

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

John Collins, Wolfgang Ketter, and Maria Gini

ABSTRACT: In multi-agent contracting, customer agents solicit the resources and capabilities of other agents, sometimes executing multistep tasks in which tasks are contracted out to different suppliers. The authors have developed a testbed for studying the decision behaviors of agents in this context. It generates sets of tasks with known statistical attributes, formulates and submits requests for quotations, generates bids with well-defined statistics, and evaluates bids according to several criteria. Each of these processes is supported by an abstract interface and a series of pluggable modules with numerous configuration parameters, and with data collection and analysis tools.

KEY WORDS AND PHRASES: Electronic marketplace, multi-agent contracting.

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. On-line marketplaces are gaining popularity with companies seeking to streamline their operations. On-line marketplaces offer benefits to both buyers and sellers. For buyers, they can significantly ease the process of searching for and comparing providers, while for sellers they can provide access to a much broader customer base [26]. Business-to-business hubs, which link buyers within an industry or across a shared need, are expected to handle as much as \$1.25 trillion by 2003 [18].

Over the past decade, the logistical complexity of 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 that demand tight integration of all processes. Thus the field is ripe for the introduction of systems that automate logistics planning among multiple entities, such as manufacturers, parts suppliers, shippers, and specialized subcontractors.

Deciding what to outsource and where, ensuring that tasks are done in the proper sequence (e.g., 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 speed is essential for made-to-order products, and needing extra time may prevent a supplier from getting a contract. The decision-maker also has to keep track of delays by suppliers that may jeopardize the completion of tasks, and renegotiate with them or others as needed.

Partial support for this research was provided by NSF under award NSF/IIS-0084202. Our thanks to the many students who contributed to the design and implementation of the MAGNET architecture.

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

Computerized agents can examine offers much more quickly than humans. They can evaluate alternative choices that are too complicated for humans because of the large number of alternatives or the complexity of the computations. For these reasons, agent-based contracting will result in cheaper, faster contract negotiations, freedom from human error, shorter lead times, prompter deliveries, and ultimately, reduced costs for both suppliers and customers.

Autonomous or semiautonomous agents can only come into use if the processes of creating, modifying, and deploying them are cost-effective. Since billions of dollars may be at stake, it is imperative that the agents exhibit the desired behavior when deployed. In short, what is needed is a cost-effective agent-development methodology that produces agents with predictably correct behavior.

This paper presents a market architecture that supports multi-agent contracting and describes the implementation of a prototype of the market architecture and the agents. The system is called MAGNET (Multi-AGent NEgotiation Testbed). It provides support for a variety of transactions, from simple buying and selling of goods and services to complex multi-agent negotiation of contracts with temporal and precedence constraints.

The major goal of the paper is to describe the main features of MAGNET and show how it can be used to develop a realistic simulation of an actual market. MAGNET is not yet a complete simulation of a market. Currently, it is focused on three areas: determining the form and content of each request for quotations (RFQ), managing the bidding process, and evaluating bids submitted by suppliers.

Market Infrastructure

Because of the advent of open standards, electronic commerce on the Internet is one of the fastest-growing segments of 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 a fast and inexpensive standardized communication infrastructure, more organizations and individuals are relying on virtual environments like 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 favors hub-and-spoke architectures, where suppliers, customers, and trading partners need only one connection to communicate with

one 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, and not transparent, and whose high entry costs tend to keep out small companies.

As briefly outlined below, there are several major advantages in providing a market infrastructure to support human decision-makers or automated agents.

Support for Multi-Agent Negotiation over Extended Periods

Negotiations can take a very long time, and during this period a context must be maintained. It may also take quite long, in the range of weeks to months, for the negotiated transaction to be completed. 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. The parties to a time-sensitive negotiation process must have a common time reference. The market can provide this, as well as methods to validate nonperformance and assess negotiated penalties.

Value-Added Services

Value-added services, such as matchmaking or publish-subscribe facilities for notification of important events, are important for two reasons [33]. 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. Second, 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 the availability of suppliers with specific capabilities, or independent ratings based on past performance. The general statistical information is useful for customer agents in formulating RFQs, while performance ratings can be used to determine the price associated with various bids.

