Dynamic Strategies for Negotiating Agents in E-commerce Applications

Nishan Jebanasam   Hana Ma   N. Parameswaran
School of Computer Science and Engineering,
The University of New South Wales
Sydney 2052 Australia
paramesh@cse.unsw.edu.au

Abstract

The world of E-commerce presents ample opportunity to fully utilize the capability of intelligent agents. The highly dynamic, fast-moving and information-rich environment can often be overwhelming for the human participant. Agents can intelligently assist users by mimicking human behaviour and adapting themselves to their client's specification. This paper explores the viability of running a society of agents in a simple marketplace to negotiate on behalf of human clients. The type of agent chosen for this role is a plan-based agent. This architecture offers the flexibility and reasoning power required to maximize the possibilities presented in the environment chosen.

Introduction

In today's information-rich world, users are faced with an increasing inflow of data. With traditional business operations now finding a new platform on the Web, there are rapidly unfolding opportunities for users to harness the Internet to maximize their transactions. However, one of the primary problems with this explosion in E-commerce is the difficulties people face with managing the overwhelming amount of content and possibilities. It is into this situation that agents can (and have) assisted.

The very term "agent" has been used rather loosely, and rightly so, as the concept can be applied to a plethora of situations and Models. However, in the context of this paper, perhaps the most accurate description would be a component of software that is capable of accomplishing tasks on behalf of its client.[1] It has been further proposed that to conform to the definition, this software must communicate in a language, either to other software components (agents) or clients.[1] Even with this in mind, there are a variety of agent architectures that can be applied to E-commerce. They can be any combination of features such as cooperative[2] vs. non-cooperative, distributed[3] vs non-distributed, plan based vs. pure reflexive and a host of other possibilities. Many of the configurations have actually been used in existing systems today. Some of these implementations are briefly mentioned in the next section.

Background

The concept of the agent can carry over to many different fields of computing, from Operating System agents to Mail Agents to Personal Assistants. These agents all have varying complexity and intelligence, depending on the task they are designed to fulfill. Some are basically processes, handling simple requests like file-retrieval or document sorting. Others are more complex, adapting their behaviour to their client's profile and yet others display almost complete autonomy, operating with little external instruction.

It is only in the last decade or so that the role of agents in E-commerce has been examined. Naturally, given the benefits of combining the two concepts, there are many systems that have been developed or are currently being developed that further explore the potential of agents in E-commerce. Outlined below are two systems that use agents aggressively to relieve human clients of some responsibility.

- SICS MarketSpace[4]

   This system provides an agent-based market infrastructure to support all users and services, and allows customers and commercial sites to find matching interests, negotiate and close deals.

- KASBAH[5]

   Also an agent-based marketplace, KASBAH is more self-contained than typical distributed systems. Each agent represents a user, who assigns a specific task. The agent then performs negotiation and settlement of deals automatically.

A Simulated Market

The domain chosen for our Model is comprised of university students who sell and buy second hand items such as books, bicycles, etc. The domain has many characteristics which are easy to define, and even predict. Additionally, the nature of the transactions will be largely second-hand goods. This lends itself to more interesting behaviour, as it is more natural to negotiate for second-hand items rather than new, which are traditionally bought at a fixed, pre-determined price. With this in mind, we can list the known properties of the domain by category, and outline the issues present therein.

The main categories to consider are the student, the item and the market itself. Students typically engage in small-scale transactions (few dollars to a few hundred dollars). These transactions can go over relatively long periods of time (days, months, years) compared to an online auction or other traditional e-commerce transactions. Furthermore, students are affected by the annual academic timetable,

with its exams, holidays etc. The items that students generally buy or sell will be simple second-hand objects, such as textbooks and furniture. They will most probably be household-related, and can in some cases be intelligently grouped for sales (e.g., all textbooks of the same subject, or putting a sofa and a coffee table together). Lastly, the marketplace itself will be a decentralized, second-hand market, with multiple simultaneous negotiations at any one time. Agreements will not be legally binding.

There are, of course, certain problems inherent in the domain outlined. Firstly, there is no real efficient organized decentralized second-hand marketplace found in most universities. At best, they may have second-hand notice boards within campus, where students can add and read notices. This in turn makes gathering complete market information prohibitively time-consuming. Additionally, there are no standardized negotiation protocols present. Traditional verbal negotiations tend to vary in their nature, from an "instant-agreement" offer to drawn out haggling, involving comparisons between likely offers, "testing the water" bids and even misrepresentation of information (e.g., pretending to have other interested parties). It is precisely these difficulties that our Model is designed to overcome.

