A Multi-Agent Negotiation Testbed for Contracting Tasks with Temporal and Precedence Constraints*

John Collins, Wolfgang Ketter, Maria Gini
Department of Computer Science and Engineering,
University of Minnesota
{jcollins,ketter,gini}@cs.umn.edu

Bamshad Mobasher
School of Computer Science, Telecommunications, and Information Systems,
De Paul University
mobasher@cti.depaul.edu

Abstract

We are interested in supporting multi-agent contracting, in which customer agents solicit the resources and capabilities of other, self-interested agents in order to accomplish their goals. Goals may involve the execution of multi-step tasks, in which different tasks are contracted out to different suppliers. We have developed a testbed that allows us to study decision behaviors of agents in this context. It can generate sets of tasks with known statistical attributes, formulate and submit requests for quotations, generate bids with well-defined statistics, and evaluate those bids according to a number of criteria. Each of these processes is supported by an abstract interface and a series of pluggable modules with a large number of configuration parameters, and with data collection and analysis tools.

1 Introduction

The business-to-business e-commerce market is expected to expand rapidly in coming years, with the global market expected to exceed $7.29 trillion in 2004, according to Gartner Group research. Online marketplaces are gaining popularity among companies seeking to streamline their operations. Online marketplaces offer benefits to both buyers and sellers. For buyers, a marketplace can significantly ease the process of searching for and comparing providers, while for sellers marketplaces provide access to much broader customer bases [26]. Business-to-business hubs, which link buyers within a particular industry or across a shared need, are expected to handle as much as $1.25 trillion by 2003 [18].

Over the past decade, the complexity of logistics involved in manufacturing and other business activities has been increasing nearly exponentially. Many processes are being outsourced to outside contractors, making supply chains longer and more convoluted. The increased complexity is often compounded by accelerated production schedules which demand tight integration of all processes.

*Partial support for this research was provided by NSF under award NSF/IIS-0084202

Thus, the field is ripe for the introduction of systems that automate logistics planning among multiple entities such as manufacturers, part suppliers, shippers, and specialized subcontractors.

Deciding what to outsource and to whom, ensuring that the tasks are done in the proper sequence (parts cannot be painted before they are finished) and that the final product is ready within the time constraints, is currently the job of a human decision-maker. A schedule with slack between tasks is less risky than a tight schedule, but in made-to-order products speed is the essence and taking extra time might prevent a supplier from getting a contract. The decision-maker also has to keep track of any delays from suppliers that could jeopardize the completion of the tasks, and renegotiate with them or others as needed.

The proliferation of business-to-business portals such as CommerceOne (`www.commerceone.com`) and VerticalNet (`www.verticalnet.com`) clearly shows the need and industry demand for value-added services such as security, match-making, and trusted intermediaries. However, a market framework for B2B interactions which can successfully address the full spectrum of the requirements mentioned above needs to provide the ability to automate contracting activities among participants, as well as provide support for automated agents that act on behalf of these participants.

Computerized agents can examine offers much more quickly than humans can. Choices that are too complex for humans, because of the large number of alternatives, or the complexity of the computations, can be programmed and executed by software agents. We believe that agent-based contracting will result in cheaper and faster contract negotiations, freedom from human errors, prompter deliveries, and thus, ultimately, in reduced costs for both suppliers and customers.

If autonomous or semi-autonomous agents are to be used, it is imperative that the creation, modification, and deployment of these agents is cost effective. Furthermore, since potentially billions of dollars are at stake, it is imperative that the agents exhibit the desired behavior when they are deployed. In short, a cost effective agent development methodology producing agents with predictably correct behavior is a necessity.

In this paper we present a market architecture that supports multi-agent contracting, and we describe the implementation of a prototype of the market architecture and of the agents. We call this system MAGNET (Multi AGent NEgotiation Testbed). MAGNET provides support for a variety of types of transactions, from simple buying and selling of goods and services to complex multi-agent negotiation of contracts with temporal and precedence constraints.

Our major goal in this paper is to describe the main features of MAGNET and to show how MAGNET can be used to develop a sufficiently realistic simulation of an actual market. MAGNET is not yet a complete simulation of a market. Currently, it is focused on the process of determining the form and content of Requests for Quotations (RFQs), on the management of the bidding process, and on the evaluation of bids submitted by suppliers.

This paper is organized as follows: Section 2 outlines the advantages of a market infrastructure. Section 3 describes the MAGNET infrastructure and the basic activities and roles of the agents. Customer agents are described more in detail in Section 4. Section 5 describes the testbed and gives some examples of the types of studies supported by this framework. Section 6 describes the architectural design and implementation of MAGNET. Section 7 provides a detailed example of a customer agent and its activities. Section 8 describes related work, and Section 9 concludes and outlines our future plans and open problems.

2 Market Infrastructure

Electronic commerce on the Internet is one of the fastest growing segments of the information and communication technology and has potential for enormous economic benefits for companies worldwide [40]. With the advent of the open standard and the growth of fast and inexpensive standardized communication infrastructure, more organizations and individuals are relying on virtual environments such as the Internet for a variety of commercial activities, including the exchange, marketing and promotion of goods or services, and contracting. This, in turn, has led to an increasing demand for innovative technologies and mechanisms that support automated transactions.

The current trend is to move towards hub and spoke architectures, where suppliers, customers, and trading partners need only one connection to communicate with each other [26]. This contrasts with the more traditional point-to-point connections supported by proprietary value-added networks, which are more expensive, harder to scale, not transparent, and whose high entry costs tend to keep out small companies.

We outline briefly several major advantages in providing a market infrastructure to support human decision makers or automated agents.

Support for multi-agent negotiation over extended time periods. Negotiations may require extended periods of time to complete, during which a context must be maintained. The time during which the negotiated transaction extends can also span significant periods of time, in the range of weeks to months. A market infrastructure simplifies the task of maintaining the state of transactions over time. Most negotiation protocols involve time limits, such as a deadline for receipt of bids. All parties to a time-sensitive negotiation process must have a common time reference. The market can provide this, as well as methods to validate non-performance, and to assess negotiated penalties.

Value-added services. Value-added services, such as matchmaking [33] or publish-subscribe facilities for notification of important events, are is particularly important for two reason. First, participants can continuously issue or retract registered capabilities in various domains. This makes it difficult for individual participating agents to keep track of and have access to the most up-to-date information. Secondly, such a facility provides a form of filtering and reduces the computational costs during bid evaluation.

Furthermore, the market may serve as a repository of statistical data about various participants. This may include general statistics about availability of suppliers with specific capabilities, or independent ratings based on past performance. The general statistical information is useful for customer agents in formulating a Request For Quotes, while performance ratings can be used to determine the price associated with various bids.

