object references. An agent identification number will be returned. It is composed of:

  <ip><date><time><copy_number>.

- An agent can remove itself by invoking the method remove(). After receiving the request to remove an agent, our system automatically invokes the agent's method beforeRemove(). An agent programmer may override this method in order to enable the agent to prepare its removal, e.g., by releasing occupied references.

- The getInfo() method can get a set of information that is associated with the agent. It includes agent's name and type. An agent identification number should be passed into this method and a string containing the agent info is returned.

- The flush() method can store the agent data to the database. It receives two arguments. One of them is the agent indentification number and the other one is the serialized agent data. It should belongs to the string type.
- The load() method can load agent data from the database. It takes an agent identification number as argument and return a string composed of agent data.
- Agent data can be flushed in database automatically. The setSaveInterval() method can set the time interval between each saving attempts.
- Agent may be idle for too long without receiving user input. It would be better for us to set a timeout period for flushing its data to the database. The setFlushTimeout() method can set the timeout period.
- The doGet() method should be overridden to read parameters from a HTML page. The method signature should be:

  ```
  public void doGet(HttpServletRequest ...);
  ```

  We can get parameters by using:

  ```
  String sign = request.getParameter("sign");
  ```

  where sign is the name of the parameter.

4.5 Summary

Our system architecture provides high uptime. By building everything in redundant, restartable small components we can provide multiple paths for the application, increase flexibility and scalability. The N-tier architecture can make layer abstraction in the system which allows administrator to maintain it easily. The agent API provides useful methods for agent developers to create and manage agent. It is architecture independent which does not require agent developers to understanding the system in depth before written any codes. However, there are two shortcomings in the system. The first one is not enough security support. We are still developing the security modules for the system. To cite an example, the HTTP server authentication component are unable to defense software agent from outside attack. It leaves a security hole in the current stage. Morever, many security measures should be enhanced in the next development stage. The second one is the lack of distributed support. Now, our system can support hundreds of users concurrently using the N-tier architecture. The local system scalability is very good but we cannot do it in a distributed fashion. The architecture does not allow us to inter-connect two or more platforms together and does not support load balancing.

For the implementation details, the combination of Linux, Apache, MySQL, and PHP was appropriate. This combination will likely not be appropriate for all projects. However, we hope that this framework is an informative first step in investigation of alternatives for the next project.

Chapter 5

Experimental Results

The following experimental results focus on comparing the expected revenue calculated by the Markov chain model and the one simulated by software agents. We want to (1) know whether the behaviour of the SA and the BA are modeled properly, (2) measure the performance of Markov chain model, and (3) find out the limitations of the Markov chain model.

5.1 Simulation Environment

The Markov chain model is a good tool to model simple BA's behaviour. On the other hand, simulation presents a particularly attractive computational alternative for investigating online auction because it averts the need for overly restrictive assumptions and because it can model a wider range of BA's behaviour than Markov chain model can cope with. Therefore, we developed a system and a simulation agent to provide a simulation environment for the BAs and SAs to run the negotiation phase automatically [17]. This is the batch mode of the system which does not require agent and it's owner communicate in an interactive environment. The tasks of simulation agent is to create a pool of BAs and SAs with starting parameters and run one of the auction types. The simulation results collected by the simulation agent will be compared to the one calculated by the Markov chain model. From the comparison, we can verify the correctness of the Markov chain model, measure the performance of the bidding behaviour, and find the best auction closing time for the seller in order to maximize his profit.

The simulation environment runs on a PC with a 300MHz PII Intel CPU, 128Mb memory and a 12Gb harddisk. Our system was built on top of Redhat Linux 6.2 with kernel-2.2.16 and the Apache web server. The experimental results are generated with the following assumptions: (1) BAs are independent to each other, meaning that they will not have any interactions, (2) The negotiation phase runs a single-item auction, (3) The SA will not advertise future items and does not contain any purchasing history. Input parameters are the arrival rate of BAs/SAs, the minimum bid increment/decrement, the reserved price of the item, the maximum bid of each BA, the valuations of BA/SA, and the closing time of the auction process. Figure 5.1 shows the simulation process.

The simulation agent will initialize the simulation environment with the input parameters provided by its owner. Then agents will be created at different time to join the auction. The arrival of agents follows the Poisson arrival model. If a BA shows interests in the item provided by the SA, they will start negotiation according to the predefined auction format. The negotiation phase will proceed until the end of the auction. The SA will record the selling price of the item after a bid is submitted. The simulation result will be used to plot a graph of expected revenue verus time at the end of the auction.

5.2 Experiment One: A small English auction

Motivations

In this experiment, we want to test the correctness of our Markov chain model for English auction. First, we apply the Markov chain model to a small size English auction and calculate the expected revenue. Then we will use similar

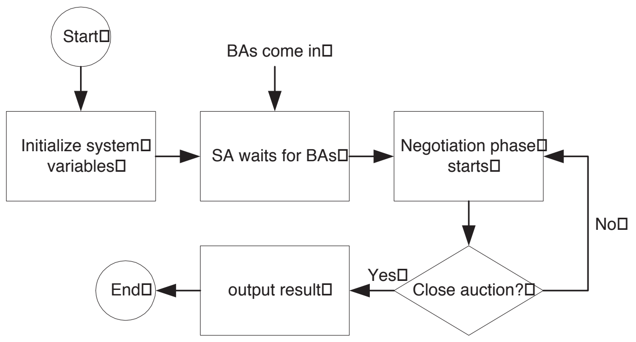


Figure 5.1: Simulation process.

parameter settings and apply the settings to agent simulation. Finally, we will compare the expected revenue predicted by the Markov chain model with the one simulated by software agents.

Experimental Settings

In the Markov chain model, we use parameter settings described in Table 5.1 and the cdf of BA's valuations for an item shown in Figure 5.2. The value of arrival rate is suggested by [52]. In that paper, the author did a study to analyze eBay auction data and found that it was reasonable to approximate the arrival rate as 1 arrival/hour.

In English auction, there is only one SA. Hence, we assume the arrival rate of SA is 0 arrival/hour and the SA is ready to serve BAs after the auction starts. Moreover, we do not have cdf for SA because it will not place any bids in the auction. Based on the previous assumptions, our agent simulation uses parameter settings as shown in Table 5.2.

Experimental Results

Figure 5.3 shows the expected revenue calculated by the Markov chain model and the expected revenue simulated by the software agents from an initial revenue of \$0 to the final revenue of \$5.

We find that the Markov chain of English model does a good job to simulate

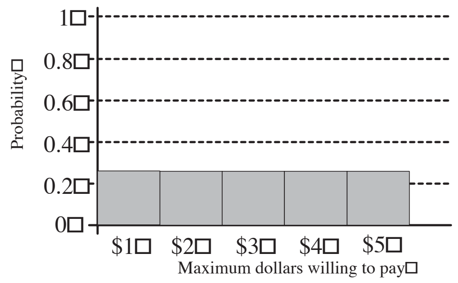


Figure 5.2: cdf of BA's valuations for an item.

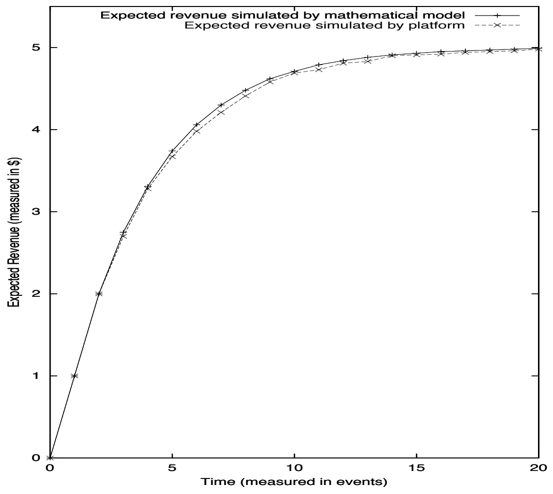


Figure 5.3: Expected revenue over time for a small English auction.