The GINA System
The name of the system we have created is GINA (Agent Negotiation with Independent Goals). GINA includes a society of intelligent agents each representing one human client, a decentralized second-hand marketplace (akin to the classified ads) in which the agents can operate, a server which maintains the agents and coordinates agent communications, and a web interface for the client.

The type of agent we chose to create is a BDI (Belief-Desire-Intention) [6.] agent. The Belief is what the world is like now (current State), the Desire is what we would like the world to be (Goals) and the Intention is what we actually choose to carry out (Plans). The agent is autonomous, plan-based, goal-oriented, heavily conversational and is designed to function over a long period of time (months) in a dynamic environment. It achieves this using a Time Line Structure (TLS), a Rule Base and an Agent Communication Language. Each of these will be described in more detail. From a client perspective, the agent is first issued tasks. *Tasks* are simple, high-level requests (eg "Buy" or "Sell" an item, given the item details). An agent can handle any number of tasks, so at a given time, a single agent can be simultaneously trying to sell a bicycle, telescope and bed and also trying to buy a lamp and a Chemistry textbook. Once given a task, an Agent will attempt to initiate a deal with another Agent. A *deal* represents a 1-1 negotiation between a buyer and a seller, and each task can have multiple ongoing deals. Therefore, for the "Buy a bicycle"

task, the agent can negotiate with several agents selling bicycles.

The marketplace is a simple, decentralized second-hand market, comprised of notices which contain information about an offer (either to buy or sell an item). The notices are partitioned into a Wanted notice board and a Selling notice board.

The System Server that is used to manage the agents essentially forwards messages from agent to agent. It also handles agent requests for market information (eg How many tables are being sold currently?) and processes client requests via the Web.

Lastly, the web front-end itself provides the necessary interface to access the Marketplace and give instructions to the client's agent.

Time Line Structure
One of the core features of our agent is the Time Line Structure (TLS). It is essentially a sequence of the agent's intentions at any given time, and its most interesting feature is the ability to change these intentions while they are still in the future. Every agent possesses its own TLS, and all plans that are generated are placed on this TLS abstracted out in the form of goals. These goals are high-level abstractions of what an agent wishes to accomplish, and as such can be broken down (expanded) into actions, which are atomic actions an agent can execute. The actual design of the TLS is a tree structure, with high-level goals forming the nodes near the top, and sub-goals branching down, and actions lying at the leaf-level. The actuator of the agent, the Executor, is essentially the "engine" that moves down the TLS at a controlled rate and executes the actions that lie along it. Thus, while these generated goals reside on the TLS, they can be postponed, brought forward or even removed. For example, if a message was received some time in the past indicating that another agent wishes to accept an offer this agent made, then this agent will have placed a goal on the TLS to close the deal with that bidder. However, if in the meantime another agent sends an even better acceptance, then the agent can "insert" another goal, which will close the deal with the second agent and reject all pending deals with other agents.

The implication of this design is that in the constantly changing world of the agent, any given agent is not "committed" to following a particular course of action. As new information becomes available, the agent can accordingly change its TLS to take advantage of the updated real-world knowledge, thereby maximizing it's flexibility. This in turn improves it's ability to negotiate, as it has the luxury of being aware of it's own plans, while at the same time it is not tied down to following them. From a programming point of view, the agent is not merely a static procedural program with one path of execution, but rather a program that can "examine" it's own path of execution and accordingly modify it.

Rules and Plans
Being a plan-based, goal-oriented agent, it necessarily follows that at the heart of the agent lies a means of generating these plans. This is the Rule Base. In essence it is a set of conditional statements that examine incoming information and generate a plan in response to them, based on both the new information available, and the present and previous states of the agent. The plan generated will in turn modify the execution of the agent by manipulating the TLS (placing new goals, postponing/deleting existing goals, etc). It is evident, therefore, that the actual rules themselves play the single largest part in determining the behaviour of the agent. It is this feature that facilitates *agent Modes*, whereby the agent can display vastly different kinds of behaviour by "tuning" some of the rules that lie at its core. As a simple example, suppose there is a rule in the Rule Base that says –