Protection Against Fraud and Misrepresentation

Participating self-interested agents will exploit every opportunity to gain advantage. The market facilitates recognizing and defending 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.
- Conducting auctions dishonestly. For example, in Vickrey-type auctions, the motivation for participants to tell the truth is predicated on their belief in the honesty of the auctioneer [36, 37].

- Miscommunication 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]. There are two types of counterspeculation. *Value-based* counterspeculation occurs when agents use their own estimates of one other's valuations to set bid prices [28, 36, 37]. *Time-based* counterspeculation opportunities fall into two classes [8]. One of these situations occurs when supplier agents are allowed to expire their bids before the RFQ 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 begin 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 higher-priced bids can be used to skew the customer's reasoning process.

The market provides a neutral third-party facility that can 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.

The MAGNET Architecture

The authors have designed a market infrastructure called 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*, *market sessions*, and *MAGNET agents*.

Market

Each market in MAGNET is a forum for commerce in a particular business area. It includes a set of domain-specific services, as shown in Figure 1, and 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 the registry include the identity of participants and a catalog (or a method for accessing a catalog) of each participant's interests, products, or capabilities that can be used for matchmaking [33].
- An *ontology* specifying the terms of discourse within the domain of the market. Agents that wish to offer resources and services do so through one or more market segments whose ontologies describe their offerings. The market that will be described further on as an

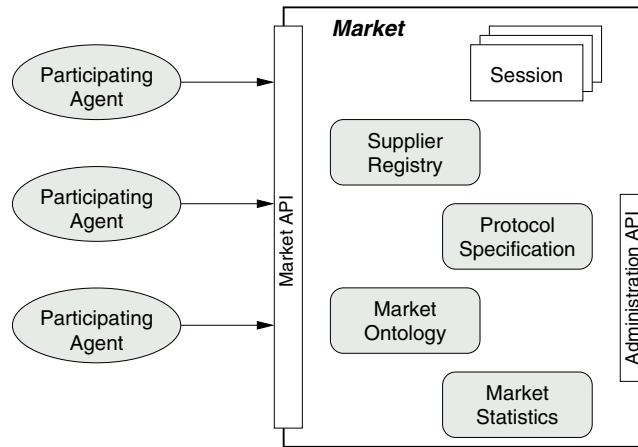


Figure 1. The Structure of a Market

© 2002 John E. Collins. Used with Permission.

example covers the 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 data on expected duration and variability, expected price and variability, and resource availability.

- A *protocol specification* that formalizes the types of negotiation supported in the market. These are limits on the parameters of the negotiation protocol, such as whether bids can be awarded before the bid deadline.

Market Sessions

An important component of every market is a set of current *market decisions* in which the actual agent interactions occur. Each session is initiated by a single agent for a particular purpose. Multiple agents can generally 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 negotiations, awards, construction work, paying of bills, and final closing. The session mechanism ensures the continuity of partially completed transactions, and relieves the participating agents of having to keep track of the details of the negotiation status.

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 initiator's standpoint. 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 client's standpoint. Proxies are market entities that act on behalf of agents and enforce market rules.

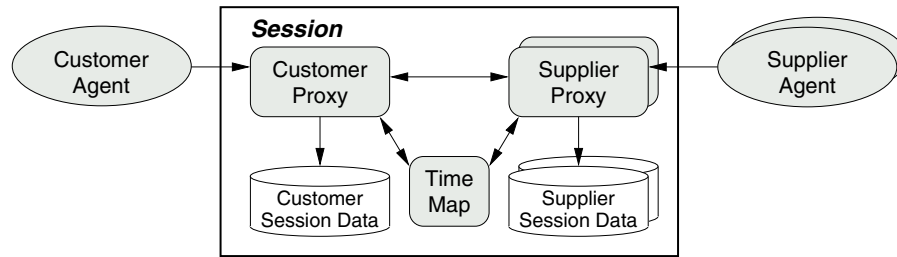


Figure 2. The Structure of a Market Session

© 2002 John E. Collins. Used with Permission.

There are two main reasons for the existence of proxy components. First, client proxy components cannot see the private data of the initiator or of other clients, which increases security. Second, in a distributed system environment, the processing and persistent data elements of the initiator and clients can occur at different locations in the network to maximize performance.

