Trading Agents

A Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Technology

by
Amit Kumar Manjhi

to the
Department of Computer Science & Engineering
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
April, 2001

Certificate

Certified that the work contained in the report entitled "*Trading Agents*", by *Amit Kumar Manjhi*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Dheeraj Sanghi

April, 2001

Abstract

The rise of electronic media and information economy heralds a future in which autonomous agents would profoundly influence electronic markets. We, in this project, have developed a platform for agent inter–communication. Agents in our framework are economically motivated and on being given the task of purchasing/selling some things, they scan all possible choices, employ various strategies and find out the most optimal peer to trade with. Our platform also allows humans to communicate their mind–set to agents in a very generic fashion and the ensuing transactions that it permits are also akin to the complex transactions that take place in the real world. A pleasant GUI has also been developed to enable first–time users to play with the software. Lastly, to demonstrate the generality of the platform and to quantify some of the likely impacts of proliferation of shopbots and pricebots – adaptive price–setting agents which firms may employ to take advantage of the growing community of shopbots, we have simulated the behavior of shopbots and pricebots and analyzed the results obtained.

Acknowledgments

I would like to express my gratitude to my supervisor Dr. Dheeraj Sanghi for his guidance, motivation and invaluable suggestions at all stages of this project. I would also like to thank him for his enthusiasm throughout the discussions I had with him during the course of this project. Working with him was a real learning experience – apart from gaining academic knowledge, it helped me hone my presentation skills and in the process, I also picked cues on effective time management.

I am also thankful to my batch mates for all the cheerful support.

Lastly, I would also like to thank my parents for their blessings, continued support and encouragement all through the project.

Contents

Chapter 1

Introduction

The rise of electronic media and information economy heralds a future in which autonomous agents would profoundly influence electronic markets. In the information economy of today, the plenitude and low cost of up–to–date information has enabled consumers, both humans and agents, to be better informed about products and prices. Search costs have reduced dramatically and it has become feasible for a buyer to compare prices across various sellers of the same good. For example, it is easy to compare prices of a book at various book–selling sites like Barnes & Nobles, Amazon.com and Borders. Typically, now the buyer visits a few sites and chooses the one with the lowest price. But the number of sites a particular human being can visit is limited by the time he/she can invest in the monotonous job of searching vis–a–vis the additional gains incurred by such an exhaustive search. A better way would be to use one of the many comparative shopping agents or *shopbots* for this purpose. Shopbots of today merely collect pricing information from vendors and then the *human master* chooses the final party with whom to transact. These represent the simplest of agents. Agents, which are basically specialized pieces of software that are semi–autonomous and suitable for a wide array of information processing tasks, can be used for a multitude of tasks – varying in complexity from doing simple exhaustive searches for a product's prices to negotiating complex transactions on behalf of their human owners.

Agents would continue to evolve, since the tremendous pressures that have fueled the rapid growth of electronic commerce in the last few years would continue to drive automation, and some of this automation will be encapsulated in the form of software agents. As they continue to evolve, growing in sophistication and variety, agents will begin to interact, not just with humans but with one another. Interaction among agents would be supported by a number of efforts that seek to standardize the agent communication languages (FIPA[21] and KQML[20]) and interaction protocols, myriad attempts to establish standard ontologies for numerous products and markets (CommerceNet[12]), and the development of various methods for e–payments such as IBM MicroPayments[13].

As a result of their continuing evolution and as they become more pervasive and intelligent, agents will invariably end up making decisions on behalf of their human owners rather than being limited to the searching job[2]. The ability of agents to process large chunks of information and their ability to respond quickly to a change in the pricing environment would prove to be immensely useful in tomorrow's world. As competitive pressure and the need to respond quickly increases, all parties (both consumers and producers) would begin using agents for carrying out transactions on their behalves. As more and more *shopbots* – buyer agents, come into play, sellers will have to rely increasingly on *pricebots* – seller agents that can adjust prices automatically on the seller's behalf in response to market conditions. In such an agent based economy, it would be both interesting and informative to observe how a group of seller and buyer agents behave when they have to achieve a certain goal.

In this thesis, we propose and implement a framework for agent inter–communication. All our agents are economically motivated, and on being given the task of purchasing/selling some things, they scan all possible choices, employ various strategies and find out the most optimal peer to transact with. A transaction in our model can be very general and it is possible to capture the associations and constraints between real life objects in our model. A GUI front–end has also been developed to make data entry and modification easier, and to increase the user–friendliness of the software. We have also developed a framework for carrying out simulations and have used it to carry out and analyze the interaction between pricebots and shopbots.

The rest of the thesis is organized as follows: in chapter 2, we discuss the general concepts behind this project and how ideas of transactions varying in their complexities can be abstracted out, chapter 3 gives an overview of the implementation, first an overview of the working of the main program and then some details and screen shots of the GUI, in chapter 4, we first talk about the simulations we have done, and then present the actual results and its analysis, chapter 5 provides the conclusion and directions in which this work could be extended.

Chapter 2

General concepts

Before looking into what is necessary for agents to transact and how agents should transact, we try to characterize a *transaction* itself.

2.1 World models

Any transaction takes place only when a buyer and a seller agree to exchange some commodity at a mutually agreeable price. Transactions would be performed differently if we consider different worlds or models[11]. Consider the following different models:

1. Every market day people (agents, traders etc.) change, while the joint distribution of their endowments and tastes is stationary.

2. People stay the same: they have been present on past market days and they except to be present on future market days. But, they neither recognize anyone not expect to be recognized by anyone.

3. People can identify at least some others and be identified by others.

World 1 is the world of instantaneous transactions. Moving to world 2, in which there are permanent actors with a past and a future, broadens the range of issues that can be tackled. Present behavior is affected by accumulated experience and by the expectation that it will have future consequences, so there is room for investment behavior and capital theory. In absence of identification, however, many activities take a restricted form or do not exist. Obviously, there can be no inter–temporal transactions; there is room only for the Robinson–Crusoe type of capital theory – unidentified individuals (or, agents) cannot take upon themselves obligations for future delivery of goods or money, so there are no financial assets or commodity markets. There is investment in human capital only to the extent that the skills acquired can be used without resort to identity. People learn the distribution of prices or trading partners confronting them, and they can search for the lowest price, the highest wage offer, or the best quality store, but they will not have much incentive to do so if they cannot identify them for the next transaction. Similarly, there is no incentive to sell more cheaply or to offer higher quality service if one knows one is not going to be identified in future. It needs but little reflection to realize that giving people identity, in addition to a past and a future, permits the discussion of much that is of interest even in narrowly defined economic activity. These economic models are for human beings. If we consider software agents transacting on behalf of human beings, additional issues need to be tackled. This software currently lies between the first and second category. There are no inter–temporal transactions, nor is identity of agents taken into account. Also, currently, the agents do not learn from their experiences. This would be possible if machine learning methods and various heuristics are employed.

Giving agents identity and recognizing their identities would be the next big step in the evolution of agents, and would mean that agents are in control for protracted lengths of time, and could be trusted to honor inter–temporal deals. The agents in this case should be very robust and reliable and they would be required to have vast amounts of world knowledge (at least of particular domains) to perform a chain of complex negotiations on their own with their peers. This would truly bring agents to a status equal to human beings and would be a fitting end to their evolutionary process.

2.2 Requirements specification

Next, we consider the requirements that such a project as ours places on its developers. The requirements that should be met are:

1. First of all, a user must be able to specify his/her constraints (or better, his mind–set) to the agent i.e. the agent should have its goal clearly chalked out by its master.

2. The agents must be able to communicate with each other in a meaningful way. This is possible only when all of the following three criteria are met.

   (a) The agents should be able to locate and exchange messages with a counterpart.

(b) The content (knowledge) that they share with each other must be in a format that is understood by all.

(c) The words that they use to communicate with each other should be understood and interpreted unambiguously by each of them (i.e. they must have a common vocabulary or ontology).

As mentioned in 1 above, the agents should be given a goal that it would seek to achieve by interaction with its peers. It is desirable that the specification mechanism should be able to model the complex requirements of human beings. In addition, the specification mechanism should also mention the strategies that an agent should follow during negotiation (how to go about achieving the goal). The specification mechanism should take care of the following things:

- It should be possible to specify some of the relationships that exist in the real world between different items (for example, whether the goods are complementary or substitute goods). Speaking more generally, users should also be able to specify their constraints (number of chairs to be bought should be exactly six times the number of tables).

- It should be possible for the user to specify the price as a function of number of items (price as a function of number of goods).

- It should be possible for the user to specify the price quote (his/her valuation) as a function of time. This, for the seller, means that if he gets a price equal to the price quote, he would award the good then and there to the bidder and would not wait for any further bids (This price would be indicative of the highest price that the seller would hope to get from the sale of the good even if the sale continued in the future). The meaning for the buyer is analogous.

For 2(a), there must be a discovery protocol to allow the agents to discover each other. Also, agents must be able to exchange messages with each other. For 2(b), there must be a common language that agents must understand. For 2(c), there must be a common vocabulary of terms. There are ARPA standards for each of the above. The standards are KQML (Knowledge Query Manipulation Language) protocol, KIF (Knowledge Interchange Format) and Ontolingua (the common conceptualization of the world).

Chapter 3

Implementation

3.1 Program

We have developed agents that can negotiate multilaterally in a peer–to–peer framework (a buyer agent can talk to several seller agents and vice–versa) for the execution of a transaction. The transactions are modeled in the same general framework as has been discussed earlier. The users can specify his/her specifications. For each item to be sold – the minimum quantity, the maximum quantity, price as a function of quantity and reserved price as a function of time could be specified. Additionally, for the whole transaction – the relations between different goods (for example, the number of chairs to be bought should be exactly six times the number of tables), the synergies between different goods, the strategy to be followed while negotiating, and the deadline by which such a deal should be completed could be specified. This is discussed in more detail in the subsection on *configuration file(3.2.1)*. Such broad definition of transaction encompasses many simple cases like purchasing/selling a single item or a group of related items.

The project consists of approximately 7000 lines of Java code and the accompanying GUI part has been implemented in Swing. All agent communication is done over TCP. The buyer agent and the seller agent are implemented as separate classes. Each object is identified by its object id that is assumed to be known unambiguously by all agents (i.e the ontology part is assumed – all agents have the same object id to object mapping). Akin to the real world, the transactions in the project are made possible by the interaction of three types of agents – *buyer agent*, *seller agent* and *central agent*. The central agent acts as a match–maker, as a sort of broker that maintains a repository of information about seller agents and helps buyer agents discover compatible seller agents.