Fig 1.

```
IF: An acceptance message is received from
    another Agent
AND: The bid lies within my Max/Min Price
THEN: - Examine all my current deals
      - Determine the most promising deal
        by considering current price and
        predicted trend
        IF: The deal being accepted is the
            most promising
        THEN: Confirm the acceptance
        ELSE: Reject the acceptance
```

This rule captures the basic, high-level plan in response to an acceptance by another agent. It exhibits a rather prudent, opportunistic approach to negotiation by always pursuing the most promising deal. However, if we wanted our agent to be more lassez-faire, we could modify the rule to read –

Fig 2.

```
IF: An acceptance message is received from
    another Agent
AND: The agreement lies within our MAX/MIN
     price
THEN: Confirm the acceptance
```

This would make the agent somewhat more "simple-minded" and easy-going, basically pursuing the most immediate deal. Both forms of behaviour can have their advantages. In the case of the former, it would ensure that if we are going to complete a deal, it would be the best possible one for our purposes. However, it'll most likely take longer to finalize a deal by necessarily being more "choosy". In the case of the latter, it will almost certainly result in a quick agreement, but not necessarily the best possible outcome.

Communication
One of the generally agreed on properties of an agent is that it communicates in an expressive language with its peers. Naturally, agents in an E-commerce environment will be no exception to this, and there has been considerable research into suitable languages (KQML[7], KIF[8] etc). However, for the purposes of our system, it was decided to use a custom language that covered all our needs. This decision was largely due to the system being closed and self-contained, negating any need for external compatibility. The language used is almost pure English, and is parsed by the agent to extract meaningful data. Hence, a typical message might be: "A:coffer:12:36.25" which would read as AGENT sends COUNTEROFFER for ITEM 12 of $36.25. Although simplistic, it covers the basic requirements of the system to conduct negotiations. The agents exchange absolutely no internal state information among themselves. This is due to the self-interest motivation, as each agent represents their own client and no one else. Any information about other agents is purely inferred (eg rate at which an agent drops his price, frequency of negotiation messages etc).

Implementation
The entire system is implemented in Java, with the Web interface using Servlets, JavaScript and standard HTML (and Forms). Each agent runs as a separate process, started by the Server process. The main component of the agent is the Goal Generator, which collects all incoming messages and parses them one by one. Each message will trigger rule in the Rule Base, and the TLS will be modified. Each agent also has an Executor, which is allowed brief control to move along the TLS and execute a few nodes. When the Executor is halted, the cycle begins again. In this way the agent conforms to the behaviour of the classical Goal-Based Intelligent Agent: sense what the world is like now, update internal state, generate a plan based on the goal desired, schedule the plan on the TLS, and finally execute the plan (actuate). The Server also runs as a process, with a separate thread to collect inbound messages and another thread to process and send them. The marketplace is implemented as an Object-Oriented database, ObjectStore PSE. Its advantage is that it allows Java Objects to be made persistent, so there is little difficulty in storing and retrieving the Notice objects. The database queries can be initiated by the Agent, the Server or the Web Servlets.

Performance/Evaluation
The agents in the system can behave in one of eight different *Modes*, specified by the client upon creation. These are Careful, Easy, Desperate, Risky, Normal,

Greedy, Opportunistic and Methodical. Each Mode is a preset combination of rules which mimic the human attributes described by the name of the Mode. The finer details regarding the tuning of the rules to achieve these Modes is too lengthy for this paper.

Furthermore, within each agent, every task can be assigned one of three different Price Curves. A *Price Curve* guides the agent's rate of bid price increase/decrease. There are currently three Price Curves available to choose from: Exponential, Linear and Logarithmic. Between them they broadly cover the ways in which a human negotiator would raise or drop his offer.

The Mode and Price Curve parameters together determine the negotiation strategy that the agent pursues. In essence, the rules are responsible for the higher-level decisions in the negotiation (e.g. which deal to favour), while the Price Curve provides the mathematical foundation for calculating bids.

All test runs consisted of running a small group of agents, each with different behavioural Modes and/or Price Curves. The agents were given tasks to buy or sell items, and left to initiate deals and negotiate among themselves.

Experiment 1
We firstly examine the effect of varying the Price Curve between the agents, with their behavioural Modes left at Normal. Each agent was given one task.