MAGNET Agents

The agents in MAGNET are independent entities acting on behalf of individuals or commercial entities that have different goals and resources.

There are two distinct agent roles, customer and supplier. A *customer* is an agent that has a goal to satisfy, and needs resources outside its direct control in order to achieve it. The goal may have a *value* that varies over time. A *supplier* is an agent that has resources and, 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 the two types of agents and their primary interactions is shown in Figure 3.

The negotiation process consists of a contracting phase and an execution phase. The *contracting* phase is a first-price, sealed-bid combinatorial auction consisting of RFQ, bidding, and bid awards. This three-step process is designed to simplify negotiations without loss of generality. It is modeled on the leveled commitment protocol proposed by Sandholm [32]. 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 and decommitments, and in some cases repeats the bidding cycle when it becomes necessary to reallocate resources that were originally committed.

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

- The customers issue requests for quotations that include specifications for 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 it can end.
- Suppliers submit bids and commit to resources (with the possibility of overcommitment) until the bids are awarded or rejected by the

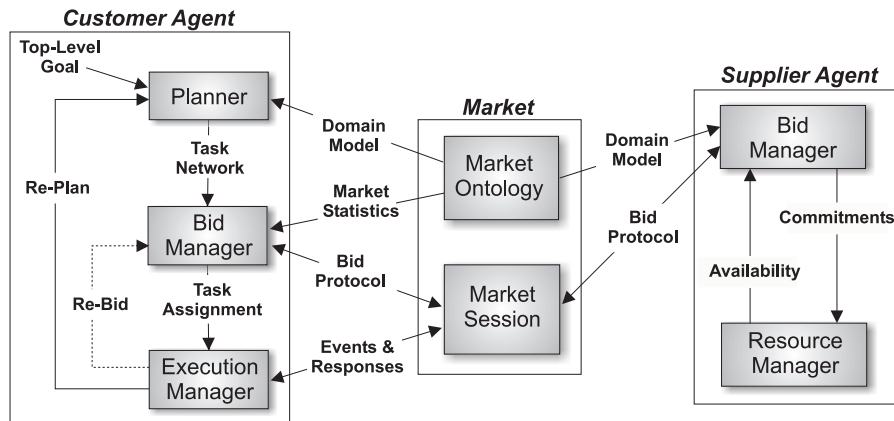


Figure 3. The MAGNET Architecture

customer. A bid includes a price for the task, a portion of the price that must be paid as a nonrefundable deposit when the bid is awarded, an estimated duration for the task, and a time window within which the task can begin.

- 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 begins to execute the tasks awarded and tries to complete them in the specified time frame. When the supplier completes a task, the customer pays the balance 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 nonperformance.

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

Customer Agents

A customer agent, as illustrated in Figure 3, has three major components: the planner, the bid manager, and the execution manager.

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

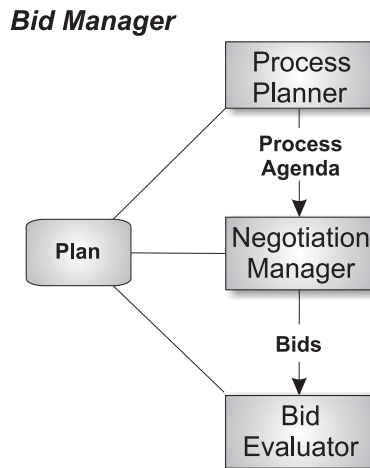


Figure 4. The Bid Manager

them, and possibly nonzero delays between tasks, to cover communication and transportation delays. An example is shown below in Figure 7.

In the current implementation, a task network is created either by selecting a predefined plan from a library of plans or by selecting randomly from a library of task types and creating random precedence relations among them. In many domains, plans will be chosen from a library or defined by a human user rather than 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.

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. It may be desirable to incorporate expected utility theory in these decision processes [5].

The first decision is to determine how much time suppliers will have to submit bids and an approximate schedule by setting limits on the start and end times for each task. This decision is made by the process planner. The more time customers keep for their own decisions, the more flexibility they have in deciding whether to accept bids or post new RFQs 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.