3.1.1 Start–up

To execute a transaction, first of all, the user prepares a *configuration file* that contains the information the agent needs to know for making the transaction (i.e. the user customizes the agent as per his needs through the configuration file). The *configuration file (3.2.1)*, is common for both buyer agents and seller agents. It basically allows the user to convey his/her mind–set to the agent. After a buyer/seller agent is invoked with the configuration file as the parameter, it first generates an unique name for itself so that its identity can be maintained across the many sets of messages it may exchange with a particular counterpart during a single session (for example, a buyer agent may contact a seller agent through different ports during its interaction and so it is necessary to identify that the buyer agent is the same in all cases). The buyer/seller agent then parses the configuration file and initializes its data structures.

3.1.2 Alternate bid

The data stored in the configuration file cannot be directly used to carry out a transaction. This is because the data is in the form of a set of inequalities. For example, *take a simple configuration file containing specification about two items, tables and chairs. Let the minimum number and maximum number of tables as specified be 2 and 4 respectively and these numbers in case of chairs be 13 and 19 respectively. Now, any of the 21 (=3*7) possibilities would satisfy the requirements of the agent (user). But, now if more constraints like the number of chairs should be strictly equal to six times the number of tables are specified, there remains only one possible tuple – (3 tables and 18 chairs) that still satisfies the requirements.* There also exists the possibility that no solution would exist that satisfies all the constraints placed by the user. Such information is not immediately obvious from the format in which specifications are made by the user. Thus, it is more practical to convert the specifications given by the user to a set of tuples that satisfy the constraints mentioned in the configuration file. It basically corresponds to finding integer solutions to a set of inequalities – i.e. Integer Linear Programming.

If there are n items,

$$i_1, i_2....i_n$$

then there are *2n* inequalities, corresponding to

$$minimum_j <= i_j <= maximum_j, 1 <= j <= n.$$

Also, there will be *one equality corresponding to each relation* mentioned in the global constraints. From these inequalities, finding all possible sets of tuples that satisfy all the constraints is termed as getting the Alternate bid representation.

3.1.3 Central server

Then, the agents contact a central server that contains the information about all seller agents. In case of a seller agent, the message it passes to the central server consists of all the objects that it intends to sell, the IP address and the port it is willing to listen on and a rough estimate of the date and the time until which it would be willing to perform the transaction. The central agent in this case first checks if the entry is present in the database. If it is not present, it inserts the entry, with all the particulars, into its register. If it is present, it overwrites the old entry with the new one. In case of a buyer agent, the message passed from the buyer agent to the central agent consists of a list of items, which the agent is interested to buy. The central agent then returns to it a list of seller agents, along with the addresses (ip address, port) of the seller agents. While searching for the appropriate seller agents, the central agent also flushes the old entries, that have become invalid.

3.1.4 Negotiation

After this, the buyer and the seller agent are ready to exchange messages and transact. The transaction consists of a set of proposals and counter proposals. Each bid (proposal/counter proposal) is accompanied with a validity period. The idea of a validity period is that a buyer/seller might not be interested in a particular transaction after some specific time period elapses (This is a more general form of bidding than what we observe in auctions, where generally a bid is necessarily valid till the auction ends). All communication is initiated by the buyer agent. Initially, the seller agent just keeps waiting for a message to arrive. The buyer agent tries to contact all the seller agents, asking for the price quote of the goods it is interested in buying and sending along its bid (which goods it wants to buy, quantity of each good to be bought; the price in this case is set to –1 because the buyer agent wants to know the price the seller has in mind before providing the idea of the valuation it has in mind, this is also true of the real world transactions – most goods have posted prices attached to them). The seller agents, for which none of the buyer's constraints interfere with their own constraints, respond to this proposal, with their bid (from this bid onwards, all bids that are exchanged necessarily have pricing information). Out of these, the buyer agent then chooses the best tuple based on how little its price differs from the price that was specified in its configuration file. It does this for whatever flexibility it has (in choosing the quantity of the goods, or the substitute good – the idea is that the buyer, if it suits its needs, can take advantage of the package deals, since for it, any of the tuples satisfy the requirements, it chooses one whose price tag (as specified by the seller) has minimal price differences with its valuation). Thus, all parameters of the transaction except the price are fixed after this.

Then, the negotiation begins. The two parties contact each other with proposals and counter proposals, depending on their bidding strategy. The bidding strategy (except for the simulation part) currently consists of a number between 1 to 100, that indicates how desperately a user wants the transaction to commit (The number is basically the maximum percentage by which the price quoted by an agent differs across two successive bids). If a transaction reaches this stage, still it is not guaranteed to occur because each agent is negotiating with more than one opposite agent. This process ends when either an *ok* message (indicating that the transaction is committed) or *withdraw message* is exchanged. The *ok* message would only be exchanged if the price quote of the parties match. A *withdraw* message is sent when a transaction has been sealed with another agent due to which the agent in question does not have sufficient available resources to successfully complete the deal.

Also, to maintain the consistency of the messages, at any time instance, the agents send only those proposals that they are able to fulfill. The agents at all time maintain information about the available resources they have with them. This amount has to be reduced if they send a bid, and has to be increased whenever a negotiation fails (on receipt of a withdraw message). Finally, if no transaction materializes, the agents contact the central agent again (In case of a seller agent, if its registration period has expired, it registers once again with the central agent). This is done iteratively till the deadline for performing the transaction, as mentioned by the user in the configuration file, expires.

3.1.5 Calculating prices

Initially, the price is calculated based on the time and quantity to be traded for each item. The prices of items which are present in some synergy collection are increased/decreased (increased for buyers, decreased for sellers) depending on the synergy value. After this, whenever the price has to be calculated, it is calculated based on the strategy value.

3.1.6 State diagrams

The following three figures show the state diagrams of buyer, seller and central agents – these just provide the outlines, the actual implementations are more complex and many other things – such as keeping track of what resources are available with an agent, have to be handled.

Step 0: start buyer agent with parameter as configuration file and address of central agent
Step 1: get Alternate bid3.1.2 representation
Step 2: get unique Name
Step 3: query central agent for seller agents
Step 4: if size of list <> 0, goto 6
Step 5: wait (Truncated binary exponential back off) and goto 3
Step 6: iterate over the list and find the seller Agent offering the best deal
Step 7: Negotiate3.1.4
step 8: if negotiation fails, goto 3
step 9: stop
Figure: LifeCycle of a Buyer–Agent

Step 0: start seller agent with parameter as configuration file, a port number and address of central agent
Step 1: get Alternate bid(3.1.2) representation
Step 2: get unique Name
Step 3: register with central agent
Step 4: wait for bid from buyer agents
Step 5: check if the proposal is still valid. if no, goto 15
Step 6: check if it is coming for the first time. if no, goto 10
Step 7: check if it is a buy message If not, goto 15
Step 8: check if any tuple in the request can be satisfied. If no such tuple can be found, goto 15
Step 9: send all tuples along with price info, goto 4
Step 10: must have only one bid, if not goto 15
Step 11: check if last message sent by the Seller–Agent contained this bid. If not, goto 15
Step 12: if it is a buy message, check if the bid can be satisfied. if not, goto 14. the bid is

guaranteed to be satisfied when the message is being received in the valid period

Step 13: find the 'next price' as per the strategy, then if it is less than the offered price =¿ send an 'ok' message, else send a new bid at 'next Price'. Store the messages sent and received, reduce the tentatively available amount. goto 16

Step 14: if date is not valid, goto 15. decrease the actual remaining amount. goto 16

Step 15: send various error messages

Step 16: check if deadline is over, or a particular tuple of items (one of the tuples that are solution of the ILP) has been sold. if no, goto 4

Step 17: if deadline is not over, re–register and goto 3.

Step 18: stop

Figure: Life cycle of a seller agent

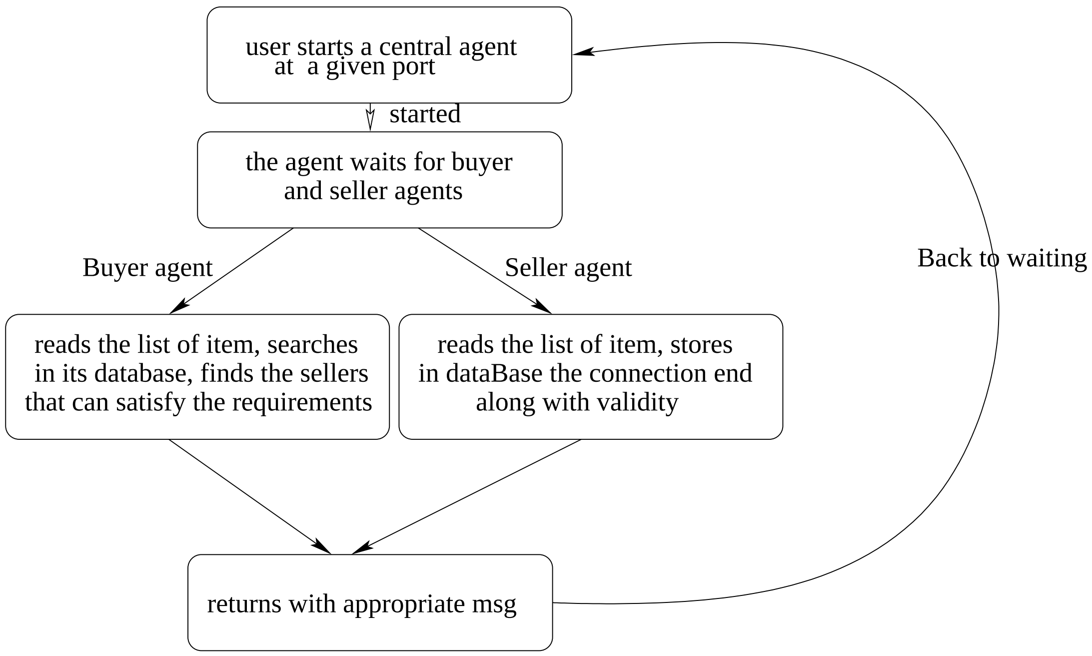

Figure: Working of a central agent

3.2 Assumptions

Following are the assumptions that the implementation makes:

- All objects can be unambiguously referred by an object id (i.e. no ontology – if two agents refer to X, do they mean the same thing, is specified). *Justification:* The implementation of ontology, though interesting, would be very complicated and trying to implement this would mean neglecting other portions of the project.

- The messages take negligible time to travel from the sender to the receiver, in relation to the validity period of the bids. *Justification:* This, though untrue in most cases, simplifies the message exchange protocol. Also, this slight anomaly can be easily rectified if both parties have a good estimate of the round–trip–time (RTT).