Protection against fraud and misrepresentation. We must assume that participating self-interested agents will exploit any opportunities that exist to gain advantage. The market facilitates recognizing and protecting against situations that allow agents to gain unfair advantage at the expense of other agents. Strategies that can result in this type of "unfair" gain include:

- Hiding one's identity or taking on the identity of another. This includes changing identity in order to escape the consequences of poor service on prior commitments.

- Dishonest auctioneer - In Vickrey-type auctions [37, 36], the motivation for truth-telling on the part of participants is predicated on their belief in the honesty of the auctioneer.
- Mis-communication of the rules under which a transaction is being conducted.
- Failure to follow through on commitments.

Discouragement of counterspeculation. Opportunities for counterspeculation arise when the rules of negotiation allow agents to gain advantage by making use of factors other than their own capabilities and valuations [20]. We are concerned with two general types of counterspeculation. Value-based counterspeculation [28, 37, 36] occurs when agents use their own estimates of each other's valuations to set bid prices. In [8], we identified two classes of time-based counterspeculation opportunities. One of these situations occurs when supplier agents are allowed to expire their bids before the Request for Quotes expires. This forces customers to make decisions without full information on other, possibly more advantageous bids. The other situation occurs when suppliers believe that the customer will start to evaluate bids before all bids are received. If the supplier believes that the customer's resource limitations will prevent full consideration of all bids, then early submission of bids, at potentially higher prices, can be used to skew the customer's reasoning process.

The market provides a neutral third-party facility that can be used to control and filter protocol exchanges to reduce both value-based and temporal counterspeculation. It is up to the agents to decide the extent to which these facilities are used, since they may slow the negotiation process or reduce the information exchange.

3 The MAGNET Architecture

We have designed a market infrastructure, which we call MAGNET [10] and implemented it as a distributed system that can be used to support electronic commerce in a variety of domains. The fundamental elements of MAGNET are the *market*, the *market sessions*, and the *agents*.

3.1 Market

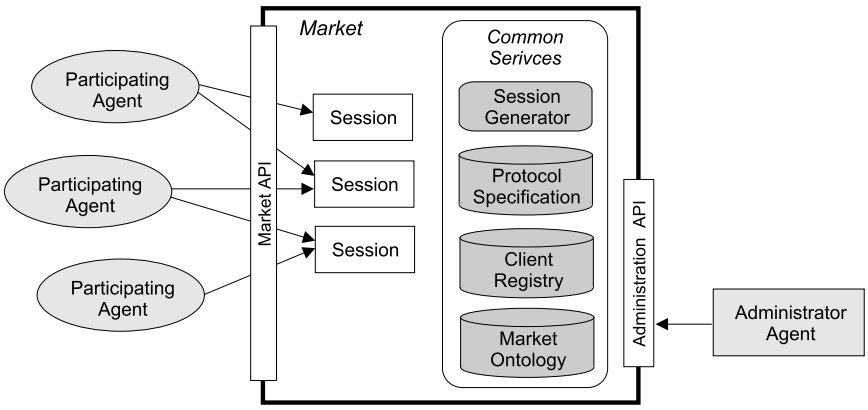

Figure 1: The Structure of a Market

Each *Market* within MAGNET is a forum for commerce in a particular business area, and includes a set of domain-specific services, as shown in Figure 1, and each market draws upon common services. Important elements of the market include:

- A *Registry* of market participants who have expressed interest in doing business in the market. Entries in this registry include the identity of a participant, a catalog (or a method for accessing a catalog) of that participant's interests, products or capabilities, which can be used for matchmaking [33].
- An *Ontology* specifying the terms of discourse within the domain of the market. Agents who wish to offer resources and services do so through one or more market segments whose ontologies describe their offerings. The market we will describe later in our example covers production, bottling, and shipment of wine. The ontology includes not only the task definitions, but statistics collected by the market about each task type. These statistics include expected duration and variability, expected price and variability, and resource availability data.
- A *Protocol Specification* that formalizes the types of negotiation supported within the market. These are limits on parameters of the negotiation protocol, such as whether bids can be awarded before the bid deadline, etc.

3.2 Market Sessions

An important component of each market is a set of current *Market Sessions* in which the actual agent interactions occur. Each session is initiated by a single agent for a particular purpose, and in general multiple agents may join an existing session as clients. The session enforces the protocol rules, and maintains its internal state according to the protocol activity and the passage of time. The session extends from the initial RFQ through the negotiation, awards, construction work, paying of bills, and final closing. The session mechanism ensures continuity of partially-completed transactions, and relieves the participating agents from having to keep track of detailed negotiation status themselves.

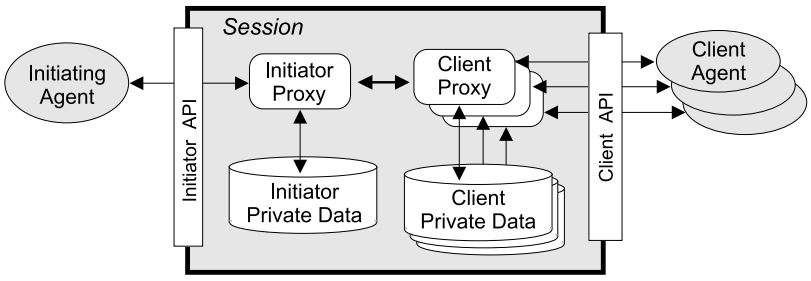

Figure 2: The Structure of a Market Session

Figure 2 shows the structure of a session. Two APIs are exposed, one for the session initiator and one for session clients. Each session contains an Initiator Proxy that implements the Initiator API and persistently stores the current state of the session from the standpoint of the initiator. A Client Proxy is provided for each client that similarly provides a Client API to the client agent,

and persistently stores the current state of the session from the standpoint of the client. Proxies are market entities that act on behalf of the agents and enforce market rules.

There are two major reasons for the existence of the proxy components. The first is related to security: client proxy components cannot see the private data of the initiator or of other clients. The second is that in a distributed system environment, the processing and persistent data elements of the initiator and clients could occur at different locations in the network to maximize performance.

3.3 MAGNET agents

Each agent in MAGNET is an independent entity, which may be acting on behalf of different individuals or commercial entities who have different goals and different resources.

We distinguish between two agent *roles*, the *Customer* and the *Supplier*. A Customer is an agent who has a goal to satisfy, and needs resources outside its direct control in order to achieve its goal. The goal may have a *value* that varies over time. A Supplier is an agent who has resources and who, in response to a request for quotes, may offer to provide resources or services, for specified prices, over specified time periods.

An architectural view of these agents and their primary interactions is shown in Figure 3.