Parameter	**Definition**	**Value**
λ	arrival rate of new bidders	1 arrival/hour
μ	departure rate of bidders from waiting queue	1/3 departure/hour
c	minimum bid increment	1
r	reserve price of the item	1
time	closing time of the auction process	20 rounds

Table 5.1: Parameter value of Experiment One (Markov model).

Parameter	**Definition**	**Value**
λ	arrival rate of Possion process	1 arrival/hour
c	minimum bid increment	1
r	reserved price of the item	1
max_bid	maximum bid of each BA	5
$prob$	probability of the BA's valuation	According to Figure 5.2
t	closing time of the auction process	20 rounds
n	number of BAs	10

Table 5.2: Parameter value of Experiment One (Simulation environment).

the auction process. The expected revenue calculated by the Markov chain is similar to the one simulated by agents. This means that the result of the Markov chain is a good indicator of expected revenue. The slope of both curves is deep near the beginning of auction and decreases towards the end of auction. The reason is that there are more bidders in the waiting queue when the expected revenue is high. According to Equation 3.3, the value of F^* will decrease which means that the probability of willing to bid is smaller. Hence, the expected revenue will not grow as fast as before. Both curves reach an equilibrium value of five dollars because they are bounded by the maximum value of the cdf. Moreover, the best auction closing time can be found from the graph. It should be around 13 events. After that, the revenue does not change too much but reach an equilibrium value. In a real auction, the seller can reference the simulation result to specify the auction closing time. Using the best auction closing time allows the seller to save the advertising expense but still make the highest profit.

5.3 Experiment Two: A large English auction

Motivations

In this experiment we test the scalability of the Markov model. A single-item English auction with a maximum bid of \$5 is considered small. Therefore, we try to apply the Markov chain model and the software agents to a similar auction process but with a larger maximum bid of \$10.

Experimental Settings

The experimental setup for agent simulation is similar to Experiment One except the use of longer auction time and different BA's cdf. The new auction closing time is 100 rounds and the new cdf can be found in Figure 5.4. On the other hand, the parameter settings for Markov chain model change from $max\_bid = 5$ to $max\_bid = 10$ and from $t = 20$ to $t = 100$ and it uses a new cdf as shown in Figure 5.4.

Experimental Results

Figure 5.5 shows the simulation result. We find that the curves of the expected revenue are close to each other. They grow exponentially and tend to the asymptote of the maximum expected revenue. Therefore, we believe that the behaviour of the SA and the BA is well modeled by the Markov chain and

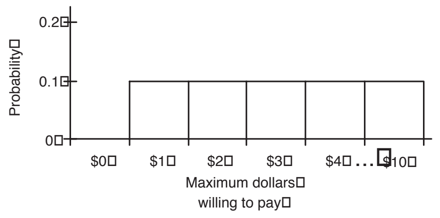


Figure 5.4: cdf of BA's valuations for an item.

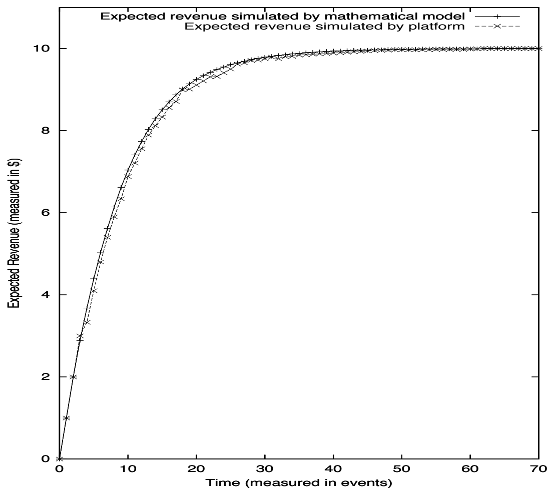


Figure 5.5: Expected revenue over time for a large English auction.

does not affect by the size of the maximum bid and the auction duration. Moreover, we find that the best auction closing time is extended compared with Experiment One. This is due to the higher maximum bid value.

5.4 Experiment Three: A small Dutch auction

Motivations

In this experiment, we try to test the correctness of the Markov chain model for Dutch auction. Again, we try to compare the prediction result with the expected revenue simulated by software agents.

Experimental Settings

We use similar parameter settings as shown in Table 5.1 and Figure 5.2 except for the value of c and r. In Dutch auction, submitting bids are considered as bid decrement. Hence, we change the value of c to -1. Moreover, the negotiation protocol of Dutch auction requires a high reserved price r. Hence, we fix r to \$5 and the selling price will drop from \$5 to \$1. The cdf of BA's valuation of an item is shown in Figure 5.2.

Experimental Results

The experimental result is shown in Figure 5.6. We find that the curves of the expected revenue are close to each other. They drop exponentially and tend to the asymptote of the minimum expected revenue. Therefore, we believe that the behaviour of the SA and the BA is well modeled by the Markov chain and does not affect by the size of the minimum bid and the auction duration. Again, we can find the best auction closing time from the graph. It is around 15 events. Seller can try to specify different selling prices at the beginning of the auction and check their auction closing time. From the result, he can find

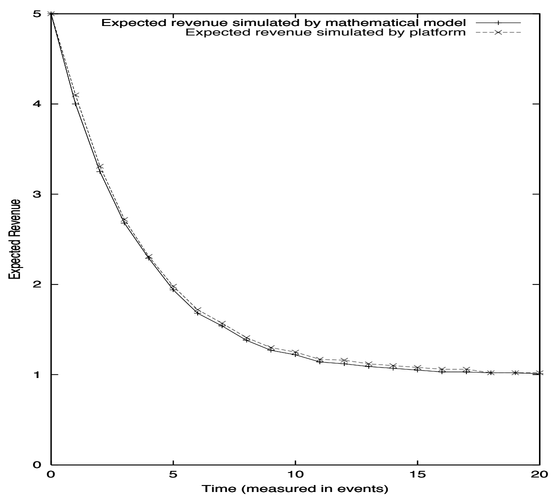


Figure 5.6: Expected revenue over time for a small Dutch auction.

out the best time for him to re-visit the auction site.

5.5 Experiment Four: A large Dutch auction

Motivations

In this experiment, we try to use another set of parameters to the Markov chain model of Dutch auction. The purpose of this experiment is to test the correctness of the Markov chain model under a large auction size.

Experimental Settings

The parameter setting is similar to the one presented in Experiment Two. The auction closing time is fixed at 100 intervals and the reserved price is fixed at \$10. The cdf of BA's valuation of an item is shown in Figure 5.4.

Experimental Results

Figure 5.7 shows the simulation result. We find that the curves are still close to each other. The approximation of Markov chain model is still valid under

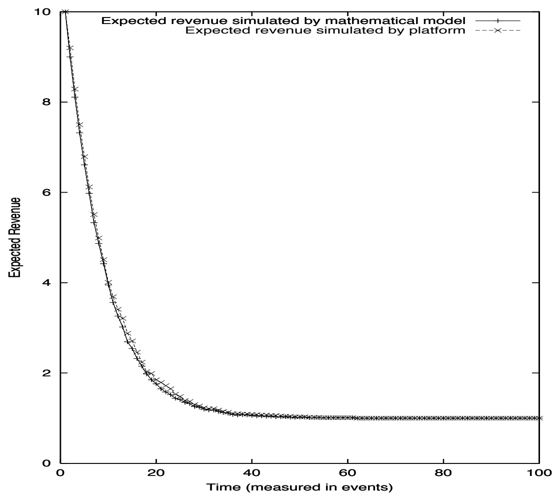


Figure 5.7: Expected revenue over time for a large Dutch auction.

a large auction. Moreover, we find that the Markov chain under estimate the expected revenue. The reason of this is due to the averaging of BA's cdf in the waiting queue.