- The underlying communication mechanism for passing of messages is reliable, and the messages arrive in the order they are sent. *Justification:* TCP is used for passing messages and it satisfies both the constraints.

- All items of a particular kind are homogeneous, there is no distinction between 2 chairs. This also implies that there would be no negotiation over the item's quality (the only negotiation basis would be the price). *Justification:* Treating all items as homogeneous makes the negotiation mechanism simple. A roundabout way of incorporating this into the present framework would be to treat two similar object with different attributes as two totally different objects.

- All price values dealt with are integral values. *Justification:* It is a simplistic assumption since all price tags must have a least count, it matters little if the least count is 1.

3.2.1 Configuration file

The configuration file contains the following information:

- **Item Section:** what objects to sell/buy and their properties (minimum quantity, maximum quantity, price as a function of number of instances, price quote as a function of time). There may be multiple such entries – as many as the number of distinct items to be traded.

- **Global constraints section**:

  – capture synergy b/w objects: expressed as a percentage on how much more the user is willing to pay above his normal valuation of these items. For example, if the user values a chair at 5 units and a table at 15 units and the synergy is 50%, then he is willing to pay 30 units for a combination of a chair and a table.

- the no. of tables must be exactly four times the number of chairs (complementary goods).
- the total no. of tea and coffee packets to be exchanged must be ten (substitute goods).

- **Strategy section:**the negotiation strategy to follow: *presently, it can take values between 1 to 100* in addition to some other values (like random buyer, bargain hunter, game theoretic, myopically optimal and derivative follower).

In the current implementation, global constraints can only be linear. Also, all global constraints must be only between two items. A part of the open configuration file can be seen in 3.3.

3.2.2 Format of message for exchanging proposals/counter proposals

() It consists of four parts,

- Name: unique identifier to identify an agent
- Type: whether it is a buy/sell/ok/withdraw message.
- Alternate bid: It contains the set of tuples that satisfy the constraints as formulated in 3.1.2. In all messages except the first message sent by either party, it consists of just a single tuple (3.1.4).
- Price: the price.
- Date: time until which the bid is valid.

3.3 GUI

The GUI, implemented in Java swing serves as a convenient front end to the program. The main window(3.3), that comes up when the program is executed as a stand–alone application, consists of the following six click–able buttons through which all the functionality provided by the program can be achieved.

1. Configure: lets the user open an existing configuration file and save an already open configuration file, that has been slightly modified, under a different name. Everywhere the integrity of the data is checked so that the data always remains consistent. Opening a

file, slightly modifying it and then saving it under a different name is a convenient way of fine–tuning the values in a configuration file so that the best values in a particular situation can be found out.

2. Start a central agent: starts a central agent at a particular port.

3. Start a buyer agent: starts a buyer agent after taking as input the configuration file. The address of the central agent is at present hard–coded but it can easily be taken through a text field.

4. Start a seller agent: starts a seller agent after taking as input the configuration file and the port where it should listen. The address of the central agent is at present hard–coded but it can easily be taken through a text field.

5. Simulate: To be implemented in the rest of the semester.

6. Help: Provides a HTML browser that provides help on both the concepts in this area as well as how to use the software. It uses a tabbed pane for this purpose and so navigation is easier.

At all places, care has been taken to validate all user inputs. Next, we have screen shots of the GUI that show a GUI at work at different stages.