Next, a decision has to be made on whether to divide the tasks among multiple requests for quotations or to submit a single RFQ and determine the 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 either to leaving large amounts of slack in the schedule, which delays the completion of the tasks, or to generating RFQs with large overlaps in task timing, which causes many bids to be unusable. When this is the case, the bidding process can be split into phases that each allow for limited schedule variability. There are several ways to do this, and experiments are being performed to understand how they perform, what the tradeoffs are, and how to recognize situations where multiphase bidding is advantageous. The preliminary results suggest that multiphase bidding can, on average, generate tighter schedules, at the same price, and substantially reduce search effort on the customer side. On the other hand, the time required for the overall bidding process may be 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 requests for quotations. These exhibit 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 in the RFQ, (2) statistical information about duration and variability for the different task types, as well as information about resource availability, and the number of vendors likely to bid, which are available from the market, and (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 requests for quotations 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. This is done 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 RFQ specifications that have no overlap (guarantees that all bids will combine feasibly).

Several plug-in versions that build RFQs have been implemented in order to test alternative approaches. The application of expected utility theory to the generation of RFQs is currently being studied [2].

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

The winner determination problem for combinatorial auctions has been shown to be complete and inapproximable [29]. This result clearly applies to the MAGNET winner determination problem, since an additional set of (temporal) constraints is simply applied to the basic combinatorial auction problem, and one 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, it is necessary to accept exponential complexity. As will be shown below, probability

distributions for search time can be determined, based on problem size metrics, and these empirically determined distributions can be used in the deliberation scheduling process (for more details, see [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. The following bid-evaluation search engines have been implemented: a highly modular simulated annealing version [9], a mixed integer programming version [6], and A* and IDA* versions that extend Sandholm's bid tree-based IDA* algorithm [31] to deal with reverse-auction problems where there are precedence constraints among items.

A wide range of conditions must be controlled in the study of bid evaluations, including:

- composition of generated plans (i.e., number of tasks, task types [which in turn control duration variability and probability of bids], density of precedence network).
- structure of RFQs (i.e., whether they cover the whole plan, amount of slack in schedule, degree to which bids are allowed to violate precedence relations).
- number, size, and composition of bids (i.e., random selections, contiguous task sets, role-based task sets).
- type of search used, search parameters.
- bid selectors and evaluators, evaluation parameters.

The MAGNET Testbed

The current implementation of MAGNET includes a fully implemented and customizable customer agent, described in the preceding 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 in a form that can be used by a standard spreadsheet or by Matlab in the case of anytime performance data.

Supplier Agents

Supplier agents were designed only recently and are still under development. They are implemented using Avalon [22] and are based on components. Using Avalon, it is straightforward to have the components of the supplier agent interact, instantiate different instances of the components, and reuse code. Avalon also allows switching components on the fly, which is very useful in testing. It can be configured using XML files that specify which components and which instances have to be included.

Since the workings of customer agents are the primary interest, a simple-

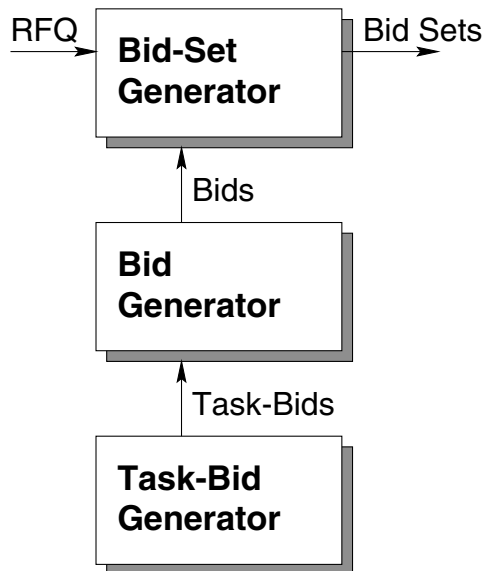


Figure 5. Simple Supplier Simulation

minded supplier has also been implemented that simply receives RFQs and responds by submitting bids. These do not maintain resource schedules and have no persistent identity. The basic structure is shown in Figure 5. Each of the three layers is implemented as an abstraction with multiple implementations.

