2002s-23

A Rule-driven Approach for Defining the Behavior of Negotiating Software Agents

Morad Benyoucef, Hakim Alj, Kim Levy, Rudolf K. Keller

Série Scientifique

Scientific Series

CIRANO
Centre interuniversitaire de recherche en analyse des organisations

Montréal
Mars 2002

ISSN 1198-8177

A Rule-driven Approach for Defining the Behavior of Negotiating Software Agents[*]

Morad Benyoucef[†], *Hakim Alj*[‡], *Kim Levy*[§] *and Rudolf K. Keller*[**]

Résumé / *Abstract*

Un des inconvénients qu'on retrouve fréquemment dans les systèmes de négociation par agents est qu'ils reposent sur des schémas ad-hoc, non adaptatifs et figés dans le code pour représenter le comportement des agents. Cette limitation est probablement due à la complexité de l'activité de négociation elle-même. En effet, au cours de la négociation, les agents logiciels (humains) ont des décisions difficiles à prendre. Ces décisions ne sont pas seulement basées sur l'information disponible sur le serveur de négociation, mais aussi sur le comportement des autres participants durant le processus de négociation. L'information et le comportement en question changent constamment et sont très incertains. Dans la première partie de l'article, nous proposons une approche à base de règles pour représenter, gérer et explorer les stratégies de négociation ainsi que l'information de coordination. Parmi les nombreux avantages de la solution proposée, on peut citer le haut niveau d'abstraction, la proximité avec la compréhension humaine, la souplesse d'utilisation et la possibilité de modifier le comportement des agents durant le processus de négociation. Pour valider notre solution, nous avons effectué plusieurs tournois entre agents et utilisé l'approche à base de règles pour implémenter des stratégies simples applicables à l'enchère anglaise et à l'enchère hollandaise. Nous avons aussi implémenté des schémas simples de coordination impliquant plusieurs enchères. Le travail de validation, en cours, est détaillé et discuté dans la seconde partie de l'article.

One problem with existing agent-mediated negotiation systems is that they rely on ad hoc, static, non-adaptive, and hardcoded schemes to represent the behaviour of agents. This limitation is probably due to the complexity of the negotiation task itself. Indeed, while negotiating, software (human) agents face tough decisions. These decisions are based not only on the information made available by

[*] This research is supported by Bell Canada, BCE Emergis, NSERC (National Sciences and Engineering Research Council of Canada), and CIRANO (Centre Interuniversitaire de Recherche en ANalyse des Organisations).

[†] CIRANO and Département d'informatique et de recherche opérationnelle Université de Montréal C.P. 6128 Succursale Centre-ville, Montréal, Québec, H3C 3J7, Canada. Email : benyouce@iro.umontreal.ca
[‡] DIRO and CIRANO. Email : aljh@iro.umontreal.ca
[§] DIRO and CIRANO. Email : levyk@iro.umontreal.ca
[**] CIRANO and Zuehlke Engineering AG Wiesenstrasse 10a, CH-8952 Schlieren, Switzerland. Emaill : ruk@zuehlke.com.

the negotiation server, but on the behaviour of the other participants in the negotiation process as well. The information and the behaviour in question are constantly changing and highly uncertain. In the first part of the paper, we propose a rule-driven approach to represent, manage and explore negotiation strategies and coordination information. For that, we divide the behaviour of negotiating agents into protocols, strategies and coordination. Among the many advantages of the proposed solution, we can cite the high level of abstraction, the closeness to human understanding, the versatility, and the possibility to modify the agents' behaviour during the negotiation process. To validate our solution, we ran many agent tournaments, and used the rule-driven approach to implement bidding strategies that are common in the English and Dutch auctions. We also implemented simple coordination schemes across several auctions. The ongoing validation work is detailed and discussed in the second part of the paper.

Mots-clés : Négociation électronique, enchères en ligne, agents logiciels, stratégie de négociation, coordination, système à base de règles, moteur de règles

Keywords: *e-negotiation, online auction, software agent, negotiation strategy, coordination, rule-based system, rule engine*

1. Introduction

According to Kephart et al., over the course of the next decade, the global economy and the Internet will merge into an information economy bustling with billions of autonomous software agents that exchange information goods and services with humans and other agents [1]. In addition to exchanging information, software agents can be programmed to search, compare, learn, negotiate, and collaborate [2], making them particularly useful for the information-rich and process-rich environment of electronic commerce (e-commerce) [3]. As e-commerce usually involves information filtering and retrieval, personalized evaluation, complex coordination, and time-based interaction (among other things), it can greatly benefit from the introduction of software agents. Therefore, we talk of agent-mediated e-commerce. A simple e-commerce transaction can be seen as a three-phase scenario: (1) finding a partner for the transaction; (2) negotiating the terms of the transaction using a recursive process; (3) and carrying out the transaction. In this research, we are interested in the negotiation phase, and particularly in its automation by way of software agents capable of mimicking the behaviour of human negotiators. It has been rightfully stated that agent-mediated negotiation absorbs many of the costs and inconveniences of manual negotiation [4].

We see the negotiation process as a form of interaction made of *protocols* and *strategies*. The protocols comprise the rules (i.e., the valid actions) of the game, and, for a given protocol, a participant (human or software) uses a strategy (i.e., a plan of action) to maximize her utility [5]. Based on this, many strategy-enabled agent-mediated negotiation systems have been described in the literature. Unfortunately, most of them use hardcoded, predefined, and non-adaptive negotiation strategies, which is evidently insufficient in regard to the ambitions and growing importance of automated negotiations research. The well-known KASBAH agent marketplace [6] is a good example of such systems. To overcome this shortcoming, we believe that negotiation strategies should be treated as declarative knowledge, and could, for instance, be represented as *if-then* rules, and exploited using inference engines.

The focus of our research is on *combined negotiations* [7], a case where the consumer combines negotiations for different complementary products that are not negotiated on the same server. For instance, a consumer may want to simultaneously purchase an item and its delivery by engaging in separate negotiations. If software agents are assigned to these negotiations, this poses a coordination problem between them. Many multi-agent negotiation systems found in the literature still rely on ad hoc schemes to solve this problem [8][9]. Again, we believe that a declarative approach should be used to describe and manage the coordination of agents across several negotiations.

To validate our approach, we designed and implemented an automated negotiation system called CONSENSUS [7] that enables a human user to instantiate one or more software agents, provide them with negotiation strategies, as well as coordination know-how, register them on corresponding negotiation servers, and launch them. The agents use the strategies to negotiate according to the protocol dictated by the server, and the coordination know-how to coordinate their actions. An example of a strategy, applicable to an English auction (one of many existing negotiation protocols) is: "*If you notice any form of jump bidding in the auction, then stop bidding and quit*". Jump bidding means making a bid that is far greater than necessary in order to signal one's interest in the auctioned item (See Section 4). An example of coordination know-how, applicable to two agents bidding as partners in two separate auctions for two complementary items is: "*If your partner looses in its auction, then stop bidding and wait for further instructions*" (See Section 4). We are currently testing various strategies and coordination schemes by way of agent tournaments. A large part of the paper is dedicated to this ongoing validation work.

In the remainder of the paper, Section 2 gives some background on strategy-enabled agent-mediated e-negotiations, then presents and justifies our solution. In Section 3, we detail our approach to the coordination of negotiating software agents after providing some background on the subject. Section 4 is dedicated to the experiments we are currently conducting in order to validate our approach. We wrap up the paper with a conclusion in Section 5.

2. Negotiation Strategies

In this section, we introduce agent-mediated e-negotiation, and then address the challenges of strategy-enabled negotiation systems. Afterward, we give our view on protocols, strategies and coordination, and present our approach in representing negotiation strategies.