Figure 1: Main window

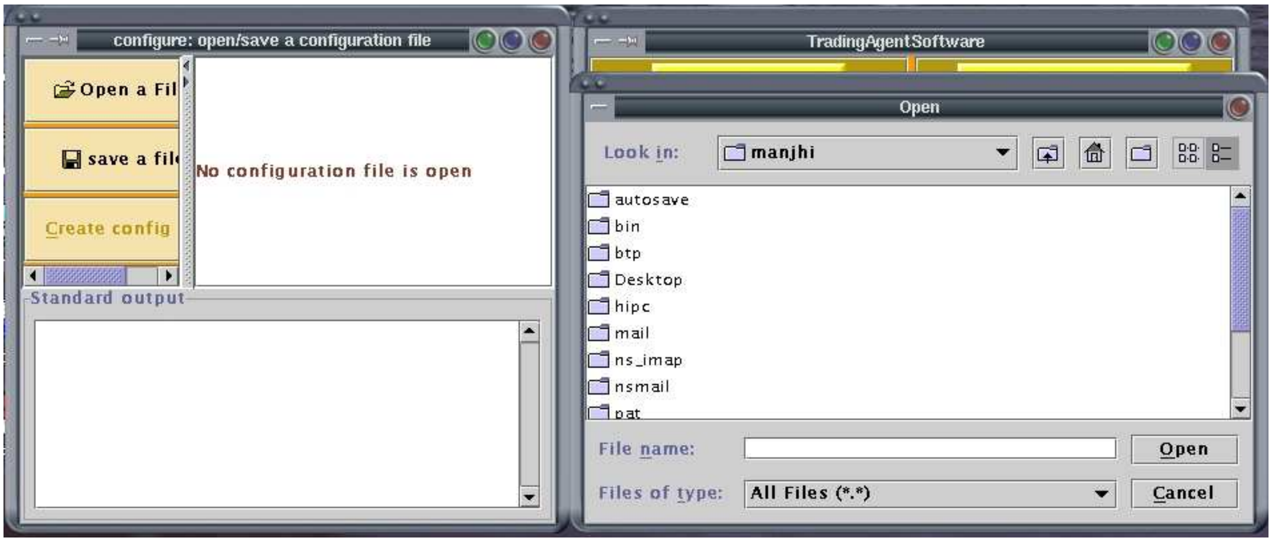

Figure 2: How to use configure tool

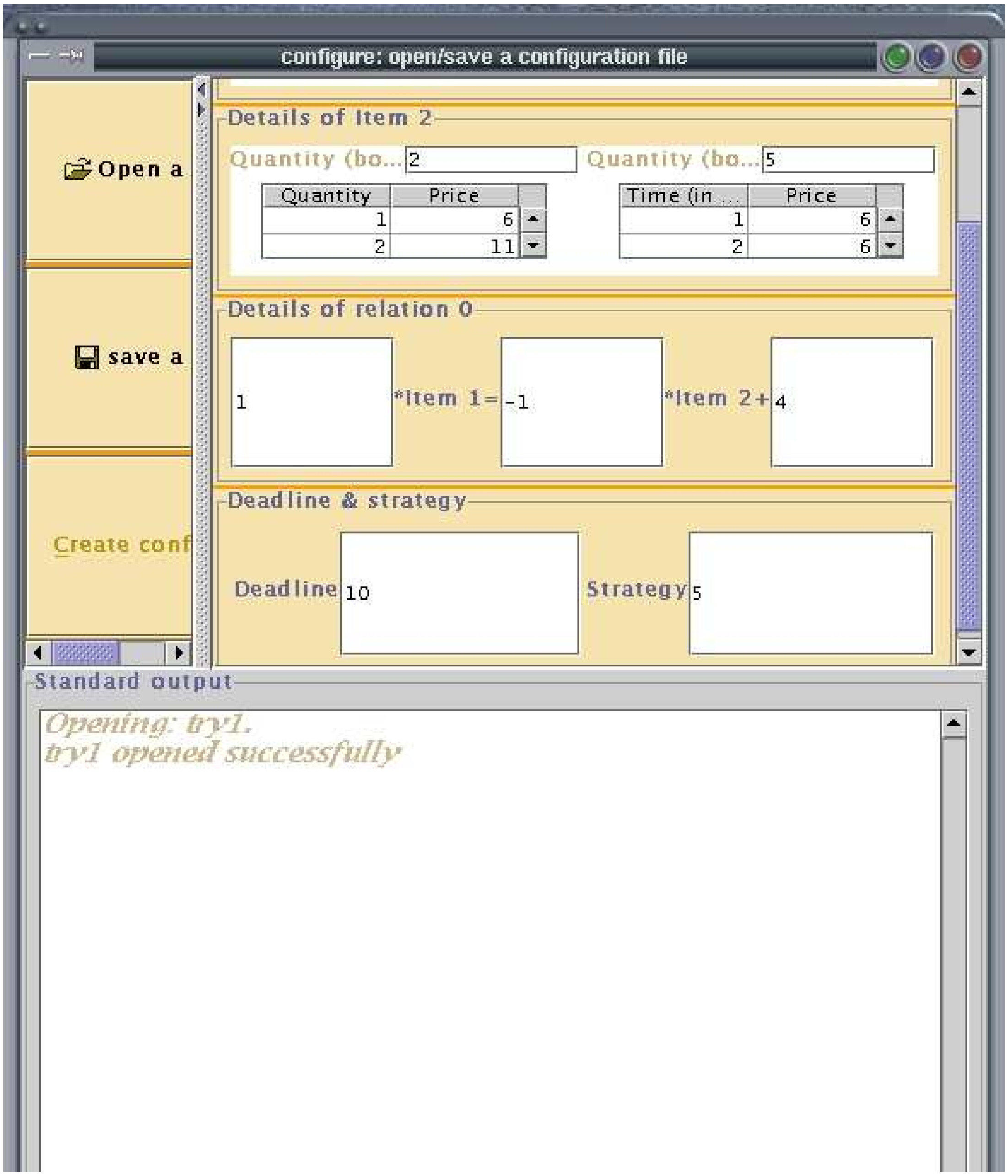

Figure 3: Figure showing part of an open configuration file

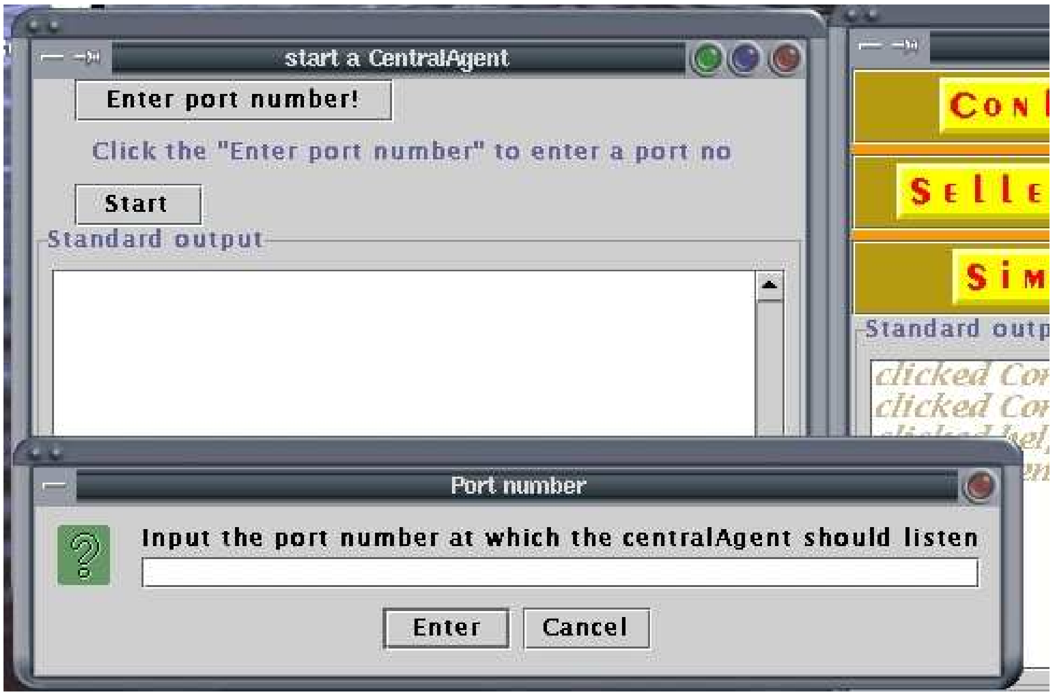

Figure 4: Starting a central Agent

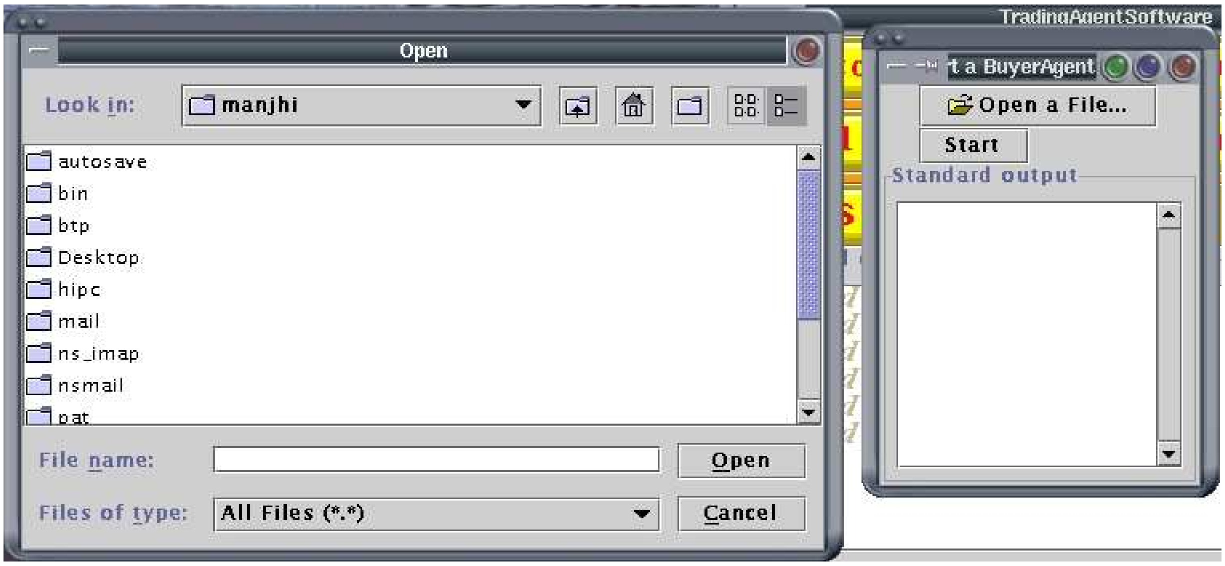

Figure 5: Starting a buyer Agent

Figure 6: Starting a seller Agent

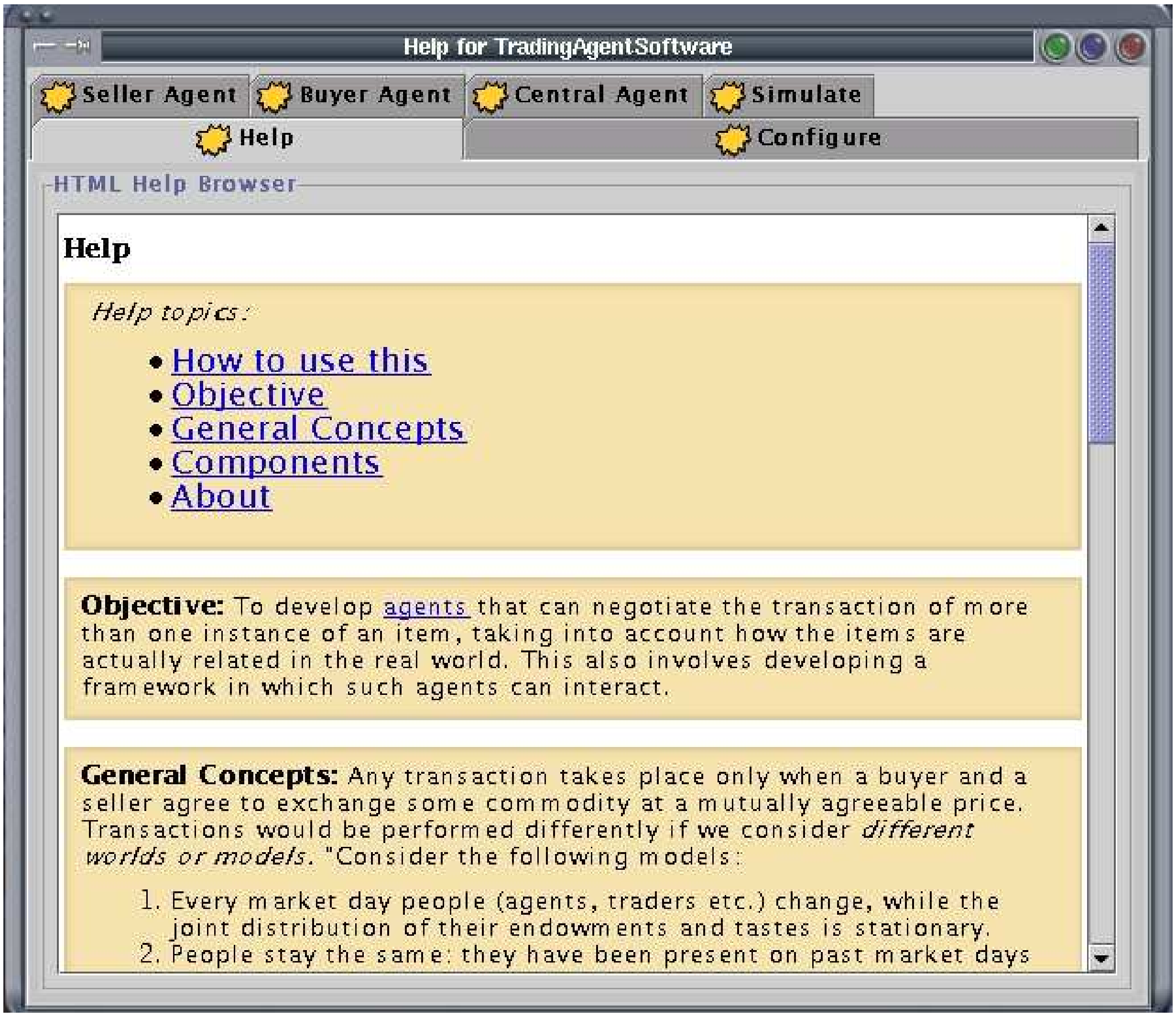

Figure 7: Browsing help with HTML browser

Chapter 4

Simulations

Our platform offers us a powerful framework to model different types of transactions and to carry simulations on those. But, it is difficult to simulate transactions in the complete generality offered by our platform, as it would involve complex modeling of many aspects of human behavior – there would be too many parameters and to vary these so as to deduce some behavioral patterns, would be a very challenging task. So, we have made many simplifications and below we present the model we have considered for carrying our simulations. We have considered a simplified dynamic pricing model[1], and in the next few sections, we state our assumptions, perform an analysis of the theoretical foundations behind it and compare this with the actual results obtained, in our effort to understand the widespread adoption of shopbots and pricebots. But before this, we look at how the simplified model of dynamic posted pricing that we consider is relevant to the world of today and the world we envision.

4.1 Dynamic posted pricing

Dynamic pricing is, take–it–or–leave–it pricing in which the seller may change the price at any time [10]. This is as opposed to the normally followed concept of fixed pricing. Dynamic pricing enables the distribution of scarce resources in the most optimal way. The adjective "posted" has been emphasized to stress that we are not considering online auctions, in which prices change during the transaction – in this case, prices change only across different transactions.

Books.com is an online bookseller, now acquired by Barnes & Nobles that employed posted pricing (i.e. it announced a non–negotiable, take-it-or-leave-it price), but used a price-comparison agent that dynamically undercut the price offered by its chief competitors: Amazon, Barnes and

Noble, and Borders. Another feature of posted pricing that makes it attractive for agents is that price determination is quicker than it is for other forms of negotiation. A human buyer may tolerate waiting two weeks for the close of an auction of an antique piece at sothebys.com. However, buyer agents will not want to have a protracted negotiation with the seller agent. It will want to settle the price and make the purchase within a fraction of a second. Multiround auctions or negotiations will almost certainly be too slow in such a situation. Posted pricing is guaranteed to be the fastest mechanism in this case.

Shopbots, or comparison shopping agents, offer an early glimpse of the vast reductions in economic friction that are likely to occur as the world makes the transition to the information economy. In this relation, it is important to consider the question – how do sellers manage to make any profit at all in a competitive market, since competitive pressures would drive prices down to the marginal production cost? The answer is the search cost involved in discovering prices – it would make buyers to forego comparing prices forever. This in turn would allow sellers to charge more than the marginal production cost. Other factors like product differentiation (such as brand name) and information bundling [1] would also contribute to sellers getting away with prices higher than their marginal cost.

This search cost has reduced drastically in the information economy of today and there are umpteen sites that offer comparative shopping (like DealPilot.com). As an increasing number of buyers begin to avail themselves of shopbots, and as shopbots become more pervasive and powerful, one of the frictional forces (involvement of high search costs) that has sustained profits for sellers in commodities markets will be reduced substantially. Price-aware buyers will be price-sensitive buyers, and they will force sellers to become extremely responsive in their pricing. We anticipate that humans will not be able to keep up with the demands of responsive pricing on millions of goods and services, and that instead sellers will rely increasingly on what we term pricebots – agents that adjust prices automatically on the seller's behalf in response to changing market conditions. We can regard Books.com's agent as an early example of a pricebot, it automatically adjusts the price to slightly less than the minimum price offered by Amazon, Barnes and Noble, and Borders.