5.6 Experiment Five: Partial knowledge on other BAs

Motivations

Next, we try to model a wider range of BA's behaviour. In the previous experiments, we assume BAs are independent to each other with no interactions between them. Now, we want to give them partial knowledge: (1) each of them knows the bid paid by the others, and (2) each of them knows other's valuations. In the auction process, the following strategy is used: When a BA is interested in bidding for an item, and it knows that no other BAs are willing to bid, the BA will raise the current bid only by the minimum bid increment. However, if there are other competitors, the BA will compare their bids and find out the maximum bid. Then, if it is still willing to bid, it will offer a new bid which is the maximum bid among other BAs plus the minimum bid increment. This complex strategy cannot be modeled by the Markov chain approach, but it can be simulated by our agents-based platform. Figure 5.8 shows the simulation result.

Experimental Settings

The parameter settings of the Markov chain model are the same as Experiment One. For agent simulation, we allow buyer agent share some knowledges about the auction and use the same parameter settings as Experiment One.

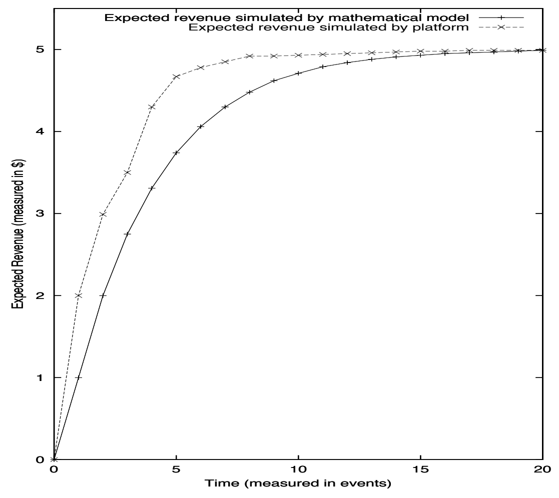


Figure 5.8: Revenue vectors under partial knowledge of other BAs.

Experimental Results

Figure 5.8 shows the simulation result. If we compare the result with the expected revenue calculated by the Markov chain model in Experiment One, we find that the curve simulated by our platform grows and approaches to the asymptote faster than the one predicted by the Markov chain model. This indicates that the auction process will speed up due to the partial knowledge about other BAs in the new bidding strategy.

5.7 Experiment Six: Partial knowledge on closing time

Motivations

In this experiment, we want to show that the expected revenue is affected by the closing time of the auction process. We try to modify the valuation function towards the end of the auction process. Again, the Markov chain model will become extremely complicated to model this behaviour, but our agent-based platform can simulate the results rather easily and faithfully.

Experimental Settings

The parameter settings of the Markov chain model is the same as Experiment One. For agent simulation, we try to update the probability of willing to bid towards to end of the auction. The new probability will be increased by 0.1 at the end of each time interval. Other parameter settings are the same as Experiment One.

Experimental Results

Figure 5.9 shows the simulation result. The experimental result shows that our agent-based platform can model a common phenomenon as in the real auction, i.e. BAs are unwilling to make new bids in the middle of the auction process, but they will submit bids at the very last moment. Figure 5.9 indicates that the growth of the expected revenue is divided into two phases. In the first phase, the expected revenue grows as usual, but it stops growing in the middle of the auction process. The second phase is triggered by the approaching

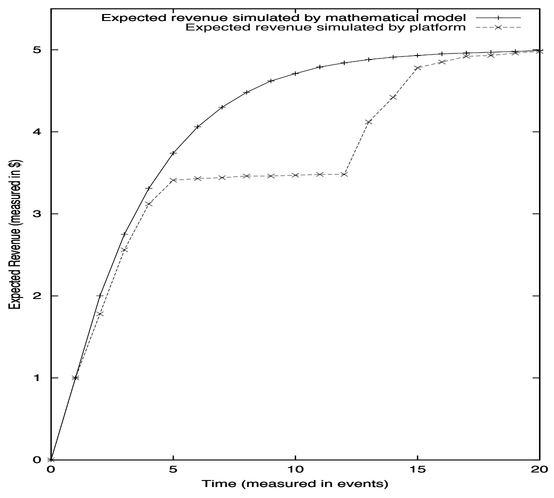


Figure 5.9: Revenue vectors under partial knowledge of closing time.

closing time of the auction process when BAs are actively taking bids, and the revenue grows at a fast speed toward the expected maximum bid.

5.8 Experiment Seven: A small Double auction

Motivations

In this experiment, we want to test the correctness of the Markov chain model for Double auction. A correct model should shows the following features in a graph of expected revenue verus time: (1) The shout prices of seller and buyer should differ significantly from an equilibrium value at the beginning of the Double auction. (2) As time passes by, the shout prices approach the equilibrium value. (3) On subsequent time intervals, the shout prices are initially nearer equilibrium, and approach equilibrium faster.

Experimental Settings

The parameter settings of the Markov chain model are listed in Table 5.3.

Parameter	**Definition**	**Value**
N	number of identical items for sale	20
$time$	number of rounds	100
λ	arrival rate of new buyers	1 arrival/hour
μ	arrival rate of new sellers	1 arrival/hour
N_b	maximum number of buyers in the auction	20
N_s	maximum number of sellers in the auction	20
$F_b(v_i^b)$	cdf of buyer valuations for this item	According to Figure 5.10
$F_s(v_i^s)$	cdf of seller valuations for this item	According to Figure 5.10

Table 5.3: Parameter value of Experiment Seven (Markov model).

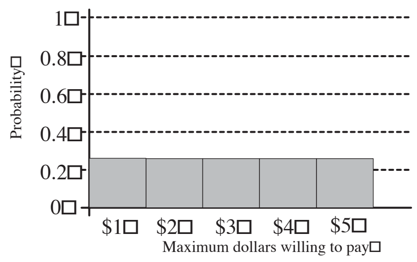


Figure 5.10: Graph of $F_b(p)$ and $F_s(p)$ for initial distribution $\sim U(1,5)$.

Experimental Results

Figure 5.11 shows the result of the Markov chain model. Shout prices of buyer are shown in triangles, while shout prices of seller are shown as squares. Filled symbols mean shouts that were accepted. Lines are drawn to join the sequence of accepted shout prices of buyer and sellers. As the experiment progresses, the offering prices approach equilibrium, and on successive time intervals, there is less variance from, and faster approach to the equilibrium value. The result

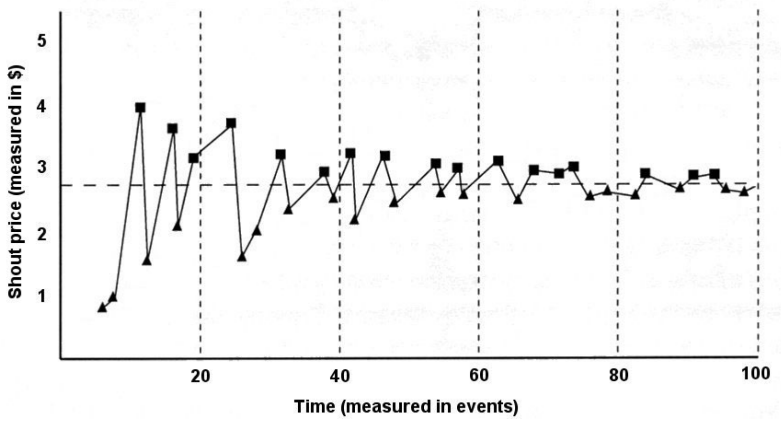


Figure 5.11: Shout price over time for a small Double auction.

matches our expectations. Markov chain does a good job to model a Double auction.