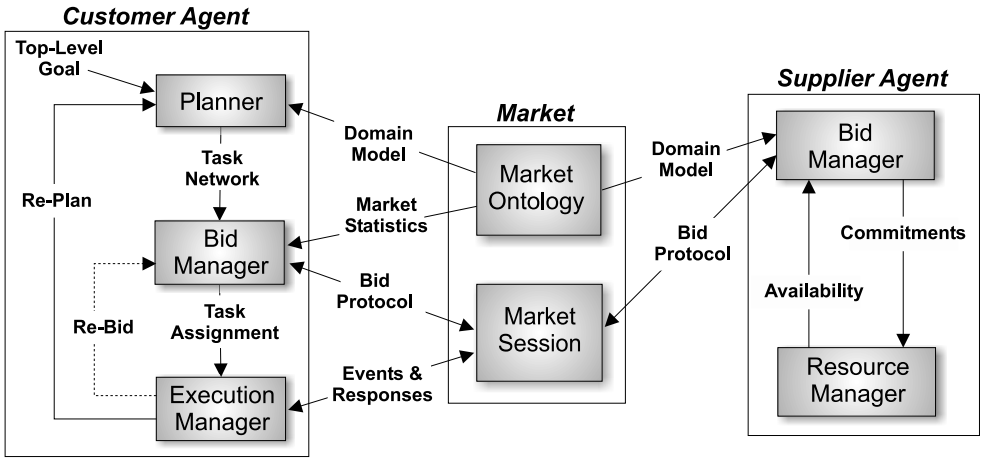

Figure 3: The MAGNET architecture

The negotiation process consists of a *contracting* phase and a *execution* phase. The contracting phase is a first-price, sealed-bid combinatorial auction consisting of a Request For Quotes, Bidding, and Bid Awards. This three-step process is designed to simplify negotiations without loss of generality. It is modeled after the leveled commitment protocol proposed by Sandohlm [31]. The resulting task assignment forms the basis of an initial schedule for the execution of the tasks. The execution phase may involve additional negotiations over schedule adjustments, decommitments, and in some cases repeating the bidding cycle when it becomes necessary to re-allocate resources that had originally been committed.

During the bidding cycle, customer and suppliers communicate by exchanging messages, which are routed by the market.

- The customer issues an RFQ which includes a specification of each task, and a set of precedence relations among tasks. For each task, a time window is specified giving the earliest

time the task can start and the latest time the task can end.

- Suppliers submit bids and commit to resources (with the possibility of overcommitment) until the bids get awarded or rejected by the customer. A bid includes a price for the task, a portion of the price required to be paid as a non-refundable deposit at the time the bid is awarded, an estimated duration for the task, and a time window within which the task can be started.
- The customer decides which bids to accept. Each task needs to be covered (i.e. no free disposal [24]) and the constraints of all awarded bids must be satisfied in the final work schedule.
- The customer awards the chosen bid combination, pays the deposit, and specifies the work schedule for the suppliers.
- Each supplier starts executing the tasks awarded and tries to complete them in the specified time frame. When the supplier completes a task, the customer pays the remainder of the price. If the supplier fails to complete a task, the price is forfeit and the deposit must be returned to the customer. A penalty may also be levied for non-performance.

Once bids have been awarded, a secondary protocol allows agents to negotiate schedule changes. This avoids outright failure and reduces risk for both parties, at the cost of complicating the behavioral requirements of agents during plan execution.

4 Customer Agents

A Customer agent, as illustrated in Figure 3, has three major components: the Planner, the Bid Manager, and the Execution Manager.

4.1 Planner

The Planner's job is to generate a task network from the top level goal. A task network consists of a set of task descriptions, the temporal constraints among them, and possibly nonzero delays between tasks, to cover communication and transportation delays. An example is shown later in Figure 7.

In our current implementation, a task network is created either by selecting a pre-defined plan from a library of plans, or by selecting randomly from a library of task types, and creating random precedence relations among them. We expect that in many domains, plans will be chosen from a library or defined by a human user rather than being generated by a general-purpose planner.

The task network generated by the Planner is a central data structure, used by the Bid Manager to generate RFQs and to evaluate timing data of bids, and by the Execution Manager to monitor and repair the ongoing execution of the plan.

4.2 Bid Manager

The high-level structure of the Bid Manager is shown in Figure 4. Each of its components is in charge of one of the major decisions the customer agent has to make during the bidding cycle. Since the customer agent is ultimately required to make decisions acceptable to a human decision-maker, it must have the ability to handle decision-making under uncertainty. We have studied how to incorporate Expected Utility Theory [5] in these decision processes.

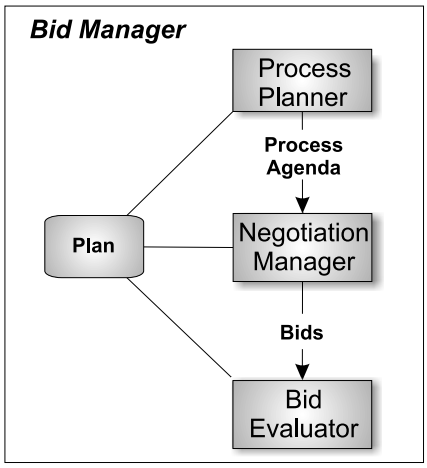

Figure 4: The Bid Manager

1. The first decision is to determine how much time suppliers are given to submit bids, and to determine an approximate schedule by setting limits on the start and end times for each individual task. This decision is made by the Process Planner. The more time the customer keeps for its own decisions, the more flexibility it has in deciding whether to accept bids, or post a new RFQ if not enough bids are submitted. However, this comes at the expense of the time devoted to executing the tasks, and reduces flexibility for the suppliers, so it is likely to increase costs.

   In the current version the Process Planner simply reads an agenda from a configuration file or a user interface. In the future it will be responsible for deciding which markets to use, when to consult local catalog and timetable databases, and how to break up the plan accordingly. If the plan has alternative branches, it may also decide which alternatives to pursue and in what order.

2. Next, a decision has to be made on whether to divide the tasks among multiple RFQs or submit a single RFQ and determine its contents. This is the job of the Negotiation Manager.

   It first decides a schedule for the bidding process, possibly subdividing the time allocated by the Process Planner to multiple RFQs. High scheduling uncertainty leads to either having to leave large amounts of slack in the schedule, which delays the completion of the tasks, or to generate RFQs with large overlaps in task timing, which causes many bids to be unusable. When this is the case, an alternative is to split the bidding process into phases, so that the schedule variability in each phase is limited. We have designed several methods to do this, and we are experimenting to understand how they perform, what the tradeoffs are, and how to recognize situations where multi-phase bidding is advantageous. Preliminary results suggest that multiphase bidding can generate tighter schedules on average, at the same price, and that search effort on the customer side is reduced substantially. On the other hand, the time required for the overall bidding process may easily become dominated by the time required for supplier deliberation, and opportunities for suppliers to submit "package" bids are reduced.

   The next step is to compose one or more RFQ. The RFQ is a structure that contains tasks and precedence relations along with a set of scheduling constraints. Scheduling constraints are

determined using (1) the precedence constraints from the task network for the tasks included inthe RFQ, (2) statistical information about duration and variability for the different task types, as well as information about resource availability, and number of vendors who are likely to bid, which are available from the market; (3) the overall schedule for the execution of the plan generated by the Process Planner. Figures 8 and 9 show examples of RFQs.