4.2 Model

We consider a simple market in which S seller agents compete to provide B buyer agents with a commodity, such as a specific book. A buyer purchases a good if his valuation (similar to the reserved price discussed before) is more than the seller's price. In the process, if it has to choose one among the many seller agents that satisfy its criterion (i.e. the price offered is less than the buyer's valuation), it follows one of the following strategies:

1. Random Buyer: buyer collects the list of sellers that sell the good it is interested in, picks out ones that offer prices less than its valuation, and then selects a seller at random from this lot (this is the same as picking a seller at random and then transacting if the price offered by seller is less than the price buyer's valuation, as suggested in [8] (since both the events are unrelated)).

2. Bargain Hunter: buyer checks the price of all sellers having prices lower than its valuation, determines the seller with the lowest price, and then purchases the good (This type of buyer agents corresponds to people who take advantage of shopbots).

The buyer population consists of a mixture of buyers employing one of these strategies, with a fraction w_A using the *Random Buyer* strategy and a fraction w_B using the *Bargain Hunter* strategy:

$$w_A + w_B = 1.$$

Buyers employing these respective strategies are referred to as type A and type B buyers.

The objective of each seller s is to set its price p so as to obtain the maximum profit, given a production cost c. The production cost c is assumed to be fixed. The profit function $\pi_s(\vec{p})$ for seller s as a function of the price vector $\vec{p}$ is expressed as follows:

$$\pi_s(\vec{p}) = (p_s - c_s)D_s(\vec{p})$$

where $D_s(\vec{p})$ is the rate of demand for goods sold by s. This rate of demand is the product of the overall rate of demand $\rho = \Sigma_b \rho_b$, the likelihood of a given buyer b selecting s as their potential seller, $h_s(\vec{p})$, and the fraction of buyers whose valuations satisfy $v_b \geq p_s$, denoted by $g(p_s)$:

$$D_s(\vec{p}) = \rho B h_s(\vec{p}) g(p_s).$$

$g(P_s) = \int_{p_s}^{\infty} \gamma(x)dx$, where $\gamma(x)$ is the probability density function describing the likelihood that a given buyer has a valuation x. For example, suppose that the buyers' valuations are uniformly distributed between 0 and v; then the integral yields $g(p_s) = 1 - p_s/v$. Alternatively, if $v_b = v$ for all buyers b, then $\gamma(x)$ is the Dirac delta function $\delta(v - x)$, and the integral yields a step function $g(p_s) = \theta(v - p_s)$ which is equal to 1 iff $p_s \leq v$ and 0, otherwise.

The probability $f_s(\vec{p})$ that buyers select seller s as their potential seller depends on the distribution of the buyer population, namely (w_A, w_B). In particular,

$$f_s(\vec{p}) = w_A f_{s,A}(\vec{p}) + w_B f_{s,B}(\vec{p})$$

where $f_{s,A}(\vec{p})$ and $f_{s,B}\vec{(p)}$ are the probabilities that seller s is selected by buyers of type A and B, respectively. The probability that a buyer of type A selects a seller s is independent of the ordering of sellers' prices, in particular, $f_{s,A}(\vec{p}) = 1/S$. Buyers of type B, however, select a seller s if and only if s is one of the lowest price sellers. Given that the buyers' strategies depend on the relative ordering of the sellers' prices, it is convenient to define the following functions:

- $\tau_s(\vec{p})$ is the number of sellers charging the same price as s, excluding s itself, and
- $\lambda_s(\vec{p})$ is the number of sellers charging a lower price than s.

Now buyers of type b select seller s iff s is s.t. $\lambda_s(\vec{p}) = 0$, in which case a buyer selects a particular such seller s with probability $1/(\tau_s(\vec{p}) + 1)$. Therefore,

$$f_{s,B}(\vec{p}) = \frac{1}{\tau_s(\vec{p}) + 1} \delta_{\lambda_s(\vec{p}),0}$$

where $\delta_{i,j}$ is the Kronecker delta function, equal to 1, whenever i=j and 0, otherwise.

4.3 Game–theoretic analysis

In this section, we determine the game-theoretic equilibrium price vectors, under various assumptions about the distribution of the buyer population. In particular, we compute the Nash equilibrium[6], which is a vector of prices $\vec{p}$ *for which sellers maximize their individual profits and from which they have no incentive to deviate.* We consider the case in which the number of sellers S > 1 and all sellers have the same production cost c. Moreover, all buyers have the same valuation v, with v > c. We also do not take into account the search cost involved in finding the optimal seller.

- All bargain hunters: This is the most widely accepted model ($w_B = 1$, $w_A = 0$). and is very simple to analyze since we are not considering the search costs involved. In this case, the prices are driven down to marginal cost for all sellers, if there is no product differentiation etc.

- All random buyers: As discussed earlier, buyers in this category do not pay attention to the prices that a seller is charging, as long as it is less than its valuation. In such a situation, tacit collusion arises, where all the sellers charge prices equal to buyer valuations since the sellers have no particular incentive to offer a price lower than v (since the number of buyers that would choose to buy from them would remain the same, no matter what price they charge, as long as the price is less than buyer's valuation). Also, a price higher than buyer's valuation is not viable since no buyer would be willing to transact at that price. So, all sellers charge a price v and this monopolistic price prevails as the equilibrium price.

- Mixed bag of buyers: One particular deduction to be made in this case is: *at equilibrium, exactly one seller charges $p < v$, whenever $w_A > 0$ and $w_B > 0$.* It follows from the following insightful observations:

  Lemma 4.1 *At equilibrium, at least one seller charges a price $p < v$.*

  Proof: Suppose that the above is not true. Then, all sellers charge a price $p = v$. This situation is not a equilibrium condition, since one of the sellers can increase its profits by charging $(v - \epsilon)$ where ϵ is the least count by which prices are allowed to change (since the profit in the new case would be $(w_B + w_A/S)B(v - \epsilon - c)$ which is greater than $(w_B + w_A)(v - c) * B/S$ (subject to ϵ being less than $\frac{w_B(s-1)(v-c)}{w_A+w_B S}$, which is true except in cases where ϵ is very coarse). Thus, the seller which first lowers its profit by ϵ stands to gain and hence, all sellers charging price $p = v$ is not an equilibrium condition. ∎

  Lemma 4.2 *At equilibrium, at most one seller charges a price $p < v$.*

  Proof: Suppose there are at least two sellers (let us call them s_1 and s_2) charging prices p_1 and p_2 respectively at equilibrium (where $p_1 < v$ and $p_2 < v$). Then, either $p_1 = p_2$ or $p_1 < p_2$ or $p_1 > p_2$. If $p_1 = p_2$ then one of the sellers can increase his profit by charging $(p_1 - \epsilon)$ (argument similar to the previous lemma) and so, this would not be an equilibrium condition. if $p_1 < p_2$ or $p_2 < p_1$ then the seller charging the higher price can increase its profit by moving to v (since it is anyway getting only random buyers as its customers, and their number remains the same at both the prices). Thus, at equilibrium, at most one seller charges a price $p < v$. ∎

Thus, at equilibrium, exactly one seller s sets its price below the buyers' valuation v (let us call it p^*), while all other sellers set their equilibrium prices at v. Next, we would like to determine the price at which this seller sets its price. For equilibrium, this seller should charge such a price p^* such that it is not possible for any of the sellers, charging v to increase their profits by beginning to charge $(p^* - \epsilon)$ i.e. the profits should be equal.

$$(w_A/S + w_B)B(p^* - c) = (w_A)(v - c)B/S$$

By setting its price to be the lowest price, while the others set their price equal to the buyers' valuation, seller s earns profits from buyers of type A who consider any seller, as well as from buyers of type B who consider only those sellers offering the lowest price.

4.4 Actual results

In the previous section, we allowed for the possibility that not all buyers would avail themselves of information about all sellers' prices (maybe due to high search costs, maybe due to lack of interest). Similarly, we cannot expect all sellers to follow game–theoretic strategy. Following game–theoretic strategy requires the sellers to have lots of knowledge – the sellers must know the user's demand function, know that all others have equivalent knowledge, must trust them that they will play fairly by the rules (i.e. compute and obey game–theoretic equilibrium). So, in addition to GT we consider two more adaptive pricing and explore the relationship between the resulting price and profit dynamics.

4.4.1 Strategies

Following are the three types of adaptive pricing strategies[1] that we consider. Each of the strategies require agents to have different levels of knowledge about their environment (the buyers and other sellers), place different levels of trust in their peers and also make very different demands on the required level of informational and computational power of agents.

- GT: The game-theoretic strategy is designed to reproduce the game-theoretic equilibrium computed in the previous section, provided that it is adopted by all sellers. It makes use of full information about the buyer population, as well as complete information about competitors' prices or pricing strategies. In essence, a game-theoretic agent assumes that all other agents abide by the prescribed game-theoretic strategy, and based on this assumption, it computes its own equilibrium price.

- MY: The myoptimal (myopically optimal) strategy makes use of information about all the buyer characteristics that factor into the buyer demand function, as well as the competitors' prices, but makes no attempt to account for competitors' pricing strategies. Instead, it is based on the assumption of static expectations: even if one seller is contemplating a price change under myoptimal pricing, this seller does not assume that this will elicit a response from its competitors; instead it assumes that competitors' prices will remain fixed.

- DF: The derivativefollowing strategy is far less informationally intensive than either the myoptimal or the game-theoretic pricing strategies. In particular, this strategy can be used in the absence of any knowledge or assumptions about one's competitors or the buyer demand function. A derivative follower simply experiments with incremental increases (or decreases) in its price, continuing to move its price in the same direction until the observed profitability level falls, at which point the direction of movement is reversed.

4.4.2 Implementation

Strategies

The strategies were implemented as follows:

- GT: The game-theoretic equilibrium requires that one seller set its price to p^*, as calculated before while the remaining sellers all charge the monopolistic price v. This computation does not address the question as to which seller volunteers to charge the low price p^*. In our implementation, we assume that the first seller which notices that no other seller is charging p^* sets its price to p^*. If such a seller exists, it sets its price to v.

- MY: The algorithm for calculating the next price offered by a myoptimal seller is based on an astute observation: *when considering only static expectations (i.e. other sellers do not change their prices in response to the price change by the seller), a seller can boost its profit either by moving to v or by undercutting the lowest price offered among all competitors (by obviously the smallest amount).* The proof can be worked out in a manner akin to deriving the game–theoretic equilibrium.

- DF: The price increment is taken to be the minimum change possible in our framework. If a seller found that the profit in previous interval was higher, the direction of change was left unchanged otherwise it was reversed.