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 an 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 the 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. Bid generators have been implemented 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 suc-

cessors 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.

Experimental Studies

The goal of the research described here 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 that is planned, the preliminary results on characterizing the performance of a winner-determination algorithm based on integer programming (IP) will be reported below. The study follows the methodology outlined by Hoos and Stützle [16].

Three measures of performance are of interest: speed, scalability, and predictability. Speed and scalability are important because combinatorial auction winner determination is known to be \mathcal{NP} -complete and inapproximable [31]. 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 the timeline must include the time allocated to winner determination.

The predictability issue is addressed by characterizing the winner-determination methods with respect to factors that can be measured or estimated when the RFQ is issued. These include (1) the number of tasks in the task network, (2) the number of bids likely to be submitted, and (3) the expected sizes of the submitted bids, that is, 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 run-time distributions for four problem sets because the size of the task network is varied. For each curve, actual observations are shown along with a logarithm of normal distribution that minimizes the χ^2 metric.¹ Typical χ^2 values range from 0.3 to 3.0 for nine degrees of freedom, a good match. Note that the first parameter of the logarithm of normal distribution is equal to the median value, which is significantly smaller than the mean for all sets. The value of data such as these is that one can now make decisions about time allocation with a specific degree of confidence. For example, one can say that for a problem the size of the present 35-task, 123-bid sample, there is 95 percent certainty that a solution can be found using the IP solver in less than 5.6 seconds.

Similar data have been generated that make it possible 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, there is a good correspondence ($0.2 < \chi^2 < 9$ degrees of freedom) between the measured data and the inferred logarithm of normal distribution.

The necessary data have now been accumulated for estimating the time

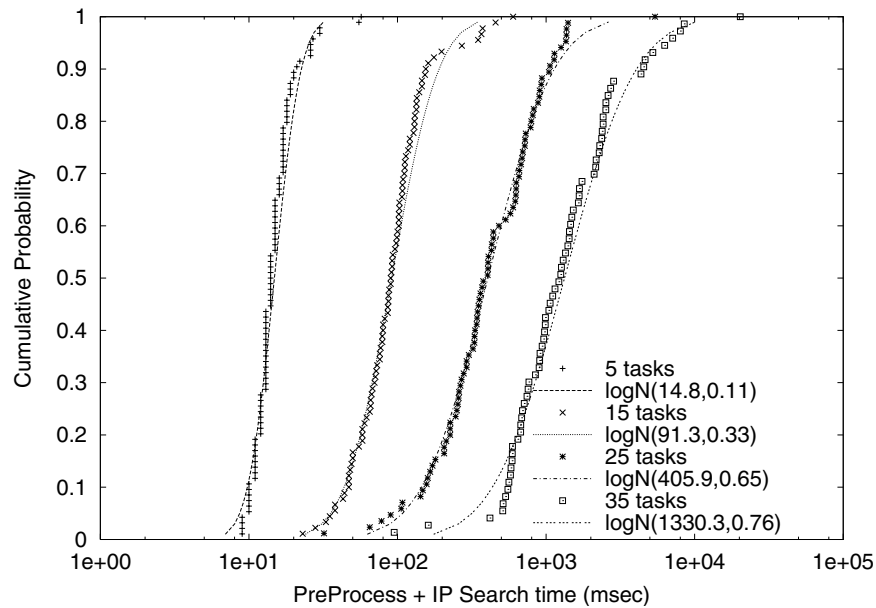


Figure 6. Observed and Inferred Runtime Distributions for the IP Solver Across a Range of Task Count Values, with a Nearly Constant Ratio of Bids to Tasks

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. Collins, Gini, and Mobasher give more details [7].

Design Principles and Implementation of the Testbed

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

Design Principles

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

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 specific 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 specific behavioral variant.
4. Virtually every feature of the system is selectable and configurable from a configuration file, and many of these 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 (e.g., one that looks up price and availability information from a catalog or timetable) and through multiple communications protocols.
6. Much of the agent’s activity 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. The longer-term goal is to support studies of mixed-initiative decision-making with experienced human users in realistic market simulations.