The primary purpose of the RFQ is to solicit the most advantageous set of bids possible. This requires finding a balance between using large windows that give flexibility to suppliers and ensuring that the bids will combine feasibly and the job will be completed by the deadline. We do this by setting early-start and late-finish times for each task. Preliminary results indicate that, given a reasonable number of bidders, some amount of overlap in the time windows between successive tasks gives better results than a RFQ specification that has no overlaps (and so guarantees that all bids will combine feasibly).

We have implemented several plug-in versions to build RFQs in order to test alternative approaches. We are currently studying how to apply Expected Utility Theory to the generation of RFQs [2].

3. At the conclusion of the bidding cycle, the agent must decide which bids to award, and exactly how to schedule the tasks in the bid awarded.

   The winner determination problem for combinatorial auctions has been shown to be $\mathcal{NP}$-complete and inapproximable [29]. This result clearly applies to the MAGNET winner determination problem, since we simply apply an additional set of (temporal) constraints to the basic combinatorial auction problem, and we cannot use the "free disposal assumption" which states that unsold goods can be disposed of freely. Because there can be no polynomial-time solution, nor even a polynomial-time bounded approximation, we must accept exponential complexity. We will see later in Section 5.2 that we can determine probability distributions for search time, based on problem size metrics, and we can use those empirically-determined distributions in our deliberation scheduling process. More details on this are in [7].

   The Bid Evaluator contains a search engine that takes a task network and a set of bids, and finds an optimal or near-optimal mapping of tasks to bids, respecting the temporal constraints. We have implemented the following bid-evaluation search engines: a highly-modular simulated annealing version [9], a Mixed Integer Programming version [6], and a A* and IDA* versions that extend Sandholm's bidtree-based IDA* algorithm [30] to deal with reverse-auction problems having precedence constraints among items.

   To study bid evaluation, we are able to control a wide range of conditions, including:

   - Composition of the generated plans: number of tasks, task types (which in turn controls duration variability and probability of bids), and the density of the precedence network;
   - structure of the RFQ: whether it covers the whole plan, amount of slack in the schedule, and the degree to which bids are allowed to violate precedence relations;
   - number and size of bids, composition of bids: random selections, contiguous task sets, role-based task sets;
   - type of search used, search parameters;
   - bid selectors and evaluators, evaluation parameters.

5 The MAGNET Testbed

Our current implementation of MAGNET includes a fully implemented and customizable customer agent, described in the previous Section, some relatively simple supplier agents, and a prototype of the market server. The testbed supports a number of measurements for evaluating search performance, including search effort, anytime performance, and solution quality, along with counts of solved, unsolved, and known unsolvable problems encountered. Output can be processed into a form that can be used by a standard spreadsheet, or Matlab in the case of anytime performance data.

5.1 Supplier Agents

Supplier Agents have been designed recently and are still under development. They are being implemented using Avalon [22] and are based on components. Using Avalon, it is straightforward to have the components of the supplier agent interact, to instantiate different instances of the components, and to reuse code. Avalon allows also to switch components on the fly, which is very useful in testing. It is also possible to configure Avalon using XML files, which specify which components and which instances have to be included.

Since our primary interest has been in the workings of the Customer agent, we have also implemented simple minded Supplier, that simply receive RFQs, and respond by submitting bids. They do not maintain resource schedules, and they have no persistent identity. The basic structure is shown in Figure 5. Each of these three layers is implemented as an abstraction with multiple implementations.

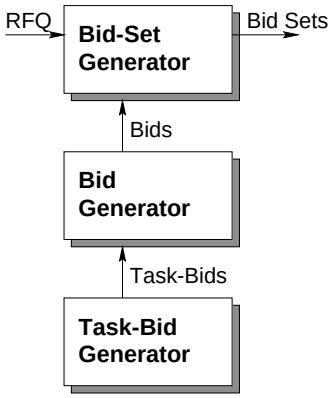

Figure 5: Simple Supplier Simulation

A Bid-Set Generator generates sets of bids and returns them to the Customer agent. Example Bid-Set Generators include one that always bids on certain task types if they are present in the RFQ, one that generates a random set of bids, and one that extends the random set generator by attempting to generate a set that covers all tasks in the RFQ.

A Bid Generator generates a single bid, possibly containing multiple individual task-bids. The average sizes, and the degree of size variability, of the bids produced are determined by configuration parameters, and in some cases by the structure of the plan and the type of Bid Generator selected. We have implemented Bid generators that can generate bids for certain types of tasks, random collections of tasks, or sets of tasks that are connected by precedence relations.

A Task-Bid Generator produces a bid for a single task. The bid specifies the task to be performed, the expected duration of the task, and early start and late finish time window data. In most cases it must also assign a cost to the task, which the Bid Generator will use in composing the overall cost for the bid. The duration and cost are selected from random distributions specified in the task-type description. The early-start and late-finish times are also randomly generated from the resource-availability data in the task-type description. The constraints on the time window for the Task-Bid come from two sources: (1) the time window specified in the RFQ, and (2) the times already specified in other Task-Bids for tasks that are immediate predecessors or successors of the current task. If the Task-Bid generator cannot fit the requested task into the time window, it fails to produce a result, and the bid will not include that particular task.

5.2 Experimental Studies

Our goal is to develop a sufficiently realistic simulation of an actual market to support evaluation of MAGNET agent performance.

As an example of the type of analysis we plan to undertake, we report preliminary results on characterizing the performance of a winner determination algorithm based on Integer Programming (IP). The study follows the methodology outlined in [16].

We are interested in three measures of performance: speed, scalability, and predictability. Speed and scalability are important because combinatorial auction winner determination is known to be $\mathcal{NP}$-complete and inapproximable [30]. Predictability is critical in the MAGNET domain because of the need to allocate time to the winner-determination process before issuing an RFQ. This is because suppliers need to know when bids will be awarded so they can calculate the cost of provisional resource reservations to cover their outstanding bids. This means that the RFQ must specify the timeline for the bidding process, and that timeline must include the time allocated to winner determination.

We address the predictability issue by characterizing our winner determination methods with respect to factors that can be measured or estimated at the time the RFQ is issued. These include (1) number of tasks in the task network, (2) number of bids likely to be submitted, and (3) expected sizes of the submitted bids, i.e. the number of tasks per bid. Although the latter two factors cannot be directly measured, they can be estimated based on historical market data.

Figure 6 shows the complete runtime distributions for four problem sets as the size of the task network is varied. For each curve, we show actual observations along with a lognormal distribution that minimizes the χ^2 metric[1]. Typical χ^2 values range from 0.3 to 3.0 for 9 degrees of freedom, a good match. Note that the first parameter of the lognormal distribution is equal to the median value, which is significantly smaller than the mean for all sets. The value of data such as this is that we can now make decisions about time allocation with a specific degree of confidence. For example, we can say that for a problem the size of our 35-task, 123-bid sample, we have a 95% confidence in finding a solution using the IP solver in less than 5.6 seconds.