	1GT	1MY	1DF
4GT	(1.5, 1.5)	(1.6, 1.3)	(1.4, 2.1)
4MY	(1.7, 1.3)	(2.3, 2.3)	(2.3, 1.2)
4DF	(1.6, 1.4)	(1.4, 7.4)	(3.3, 3.3)

Table 1: 4-on-1 payoff matrix. The numbers show the average profit obtained when the row players were pitted against the column players. The first coordinate of each tuple gives the average payoff of the row players.

Other details

Our platform which allows for protracted negotiation was easily adapted to simulating dynamic posted pricing by just implementing the method to get next prices in case any of the above strategies has to be followed (The negotiation consists of just two bids exchanged from either side). We performed simulations with $v = 90, c = 70$ and

$$gt + my + df = 5; gt, my, df \in \{0, 1, 4, 5\}$$

where gt denotes the number of sellers following game theoretic strategy, my denotes the number of sellers following myopically optimal strategy and df denotes the number of sellers following the derivative follower strategy. The maximum average profit that could be obtained per buyer by each seller would be 4. The ratio of number of random buyers to the number of bargain hunters was kept at $\frac{2}{3}$. The minimum price change being 1, p^* as referred to in above works out to be $70 + (\frac{20}{68})8$ which was set at 72 (though 73 would have been more appropriate since p^* should be the maximum price where undercutting does not pay off). Next, we attach the actual results of the simulation, the table showing the payoffs in various cases, and the graphs showing the pricing behavior of the pricebots during the simulation. In all graphs, where 4 pricebots of one kind are pitted against 1 pricebot of a different kind, only two colors (two types of dots) are used for the sake of clarity.

Analysis

The pure combinations (5 GT, 5 MY, and 5 DF sellers) lie along the diagonal of Table 1. First, consider the 5 GT sellers. Since GT does not observe its opponent's choice of price prior to making its own, and since an under-cutter can grab a significant portion (68%) of the market share, there is a strong incentive to price low. Whenever a GT comes into play, it prices itself (unless some agent is already operating at that price) such that it is not feasible to undercut