Architectural Design

Web-based technology is used for the market infrastructure. The server is implemented in Enterprise JavaBeans [12], and Apache SOAP is used for communication between agents and the server.

The architectural style selected for MAGNET is based on Web services and messaging. MAGNET components are expected 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 of using 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.

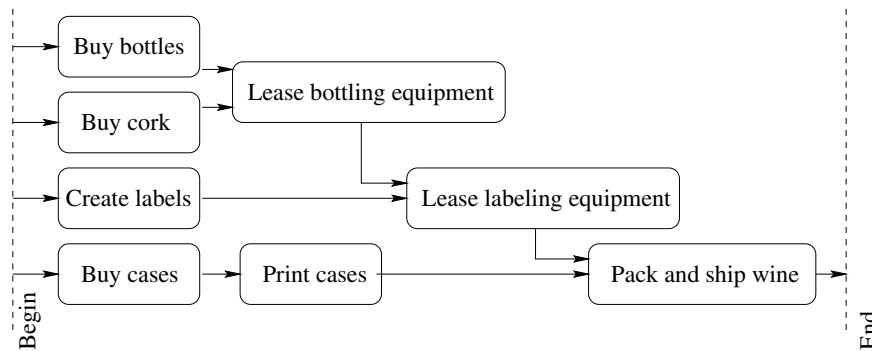


Figure 7. Plan for Bottling Wine

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 RFQs, and awarding bids to suppliers. Execution monitoring services will soon be added to monitor task execution. A supplier agent interacts in a similar way, using services like supplier registration and bid submission.

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

The customer agent and supporting infrastructure have been released as open source at www.cs.umn.edu/magnet.

An Example

As an example, imagine that a small winery needs to bottle wine and ship it.² Figure 7 shows a plan for completing the bottling process. The plan is complicated by a couple of factors. If bottling is done in spring, the peak season, there are often shortages of supplies (e.g., wine bottles, wine filters, corks) and equipment (e.g., bottling and corking machines). If the wine is going to be sold immediately, labels and cases have to be made. Every winery has its own distinctive labels and cases. Around Christmas, wine cases are in short supply, and shipping resources are overbooked.

There are other complicating factors. For example, different types of wine (white or red) require different types of bottles, and some wine bottles require special corks.

The example assumes that bottling is done in spring. This means that there is a shortage of supplies and equipment. Since the bottled wine has to be sold immediately, the deadline for shipping is short.

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 like the one shown in Figure 7. The task network is generated by the component of the customer agent that 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 are minimized. The cost must 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 either because the execution manager is using 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 the bidding is being carried out in stages. For example, a customer able to create the labels for its bottles may not contract for that task.

Before bids can be solicited in the market, requests for quotations must be composed. RFQs are a structure that contains some portion of the plan data (tasks and precedence relations) along with a set of scheduling constraints. To compose an RFQ, the bid manager must not only know what tasks to specify, but also must decide when to schedule each task and how much flexibility to allow in the schedule.

In the winery example, if it is learned from the market that bottling resources are thin, the bid manager might decide to wait until bottling is underway before ordering the labels. However, this choice would only be feasible if the wine is not to be shipped immediately.

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

Figure 8 shows two alternative ways to schedule and compose RFQs for the winery project. In version A, there are five weeks to finish the task, and the scarce resources are bottling and labeling equipment. Therefore, a week and a-half is allotted for the one-week bottling job. Since time windows do not overlap, one is guaranteed that if bids on all tasks are received, they can be combined feasibly. In version B, the bottled wine has to be shipped as soon as possible. Therefore, the bottling equipment and supplies are bid out first in RFQ B1, and the remainder of the tasks, labeling and pack/shipping, are bid out in RFQ B2 after a bid that finishes the labels by the end of week 1 has been received and accepted. Splitting the RFQ makes it possible to reduce the overall duration of the tasks and finish in four weeks instead of five.

Figure 9 shows yet another way to decompose the request for quotations into 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 workable schedule that respects all the precedence constraints. Clearly, deciding whether and how to split RFQs is, in itself, a complicated problem.

Once bids have been submitted, it is up to the bid evaluator to evaluate them and decide which 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.