We have generated similar data that allow us to infer probability distributions for a problem set in which the bid count is varied over a 3:1 range, with the size of the task network held constant at 20 tasks, and for another set in which the sizes of bids is varied over a 5:1 range. Again, we

[1]The χ^2 metric is determined by dividing the inferred density function into a number of equal areas, and counting the number of observations that fall into each of those zones. The χ^2 value is the mean square deviation from a "perfect fit".

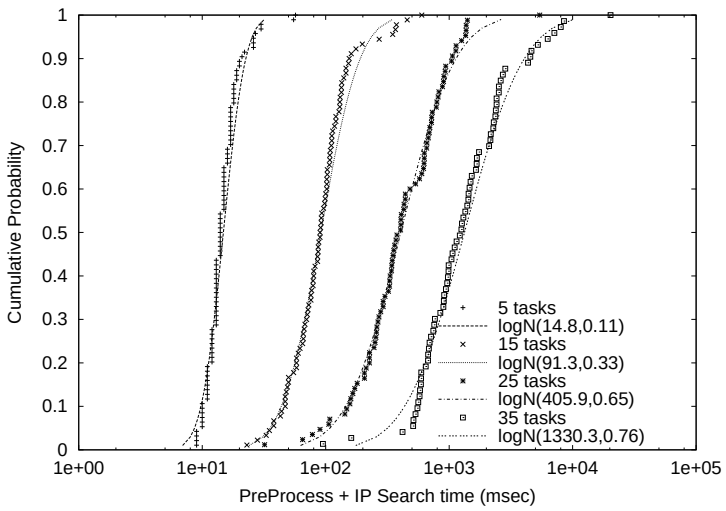

Figure 6: Observed and inferred runtime distributions for the IP solver across a range of task count values, with a nearly constant ratio of bids to tasks.

have found good correspondence ($0.2 < \chi^2 < 3.0$, 9 dof) between the measured data and inferred lognormal distributions.

We now have the necessary data to estimate the time that must be allocated to the winner determination process using the IP solver. The process is to choose a desired "probability of success", and then to estimate parameters for a function that reasonably matches the inferred distributions at that level of probability. More details are in [7].

6 Design Principles and Implementation of the Testbed

The previous sections described the design of MAGNET and of the agents. This section deals with the overall system architecture and the implementation of the market infrastructure.

6.1 Design Principles

In order to maximize the usefulness of the MAGNET testbed as a research tool, we have adopted several design principles that make it easy to plug together and reconfigure, and that enhance its transparency. Examples are:

1. The system is written in Java, and has been tested on multiple platforms.
2. The system is organized into a set of components, and a set of systems that can be constructed from various subsets of components. Each system is constructed to serve a particular experimental purpose.
3. All the major behavioral modules are written as abstract classes, with (at least potentially) multiple implementations that can be "plugged in" to implement a particular behavioral variant.
4. Virtually every feature of the system is selectable and configurable from a configuration file, and many of them can be viewed and changed from a user interface. This includes the choice of behavioral plug-ins.

5. The interface between the agents and the Market is also abstracted. This allows connection with multiple types of markets (such as one that looks up price and availability info from a catalog or timetable) and through multiple communications protocols.
6. Much of the activity of the agent is agenda-driven, and development and maintenance of the agenda is an important activity in its own right. Agenda items can select plug-ins, update configuration details, evaluate options, interact with the market or other agents, update the agenda, and record results.
7. A pervasive logging and data collection system allows for both detailed examination of behavior and the generation of experimental data. The level of logging detail may be independently configured for different modules, and the various logging levels have well-defined meanings.

The system in its current form is useful for several types of studies. Recent work includes experiments with bid evaluation performance, and studies of the RFQ composition problem. Our longer-term goal is to support studies of mixed-initiative decision making with experienced human users in realistic market simulations.

6.2 Architectural Design

We use Web-based technology for the market infrastructure. In particular, the server is implemented in Enterprise JavaBeans [12], and we use Apache SOAP for communication between agents and the server.

The architectural style we selected for MAGNET is based on Web services and messaging. We expect MAGNET components to require both synchronous and asynchronous communications, so the ability to combine the Web Services architectural style with messaging is important.

From the architecture point of view, an agent is modeled as a service requester, requesting some service from a provider, which in MAGNET is an interface to the market. The agent has the choice to use a web-based synchronous approach based on HTTP, or a message based approach using SMTP e-mail. Once the agent chooses the service and makes the request, it will receive a reply, either synchronously or asynchronously depending on the service selected.

A Customer agent interacts with the market as a SOAP service requester. The requester's API contains separate classes that act as SOAP service requesters for each of the SOAP services available. The services include finding available markets in MAGNET, an ontology service, submitting the RFQs, and awarding bids to suppliers. We will soon add execution monitoring services to monitor task execution. A Supplier agent interacts in a similar way, using services such as supplier registration and bid submission.

The MAGNET market exists as an Enterprise JavaBean (EJB) and interacts with customer and supplier agents through the use of the SOAP services described earlier. Market sessions exist also as EJBs and are created and accessed through the use of market-related SOAP services. Session persistence is addressed through the use of entity beans, which are persistent as needed through the use of an application server's database. Session related EJBs are used by the RFQ, bid submission, and bid award SOAP services.

The customer agent and supporting infrastructure have been released as open source at `http://www.openchannelsoftware.com/projects/MAGNET`.

7 An Example

As an example, let's imagine that a small winery needs to bottle wine and ship it[2]. Figure 7 shows a plan to complete the bottling process. The plan is complicated by a couple of factors. If bottling is done in spring, which is the peak season for bottling, often there is a shortage of supplies, such as wine bottles, wine filters, and corks, and there is a shortage of equipment, such as bottling and corking machines. If the wine is going to be sold immediately, then labels and cases have to be created. Each winery uses distinctive labels and cases. Experience also shows that around Christmas wine cases are in short supply, and that shipping resources are over booked.

There are other complicating factors, such as the fact that different types of wine (white or red) require different types of bottles, and that some wine bottles require special corks.

In our example, let's assume that bottling is done in spring. This means there is a shortage of supplies and equipment. Since the bottled wine has to be sold immediately, the deadline for shipping is short.