5.9 Comparisons and Discussions

From the above experimental results, we find that the analytical approach based on the Markov chain model has some limitations:

- It does not allow BAs to interact with one another.
- It does not consider the case where the market value of the item may decline over time. In this case, a depreciation term should be included.
- It cannot model whether the auctions run on a weekend or a weekday, if the results can be distinguished.

On the other hand, the advantages of our agent-based simulation environment are:

- It does well in approximating the expected revenue in a single-item auction.

- It allows the analyst to easily scale up the auction complexity in the agent-based simulation with a higher maximum bid value and more BAs, while the result still matches well with that obtained from a mathematical analysis.
- It can model simple as well as complex BA's behaviour and the assumptions used in the simulation are realistic.
- It can be applied to other complex scenarios in an auction process. For example, multi-item auctions, multi-seller auctions, different cdf distributions for BA's valuations, different arrival rates for BAs, and different negotiation constraints.

5.10 Summary

In summary, we find that it is feasible and tractable to model an online auction using a Markov chain model. In an online auction, an agent in its lifetime meets many different agents, and as a result modeling individual agent with simple behaviours tends to be incorrect and incomplete. In such a case, modeling the overall auction process as a Markov chain is more accurate than modeling the interior reasoning of each agent participating in the auction. However, Markov chain cannot model complex BA's behaviours. Our agent-based platform, on the other hand, can provide a more sophisticated simulation environment. We can model a wider range of BA's behaviour (and even seller's behaviour) than what the Markov chain model can cope with. Furthermore, the assumptions used in the simulation approach can be made as realistic as possible.

Chapter 6

Conclusion

In this thesis, (1) we have presented the architecture for an agent-based platform which can perform online auctions. The architecture design focuses on scalability, flexibility, and reusability. Agent developers don't have to understand the system flow and system components in advance. The only thing they should understand is the agent API provided by the platform. Actually, the agent API is a Java package for agent developers to import in their agent classes which can shorten the development time. Moreover, the agent API can be extended easily to support other E-commerce applications. (2) We discussed the negotiation protocol between agents. It includes English auction, Dutch auction, and Double auction. A set of negotiation messages have been developed according to the auction properties. Agents can follow the negotiation messages to communicate, and perform auctions in the interactive mode. (3) Moreover, we used software agents in batch mode to simulate the expected revenue on the platform and compared it with the value calculated by Markov chain model. The batch mode processing is equivalent to the creation of a pool of agents and give them parameters. Then they will perform computation without any human interventions. Seller can use this feature to find out the best closing time of the auction so as to maximize his profit. (4) Experimental results show that simple bidder's behaviour can be well analyzed using a Markov chain model as well as our agent-based simulation approach.

However, it is difficult to extend the Markov chain model to cover a wide range of bidder's behaviour. Nonetheless, our platform can deal with this problem easily. Furthermore, the agent-based approach can be extended and refined with more realistic scenarios for automatic agent-based simulations, allowing us to construct a dynamic and diverse environment for a variety of E-commerce applications.

Appendix A

Markov Chain for English Auction

A.1 Graphs of F and F^* Functions

Figure A.1 shows $F(p) \sim U(1,5)$.

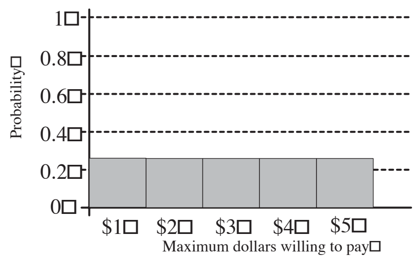


Figure A.1: Graph of $F(p)$ for initial distribution $\sim U(1,5)$.

For $F^*(p=2)$, $L_{max} = 0$. Hence, $F^*(p=2)$ equals to $F(p)$, i.e. If there are no bidders in the waiting queue, we have no prior information about them. The result is shown in Figure A.2.

For $F^*(p=3)$, $L_{max} = 1$. We must have the following sequence of events:

- b_1 arrives and, finding he can afford a successful bid, bids $\$r = \$1$.

- b_2 arrives and, finding he can afford a successful bid, bids $\$r + \$1 = \$2$, thus bumping b_1 to the waiting queue.

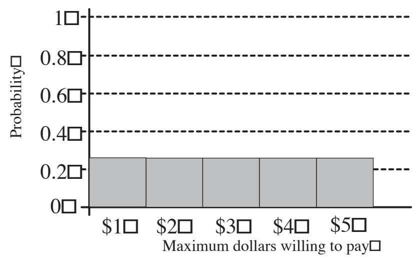


Figure A.2: Graph of $F^*(p=2)$.

- The only information we have about b_1 from this scenario is that he was willing to pay \$1; since all bidders have $F(p) \sim U(1,5)$, this tell us nothing new. Hence, $F^*(p=3) = F(p)$.

Mathematically,

$$\begin{aligned}
F^*(p=3) &= \frac{L_{max}-1}{L_{max}} F^*(p-1) + \frac{1}{L_{max}} F(v_0 | v_i \geq p-1) && \text{(A.1)}\\
&= \frac{1-1}{1} F^*(p=2) + \frac{1}{1} F(v_0 | v_i \geq p-1) && \text{(A.2)}\\
&= 0 + F(v_0 | v_i \geq p-1) && \text{(A.3)}\\
&= F(p) && \text{(A.4)}\\
&\sim U(1,5). && \text{(A.5)}
\end{aligned}$$

The result is shown in Figure A.3.

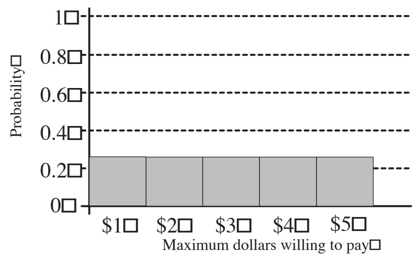


Figure A.3: Graph of $F^*(p=3)$.

For $F^*(p=4)$, $L_{max}=2$. There are two bidders in the waiting queue. In a process similar to that described for $F^*(p=3)$,

- b_1 bids \$1.

- b_2 bids \$2, sending b_1 to the waiting queue.
- b_3 bids \$3, sending b_2 to the waiting queue.
- the next awakener from the waiting queue will be b_1 with probability $\frac{1}{2}$ and b_2 with probability $\frac{1}{2}$. Hence, the distribution function of the next awakener's valuation equals:

$$\begin{aligned} F^*(p=4) &= \frac{L_{max}-1}{L_{max}}F^*(p-1)+\frac{1}{L_{max}}F(v_0|v_i \geq p-1) && \text{(A.6)} \\ &= \frac{2-1}{2}F^*(p=3)+\frac{1}{2}F(v_0|v_i \geq 4-1) && \text{(A.7)} \\ &= \frac{1}{2}F^*(p=3)+\frac{1}{2}F(v_0|v_i \geq 3). && \text{(A.8)} \end{aligned}$$

The result is shown in Figure A.4.

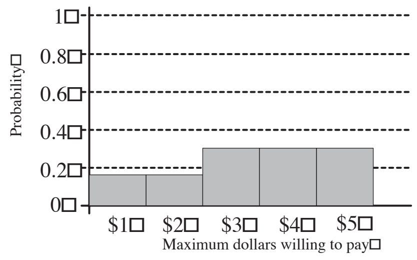


Figure A.4: Graph of $F^*(p=4)$.

For $F^*(p=5)$, $L_{max}=3$. There are three bidders in the waiting queue. In a process similar to that described for $F^*(p=4)$,

- b_1 bids \$1.
- b_2 bids \$2, sending b_1 to the waiting queue.
- b_3 bids \$3, sending b_2 to the waiting queue.
- b_4 bids \$4, sending b_3 to the waiting queue.
- the next awakener from the waiting queue will be b_1 with probability $\frac{1}{3}$, b_2 with probability $\frac{1}{3}$ and b_3 with probability $\frac{1}{3}$. Hence, the distribution function of the next awakener's valuation equals:

$$F^*(p=5) = \frac{L_{max}-1}{L_{max}} F^*(p-1) + \frac{1}{L_{max}} F(v_0|v_i \geq p-1) \quad (A.9)$$
$$= \frac{3-1}{2} F^*(p=4) + \frac{1}{2} F(v_0|v_i \geq 5-1) \quad (A.10)$$
$$= \frac{2}{3} F^*(p=4) + \frac{1}{2} F(v_0|v_i \geq 4). \quad (A.11)$$

The result is shown in Figure A.5.