2.1. Agent-mediated E- negotiation

Electronic negotiation (e-negotiation) takes place when the negotiating function is performed by (networked) computers. Fully automated e-negotiation requires that all parties involved be software agents, semi-automated e-negotiation involves a human negotiating with a software agent, and manual e-negotiation refers to processes in which all parties are human [10]. Within fully automated e-negotiation, Parkes et al. identify autonomous and semi-autonomous agents. An autonomous agent requires a complete set of preferences in order to represent the user in all situations. On the other hand, a semi-autonomous agent will bid when it has enough knowledge to proceed, and query the user when its best action is ill-defined given the current information [4]. Finally, software agents can be involved in competitive negotiations when they are adversaries with conflicting interests. They can be involved in cooperative negotiations when they all aim at satisfying the same interest [11]. The most popular negotiating agents on the Internet are reservation-price agents, which are capable of bidding in online auctions on behalf of a human user. Commercial online auctions such as eBay[1] refer to this as *proxy bidding*.

In addition to their use in e-negotiations, agents can be, and are actually used, in other phases of an e-commerce transaction, including shopping, advertising, delivery, marketing and sales analysis [1]. Pricebots, for instance, are software agents that employ price-setting algorithms in an attempt to maximize profits, thus helping sellers to increase flexibility in their pricing strategies. An example taken from [12] points to books.com[2], which uses a pricebot to monitor prices on competitor sites and offer the customer a lower price than what the competitors ask for. This is called real-time dynamic pricing. Shopbots, to cite another example, are software agents that automatically gather information from multiple on-line vendors about the price and quality of consumer goods and services [12]. A successful shopbot on the web today is mysimon.com[3].

2.2. Challenges of Strategy-enabled Negotiation Systems

Designing, building, and tuning software agents before letting them loose in widely competitive scenarios like e-negotiations, inhabited by (human and software) expert negotiators, happens to be an arduous task [13]. This is why, according to Wong et al. [14], most strategy-enabled agent-based systems use predefined and non-adaptive negotiation strategies in the generation of offers and counteroffers. Commercial online auction sites such as eBay, for instance, offer the possibility of proxy bidding (see Section 2.1) which is actually a software agent using a straightforward strategy: "*bid until you reach your reserve price, by going up each time with a certain bid increment.*" On the academic front, the buying (selling) agents in the KASBAH marketplace (see Section 1) can choose between three negotiation strategies: anxious, cool-headed and frugal, corresponding to linear, quadratic, and exponential functions, respectively, for increasing (or decreasing) their bid (or ask price) for an item over time [6].

Negotiating agents face tough decisions such as whether or not to accept an offer, whether or not to bid, how much to bid, and whether or not to quit. Those decisions must profit from all the information available in the marketplace: available goods and their expected resale value, historical experience on prices, participant's identity and behaviour, etc. [13]. The decisions are made even harder to take by the fact that the market information is constantly changing and highly uncertain – new goods become available, other buyers come and leave, prices keep on changing, no one knows for sure what utility functions other agents have, etc. To complicate things further, the participants' goals, beliefs, intentions are expected to change with time [15].

One piece of information that is vital to the bidder is the valuation of the item at stake. In an auction where the market price is a common valuation among bidders (e.g., an auction for personal computers), a bidder who wins the auction is the one with the highest yet possibly overrated valuation. This is called the *winner's curse*, and it can be avoided by predicting the market price of the item by monitoring its prices in several online auctions [8].

In addition to private and public information, a successful strategy must take into account the strategies of the opponents [13] as well as their reputation [1]. It must also protect against the opponents trying to extract the agent's private information. In a bargaining situation (a bilateral negotiation) for instance, the buyer agent usually knows its

[1] http://www.ebay.com
[2] http:// books.com
[3] http:// mysimon.com

owner’s willingness-to-pay for an item [16]. If the seller agent is made aware of this information, it can make a take-it-or-leave-it offer that will extract the buyer’s entire surplus. As another example, a seller agent knows its owner’s minimum acceptable price and keeps it secret, and a buyer agent starts bidding at zero and offers a sequence of incremental bids until it gets the item for a price slightly more than the minimum acceptable price [16]. This is not to the seller’s advantage. Both examples point to the need for negotiating agents to guard against strategies that extract their private information. Finally, we should mention that, with eBay’s proxy bidding, one must reveal the highest price one is willing to pay, which gives the auction site information that could be used to cheat the bidder [9].

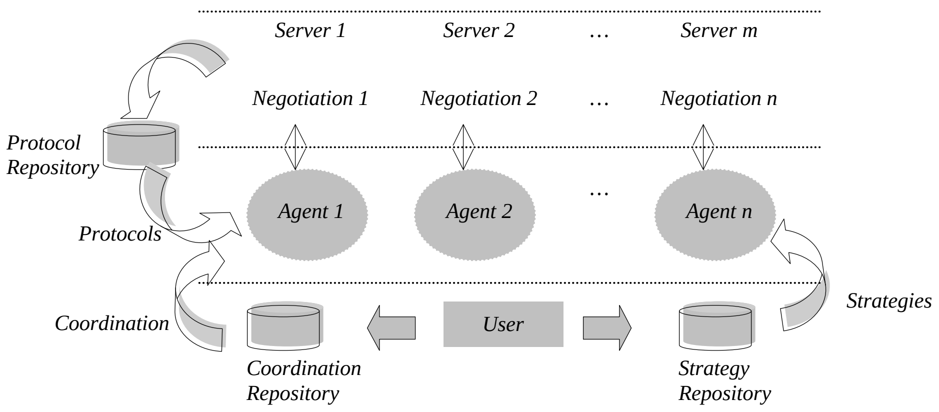


Figure 1: Protocols, Strategies and Coordination in CONSENSUS

2.3. Protocols, Strategies and Coordination

As mentioned in Section 1, the basic components of automated negotiation systems are the protocol and strategies [17]. The protocol defines how the interaction between the agents is to be carried out. The strategies specify the sequence of actions the agent plans to make during the negotiation. In CONSENSUS, the protocol (i.e., the negotiation rules) is communicated to the participant before she engages in the negotiation. A human negotiator would just consult the protocol; a software agent would download it in a usable format (in XML for instance). On the other hand, the strategies are the responsibility of the negotiator. A human negotiator would use her strategies to make offers or to respond to her opponents’ offers; a software agent would do the same, based on the instructions provided by its creator. In addition to protocols and strategies, we introduce a third component: *coordination*. This is the information that the agents need in order to coordinate their actions whenever they participate in a combined negotiation (See Section 1). Figure 1 shows our view of the three components.

2.4. The CONSENSUS Approach

Contrary to most existing approaches in which strategies are coded into negotiation agents, Su et al. use a high-level rule specification language and GUI tools to allow human negotiation experts to dynamically add and change negotiation strategies at run-time [18]. Each strategy is expressed in terms of an Event-Trigger Rule (ETR), which specifies the condition to be checked and the actions to be taken by the negotiation server. Rules are activated upon the occurrence of specific events during the negotiation process. An ETR server is used to manage events and trigger rules that are relevant to the posted event.

A second approach, although not directly related to e-negotiations, is the work of Grosof et al. [19]. It is concerned with the representation of business rules in contracts and it relies on the fact that many contract terms involve conditional relationships, and can conveniently be expressed as rules. An example of a business rule is [20]: “*If the buyer returns the purchased good for any reason, within 30 days, then the purchase amount, minus a 10% restocking fee, will be refunded.*” A declarative approach called Courteous Logic (CL) was devised to represent the rules. CL is an extension of ordinary logic achieved by adding the possibility to express prioritized conflict handling as rules.

Various other approaches are being explored to provide negotiating agents with an adaptive behaviour. They include logic, case-based reasoning (CBR), and constraint-directed search [15]. The use of CBR, for instance, is justified by the fact that negotiation skills in humans seem to come from experience [14]. Bayesian learning is also used to make software agents learn negotiation strategies [20]. According to Kephart et al, agents should learn, adapt, and anticipate, and in order to do so they will use a variety of machine learning and optimization techniques [1].