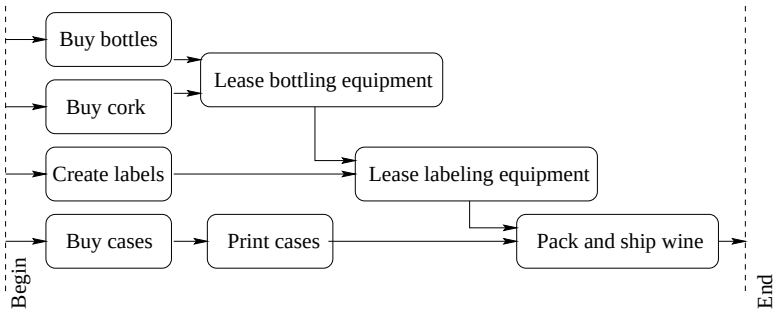

Figure 7: Plan for bottling wine

The first task for the Customer agent is to plan for all the tasks needed to accomplish the goal of bottling and shipping wine. The plan is expressed as a task network, such as the one shown in Figure 7. The task network is generated by the component of the Customer agent which is labeled Planner in Figure 3.

The task network is then passed to the Bid Manager. The Bid Manager is responsible for ensuring that resources are assigned to each of the tasks, that the assignments taken together form a feasible schedule, and that the cost and risk of executing the plan is minimized. This cost must also be less than the value of the goal at the time the goal is reached.

When the Bid Manager is invoked, some tasks may already be assigned. This can occur because the Execution Manager may use the Bid Manager to repair a partially-completed plan in which previously determined assignments have failed, because the agent will perform some of the tasks itself, or because bidding is being carried out in multiple stages. For example, the customer may be able to create the labels for the bottles, and so it might not contract for that task.

Before bids can be solicited in the market, an RFQ must be composed. The RFQ is a structure that contains some portion of the plan data (tasks and precedence relations) along with a set of scheduling constraints. To compose the RFQ the Bid Manager needs not only to know what tasks should be specified in the RFQ, but also decide when each task should be scheduled and how much flexibility to allow in the schedule.

[2]The example is taken from the operations of the "Weingut W. Ketter" winery, Kröv, Germany.

In our winery example, we would find out from the market that bottling resources are thin. The Bid Manager might decide to wait until the bottling is under way before ordering the labels. However, this choice is possible only if the wine does not need to be shipped immediately.

Dividing the bidding process into multiple phases can be an important strategy to reduce the level of uncertainty. For example, we might not want to take bids on the labeling equipment until we have firm dates for the bottles and corks.

Figure 8 shows two alternative ways to schedule and compose the RFQs for our winery project. In version A, we believe we have 5 weeks to finish our task, and the scarce resources are bottling and labeling equipment. Therefore, we allow a week and a half for the one-week bottling job. Since time windows do not overlap, we are guaranteed that if we receive bids on all tasks, they can be combined feasibly. In version B, we are interested in shipping the bottled wine as soon as possible. Therefore, we bid out the bottling equipment and the supplies first in RFQ B1, and we bid out the remainder of the tasks, labeling and pack/shipping, in RFQ B2 after we received and accepted a bid that finishes the labels by the end of week 1. Splitting the RFQ gives us the opportunity to reduce the overall duration for the tasks, and to get done in 4 weeks instead of 5 weeks.

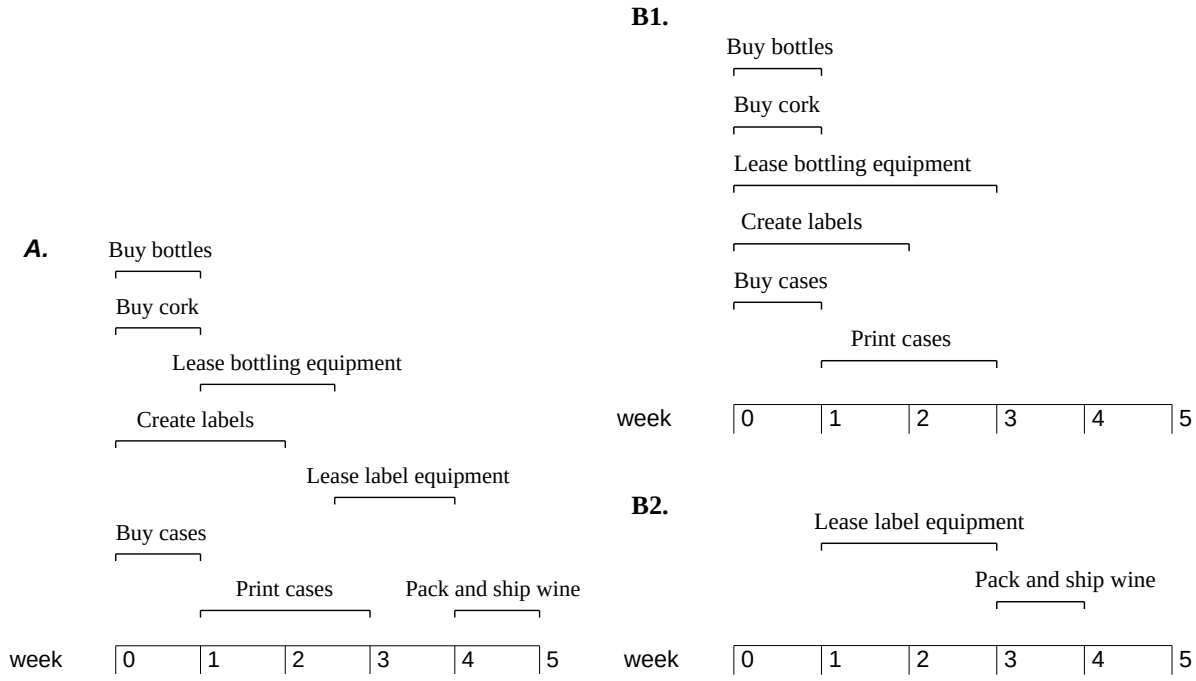

Figure 8: RFQ Example. A shows a single RFQ, B1 and B2 are two separate RFQs for the same tasks, with different time windows

Figure 9 shows yet another way to decompose the RFQ in two parts with larger time windows and greater overlap between them. The tasks are divided between the two RFQs differently and there is a much larger overlap of the time windows. This gives more flexibility to the suppliers, but makes it harder to combine the bids into a feasible schedule, which respects all the precedence constraints. Clearly, deciding whether to split RFQs and how to split them is, by itself, a complicated problem.

Once bids have been submitted, it is up to the Bid Evaluator to evaluate them and decide

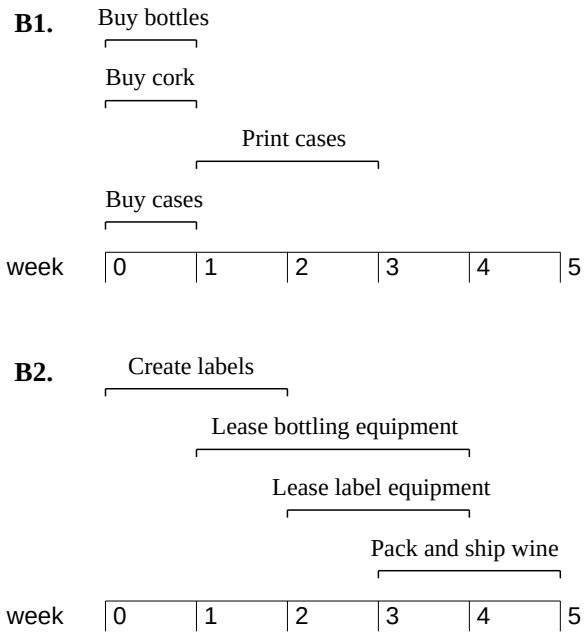

Figure 9: A different partitioning of tasks in two RFQs. The total completion time is the same, but some time windows are different.

which ones to accept (if any). The Bid Evaluator includes a search engine that, given a task network and a set of bids, attempts to find an optimal or near-optimal mapping of bids to tasks, respecting temporal constraints. This includes solving an extended version of the combinatorial auction winner-determination problem [1, 29].