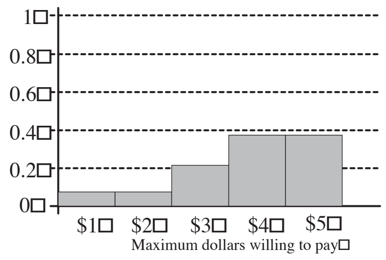


Figure A.5: Graph of $F^*(p=5)$.

For $F^*(p=6)$, it is undefined because there will never be a going price of \$6 from the current winner.

A.2 Solution of a Small Example

This section will solve a small example using the previous formulation. First, it will outline the variables and their values; Second, it will calculate the F^* functions for approximating the waiting queue behavior; Third, it will show some progressions of possible price vectors.

Variables:

N = 1 item for sale

p = going price

r = \$1

c = \$1 minimum bid increment

t = the index of events

$L = 0$

$\lambda = 1$

$\mu = 1/3$

$F(p) \sim U(1,5)$

The state space is the space $\{p \times L \times t\}$, where $p \in (0,5)$; $L \in (0,4)$ and $t \in (0,\infty)$.

A.2.1 Computation of Transition Probabilities

The F and F^* functions are calculated in the previous section. Using these values of F and F^* to calculate the transition probability as outlined in Table 3.2, the **P** matrix can be computed.

Several states in $\{p \times L\}$ are infeasible, because the price could not reach the level of \$$p$ without having at least $\frac{p-r}{c}$ bids. Hence, we can collapse the **P** matrix without showing the infeasible areas and gives the matrix shown in Figure A.6.

Calculated for $\lambda = 1$ and $\mu = 1/3$.

State space is denoted by (p, L).

$$\begin{array}{c|ccccccccccc|}
 & (1,0) & (2,0) & (3,0) & (3,1) & (4,0) & (4,1) & (4,2) & (5,0) & (5,1) & (5,2) & (5,3) \\
(1,0) & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
(2,0) & 0.00 & 0.25 & 0.00 & 0.75 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
(3,0) & 0.00 & 0.00 & 0.50 & 0.00 & 0.00 & 0.50 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
(3,1) & 0.00 & 0.00 & 0.13 & 0.38 & 0.00 & 0.13 & 0.38 & 0.00 & 0.00 & 0.00 & 0.00 \\
(4,0) & 0.00 & 0.00 & 0.00 & 0.00 & 0.75 & 0.00 & 0.00 & 0.00 & 0.25 & 0.00 & 0.00 \\
(4,1) & 0.00 & 0.00 & 0.00 & 0.00 & 0.18 & 0.56 & 0.00 & 0.00 & 0.07 & 0.19 & 0.00 \\
(4,2) & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.28 & 0.45 & 0.00 & 0.00 & 0.12 & 0.15 \\
(5,0) & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
(5,1) & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
(5,2) & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\
(5,3) & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00
\end{array}$$

Figure A.6: **P** matrix with transition probabilities of a small auction.

A.2.2 Progression of the Price Vector

Let π_t be the probability vector describing the state of the auction after the t^{th} event has occurred (with an event being either a new arrival or an awakening from the queue, regardless of whether he decides to bid or not).

At $t = 0$, the auction state is:

	(1,0)	(2,0)	(3,0)	(3,1)	(4,0)	(4,1)	(4,2)	(5,0)	(5,1)	(5,2)	(5,3)
π_0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure A.7: Auction state at t=0.

In other words, this type of auction will begin with probability 1 in state (0,0).

At $t = 1$, the auction state is represented by $\pi_1 = \pi_0 \cdot \mathbf{P}$.

	(1,0)	(2,0)	(3,0)	(3,1)	(4,0)	(4,1)	(4,2)	(5,0)	(5,1)	(5,2)	(5,3)
π_1	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure A.8: Auction state at t=1.

At $t = 2$, the auction state is represented by $\pi_2 = \pi_0 \cdot \mathbf{P}^2$.

	(1,0)	(2,0)	(3,0)	(3,1)	(4,0)	(4,1)	(4,2)	(5,0)	(5,1)	(5,2)	(5,3)
π_2	0.00	0.25	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure A.9: Auction state at t=2.

The probabilistic progression of the auction for the next several time periods can be calculated in a similar way. Figure A.10 shows the calculation for the first 20 time intervals of the auction.

Figure A.10 provides valuable insight to the dynamics nature of an online English auction. The π_6 row (shaded) gives the probability distribution of the going price of the auction after the sixth event has occurred. From this result, we can calculate the probability that the going price will be \$4 after the sixth event has occured, which will be $0.09 + 0.28 + 0.13 = 0.50$. For an one item auction, the selling price will be one bid increment below the going price, so a revenue of \$3 would be expected with the probability of 0.50.

If we perform similar summations over the 20 time intervals for all possible price levels, we can get the following information which is shown in Figure A.11.

State space is denoted by (p, [illegible])

Time	(1,0)	(2,0)	(3,0)	(3,1)	(4,0)	(4,1)	(4,2)	(5,0)	(5,1)	(5,2)	(5,3)
0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.25	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.06	0.09	0.47	0.00	0.09	0.28	0.00	0.00	0.00	0.00
4	0.00	0.02	0.11	0.22	0.02	0.24	0.30	0.00	0.01	0.05	0.04
5	0.00	0.00	0.08	0.10	0.05	0.30	0.22	0.01	0.07	0.12	0.05
6	0.00	0.00	0.05	0.04	0.09	0.28	0.13	0.08	0.16	0.13	0.03
7	0.00	0.00	0.03	0.02	0.12	0.23	0.08	0.24	0.17	0.10	0.02
8	0.00	0.00	0.02	0.01	0.13	0.17	0.04	0.41	0.15	0.07	0.01
9	0.00	0.00	0.01	0.00	0.13	0.11	0.02	0.56	0.12	0.05	0.01
10	0.00	0.00	0.00	0.00	0.12	0.08	0.01	0.67	0.09	0.03	0.00
11	0.00	0.00	0.00	0.00	0.10	0.05	0.00	0.76	0.06	0.02	0.00
12	0.00	0.00	0.00	0.00	0.08	0.03	0.00	0.82	0.05	0.01	0.00
13	0.00	0.00	0.00	0.00	0.07	0.02	0.00	0.87	0.03	0.01	0.00
14	0.00	0.00	0.00	0.00	0.05	0.01	0.00	0.91	0.02	0.00	0.00
15	0.00	0.00	0.00	0.00	0.04	0.01	0.00	0.93	0.02	0.00	0.00
16	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.95	0.01	0.00	0.00
17	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.96	0.01	0.00	0.00
18	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.97	0.01	0.00	0.00
19	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.98	0.01	0.00	0.00
20	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.98	0.00	0.00	0.00

Figure A.10: Probabilistic progression of an Auction across its state space.

The progression from an initial revenue of \$0 (corresponding to a going price of \$1) to the final revenue of \$4 (corresponding to a going price of \$5) can be clearly seen. The different price vectors can be plotted over time, as shown in Figure A.12.