Fig 3.

| | Seller 0 |
	Buyer 1
	Buyer2
	Buyer3
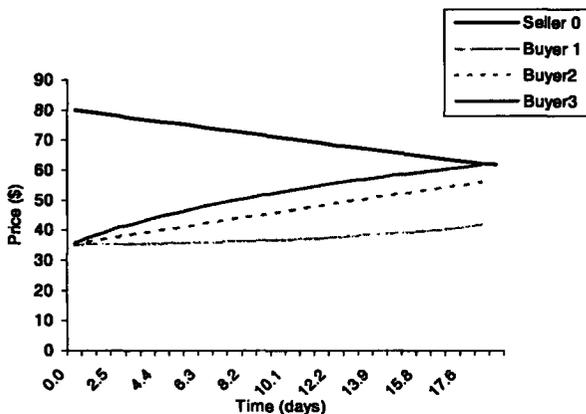
In the above graph, the y-axis represents an offer price, and the x-axis shows the time elapsed since the start of the negotiation. As can be seen above, Buyer 3 clinched the deal first, since the logarithmic curve forced him to raise his price more in the early stages of negotiation, while the linear rose slower, and the exponential slowest of all. Varying only the Price Curve merely reveals the agent's mathematical calculations, however. It is the behavioural Modes that reveal how changes to the Rule Base affect the Agent.

Experiment 2

The following tests show how a particular behaviour Mode performs in a given society. In each test, we have a configuration of three Control Agents each with a set pre-determined behaviour and a fourth Variable Agent whose behaviour is changed in each test run. The Control Agents are all selling chairs, of which the Variable Agent wishes to buy one. Furthermore, the Variable Agent is selling a lamp, which the Control Agents are competing to buy.

Fig 4.
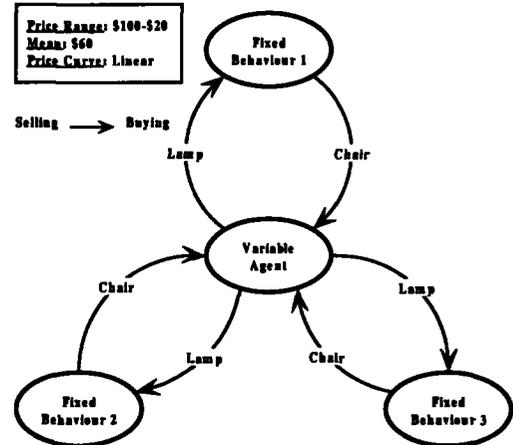
All agents are negotiating between $100-$20, with a Linear Price Curve (the mean price, therefore, being $60). The results are shown by the final agreed price.

In environment A, the three Control Agents are Greedy, Methodical and Desperate. This represents a "random" society with two tough agents and one weak. In environment B, they are Careful, Easy and Opportunistic. This is also a random society with two weak agents and one tough. Environment C is a "tough" environment as the three Control Agents are Opportunistic, Greedy and Methodical. Lastly, environment D is a "weak" environment where the three Control Agents are Careful Easy and Desperate, creating a less aggressive society. Table 1 illustrates how the Variable Agent has performed in the different environments, with each column representing a particular society, and each row being a Variable Agent within that society. *The figures in the table show the final agreed bid price.*

Table 1.

Variable Agent Mode	Seller's Performance ($)				Buyer's Performance ($)			
	A	B	C	D	A	B	C	D
Careful	61.40	60.10	59.41	61.15	58.90	59.23	61.08	58.89
Desperate	60.65	60.33	58.22	60.32	60.33	60.73	62.37	59.84
Easy	60.40	60.74	60.76	60.54	60.01	59.81	62.17	59.84
Greedy	61.82	61.08	59.40	62.12	59.32	60.38	Failed	58.13
Methodical	62.73	62.36	60.66	62.21	57.70	59.21	60.27	57.53
Normal	61.32	61.48	58.73	60.92	57.78	60.26	60.76	59.30

Opportunistic	61.96	61.49	60.50	62.24	57.81	59.22	60.10	57.85
Risky	58.66	61.10	59.06	60.99	61.38	58.91	61.12	58.92

By testing the eight types of Agents in the different environments, it can be seen how their negotiation strategies performed (in terms of final price). In general the weaker agents finalized deals at poorer prices (selling cheap and buying expensive), while the intermediate agents fared better, and the tougher agents usually closing very good deals indeed.