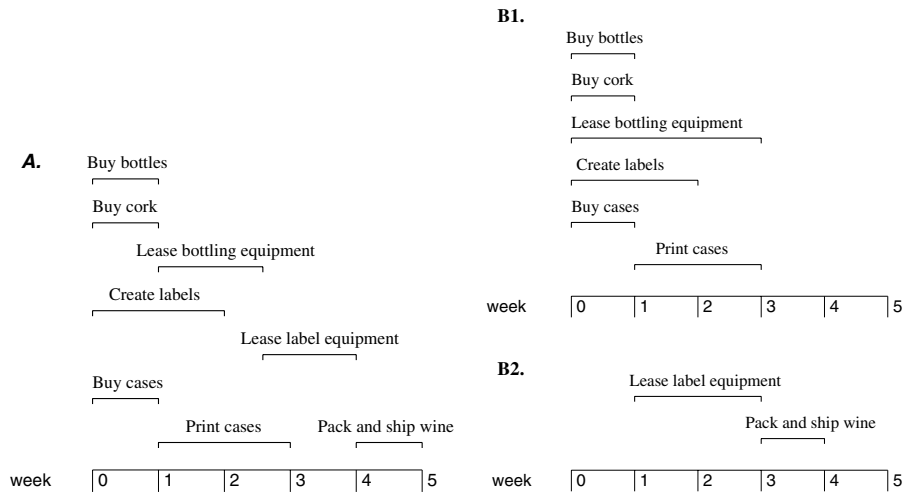


Figure 8. RFQ Example

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

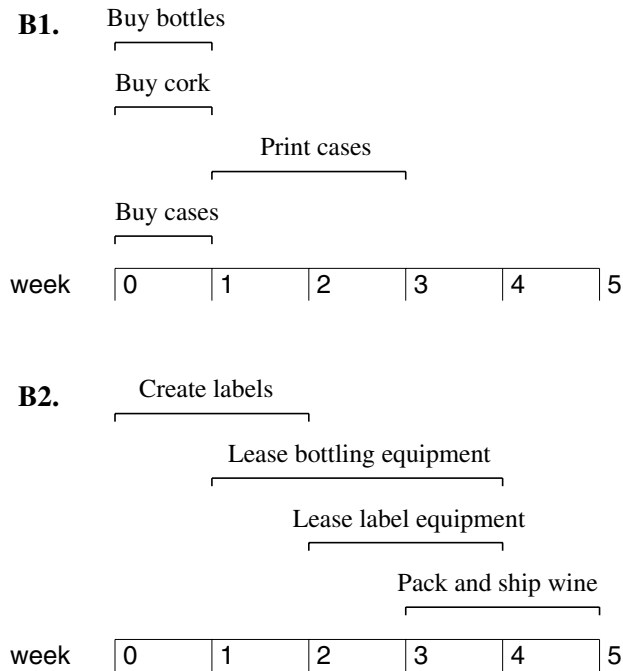


Figure 9. A Different Partitioning of Tasks in Two RFQs

The total completion time is the same, but some of the time windows are different.

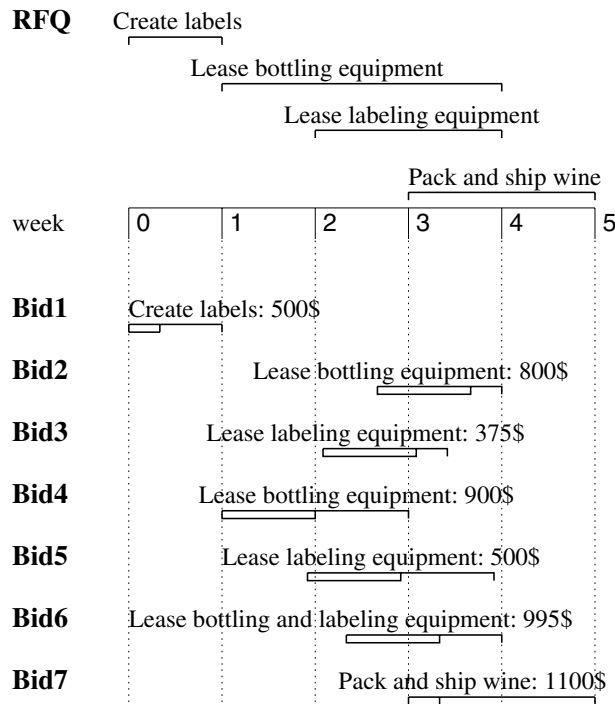


Figure 10. Bid Example

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. The RFQ was composed with a sizable overlap between the labeling and bottling tasks, perhaps in the expectation that there would be large numbers of bidders with a wide variation in lead times.