Time	Prob (rev=\$0)	Prob (rev=\$1)	Prob (rev=\$2)	Prob (rev=\$3)	Prob (rev=\$4)
0	1.00	0.00	0.00	0.00	0.00
1	0.00	1.00	0.00	0.00	0.00
2	0.00	0.25	0.75	0.00	0.00
3	0.00	0.06	0.56	0.38	0.00
4	0.00	0.02	0.33	0.56	0.10
5	0.00	0.00	0.18	0.57	0.25
6	0.00	0.00	0.09	0.51	0.40
7	0.00	0.00	0.05	0.42	0.53
8	0.00	0.00	0.02	0.34	0.64
9	0.00	0.00	0.01	0.26	0.73
10	0.00	0.00	0.01	0.20	0.79
11	0.00	0.00	0.00	0.15	0.84
12	0.00	0.00	0.00	0.12	0.88
13	0.00	0.00	0.00	0.09	0.91
14	0.00	0.00	0.00	0.07	0.93
15	0.00	0.00	0.00	0.05	0.95
16	0.00	0.00	0.00	0.04	0.96
17	0.00	0.00	0.00	0.03	0.97
18	0.00	0.00	0.00	0.02	0.98
19	0.00	0.00	0.00	0.02	0.98
20	0.00	0.00	0.00	0.01	0.99

Figure A.11: Probabilistic progression of auction revenue over time.

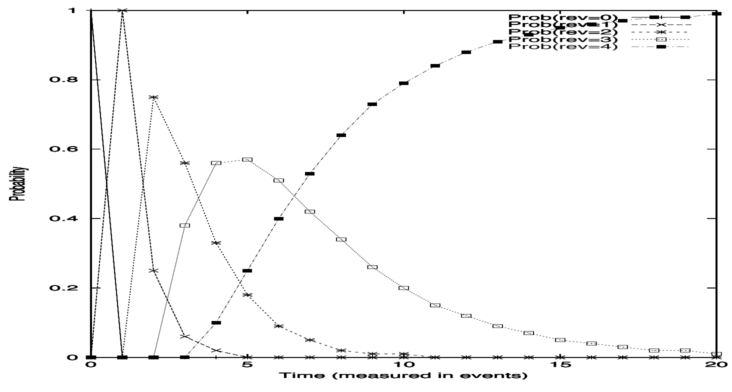


Figure A.12: Progression of individual revenue vectors over time.

Appendix B

Markov Chain for Double Auction

B.1 Graphs of $F_b(v_i^b)$ and $F_s(v_i^s)$ Functions

Figure B.1 shows $F_b(v_i^b)$ and $F_s(v_i^s) \sim U(1,5)$.

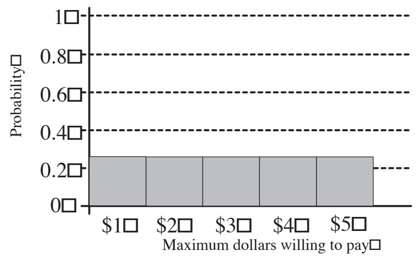


Figure B.1: Graph of $F_b(v_i^b)$ and $F_s(v_i^s)$ for initial distribution $\sim U(1,5)$.

$F_b(v_i^b) \sim U(1,5)$ means that buyer has the same probability of valuation for an item at different prices. The lowest value of accepting a sell offer is \$1 and the highest value of accepting a sell offer is \$5.

In contrast, $F_s(v_i^s) \sim U(1,5)$ means that seller has the same probability of valuation for an item at different prices. The lowest value of accepting a buy offer is \$1 and the highest value of accepting a buy offer is \$5.

B.2 Solution of a Small Example

This section will solve a small example using the previous formulation. First, it will outline the variables and their values; Second, it will compute the transition probability matrix **P**.

Variables:

$N = 5$ item for sale

$p =$ current shout price

$N_b = 5$

$N_s = 5$

$t =$ the index of events

$L = 0$

$\lambda = 1/3$

$\mu = 1/3$

$F_s(v_i^s) \sim U(1, 5)$

$F_b(v_i^b) \sim U(1, 5)$

The state space is the space $\{N_s \times N_b \times t\}$, where $N_s \in (0, 5)$; $N_b \in (0, 5)$ and $t \in (0, \infty)$.

B.2.1 Computation of Transition Probabilities

Using the above parameters and the transition probability as outlined in Table 3.7, the **P** matrix can be computed. Figure B.2 shows the corresponding state space in the **P** matrix. It is represented as five smaller matrix which are $A1$, $A2$, $A3$, $A4$ and $A5$. Figure B.3 shows the complete **P** matrix which is composed of $A1$, $A2$, $A3$, $A4$ and $A5$.

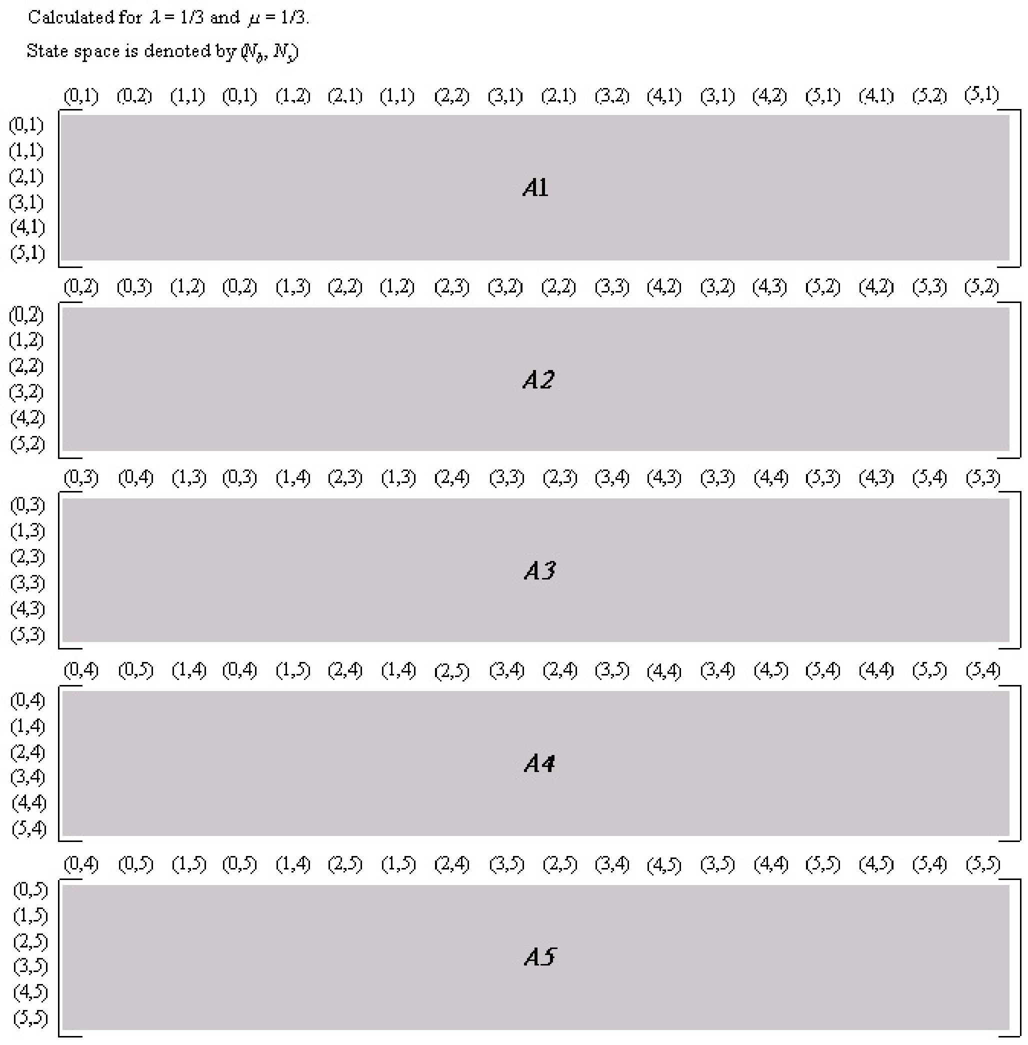


Figure B.2: State space of **P** matrix.