Figure 10 shows a very small example of the problem the Bid Evaluator must solve. We composed the RFQ with a large overlap between the labeling and bottling tasks, perhaps because we believed there would be large numbers of bidders with a wide variation in lead times.

Bid 2 indicates bottling could start late in week 2, would take a week, and the supplier was willing to shift that out 2 more days to accommodate our schedule. Bid 3 shows that labeling could start through week 2, would take a eek, and needs to finish mid of week 3. Clearly these two bids cannot be combined, since labeling must at least finish when bottling finishes or later, but cannot finish before bottling is finished. Bid 4 shows a more expensive bottling task which could start earlier, and also needs a week to finish. This can be combined with Bid 3, with one day slack to accommodate contingencies. Bid 5 gives us a large enough time window for the labeling task to be combined with Bid 4 but not with Bid 2. Bid 6 shows one special characteristic of the MAGNET system. One supplier can bid for multiple tasks in one bid, even if the RFQ specified those tasks separately. In this example this proves to be cheaper but not very efficient time-wise. The supplier offers to do the two tasks cheaper then others do, but the start time is later. Bid 3 has a lower cost, but it leaves no time for the successive labeling task, and so it cannot be accepted. The best combination for an early finish is Bid 1, Bid 4, Bid 5 and Bid 7. This combination finishes at the end of the second day of week 3, which means it finishes one week and three days earlier than originally specified in the RFQ. The cost is 3000. If saving money is more important than finishing earlier, Bid 1, Bid 6, and Bid 7 will be selected, for a total cost of 2895.

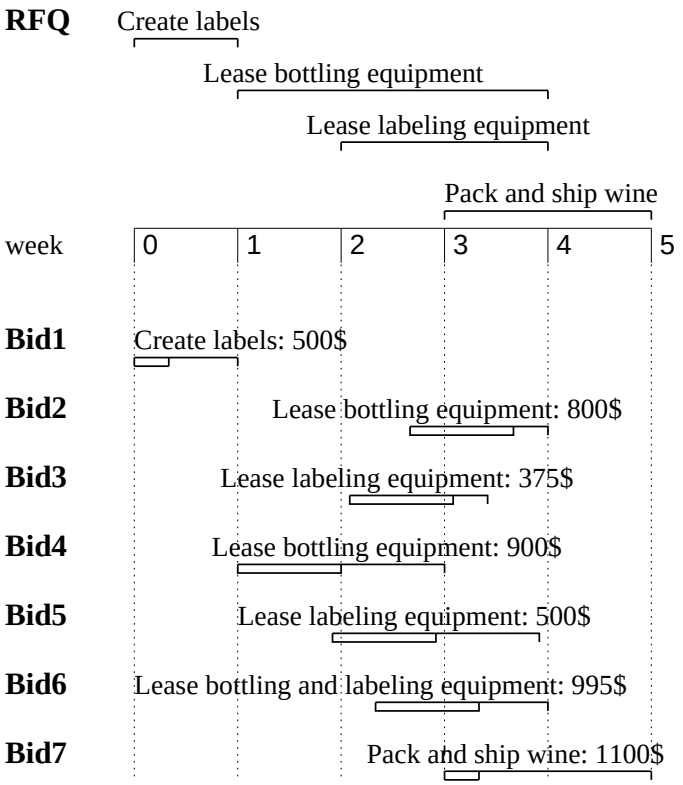

Figure 10: Bid Example

8 Related Work

Markets play an essential role in the economy, and market-based architectures are a popular choice for multiple agents (see, for instance, [3, 34, 38, 35, 19, 4]). Most market architectures limit the interactions of agents to manual negotiations, direct agent-to-agent negotiation [32, 11], or various types of auctions [39].

Existing architectures for multi-agent virtual markets typically rely on the agents themselves to manage the details of the interaction between them, rather than providing explicit facilities and infrastructure for managing multiple negotiation protocols. In our work, agents interact with each other through a market. The market infrastructure provides a common vocabulary, collects statistical information that helps agents estimate costs, schedules, and risks, and acts as a trusted intermediary during the negotiation process.

Auctions are becoming the predominant mechanism for agent-mediated electronic commerce [15]. The Fishmarket [27], AuctionBot [39], and eMEDIATOR [29] are examples of multi-agent auction systems. The determination of winners of combinatorial auctions [23] is hard. Methods for improving the efficiency of combinatorial auctions have been developed in the last few years, among others, by Sandholm [29] and Fujishima [13]. Mixed integer programming has been demonstrated to work extremely well even on large problems by Andersson [1]. We have demonstrated how to use Integer Programming within the constraints posed by our formulation of the problems in MAGNET [6].

Several systems have attempted to organize task-oriented work among multiple agents. Parkes [25] describes an auction-based system for controlling access to shared railroad resources. It uses a

mixed-integer approach, with many domain-specific optimizations. In [17] combinatorial auctions are used for the initial commitment decision problem, which is the problem an agent has to solve when deciding whether to join a proposed collaboration. Their tasks have precedence and hard temporal constraints. However, to reduce search effort, they use domain-specific *roles*, a shorthand notation for collections of tasks. In their formulation, each task type can be associated with only a single role. MAGNET agents are self-interested, and there are no limits to the types of tasks they can decide to do. In [14] scheduling decisions are made not by the agents, but instead by a central authority. The central authority has insight to the states and schedules of participating agents, and agents rely on the authority for supporting their decisions.

9 Conclusions and Future Work

The MAGNET automated contracting environment is designed to support negotiation among multiple, heterogeneous, self-interested agents over the distributed execution of complex tasks. The MAGNET testbed is a prototype implementation of a Customer agent, a market server, and a population of Supplier agents. It is highly configurable and extensible, and has been used for several statistical studies aimed at understanding the decision processes for a Customer agent.

The current system has proven to be very useful for the types of statistical studies we have pursued so far. Future plans call for more focus on mixed-initiative interaction, and our current user interface is too primitive to support that work.

A major need in this area of research is the establishment of a set of benchmark problems by which different strategies can be compared. Leyton-Brown et al [21] have proposed a test suite called CATS for testing combinatorial auction systems. It solves part of the problem, but it only deals with bids, not the RFQ, and it does not handle the precedence relations and temporal constraints needed in the MAGNET environment.