Inspired by the work of Su et al. [18] and Grosof at al. [19], we take a rather pragmatic approach to the issue of negotiation strategies: we consider them as declarative knowledge that is given to the agents before and/or during the negotiation process. Strategies such as: "*if the bidding gets too intense, then abandon the negotiation*", "*if there is jump-bidding, then wait for further instructions*", or "*if there are snipers, then snipe-back*" can indeed be nicely coded as rules. According to McClintock et al. [21], rules drive the activity in a software application by describing the action(s) to take when a specified set of conditions is met. They can be coded as stand-alone atomic units, separate from the rest of the application logic. Rule-based systems (RBS) are known for their advantages over other software specification approaches. First, rules are at a relatively high level of abstraction, and are closer to human understanding, especially by business domain experts who are typically non-programmers. Second, they are relatively easy to modify dynamically [22]. It is widely recognized in modern software design that a vital aspect of modifiability is modularity and locality in revision. New behaviour can be specified by simply adding rules, without needing to modify the previous ones. Furthermore, rules can be augmented with procedural attachments (e.g., a Java method) so that they have an effect beyond pure-belief inferring [23]. Furthermore, they are executable by custom-made or off-the-shelf rule engines, and can be communicated over a network in adequate formats (XML for instance).

```
   // If you are not leading and you have
not
   reached your reserve price then bid.
Rule Rule1
{
  When {?x: TheElement(iLead == false;
              (highestBid+myIncrement)<=
               myReservePrice);}
  Then {modify? x
                 {action = "BID";}
  }
};
```

```
// If you are not leading and your bid is
greater then the reserve price, then quit.
Rule Rule2
{
   When {?x: TheElement(iLead == false;
            (highestBid + myIncrement) >
             myReservePrice);}
   Then {modify ?x
                {action = "DROP";}
   }
};
```

Figure 2: Proxy bidding in the JRules Syntax

The architecture of CONSENSUS [7] is built on ILOG JRules [24], an off-the-shelf rule engine. The rules are written in a Java-looking syntax, making them relatively self-explanatory. Figure 2 shows two rules in the JRules syntax that implement proxy bidding in an English auction. Rule1 is triggered when the agent is trailing and its reserve price is not met. In this case the agent places a bid equal to the actual bid plus the bidding increment. Rule2 enables the agent to quit the auction whenever its reserve price is met.

3. Coordination of Negotiating Agents

In this section, we define the need for coordination in CONSENSUS, present some approaches to solve the coordination problem, and detail our own approach.

3.1. Background and Related Work

The interaction between agents can vary from simple information interchanges, to requests for particular actions to be performed and on to cooperation (working together to achieve a common objective) and coordination (arranging for related activities to be performed in a coherent manner) [25]. Multiple agents need to be coordinated for the following reasons: preventing anarchy and chaos; meeting global constraints; distributing expertise, resources or information; dealing with dependencies between agents; and ensuring efficiency [15].

In CONSENSUS, there is a clear need to coordinate the work of software agents (see Section 1). First, in a situation where many agents participate in separate negotiations with the goal of purchasing only one item (e.g., they engage in many concert ticket auctions with the goal of purchasing just one ticket), we need to make sure that only one agent finalizes its deal, and that only the agent in the cheapest auction (i.e., the one with the smallest asking price) is the one who bids. It is common in online auctions to forbid bidders from breaking their commitments. Thus, if more than one winning agent end up winning their auctions, they would not have the possibility to retract, or at best they would be allowed to do it, but penalized for it. Second, in a situation where multiple agents negotiate a package of complementary items (i.e., there is no use for one item without the others), we need to make sure that all the items or none are purchased in order to avoid *exposure*. Exposure occurs when we need many items, enter negotiations for all items, and end up making a deal on some but failing to make a deal on others. Coordination in CONSENSUS will be discussed further, but first, let us review some related work.

The Biddingbot [8] is a Multi-Agent System (MAS) that supports users in attending, monitoring, and bidding in multiple auctions. Agents simultaneously monitor prices of the item in several auctions. The system consists of one leader agent and several bidder agents, each one being assigned to an auction site. Bidder agents cooperatively gather information, monitor, and bid in the multiple sites simultaneously. The leader agent facilitates cooperation among bidder agents as a matchmaker, sends user requests to the bidder agents, and presents bidding information to the user.

Another system, designed at the HP Laboratory in Bristol, England, takes a different approach. Instead of a MAS, it uses one single agent [9]. The agent participates in many auctions for one item, and coordinates bids across them to hold the lowest bids. As auctions progress, and it is outbid, the agent may bid in the same auction or choose to bid in a different auction. The algorithm used by the agent consists of two parts: (1) a coordination component, which ensures it has the lowest leading bids possible; and (2) a belief-based learning and utility analysis component to determine if it should deliberately lose an auction in the hope of doing better in another one later.

Not directly related to e-negotiation research, but highly relevant in the way it deals with coordination in MAS is the ARCHON project [26]. Software agents are divided into two layers: an ARCHON Layer and an application program. The former encapsulates generic knowledge about cooperation, which is domain independent and encoded in terms of production rules. The latter is a special problem solving application, which performs the problem solving process. Cooperation know-how is encoded in an ARCHON layer by rules such as the following: " *If an agent has generated a piece of information, and it believes that it is of use to an acquaintance, then send it to the acquaintance.*" ARCHON separates cooperation knowledge from application knowledge, thus enabling generic cooperation features to be reused for other applications.

3.2. The CONSENSUS approach

Inspired by the ARCHON approach, we treat coordination information as declarative knowledge, and represent it as if-then rules which the agents exploit using an inference engine. Since our agents are already coupled with rule engines (for their strategy component), it is convenient to use the same rule engine to execute the coordination. We distinguish between strategy rules (described in Section 2), used to determine the action to take based on the information about a particular negotiation, and coordination rules, which are used to determine the action to take based on information about other negotiations (possibly combined with information about the negotiation itself). Coordination as well as strategy rules have conditions that are built on the state of the agent, the amount (spent, to spend, committed, remaining, etc.), the remaining time of the negotiation, the frequency of bids, etc.

Suppose we have two agents participating in separate negotiations with the goal of purchasing just one item. The following rule ensures that the agents make no more than one commitment at the same time: " *If Agent2 is leading or in the process of bidding, then Agent1 should wait.*" Figure 3 shows the same rule in the JRules syntax.

Note that this rule is too restrictive since an agent cannot make a bid if the other agent is leading (i.e., holds the highest bid). Evidently, this should not always be the case. If breaking commitments is allowed in the auctions (perhaps at a certain cost) the rule in question can be relaxed, and we may have the two agents leading at the same time while making sure that one of them drops out of the auction before it is too late.

Suppose now that we have two agents participating in separate negotiations with the goal of purchasing two complementary items. The following rule minimizes the risk of exposure (see Section 3.1): "*If Agent2 is trailing, and its chances of making a deal are slim, then Agent1 should wait for further instructions.*"

The intervention of the user in complex coordination situations should be permitted. When the agent cannot determine the action to take, it can query the user using a rule such as: "*If condition, then ask the user.*"

Finally, for practical reasons, we adopted a *blackboard* coordination approach where the agents post and read from a general blackboard. Coordination rules are therefore triggered by information that the agents make available in the blackboard. This approach is suitable only if the tasks are assigned, a priori, to the agents, if the number of agents is small and if the agents share a common domain understanding. Evidently, all requirements are satisfied in our case. Otherwise, the blackboard coordination scheme, with no direct agent-to-agent communication, would result in a severe bottleneck.

```
// If Agent2 is leading, or is in the process of bidding, then Agent1 waits
Rule coordinate1
{
  Priority = high;
  When {
            ?y: TheElement(iLead == false);
            Blackboard( ?b1: get("u2","iLead"); ?b2:
                        get("u2","iBid")) from getBlackboard(); evaluate(?b1 || ?b2);
            }
  Then { modify ?y
            { action = "DO NOTHING";}
             assert logical BidBloc() {ref = new Integer (0);}
  }
};
```

Figure 3: A coordination rule in the JRules syntax

4. Validation

In general, economists model an auction as a game with bidders playing against each other. The point of the game is to win the object at the lowest possible price; each bidder devises a strategy with this in mind [27]. An auction strategy is often a function of: (1) the bidder's private valuation, (2) an estimate of the opponents' valuation (if available), and (3) the past bids of the opponents (if known). As this information changes, the strategy is updated accordingly [28]. Along these lines, in order to validate our choice of representation for strategies and coordination, we used our agent-mediated negotiation system (i.e., CONSENSUS) to conduct agent tournaments. Agents function in repeated cycles we call *pulses*. A pulse is made of four steps as described in the following piece of code:

```
Repeat
        Sleep (p);
        GetInformation ();
        Think ();
        Act ();
Until (State = winning or State = loosing or State = dropping)
```

The pulse's steps are:

- Sleep (p): the agent goes to sleep for a period p that can be fixed or determined dynamically.
- GetInformation (): the agent queries the server and updates its private information.
- Think (): the agent applies the rules to determine the action to take.
- Act (): the agent takes the appropriate action.