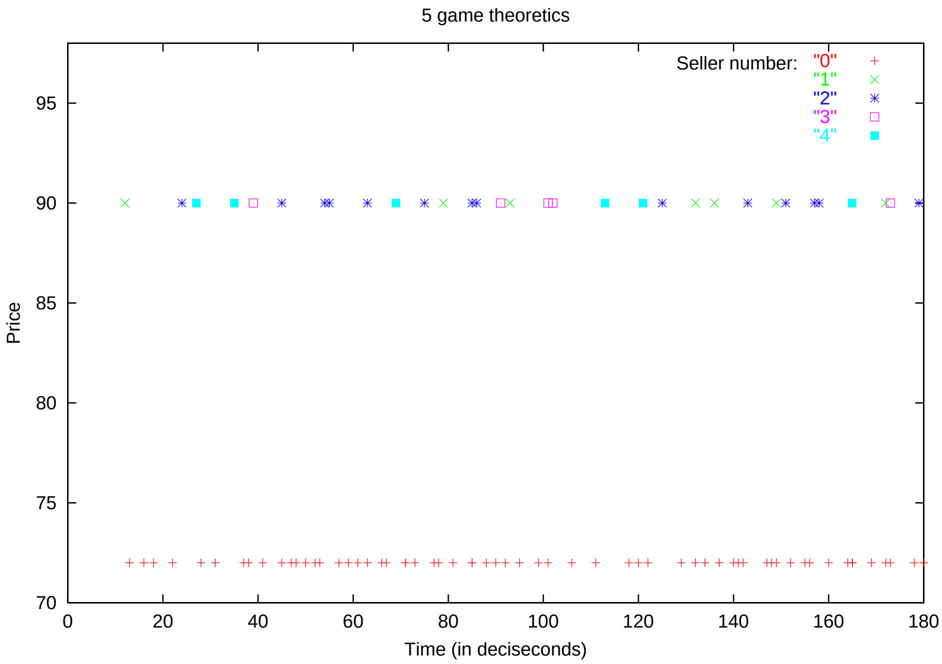

Figure 8: 5 game theoretic

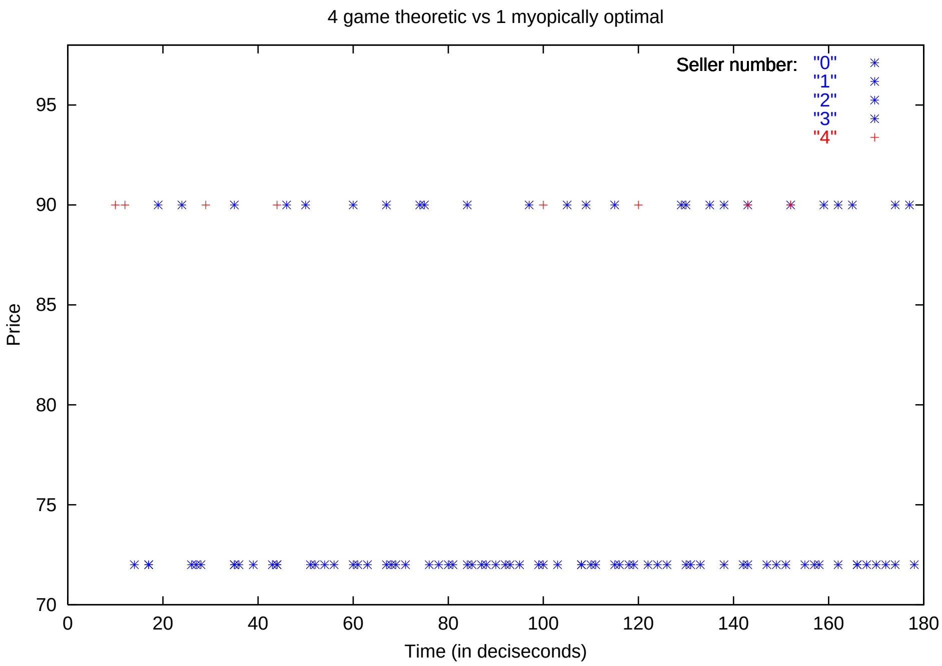

Figure 9: 4 game theoretic vs 1 myopically optimal

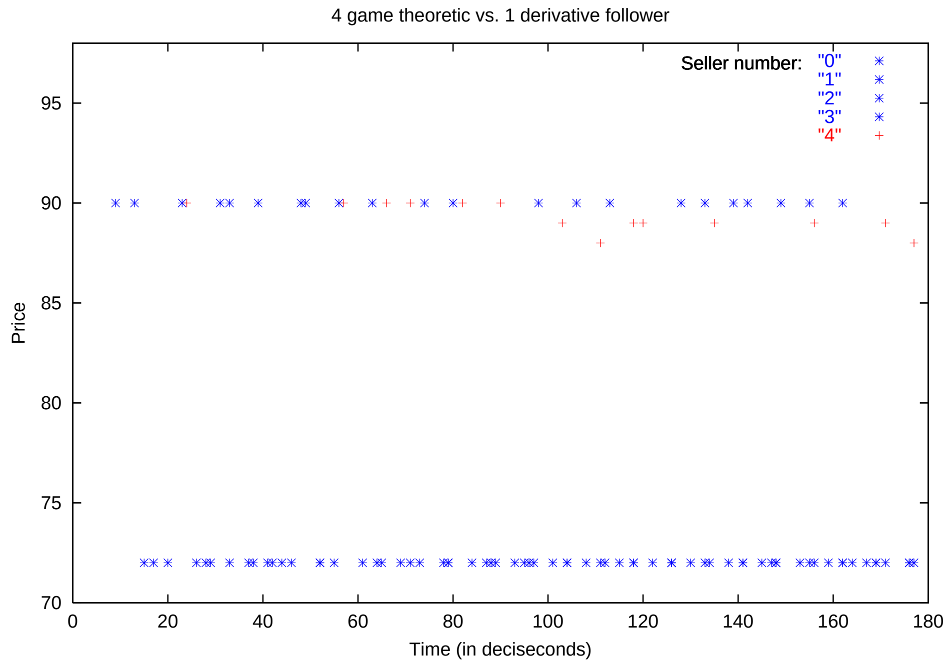

Figure 10: 4 game theoretic vs 1 derivative follower

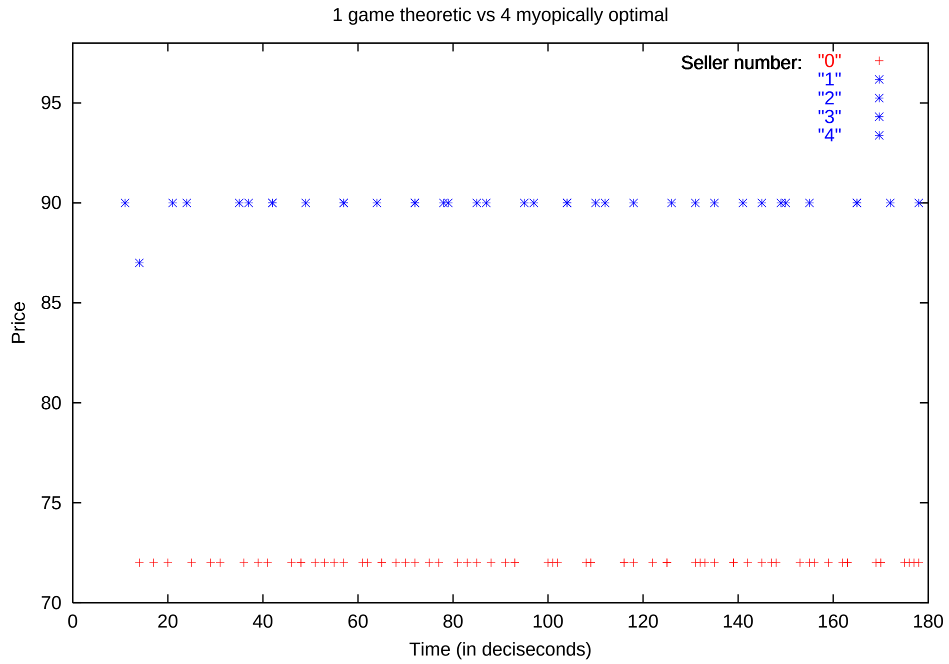

Figure 11: 4 myopically optimal vs 1 game theoretic

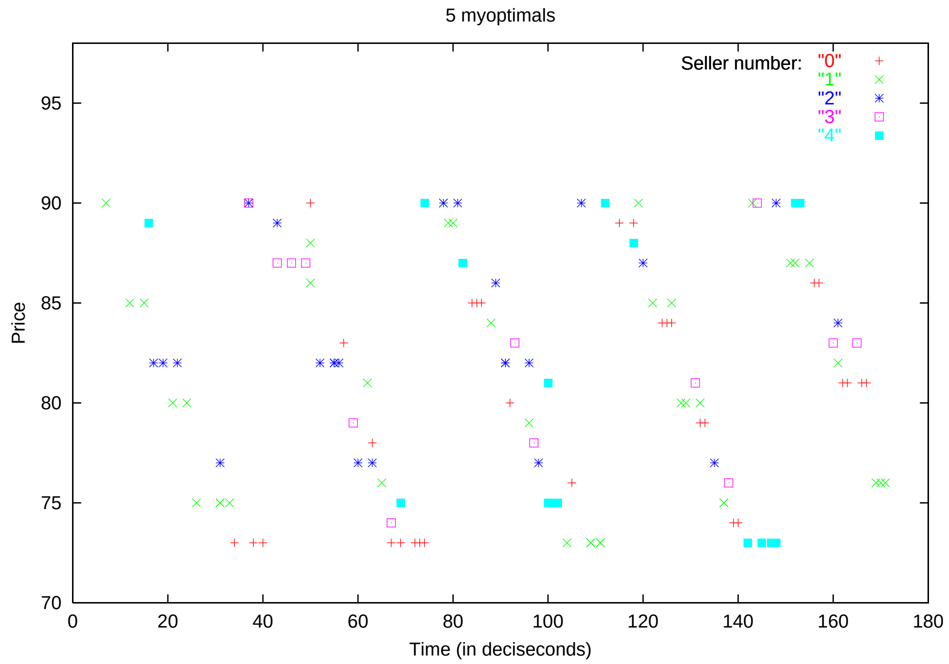

Figure 12: 5 myopically optimal

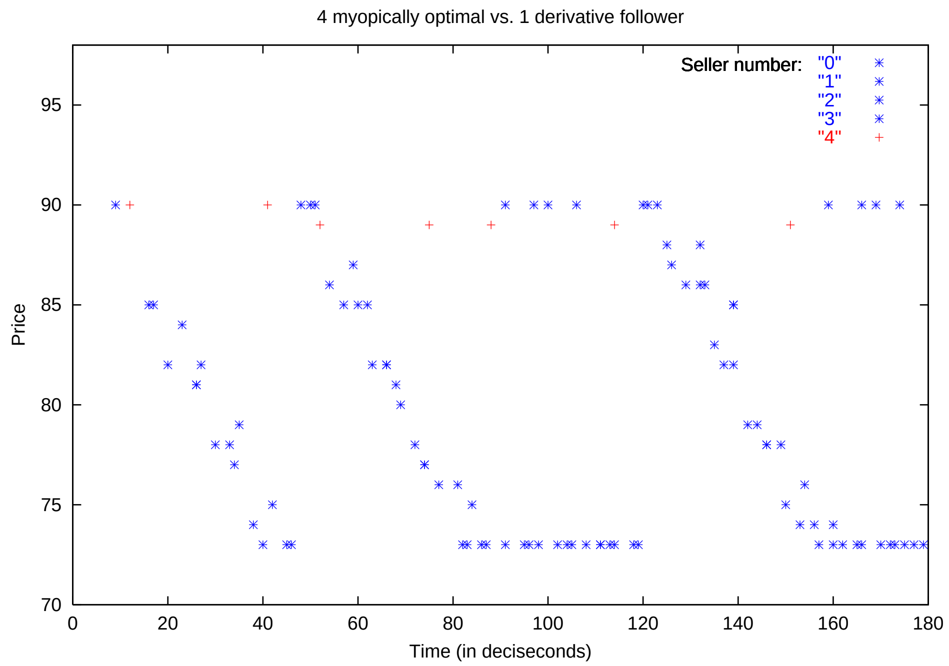

Figure 13: 4 myopically optimal vs 1 derivative follower

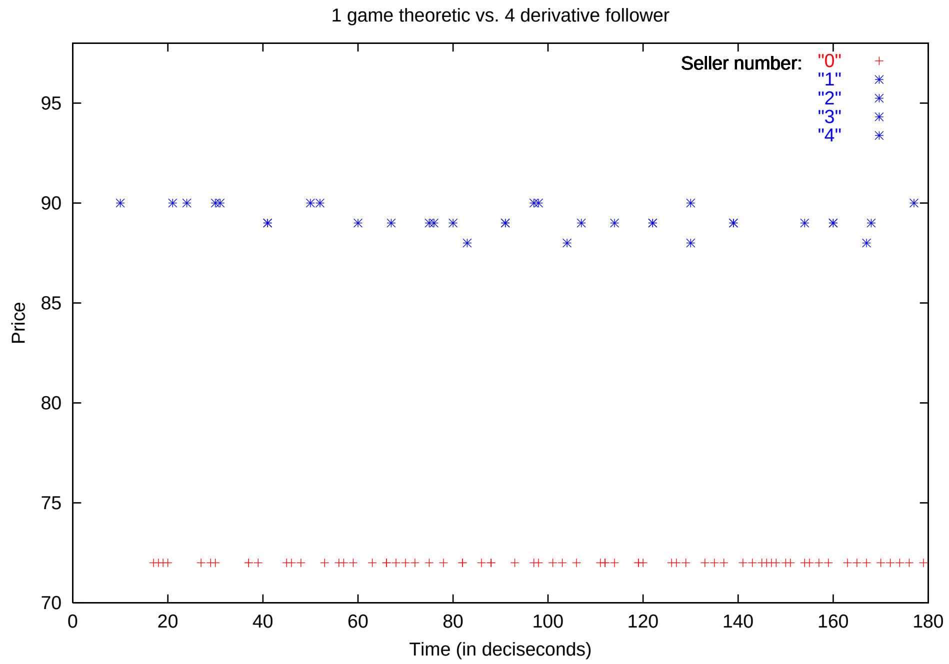

Figure 14: 4 derivative followers vs 1 game theoretic

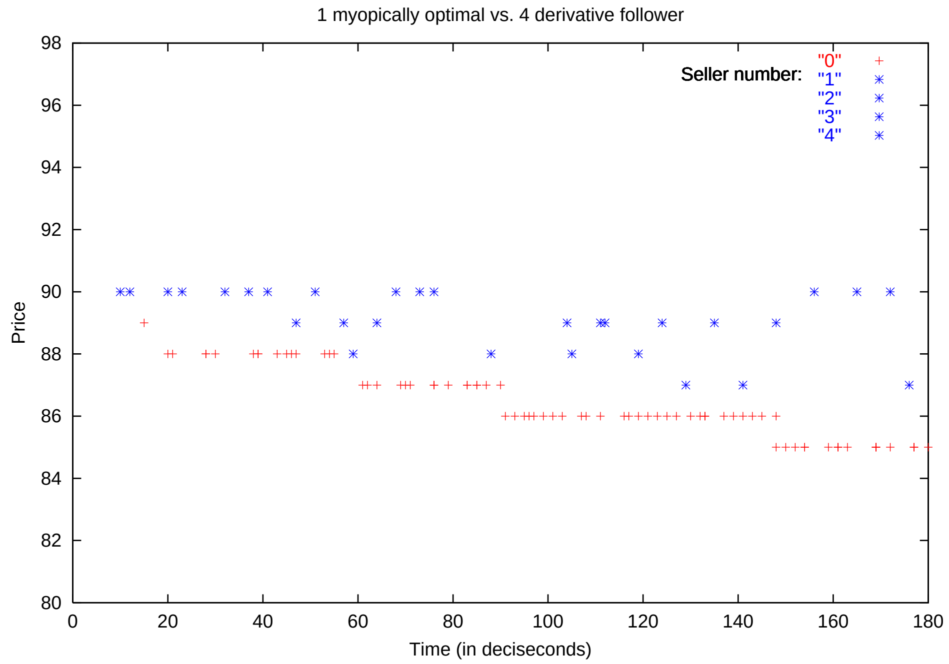

Figure 15: 4 derivative followers vs 1 myopically optimal

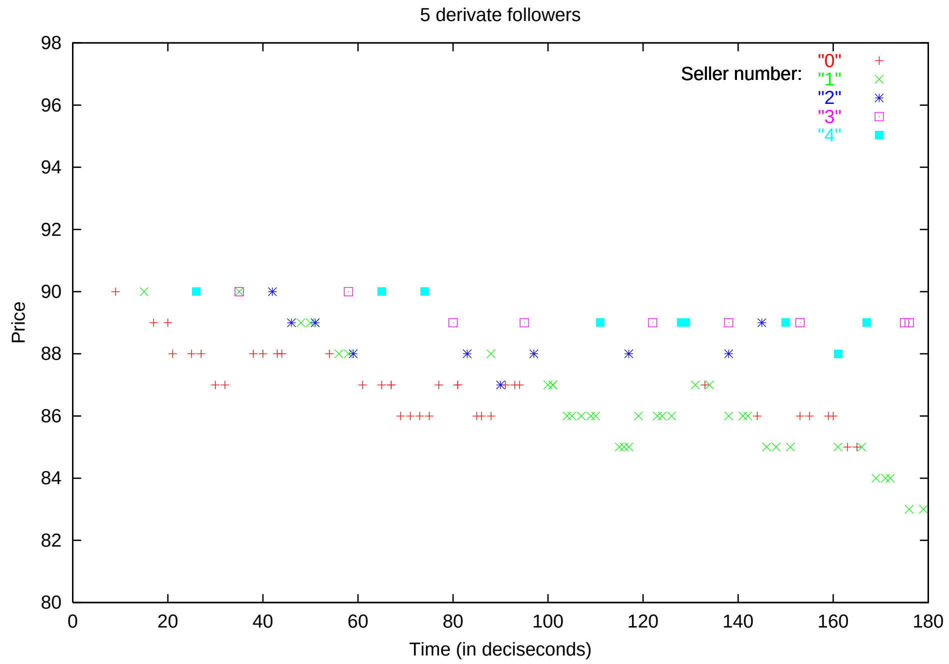

Figure 16: 5 derivative followers

Below the price of (p^*=72), undercutting is not worthwhile because the profit margins are so low that it is better to charge the monopolist price $p_m = 90$ and accept the low 8% market share. Due to the generally low prices, each GT pricebot earns relatively low profits of 1.5 (would have been higher by 0.1 if the more appropriate p^*=73 was used) on average, as compared to the theoretical maximum of 4 that could be obtained by a collusive cartel in which each seller charges the monopolistic price of 90. Now consider the 5 MY sellers. As is clear from the figure, they undercut one another until the price falls to p^* , at which point undercutting becomes less profitable than charging the monopolistic price of 90. Thus prices suddenly jump up to this level. However, as soon as this occurs, undercutting once again becomes attractive, and the price war cycle begins anew. Although the MY sellers fall into endless price wars, their average prices are higher than those of GT sellers. This is reflected in their much higher average profit: 2.3 vs. 1.5 in this example. Now consider the 5 DF sellers. Interestingly, although they are the least informed, they maintain the highest prices and therefore the highest profits. Their average profit of 3.3 is very close to the optimal value of 4 that could be attained by a cartel.