Acknowledgments

Our thanks to the many students who have contributed to the design of the architecture and to the implementation of MAGNET.

References

[1] Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for combinatorial auction winner determination. In *Proc. of 4th Int'l Conf on Multi-Agent Systems*, pages 39–46, July 2000.

[2] Alex Babanov, John Collins, and Maria Gini. Risk and expectations in a-priori time allocation in multi-agent contracting. In *Proc. of the First Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

[3] Anthony Chavez and Pattie Maes. Kasbah: An agent marketplace for buying and selling goods. In *Proc. of the First Int'l Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, April 1996.

[4] Samuel P. M. Choi and Jiming Liu. A dynamic mechanism for time-constrained trading. In *Proc. of the Fifth Int'l Conf. on Autonomous Agents*, pages 568–575, 2001.

[5] John Collins, Corey Bilot, Maria Gini, and Bamshad Mobasher. Decision processes in agent-based automated contracting. *IEEE Internet Computing*, pages 61–72, March 2001.

[6] John Collins and Maria Gini. An integer programming formulation of the bid evaluation problem for coordinated tasks. In Brenda Dietrich and Rakesh V. Vohra, editors, *Mathematics of the Internet: E-Auction and Markets*, volume 127 of *IMA Volumes in Mathematics and its Applications*, pages 59–74. Springer-Verlag, New York, 2001.

[7] John Collins, Maria Gini, and Bamshad Mobasher. Multi-agent negotiation using combinatorial auctions with precedence constraints. Technical Report 02-009, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, Minnesota, February 2002.

[8] John Collins, Scott Jamison, Maria Gini, and Bamshad Mobasher. Temporal strategies in a multi-agent contracting protocol. In *AAAI-97 Workshop on AI in Electronic Commerce*, July 1997.

[9] John Collins, Rashmi Sundareswara, Maria Gini, and Bamshad Mobasher. Bid selection strategies for multi-agent contracting in the presence of scheduling constraints. In A. Moukas, C. Sierra, and F. Ygge, editors, *Agent Mediated Electronic Commerce II*, volume LNAI1788. Springer-Verlag, 2000.

[10] John Collins, Ben Youngdahl, Scott Jamison, Bamshad Mobasher, and Maria Gini. A market architecture for multi-agent contracting. In *Proc. of the Second Int'l Conf. on Autonomous Agents*, pages 285–292, May 1998.

[11] Peyman Faratin, Carles Sierra, and Nick R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1997.

[12] David Flanagan, Jim Farley, William Crawford, and Kris Magnusson. *Java Enterprise in a nutshell*. O'Reilly & Associates, Inc., Sebastopol, CA, 1999.

[13] Yuzo Fujishjima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. of the 16th Joint Conf. on Artificial Intelligence*, 1999.

[14] Alyssa Glass and Barbara J. Grosz. Socially conscious decision-making. In *Proc. of the Fourth Int'l Conf. on Autonomous Agents*, pages 217–224, June 2000.

[15] Robert H. Guttman, Alexandros G. Moukas, and Pattie Maes. Agent-mediated electronic commerce: a survey. *Knowledge Engineering Review*, 13(2):143–152, June 1998.

[16] Holger H. Hoos and Thomas Stützle. Evaluating Las Vegas algorithms – pitfalls and remedies. In *Proc. of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 238–245. Morgan Kaufmann Publishers, 1998.

[17] Luke Hunsberger and Barbara J. Grosz. A combinatorial auction for collaborative planning. In *Proc. of 4th Int'l Conf on Multi-Agent Systems*, pages 151–158, Boston, MA, 2000. IEEE Computer Society Press.

[18] Sari Kalin. Trading places. *CIO Magazine*, 13(9):96, 2000.

[19] Nikos Karacapilidis and Pavlos Moraïtis. Intelligent agents for an artificial market system. In *Proc. of the Fifth Int'l Conf. on Autonomous Agents*, pages 592–599, 2001.

[20] D. M. Kreps. *A Course in Microeconomic Theory*. Princeton University Press, 1990.

[21] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proc. of ACM Conf on Electronic Commerce (EC'00)*, Minneapolis, MN, October 2000.

[22] B. Loritsch. Developing with Apache Avalon. Apache Software Foundation, 2001.

[23] R. McAfee and P. J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[24] Noam Nisan. Bidding and allocation in combinatorial auctions. In *1999 NWU Microeconomics Workshop*, 1999.

[25] David C. Parkes and Lyle H. Ungar. An auction-based method for decentralized train scheduling. In *Proc. of the Fifth Int'l Conf. on Autonomous Agents*, pages 43–50, Montreal, Quebec, May 2001. ACM Press.

[26] Charles Phillips and Mary Meeker. The B2B internet report – Collaborative commerce. Morgan Stanley Dean Witter, April 2000.

[27] J. A. Rodriguez, P. Noriega, C. Sierra, and J. Padget. FM96.5 - A Java-based electronic auction house. In *Second Int'l Conf on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97)*, London, April 1997.

[28] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter*. MIT Press, Cambridge, MA, 1994.

[29] Tuomas Sandholm. An algorithm for winner determination in combinatorial auctions. In *Proc. of the 16th Joint Conf. on Artificial Intelligence*, pages 524–547, 1999.

[30] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[31] Tuomas Sandholm and Victor Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *1st Int'l Conf. on Multiagent Systems*, pages 328–335, San Francisco, 1995.

[32] Tuomas W. Sandholm. *Negotiation Among Self-Interested Computationally Limited Agents*. PhD thesis, Department of Computer Science, University of Massachusetts at Amherst, 1996.

[33] Katia Sycara, Keith Decker, and Mike Williamson. Middle-agents for the Internet. In *Proc. of the 15th Joint Conf. on Artificial Intelligence*, pages 578–583, 1997.

[34] Katia Sycara and Anandeep S. Pannu. The RETSINA multiagent system: towards integrating planning, execution, and information gathering. In *Proc. of the Second Int'l Conf. on Autonomous Agents*, pages 350–351, 1998.

[35] Maxsim Tsvetovatyy, Maria Gini, Bamshad Mobasher, and Zbigniew Wieckowski. MAGMA: An agent-based virtual market for electronic commerce. *Journal of Applied Artificial Intelligence*, 11(6):501–524, 1997.

[36] Hal R. Varian. Economic mechanism design for computerized agents. In *USENIX Workshop on Electronic Commerce*, New York, NY, July 1995.

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

[38] Michael P. Wellman and Peter R. Wurman. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24:115–125, 1998.

[39] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second Int'l Conf. on Autonomous Agents*, pages 301–308, May 1998.

[40] V. Zwass. Electronic commerce: structures and issues. *Int'l Journal of Electronic Commerce*, 1(1):3–23, 1996.