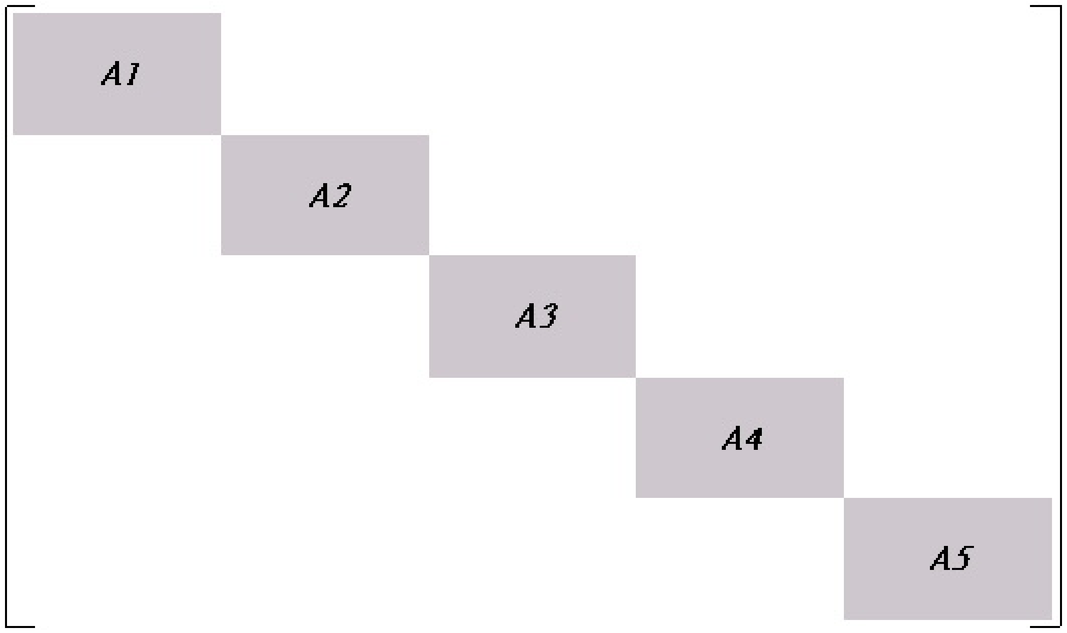


Figure B.3: The complete **P** matrix.

Appendix C

Implementation Details

C.1 Tables Design in MySQL

The following tables are stored in the MySQL database. Some of them are needed in transactions and others are for storing system parameters.

Table of adminblock : This table stores the setting of the administrator block.

Field	**Type**	**Key**	**Default**
title	varchar(60)	YES	NULL
content	text	YES	NULL

Table C.1: Table of admin block.

- title : Title of the admin block
- content : Description of the admin block

Table of agent : This table stores agent data.

- agent_owner : User id of the owner of this agent
- agent_type : Specify the type of agent

Table of mainblock : This table stores the setting of the main block.

Field	**Type**	**Key**	**Default**
agent_owner	varchar(25)	NULL	NULL
agent_type	char(1)	NULL	NULL

Table C.2: Table of agent.

Field	**Type**	**Key**	**Default**
title	varchar(60)	YES	NULL
content	text	YES	NULL

Table C.3: Table of mainblock.

- title : Title of the main block
- content : Description of the main block

Table of settings : This table stores the setting of the web page.

- sitename : Name of the web site
- site_url : URL of the web site
- site_logo : File name of the site logo
- site_slogan : Slogan of the web site
- startdate : Starting date of the web site
- adminmail : Email address of the administrator
- Default_Theme : Name of the default theme
- myIP : IP address of the web site
- language : Default langauge of the web site
- locale : Default locale of the web site

Field	**Type**	**Key**	**Default**
sitename	varchar(100)	NULL	NULL
site_url	varchar(200)	NULL	NULL
site_logo	varchar(25)	NULL	NULL
site_slogan	varchar(100)	NULL	NULL
startdate	varchar(30)	NULL	NULL
adminmail	varchar(100)	NULL	NULL
Default_Theme	varchar(25)	NULL	NULL
myIP	varchar(25)	NULL	NULL
language	varchar(30)	NULL	NULL
locale	varchar(10)	NULL	NULL
setCookies	int(1)	NULL	0
userimg	varchar(100)	NULL	NULL
adminimg	varchar(100)	NULL	NULL
site_font	varchar(100)	NULL	NULL

Table C.4: Table of settings.

- setCookies : 1 = set, 0 = not set yet
- userimg : File name of the user image
- adminimg : File name of the administrator image
- site_font : Default font of the web site

Table of users : This table stores the setting of each user.

Field	**Type**	**Key**	**Default**
uid	int(11)	PRI	0
name	varchar(60)	NULL	NULL
uname	varchar(25)	NULL	NULL
email	varchar(60)	NULL	NULL
url	varchar(100)	NULL	NULL
password	varchar(40)	NULL	NULL
theme	varchar(255)	NULL	NULL

Table C.5: Table of users.

- uid : User Identifier. It is assigned by system.
- name : Last name.
- uname : First Name.
- email : Email address of the user.
- url : URL of the user's home page.
- password : User Password. It is encrypted by DSE.
- theme : User's default theme.

Table of product : This table stores settings of each product.

Field	**Type**	**Key**	**Default**
itemID	varchar(15)	PRI	NOT NULL
title	varchar(45)	NULL	NULL
description	varchar(100)	NULL	NULL
quant	char(2)	NULL	NULL
minbid	char(3)	NULL	NULL
duration	char(2)	NULL	NULL
reserve	charchar(6)	NULL	NULL
startDay	char(2)	NULL	NULL
startHour	char(2)	NULL	NULL
startMinute	char(2)	NULL	NULL
startSecond	char(2)	NULL	NULL
numOfBid	char(2)	NULL	NULL
seller	varchar(10)	NULL	NULL

Table C.6: Table of product.

- itemID : Product ID. It is assigned by system.
- title : Product Name.
- description: Product Description.

- quant : Product Quantity.
- minbid : Minimum Bid of that product.
- duration : Product selling period.
- reserve : Reserved price of that product.
- startDay : Selling Day for calculating closing time.
- startHour: Selling Hour for calculating closing time.
- startMinute : Selling Minute for calculating closing time.
- startSecond : Selling Second for calculating closing time.
- numOfBid : Total number of bids so far.
- seller : Name of the Seller.

C.2 Sample Apache Configuration File

```
###############################################################
#          Apache-Tomcat Smart Context Redirection           #
###############################################################
LoadModule jserv_module modules/ApacheModuleJServ.dll
<IfModule mod_jserv.c>
ApJServManual on
ApJServDefaultProtocol ajpv12
ApJServSecretKey DISABLED
ApJServMountCopy on
ApJServLogLevel notice

ApJServDefaultHost localhost
```

```
ApJServDefaultPort 8007


#

# Mounting a single smart context:

#

# (1) Make Apache know about the context location.

Alias /examples c:/jakarta-tomcat/webapps/examples

# (2) Optional, customize Apache context service.

<Directory "c:/jakarta-tomcat/webapps/examples">

    Options Indexes FollowSymLinks

# (2a) No directory indexing for the context root.

#     Options -Indexes

# (2b) Set index.jsp to be the directory index file.

#     DirectoryIndex index.jsp

</Directory>

# (3) Protect the WEB-INF directory from tampering.

<Location /examples/WEB-INF/>

    AllowOverride None

    deny from all

</Location>

# (4) Instructing Apache to send all the .jsp files under

# the context to the jserv servlet handler.

<LocationMatch /examples/*.jsp>

    SetHandler jserv-servlet

</LocationMatch>

# (5) Direct known servlet URLs to Tomcat.

ApJServMount /examples/servlet /examples


# (6) Optional, direct servlet only contexts to Tomcat.
```

```
ApJServMount /servlet /ROOT

</IfModule>
```

Bibliography

[1] R. Hake, E-commerce, a world of opportunity, in *The British Computer Society*, 1999.