Next, by pitting different pricebots against each other, we try to find the best of the three strategies.

From the figures, it is clear that when a single GT or MY agent is pitted against 4 DFs, it fares much better than does a DF. GT consistently undercuts the 4 DFs because it is biased toward low prices (though it is not able to garner more profits), MY is even more effective because it undercuts the 4 DFs by the minimal amount necessary to grab the 68% market share. Thus, if agents were permitted to select pricing strategies on the basis of expected longterm payoff, a society of 5 DFs would be unstable. The first agent to reconsider its strategy choice would switch to MY, as would each successive agent until all were converted to MY. The situation is analogous to the Prisoner's Dilemma: *self-interest compels all of the sellers to defect to the MY strategy, even though this leads to a lower profit than would be obtained if they were to adhere to the DF strategy.* A DF is also exploited when it is pitted against 4 MYs. GTs, because of their penchant for pricing low, neither are exploited, nor are able to exploit other pricebots. Of all the three strategies considered, MY should be the algorithm of choice whenever detailed buyer information is available. Otherwise, a simpler strategy like DF may be the only choice. In practice, the detailed buyer information required by MY and GT is unlikely to be obtained very easily. Next, we discuss techniques that could make MY better – essentially by making it faster and making it less myopic.

▮ *Fast Myoptimal*

When MY's re–pricing rate is increased, what should happen? Intuitively, it is clear that such a pricebot would spend more of its time undercutting its competitors and so, should be better off. Simulations confirm that, if all sellers adopt MY then the one that re–prices fastest makes the maximum profit.

Developing this situation further would imply that each pricebots needs to get pricing information of its peers faster. And the only way to get this information is to employ shopbots for the purpose. If all such pricebots are involved in a race of who could re–price fastest, it would enormously load pricebots (because in the end, they are being queried) and shopbots (used as intermediaries to query pricebots). For this, the shopbots would gain little (since the pricebot which submits a query to it does not read the advertisement posted on the page). Similarly, pricebots would gain little in–spite of the tremendous load being put on them (since no actual deals are taking place). A natural conclusion of all this would be that both shopbots and pricebots would start charging for the pricing information they provide, thus, re–introducing the search cost, which we assumed previously to have been eliminated due to information economy. There is still no denying that the search cost would be a small fraction of what it was previously and would largely be negligible in case of medium to high value items.

Another likely development is that sellers would price their wares only on demand since it is pointless and inefficient to change the price of an item more frequently than it is being requested by buyers. But, this could generate an endless pricing loop unless care is taken. For example, when a buyer expresses interest, seller s_1 starts asking other sellers (since it does not have a ready–made price), which in turn would again query s_1 if they follow a similar strategy. One way to avoid such problems would be to set up a reverse auction whenever a buyer expresses interest. Sellers would bid against one another, and the lowest bidder would be obligated to sell the commodity to the buyer for the agreed upon price. Thus the natural limit of ever faster *non–negotiable dynamic pricing* would appear to be *negotiable dynamic pricing mechanisms* such as auctions, in which the dynamics may occur over the course of the transaction.

- *Foresight and machine learning*

The seller instead of being myopic must learn to have more foresight. The algorithm would improve if the seller agent while deciding its next price could also take into account the anticipated pricing behavior of its competitors. One promising method research on which is carried out is $Q - learning$[1].

4.4.3 Simple extensions

Some simple extensions to our simulations would be desirable and can be easily implemented. Instead of the buyer agent being either black or white (either a random buyer or a bargain hunter), it would be more desirable if it has shades of grey i.e. the buyer could choose to compare any number of sellers between 1 and S (the total number of sellers). This could be a function of how costly searching is and how much time and effort does a buyer agent have to spare on searching.

The valuation of the buyer's valuation instead of remaining fixed, should be represented by a probability distribution function, indicating how dearly a buyer wants to purchase a good.

Chapter 5

Conclusions and future works

The information economy of tomorrow promises to be the largest multi-agent system ever envisioned, with numbers of agents running into the billions. Building economic behavior into the agents offers a twofold promise: their myriad interactions and conflicts will be governed by the same economic principles that have lent coherency to the activity of billions of humans, and they will be able to operate in the same economic space as humans, and thus can interact profitably, both with their peers and human beings. But, before indulging in an all–agent economy, we need to simulate agents' behavior and find out the ramifications when all of them follow some pre–defined strategies. Or else, wide chaotic economic swings like cyclical price wars could become more frequent (since agents are not subject to the restraints that normally rein economic activity – their transactions take place almost instantaneously, cost is next to nothing and distance is irrelevant).

This project is basically an attempt to do the above. We have tried to model complex human transactions in our framework – we allow the user to specify a set of items for a transaction instead of specifying each item in a separate transaction and as a result, the optimal transaction can be chosen from a larger search space. As a testimony to the generality of our framework, we have successfully modeled and simulated the interactions of shopbots and pricebots in a *dynamic posted pricing* environment. The results of this simulation are promising and show that collective behavior of agents may not resemble that of humans. Learning does not come intuitively to them, and as a result, they are involved in cyclical price wars.

Through automated trading by software agents, we expect improvements in the quality of existing markets, such as consumer goods markets, and to reap the benefits of markets as effective instruments of resource allocation also in non-traditional domains, such as fine-grained markets for electric power[10]and communication bandwidth, selling excess CPU time or storage capacity([14], [15], [16]), constraining peak usage of networked resources and improving network utilization([17], [19], [18]). Future work, in form of applying our framework to these non–traditional domains, could be carried out.

That to evolve, agents will have to learn, adapt, and anticipate is a statement that needs but little reflection to agree with. For this, they would have to use a variety of machine learning and optimization techniques. Further work needs to be done in this direction and the complexity here is in learning when the environment is constantly changing. Also, the questions of how simulations could be carried out in a more complex environment needs to be thought of. The challenge is to have buyer and seller agents mimic the complex behavior and strategic thinking of humans and then, explore and analyze the resulting interactions among them. Further, we would want that agents change their strategies dynamically on the basis of results obtained by performing data mining on the bids received till then. Also, in our framework, deadline has been used primarily to denote the validity period of bids. Studies on how to respond to bids in an optimal fashion keeping the deadlines in mind could be formulated. A slightly different direction of future work could be the development of an ontology for tradable goods[12] and integrating it with our framework.

Coming back to simple but highly usable features of our project, the GUI provides a chance to play with the software. Also, features like opening, slightly modifying and then re–saving a configuration file are useful for fine–tuning the values in a configuration file and finding the optimum configuration when other peers follow some fixed strategies. The peer–to–peer framework also ensures that the central agent is not excessively loaded.

Bibliography

[1] J. Kept, J.E. Hanson and A.R. Greenwald. "Dynamic pricing by Software Agents", IBM Institute for Advanced Commerce, T.J. Watson Research Center

[2] R. Guzman and P. Maes. "Agent–mediated Integrative Negotiation for Retail Electronic Commerce." Appeared in the *Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET '98).*

[3] A. Chavez and P. Maes "Kasbah: An Agent Marketplace for Buying and Selling Goods." *Proceedings of the First International Conference on the Application of Intelligent Agents and Multi–Agent Technology (PAAM '96)* pp. 75–90 . London, UK, April 1996. Available from URL: <http://agents.www.media.mit.edu/groups/agents/publications>.

[4] Trading Agents, by Dianand Roeland available from the URL: <http://cs.rug.nl/ ben/agents/trading-agents.html>.

[5] M. N. Huhns. An Agent–based Global Economy *appeared in Internet Computing.*

[6] J. Nash. Non–cooperative games. Annals of Mathematics, 54:286–295, 1951.

[7] J. Kephart and A.R. Greenwald, Shopbot Economics. *In Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*

[8] A.R. Greenwald and J.O. Kephart. Shopbots and Pricebots, *Proceeding of the First ACM Conference on Electronic Commerce.* ACM press, November 1999.

[9] A.R. Greenwald, J.O. Kephart, and G.J. Tesauro. Strategic pricebot dynamics, In M. Wellman, editor, *Proceeding of the First ACM Conference on Electronic Commerce.* ACM press, November 1999.

[10] H.Green. Good–bye to fixed pricing? *Business Week*, pages 71–84, May 4 1998.

[11] The F–Connection: Families, Friends, and Firms and the Organization of Exchange, *an old paper from the 1980's that talks about how identity is important for transactions.*

[12] CommerceNet homepage, http://www.commerce.net

[13] A. Herzberg and H. Yochai. MiniPay: Charging per click on the web. *In proceedings of the Sixth International World Wide Web Conference*, April 1997. URL: http://www.hr.il.ibm.com/mpay

[14] Yair Amir, Baruch Awerbuch, R. Sean Borgstrom, A Cost-Benefit Framework for Online Management of a Metacomputing System, *Proceedings of the First Int. Conf. on Information and Computation Economics ICE'98, Charleston, SC, USA, October, 1998.*

[15] Ori Regev and Noam Nisa, The POPCORN Market - an Online Market for Computational Resources, *Proceedings of the First Int. Conf. on Information and Computation Economics ICE'98, Charleston, SC, USA, October, 1998.*

[16] Jonathan Bredin, David Kotz and Daniela Rus, Market-based Resource Control for Mobile Agents, *Proc. of the Second Int. Conf. on Autonomous Agents, AA'98, Minneapolis/St. Paul, May, 1998.*

[17] Spyros Lalis, Christos Nikolaou, Dimitris Papadakis and Manolis Marazakis, Market- driven Service Allocation in a QoS-capable Environment, *Proceedings of the First Int. Conf. on Information and Computation Economics ICE'98, Charleston, SC, USA, October, 1998.*

[18] D. Reininger, D. Raychaudhuri and M. Ott, Market Based Bandwidth Allocation Policies for QoS Control in Broadband Networks, *Proceedings of the First Int. Conf. on Information and Computation Economics ICE'98, Charleston, SC, USA, October, 1998.*

[19] C. Courcoubetis, V.A. Siris and G.D. Stamoulis, Network control and usage-based charging: Is charging for volume adequate?, *Proceedings of the First Int. Conf. on Information and Computation Economics ICE'98, Charleston, SC, USA, October, 1998.*

[20] KQML: An agent communication language http://www.cs.umbc.edu

[21] Foundations for Intelligent Physical Agents web page. URL: http://www.fipa.org.