The agent's states are:

- Trailing: the agent does not hold the highest bid.
- Bidding: the agent is in the process of making a bid.
- Leading: the agent holds the highest bid.
- Winning: the auction ended while the agent was leading.
- Loosing: the auction ended while the agent was trailing.
- Dropping: the agent quits voluntarily.

The actions the agent might take are:

- Do nothing: take no action for the time being.
- Bid (v): bid the amount v.
- Drop: quit the auction permanently.

4.1. English Auction Tests

In an English auction, the participant has to decide whether or not to make a bid, how much to bid, and whether to continue or to abandon the auction (see the actions above). Observing the opponents (as in real life auction houses) is essential in taking such decisions, and by doing so, the participant gains two types of information: (1) how many bidders have dropped out of the auction (since they have lower valuations than the current bid); and (2) at what prices they dropped out [27]. Good strategies are also the result of correct predictions about the behaviour of the opponents and sometimes that means guessing someone else's private information. Finally, it might be helpful for a bidder to know if the opponents are *symmetric* (i.e., they use the same measurements to estimate their valuations), and if they have secret information about the item [28]. Using the rule-based approach, we provided our agents with common bidding strategies and made them participate in bidding tournaments. Here are some of the results.

4.1.1. Optimal Bidding. Optimal bidding (also called proxy bidding) means bidding a small amount more than the high bid until you reach your valuation and then stop. This strategy, described by the rules in Figure 2, has the effect shown in Figure 4. The Agent and its opponents are provided with optimal bidding rules. The time is in seconds but in reality, an auction like this might take a week to ten days. The winner (Agent in this case) is the one with the highest valuation (i.e., reserve price).

4.1.2. Update Rate. Since online auctions take place over a long period of time, it is difficult for the participants to monitor the bidding process. The reasons are: (1) there is a cost every time you enter a bid because of Internet connectivity cost; and (2) there is an opportunity cost associated with logging on, filling the form, and entering the bid. An agent should not maintain a connection to the server at all times. It should optimize connections by connecting only when it is relevant to do so, and should be able to determine rapidly whether or not the auction suits its needs. Dropping out early means you can enter another auction early. Figure 5 shows our agent adjust its update rate to the average bidding rate. At the start of the auction, it makes few interventions, and as the activity increases towards the end of the auction, it participates more often (monitors the auction closely). Note that in this case, the agent makes bids every time it updates its information, but this is not an obligation.

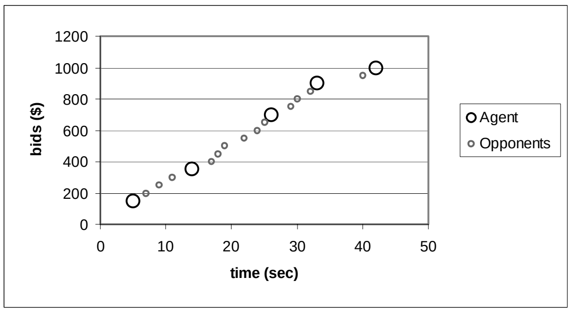


Figure 4: Optimal bidding

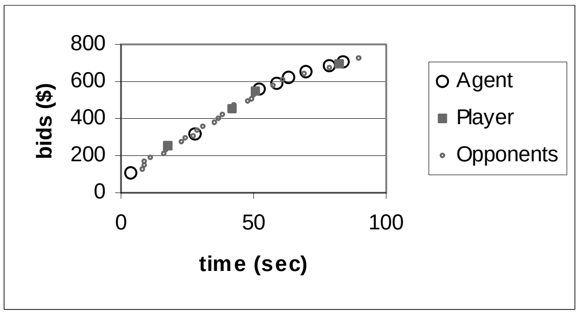


Figure 5: Adjust update rate

4.1.3. Adapt Increment. In auctions where there no is minimum increment enforced by the server, we talk of *spontaneous bidding*. In this case, it might be useful to observe the bidding process and adapt one's bidding increment to that of the opponents. Figure 6 shows our agent doing exactly that.

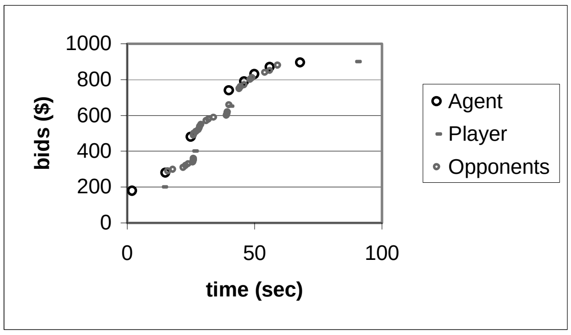


Figure 6: Adapt increment - spontaneous bidding

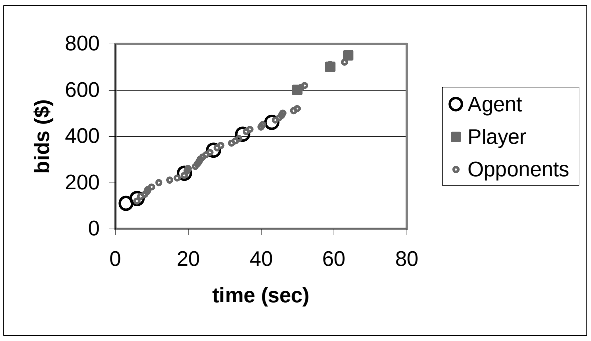


Figure 7: Jump bid – Detect and quit

4.1.4. Jump Bidding. This strategy consists on entering a bid larger than necessary to become a leader, which sends a signal to potential bidders. Jump bidding is credible because it is costly (i.e., money can be left on the table). The sender and the recipient of the signal may be better off in a jump bidding equilibrium: the sender saves bidding costs by deterring potential competition; and the recipient saves the costs of fruitlessly bidding against a strong opponent [29]. Figure 7 shows our agent detecting a jump bid made by Player and deciding to quit. Some opponents continue to bid but Player finally wins, as it was not bluffing. In Figure 8, our agent decides to respond with jump bids until Player (and everyone else) quits. In Figure 9, our agent detects a jump bid, and decides to wait until bidding goes back to normal before bidding again.

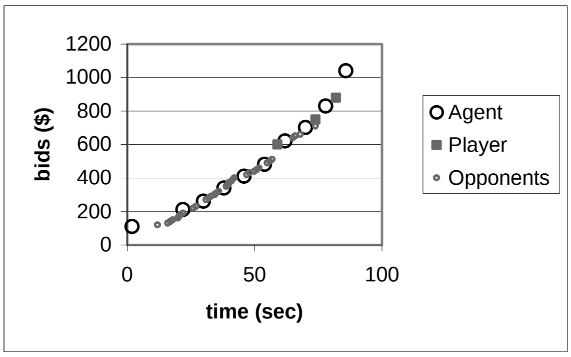


Figure 8: Jump bid – Detect and respond

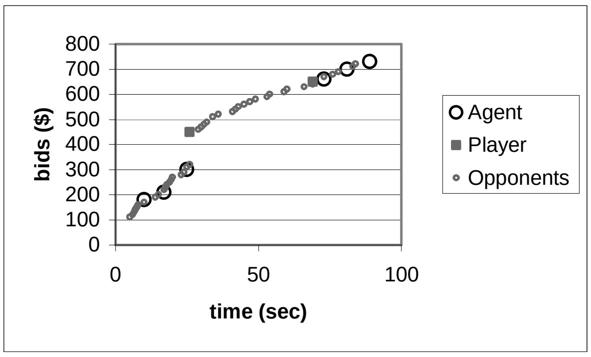


Figure 9: Jump bid – Detect and wait

4.1.5. Sniping. This strategy consists of waiting until the last minute before the auction ends, and trying to submit a bid, which just barely beats the high bid and gives the opponents no time to respond. Obviously, if the auction closes at a fixed date, then what incentive does a bidder have to place any bids early in the auction? Submitting an early bid is dominated by the strategy of submitting the same bid just before the auction ends. If all bidders were to follow the sniping strategy, the game would become equivalent to a first-price sealed-bid auction, with all the bids submitted at the end. In Figure 10 our agent snipes and wins. Figure 11 shows a sniping war between three agents.