[2] D. Lucking-Reiley, Auctions on the internet: What's being auctioned, and how?, in *Journal of Industrial Economics*, volume 48, pages 227–252, 2000.

[3] http://www.ebay.com, eBay.

[4] T. Sandholm and V. Lesser, Issues in automated negotiation and electronic commerce: Extending the contract net framework, in *Proceedings of the 1st International Conference on Multiagent Systems*, pages 328–335, San Francisco, California, 1995.

[5] J. Oliver, A machine learning approach to automated negotiation and prospects for electronic commerce, in *Journal of Management Information Systems*, volume 13, pages 83–112, 1996.

[6] C. Sierra, P. Faratin, and N. Jennings, A service-oriented negotiation model between autonomous agents, in *Proceedings of the 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, pages 17–35, Ronneby, Sweden, 1997.

[7] J. Lee, ICOMA: an open infrastructure for agentbased intelligent electronic commerce on the internet, in *Proceedings of the International Conference on Parallel and Distributed Systems*, pages 648–655, Iwate, Japan, 1997.

[8] H. L. Cardoso, A multi-agent system for electronic commerce including adaptive strategic behaviours, in *Proceedings of the 9th Portuguese Conference on Progress in Artificial Intelligence*, pages 256–266, Evora, Portugal, 1999.

[9] P. Wurman and M. Wellman, A control architecture for flexible internet auction servers, in *Proceedings of the First IAC Workshop on Internet-based Negotiation Technologies*, Yorktown Heights, 1999.

[10] S. Su, C. Huang, and J. Hammer, A replicable web-based negotiation server for ecommerce, in *Proceedings of the 33rd International Conference on System Sciences*, Hawaii, 2000.

[11] H. Leung and M. He, Agents in e-commerce: State of the art, under consideration for publication in Knowledge and Information Systems.

[12] G. Weiss, *Multiagent systems*, The MIT Press, Cambridge, London, 2000.

[13] B. Carrie and A. Segev, Electronic negotiation through internet-based auctions, in *CMIT Working Paper 96*, 1996.

[14] http://www.priceline.com, Priceline.com.

[15] http://www.cathypacific.com, Cathy Pacific.

[16] A. Lee, M. Lyu, and I. King, Agent-based multimedia data sharing platform, in *Proceedings of the International Symposium on Information Systems and Engineering*, Las Vegas, 2001.

[17] A. Lee, M. Lyu, and I. King, An agent-based platform for online auctions, in *Proceedings of the International Conference on Internet Computing*, Las Vegas, 2001.

[18] J. L. E. Turban and D. Chung, *Electronic Commerce A Managerial Perspective*, chapter 1, Prentice Hall, 1999.

[19] R. Murch and T. Johnson, *Intelligent Software Agents*, chapter 14, pages 131–137, Prentice Hall, 1999.

[20] N. Jennings, K. Sycara, and M. Wooldridge, A roadmap of agent research and development, in *Journal of Autonomous Agent and Multi-Agent Systems*, pages 7–38, 1998.

[21] J. R. Oliver, A machine learning approach to automated negotiation and prospects for electronic commerce, in *Journal of Management Information Systems*, volume 13, pages 83–112, 1997.

[22] http://www.jango.com, Jango.

[23] http://www.personalogic.com, Personalogic.

[24] http://www.firefly.com, Firefly.

[25] http://www.csta.ac.com, Bargainfinder.

[26] http://kasbah.media.mit.edu, Jango and kasbah.

[27] M. Ma, Agents in e-commerce, in *Communications of the ACM*, volume 42, pages 78–80, 1999.

[28] http://www.onsale.com, Onsale.

[29] H. J. Muller, *Negotiation Principles*, chapter 7, Foundations of Distributed Artificial Intelligence, 1996.

[30] R. P. McAfee and J. McMillan, Auctions and bidding, in *Journal of Economic Literature*, pages 699–738, 1987.

[31] P. Milgrom, Putting auction theory to work: the simultaneous ascending auction, in *Journal of Political Economy*, volume 108, Dept. Economics, Standford University, 2000.

[32] J. Rust and D. Friedman, *The Double Auction Market: Institutions, Theories and Evidence*, chapter 1, pages 1–25, Addison-Wesley, 1993.

[33] W. P. B. Sunju Park, Edmund H. Durfee, An adaptive agent bidding strategy based on stochastic modeling, in *Proceedings of the Third International Conference on Autonomous Agents*, Seattle, Washington, 1999.

[34] Y. Fujishima, D. McAdams, and Y. Shoham, Speeding up ascending-bid auctions, in *Proceedings of the Thirteenth European Conference on Artifical Intelligence*, pages 554–559, Bringhton, UK, 1998.

[35] R. Wilson, Strategic analysis of auctions, in *Handbook of Game Theory*, volume 1, 1992.

[36] R. Gray, Agent tcl: A flexible and secure mobile-agent system, in *Proceedings of the Fourth Annual Tcl/Tk Workshop*, pages 9–23, Monteray, California, 1996, http://www.cs.dartmouth.edu/agent/papers.html.

[37] D. Johanson, An introduction to the tacoma distributed system, Technical report, Univ. of Tromso, Dept. of CS, June 1995.

[38] D. Lange and D. Chang, Aglets workbench - programming mobile agents in java, in *White Paper, IBM Corporation, Japan*, 1996.

[39] J. Conlisk, The law of supply and demand as a law of markov chains, in *Jorunal of Economic Theory*, volume 26, pages 1–16, 1982.

[40] C. Turner, P. Startz, and C. Nelson, A markov model of heteroskedasticity, risk, and learning in the stock market, in *Jorunal of Financial Economics*, volume 25, pages 3–22, 1989.

[41] T. Kimoto and K. Asakawa, Stock market prediction system with modular neural networks, in *Proceedings of the IJCNN*, pages 1–6, San Diego, 1990.

[42] B. Freisleben, Stock market prediction with backpropagation networks, in *Proceedings of the 5th International Conference of Industrial and Engineering Applications of Artificial Intelligent and Expert System*, pages 451–460, Paderborn, Germany, 1992.

[43] G. Jang and F. Lai, Intelligent stock market prediction system using dual adaptive structure neural networks, in *Proceedings of the Second International Conference on Artificial Intelligent Applications*, pages 88–97, Gaithersbury, 1993.

[44] P. Wurman, M. Wellman, and W. Walsh, The michigan internet auctionbot: A configurable auction server for human and software agents, in *Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis/St. Paul, 1998.

[45] R. Glushko, J. Tenenbaum, and B. Meltzer, An xml framework for agent-based e-commerce, in *Communications of the ACM*, volume 42, pages 106–114, 1999.

[46] R. Guttman and P. Maes, Agent-mediated integrative negotiation for retail electronic commerce, in *Agent Mediated Electronic Commerce*, volume 1571, pages 70–90, 1998.

[47] R. Guttman, A. Moukas, and P. Maes, Agent-mediated electronic commerce: A survey, in *The Knowledge Engineering Review*, pages 147–159, 1998.

[48] T. Liang and J. Huang, A framework for applying intelligent agents to support electronic trading, in *Decision Support Systems*, pages 305–317, 2000.

[49] M. Breugst, I. Busse, S. Covaci, and T. Magedanz, Grasshopper - a mobile agent platform for in based service environments, in *Proceedings of the IEEE IN Workshop 1998*, pages 279–290, Bordeaux, France, 1998.

[50] http://www.php.net, PHP.

[51] http://www.mysql.com, MySQL.

[52] B. Carrie and A. Segev, Optimal design of internet-based auctions, in *CMIT Working Paper 99*, 1999.