The society that forms the agent environment also has an impact on the deals, since the results indicate that the tougher the environment, the poorer most of the Variable Agents perform. Therefore, it would appear that in a strong Market, with plenty of supply/demand and competition, a tougher agent would be an ideal choice. However, if the Market is weak, then weaker Agents will at least close deals by attracting negotiations with their willingness to finalize at poorer prices for themselves. This can be seen in the random societies A and B, where the Variable Agent "targeted" the weakest Control Agent to finalize a deal with, although this detail is not shown in the table. These trends are what we would expect from the real world this system simulates.
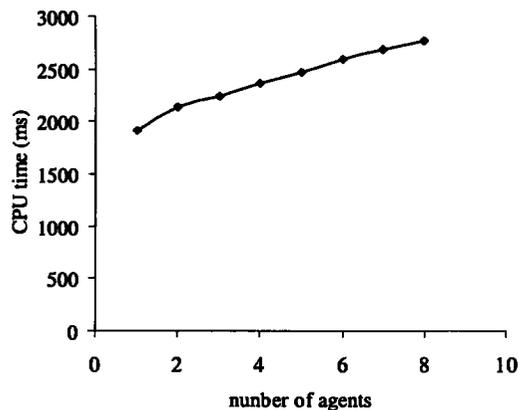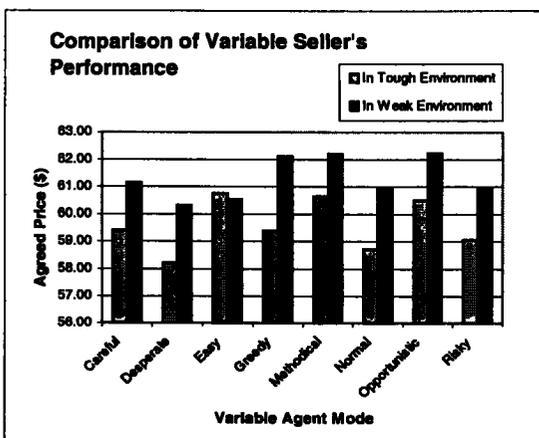
Fig 5.

Fig 6.

These graphs further outline the impact of the tough and weak Market environments (C and D) on the Variable Agent. Again, the final agreed price is the evaluating factor, and it can be seen how the Variable Agent took advantage of the weaker Control Agents or were taken advantage of by the stronger Control Agents.

Conclusion

As can be seen, the Model implemented has the basic structure to support many more interesting features. Adding sophisticated functionality like allied-agent negotiations, grouping of items and variable autonomy (degree of agent freedom in decision making) can all be built in by adding to the Rule Base. Furthermore, the entire agent can be carried over to other domains apart from E-commerce by "plugging in" a different Rule Base. The underlying architecture need not change. By using the concept of the TLS, the agent can plan ahead while remaining extremely flexible to changes in the dynamic environment.

References

1. Genesereth, Michael R., and Ketchpel, Steven P. July 1994. Software Agents *Communications of the Associations for Computing Machinary*, pp 48-53
2. Guttman, Robert H., and Maes, Pattie 1998. Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce *Cooperative Information Agents* (p135-147)
3. Dr Gauch, Susan, and Zhu, Xiaolan Distributed Intelligent Search Agents for the World Wide Web, http://www.ittc.ukans.edu/~sgauch/agentdetails.html
4. Chavez, Anthony, and Maes, Pattie 1996. KASBAH: An Agent Marketplace for Buying and Selling Goods. *Proceedings of PAAM '96* Practical Applications Company
5. Eriksson, Joakim, and Finne, Niclas and Janason, Sverker May 1998. SICS Marketspace – An Agent-Based Marketplace Infrastructure Proc. of The Workshop on Agent Mediated Electronic Trading, AMET-98, Minneapolis/St. Paul
6. Rao, A. S. and Georgeff, M. P. 1995. BDI agents from Theory to Practice Tech. Rep. 56, Australian Artificial Intelligence Institute, Melbourne, Australia
7. Finin, Tim, and Labrou, Yannis 1994. A Semantics Approach for KQML – A General Purpose Communication Language for Software Agents. In *Third International Conference on Information and Knowledge and Management* (p447-455)
8. KIF http://logic.stanford.edu/kif/kif.html