To encourage early participation, a short *extension period* (five minutes for instance) might be added to the auction. If there is bidding activity in the last five minutes of the auction, then the closing time will be extended by an additional five minutes. The auction does not end until five minutes have passed without a new bid, which gives bidders the opportunity to protect themselves against snipers. We intend to implement this feature into our auction server and experiment with some strategies that take advantage of the extension period.

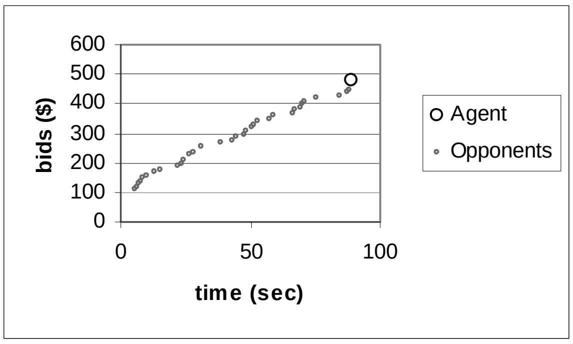


Figure 10: Sniping and winning

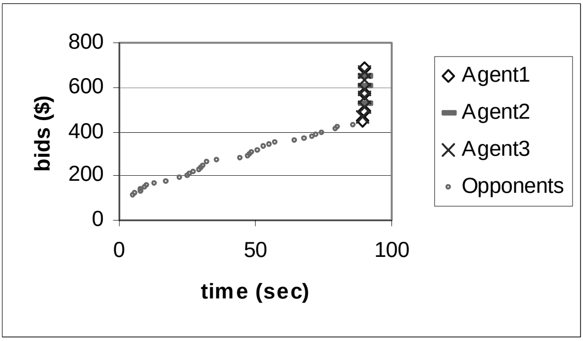


Figure 11: Sniping war

4.1.6. Shielding. Bid shielding involves making artificial high bids. The bidder puts in an early low bid (say $10) on an item, and then gets a friend (or a false identity) to put in an extremely high bid (say $500) on the same item. The high bid acts as a *shield* for the low bid, keeping anyone else from bidding. Just before the end of the auction, the bidder retracts the $500 bid, leaving the $10 bid as the winning bid on an item that should have gone for a higher price. Figure 12 shows Player 1 and Player 2 performing a shielding and Agent detecting the shield. Player 1 wins after Player 2 retracts its bid. Agent could not continue bidding because the shield exceeded its valuation. In Figure 13, Agent detects a shield and waits for it to be removed to snipe and win.

4.1.7. Increase Valuation. It can happen that a bidder reconsiders her valuation of an item after observing how the others bid on it. Figure 14 shows our agent increase its valuation when it sees that the possibilities of winning are good. It does so when its reserve price is reached, the auction is about to close, and there are few participants remaining. In the figure, the agent changes the reserve price from 600$ to 900$ and stays in the game to win.

4.1.8. Combination of Rules. It is possible to combine several strategies. Figure 15 shows the result of a "cautious" behaviour combined with an "aggressive" one. The agent avoids jump bidding by entering a waiting state, and at the end snipes and wins. This is a way to hide your real intentions so that your opponents cannot guess your strategy.

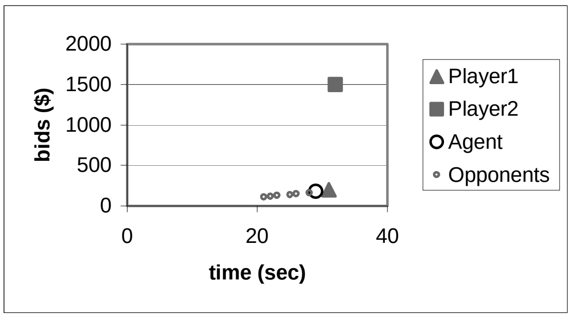


Figure 12: Shielding – detect and drop out

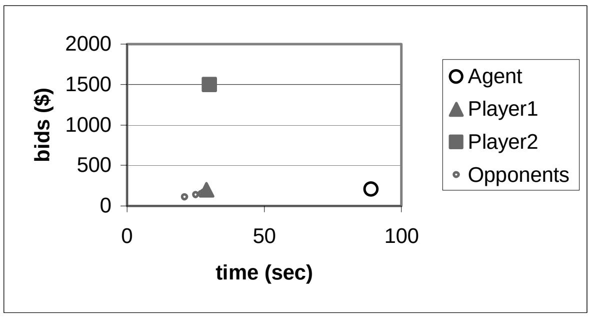


Figure 13: Shielding – detect and snipe

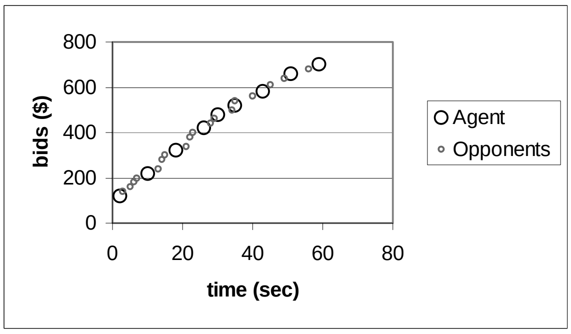


Figure 14: Increase valuation

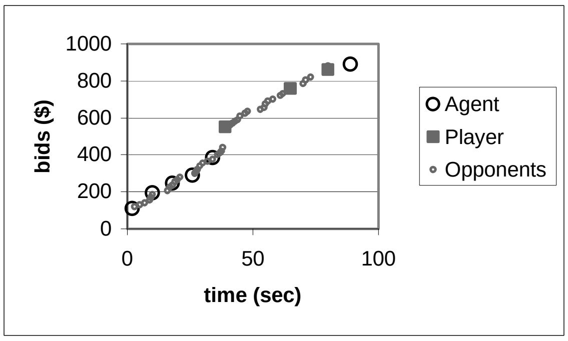


Figure 15: Detect jump bid & wait & Snipe

4.2. Dutch auction

In a Dutch auction, a bidder selects a cutoff price at which to claim the object so long as no one else has already claimed it. Usually, all bidders have the same strategy, which consists of *shading down* their bids slightly below their valuations. This means biding less than what you think the object is worth to avoid the winner's curse (see Section 2) [27]. No relevant information is disclosed in the course of the auction, only at the end when it is too late. This makes designing strategies a futile task. However, this is not the case with the multi-item Dutch auction. With many items for sale, bidders can watch the progress of the auction, look at the transaction prices, the number of remaining items (if available), the remaining time (if available), and decide whether to bid (bid in this case means to claim a number of items at the current asking price), wait for the price to drop, or quit.
We designed a Multi-item Dutch auction as follows:

- One ore more identical items are put on sale and the unit price is made public.
- As time passes, the seller decreases the unit price to generate interest.
- A buyer's bid is the quantity she wants to purchase at the current price.

The basic behaviour of our bidding agents in this case is not different from that of the English auction. Instead of bidding a price, they bid a quantity of items. At the time of its creation, an agent is given the following information: the number of items to buy, the minimum acceptable number of items, and how much the user values the item.
Simple bidding tactics were coded as rules and given to agents, and the agents were made to participate in Dutch bidding tournaments. Here are some of the results.

4.2.1. Safe Buying. As the price keeps decreasing, when it equals your valuation, buy the minimum number of items. Keep watching the auction, and before it closes, buy the remaining items (possibly at a smaller price that your valuation). The effect of this strategy is shown in Figure 16 (right). The agent bought the 4 minimum items, and later, 3 more items to reach its maximum number of 7.

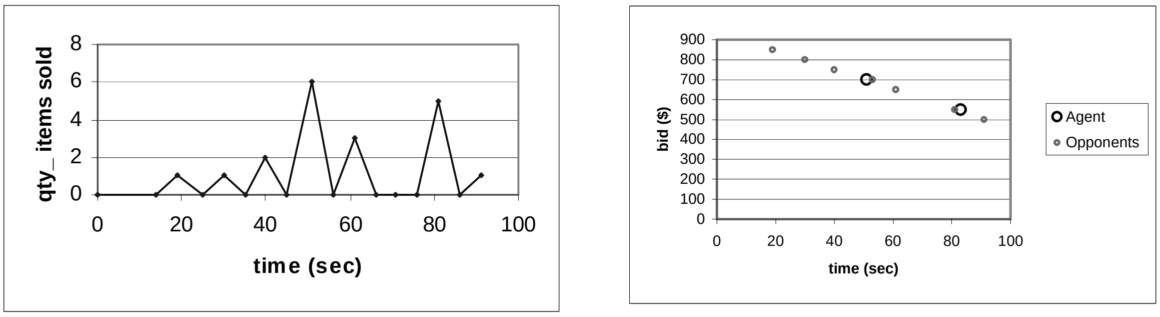


Figure 16: Multi-item Dutch Auction - Safe buying

4.2.2. Panic Buying. As prices go down, if you see that a big number of items have been sold, and if your valuation is not reached yet, then it might mean that your valuation is too low. Adjusting your valuation up will permit you to buy at least your minimum number of items even at a higher price. In Figure 17, the Agent buys the minimum 4 items needed. The risk of missing the minimum number of items is small but the risk of the winner's curse is high.

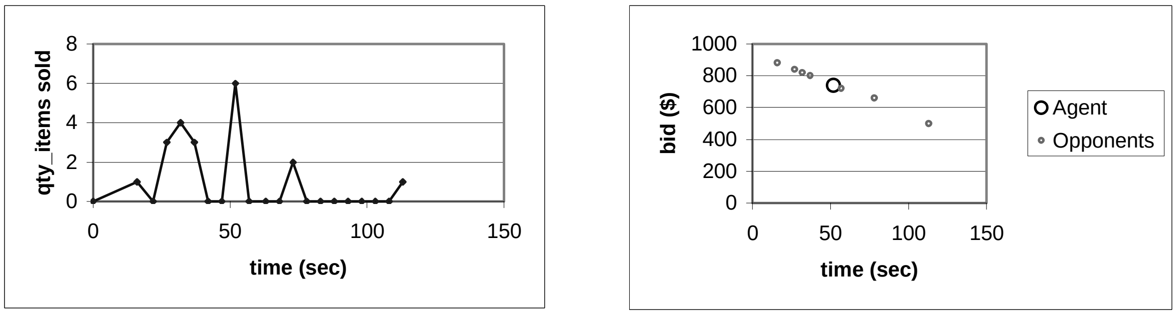


Figure 17: Multi-item Dutch Auction – Panic buying

4.2.3. Patient Buying. Figure 12 shows our agent decrease its valuation using the following rule: "if the price decrease rate is high, the selling rate is low, and you reach your valuation, then lower your valuation". This strategy might save the buyer from the winner's curse as it helps get the item at a lower price. There is however a risk that someone else decides to buy all the items just before our agent makes its move.

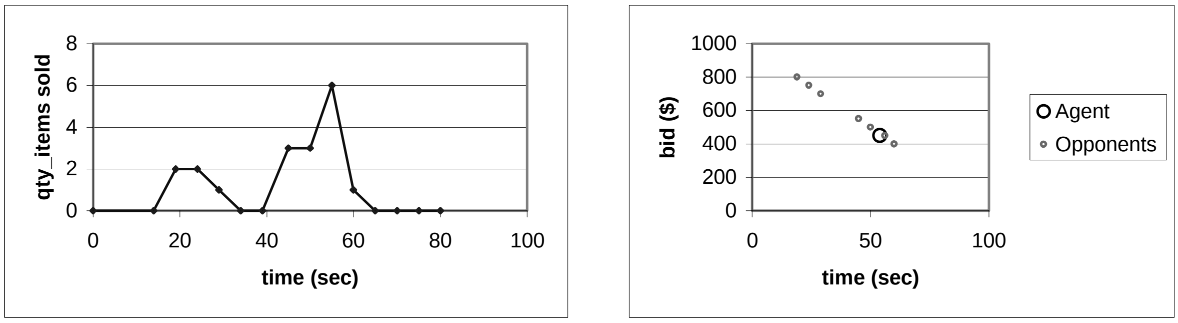


Figure 18: Multi-item Dutch Auction – Patient buying

4.3. Coordination

Using the same setting as before, we made many agents participate in separate auctions at the same time, and used the rule-based approach to manage their coordination.

4.3.1. Coordination across two English auctions. We want two complementary items A and B. Agent1 is launched in an English auction for item A, and Agent2 in another one for item B. The agents negotiate separately, but they coordinate their actions using coordination rules. Figure 19 shows one simple coordination scheme: "If Agent1 (left) detects a jump bid (i.e., the opponent is serious about winning and there is a risk that Agent1 may loose its auction) then Agent1 must quit. In this case, Agent2 (right) must also quit to avoid exposure."

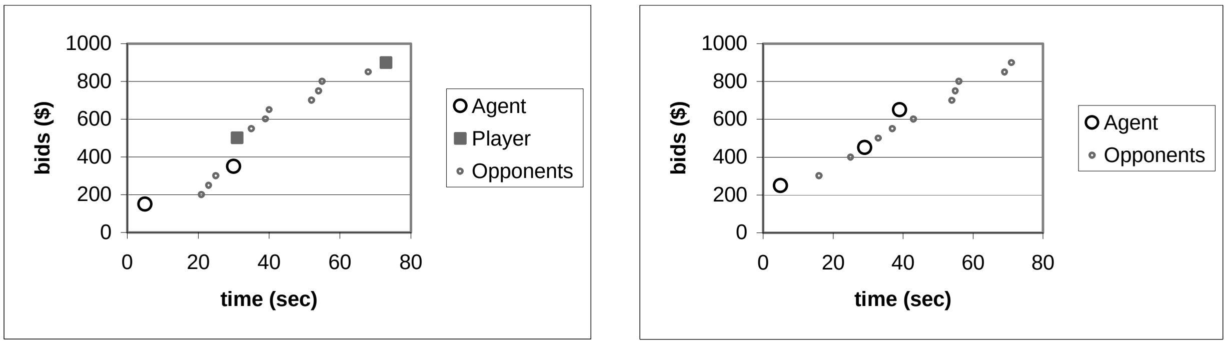


Figure 19: Coordination across two English auctions

4.3.2. Coordination across an English and a Dutch auction. We want either item A or item B. Agent1 is launched in an English auction for item A, and Agent2 in a Dutch auction for item B. Figure 20 shows the effect of the following coordination scheme: "if the going price in the Dutch auction is lower than or equal the current bid in the English auction, and your valuation is higher than the going price, then buy in the Dutch auction", and "if the going price in the Dutch auction is lower or equal to the current bid in the English Auction, then quit the English auction."

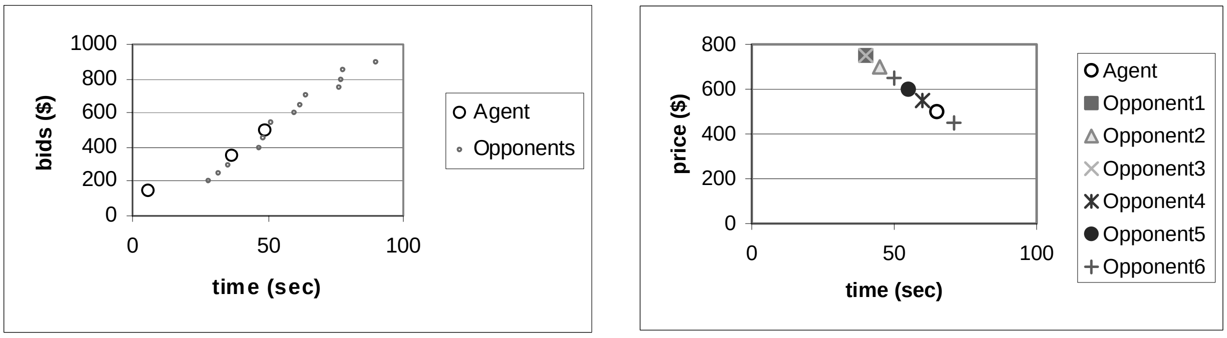


Figure 20: Coordination across an English and a Dutch auction

4.3.3. Coordination across three English auctions. For this test we have three agents Agent1, Agent2, and Agent3 participate in three English auctions for items B, C and A respectively. The auctions are independent. The agents and their opponents are provided with optimal bidding strategies. The goal is to win "(B or C) and A", that is: items B and A, or items C and A. Since the three agents are launched in parallel, there is a need to coordinate them.

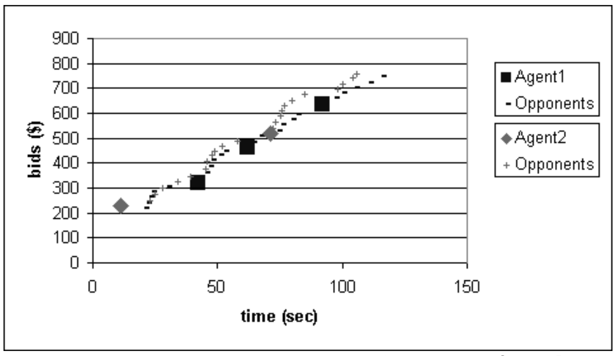
Figure 21 (left) shows the result of Agent1 and Agent2 bidding in parallel. Only one of them is leading its auction at the same time, and bids are always made in the cheapest auction.

Figure 21 (right) shows Agent 3 quitting (its reserve price is met), causing Agent1 and Agent2 (left) to quit.

Figure 22 (left) shows both Agent 1 and Agent2 loose their auctions (because they reached their valuations). At the same time, Agent3 (right) drops out of its auction.

Figure 23 shows Agent3 (right) snipe and win, and Agent1 and Agent 2 (left) continue until Agent2 wins.

Figure 24 shows Agent 3 (right) engage in a sniping war and loosing to another sniper. Agent1 and Agent2 (left) have no choice but to drop out of their auctions.



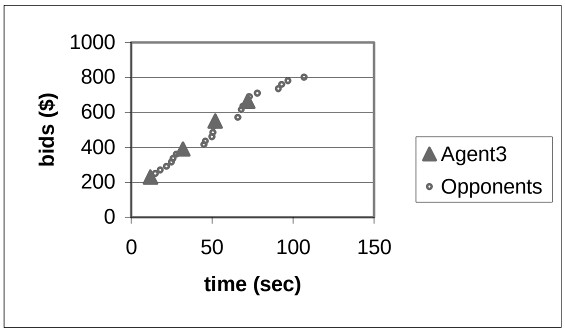
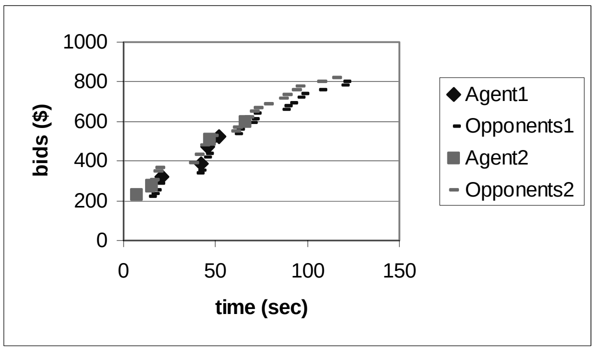


Figure 21: If you loose item A, stop everything



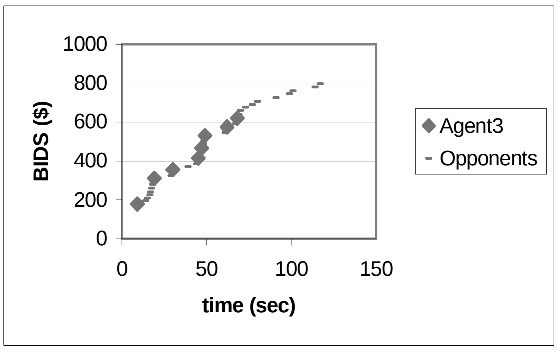
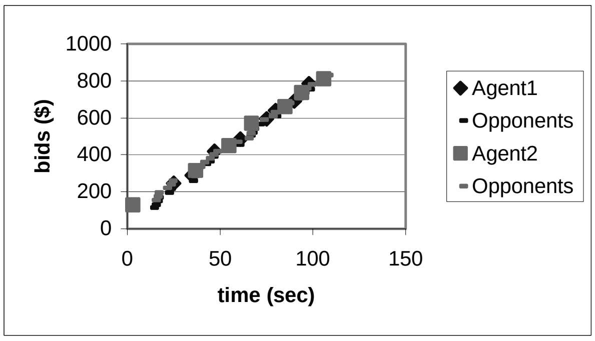


Figure 22: Agent1 and Agent2 loose, Agent3 quits



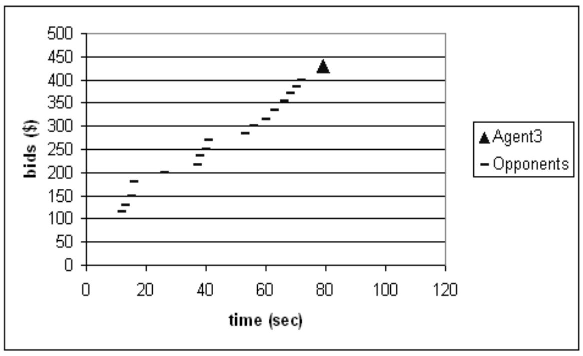
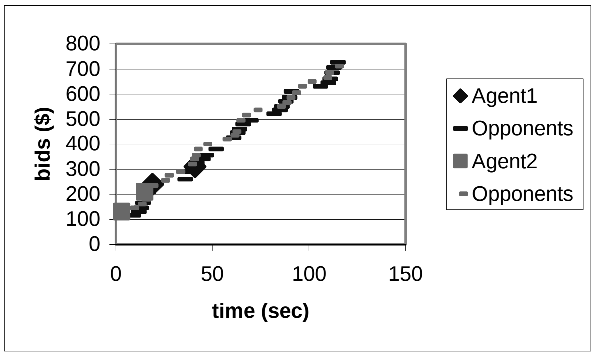


Figure 23: Agent3 snipes and wins, Agent1 and Agent2 keep going



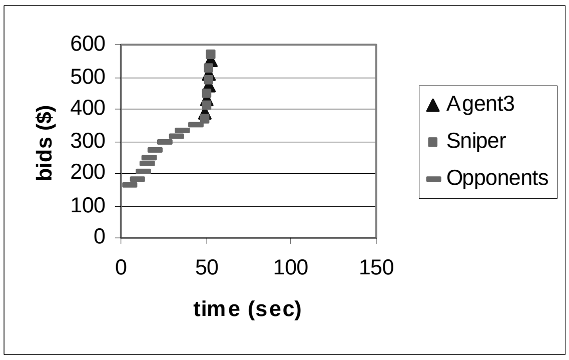


Figure 24: Agent3 snipes and looses, Agent1 and Agent2 drop out

5. Conclusion

The aim of the paper was to demonstrate that we could capture a wide range of bidding strategies and bid coordination schemes, using a rule-based approach, while supporting a wide spectrum of negotiation types. The well-known advantages of rule-based systems are the modularity and the uniformity (knowledge is represented using the same format). The possible inefficiency of rules and their eventual slow processing can be overcome by compiling the rules. Graphical tools can also be used to browse the rule base in order to verify its coherence. We use a blackboard coordination scheme for it is a simple and proven solution.

Basic behaviour of agents, various bidding tactics as well as coordination schemes across multiple auctions were implemented using our representation. They were tested in agent tournaments within simulated markets. So far, the results are encouraging and other possibilities are yet to be explored. We are in the process of evaluating this representation by testing its flexibility and seeking the limits of its expressiveness. As future points of interest we see the following:

(1) It is no secret that rules are a bit restricted in their ability to be adapted automatically, as agents are usually expected to adapt over time to improve their performance. For now, our agents lack learning capabilities, but we will investigate how our approach can lend itself to machine learning techniques.

(2) Bid strategies usually involve fairly complex optimization algorithms and forecasting that could not be expressed directly as rules. We propose to make use of optimization algorithms and forecasting as procedural attachments in the action part of the rules.

(3) We have been considering negotiations where the only negotiable attribute is the price. In the case of a plane ticket, for instance, the attributes could be the price, the date of the flight, the airports (departure and destination), the class of the seat, etc. Coordination would be more interesting (and complex) if many attributes of the item were negotiable (i.e., multi-attribute negotiations).

References

[1] J. O. Kephart, J. E. Hanson, and A. R. Greenwald. Dynamic pricing by software agents. Computer Networks, 2000. To appear.

[2] K. Jonkheer. Intelligent agents, markets and competition: consumers' interests and functionality of destination sites. Published Electronically on FirstMonday (peer-reviewed journal on the internet), 1999.

[3] A. Moukas, R. Guttman, and P. Maes. Agent-mediated electronic commerce: An MIT media laboratory perspective. In International Conference On Electronic Commerce, ICEC98, April 1998.

[4] D. C. Parkes, L. H. Ungar, and D. P. Forster. Agent Mediated Electronic Commerce, chapter Accounting for Cognitive Costs in On-line Auction Design, pages 25-40. Springer-Verlag, 1999.

[5] R. H. Guttman, A. G. Moukas, and P. Maes. Agent-mediated ecommerce: A survey. Knowledge Engineering 13(3), 1998.

[6] A. Chavez and P. Maes. Kasbah: An agent marketplace for buying and selling goods. In The First International Conference on the Practical Application of Intelligent Agents and Multi-agent Technology, April 1996.

[7] M. Benyoucef, H. Alj, M. Vézeau, and R. K. Keller. Combined Negotiations in E-Commerce: Concepts and Architecture. Electronic Commerce Research Journal, 1(3):277-299, July 2001. Special issue on Theory and Application of Electronic Market Design. Baltzer Science Publishers.

[8] T. Ito, N. Fukuta, T. Shintani, and K. Sycara. Biddingbot: A multi-agent support system for cooperative bidding in multiple auctions. In 4th Intl Conference on Multi-agent Systems (ICMAS-2000), Boston. MA, July 2000.

[9] C. Priest. Algorithm design for agents which participate in multiple simultaneous auctions. Technical Report HLP-2000-88, Hewlett-Packard, Bristol, England, July 2000.

[10] C. Beam, A. Segev, and G. Shanthikumar. Electronic negotiation through internet-based auctions. Technical Report 96-WP1019, Haas School of Business, UC Berkeley, December 1996.

[11] C. Beam and A. Segev. Automated negotiations: A survey of the state of the art. Technical Report 97-WP-1022, Haas School of Business, UC Berkeley, 1997.

[12] A. R. Greenwald and J. O. Kephart. Shopbots and pricebots. In16th International Joint Conference on Artificial Intelligence, volume 1, pages 506-511, Stockholm, Sweden, August 1999.

[13] E. Gimenez-Fuentes, L. Godo, and J. A. Rodriguez-Aguillar. Designing bidding strategies for trading agents in electronic commerce. In Third Intl Conference on Multi-Agents Systems (ICMAS-98), Paris, France, July 1998.

[14] W. Y. Wong, D. M. Zhang, and M. Kara-Ali. Negotiating with experience. In KBEM-2001, Austin, TX, 2000.

[15] H. S. Nwana, L. Lee, and N. R. Jennings. Co-ordination in software agent systems. BT Technology Journal, 14(4):79-89, October 1996.
[16] H. R. Varian. Economic mechanism design for computerized agents. In USENIX Workshop on Electronic Commerce, New York, NY, July 1995.
[17] A.R. Lomuscio, M. Wooldridge and N.R. Jennings. A classification Scheme for negotiation in electronic commerce. In Agent-mediated e-commerce: a European AgentLink Perspective, pages 19-33, Springer-Verlag, 2001.
[18] S. Y. Su, C. Huang, and J. Hammer. A replicable web based negotiation server for e-commerce. In 33rd International Conference on System Sciences, Hawaii, 2000.
[19] B. N. Grosof. Courteous logic programs: Prioritized conflict handling for rules. Technical Report RC 20836, IBM Research Division, T.J. Watson Research Center, May 1997.
[20] D. Zeng and K. Sycara. Bayesian learning in negotiation. International Journal of Human-Computer Studies, (48):125-141, 1998.
[21] C. McClintock and C. A. Berlioz. Implementing business rules in java. Java Developers Journal, May 2000.
[22] D. M. Reeves, B. N. Grosof, Michael P. Wellman, and Hoi Y. Chan. Toward a declarative language for negotiating executable contracts. In Workshop on AI in Electronic Commerce, Menlo Park, CA, 1999.
[23] B. N. Grosof, Y. Labrou, and H. Y. Chan. A declarative approach to business rules in contracts: Courteous logic programs in XML. In 1st Conference on Electronic Commerce, Denver, Colorado, November 1999.
[24] Ilog Jrules. http://www.ilog.com/products/jrules.
[25] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. International Journal of Group Decision and Negotiation, 10(2), 2001. To appear.
[26] M Berndtsson, S. Chakravarthy, and B. Lings. Coordination among agents using reactive rules. Technical Report HS-IDA-TR-96-011, Department of Computer Science, University of Skovde, Sweden, October 1996.
[27] L. J. Mester. Going, Going, Gone: Setting Prices with Auctions. Federal Reserve Bank of Philadelphia Business Review (March/April):3-13 1988
[28] http://www. agorics.com
[29] R. F. Easly and R. Tenorio. Bidding Strategies in Internet Yankee Auctions. Working paper, University of Notre Dame, 1999.

Liste des publications au CIRANO*

* Consultez la liste complète des publications du CIRANO et les publications elles-mêmes sur notre site Internet :

http://www.cirano.qc.ca/publication/documents.html

2001s-50	The Underpricing of Initial Public Offerings: Futher Canadian Evidence / Maher Kooli et Jean-Marc Suret
2001s-49	How Innovative Are Canadian Firms Compared to Some European Firms? A Comparative Look at Innovation Surveys / Pierre Mohnen et Pierre Therrien
2001s-48	A Tale of Two Ports / Ngo Van Long et Kar-yiu Wong
2001s-47	Wage Policy of Firms: An Empirical Investigation / Stéphanie Lluis
2001s-46	Forecasting Some Low-Predictability Time Series Using Diffusion Indices / Marc Brisson, Bryan Campbell et John W. Galbraith
2001s-45	The Importance of the Loss Function in Option Pricing / Peter Christoffersen et Kris Jacobs
2001s-44	Let's Get "Real" about Using Economic Data / Peter Christoffersen, Eric Ghysels et Norman R. Swanson
2001s-43	Fragmentation, Outsourcing and the Service Sector / Ngo Van Long, Ray Riezman et Antoine Soubeyran
2001s-42	Nonlinear Features of Realized FX Volatility / John M. Maheu et Thomas H. McCurdy
2001s-41	Job Satisfaction and Quits: Theory and Evidence from the German Socioeconomic Panel / Louis Lévy-Garboua, Claude Montmarquette et Véronique Simonnet
2001s-40	Logique et tests d'hypothèse : réflexions sur les problèmes mal posés en économétrie / Jean-Marie Dufour
2001s-39	Managing IT Outsourcing Risk: Lessons Learned / Benoit A. Aubert, Suzanne Rivard et Michel Patry
2001s-38	Organizational Design of R&D Activities / Stefan Ambec et Michel Poitevin