Bid 2 indicates that bottling could start late in week 2, would take a week, and the supplier is willing to shift that out two more days to accommodate the schedule. Bid 3 shows that labeling could start through week 2, would take a week, and needs to finish in the middle of week 3. These two bids cannot be combined, because labeling must finish when bottling finishes or later, but cannot finish before bottling is finished. Bid 4 shows a more expensive bottling task that could start earlier and also needs a week to finish. This can be combined with Bid 3, with one day of slack to accommodate contingencies. Bid 5 gives a large-enough time window for the labeling task to be combined with Bid 4 but not with Bid 2. Bid 6 shows a special characteristic of the MAGNET system. A supplier can bid for multiple tasks in one bid, even if the RFQ specified those tasks separately. In the example this proves to be cheaper but not very efficient time-wise. The supplier offers to do the two tasks less expensively than others, but the start time is later. Bid 3 has a lower cost but cannot be accepted because it leaves no time for the subsequent labeling task. 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.

Related Work

Markets play an essential role in the economy, and market-based architectures are a popular choice for multiple agents [3, 4, 19, 34, 35, 38]. Most market architectures limit the interactions of agents to manual negotiations, direct agent-to-agent negotiations, or auctions [11, 30, 39].

Existing architectures for multi-agent virtual markets typically rely on the agents themselves to manage the details of their interactions rather than provide explicit facilities and infrastructure for managing multiple negotiation protocols. In the work described here, agents interact with one 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]. Fishmarket, AuctionBot, and eMEDIATOR are examples of multi-agent auction systems [27, 29, 39]. Determining the winners of combinatorial auctions is difficult [23]. Methods for improving the efficiency of combinatorial auctions have been developed in the last few years by Fujishima and Sandholm, among others [13, 29]. Andersson has demonstrated that mixed-integer programming works extremely well even on large problems [1]. Collins and Gini have shown how to use integer programming within the constraints posed by the formulation of problems in MAGNET [6].

Several systems have attempted to organize task-oriented work among multiple agents. Parkes describes an auction-based system for controlling access to shared railroad resources [25]. It uses a mixed-integer approach, with many domain-specific optimizations. Hunsberger and Grosz used combinatorial auctions for the initial commitment decision problem, which is the problem an agent has to solve when deciding whether to join a proposed collaboration [17]. 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 the study by Glass and Grosz, scheduling decisions are made not by the agents, but by a central authority [14]. The central authority has insight on the states and schedules of participating agents, and agents rely on the authority to support their decisions.

Conclusions and Future Work

The MAGNET automated contracting environment is designed to support negotiations among 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 of a customer agent.

The current system has proven very useful for the types of statistical studies pursued so far. Future plans call for more focus on mixed-initiative interaction, a type of work for which the current user interface is too primitive.

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

NOTES

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

REFERENCES

1. Andersson, A.; Tenhunen, M.; and Ygge, F. Integer programming for combinatorial auction winner determination. In *Proceedings of the Fourth International Conference on Multi-Agent Systems*. Boston: IEEE, 2000, pp. 39–46.
2. Babanov, A.; Collins, J.; and Gini, M. Risk and expectations in a-priori time allocation in multi-agent contracting. In *Proceedings of the First International Conference on Autonomous Agents and Multi-Agent Systems*. Bologna: ACM Press, 2002, pp. 53–60.
3. Chavez, A., and Maes, P. Kasbah: An agent marketplace for buying and selling goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*. London: Practical Applications Company, 1996, pp. 75–90.
4. Choi, S.P.M., and Liu, J. A dynamic mechanism for time-constrained trading. In *Proceedings of the Fifth International Conference on Autonomous Agents*. Montreal: ACM Press, 2001, pp. 568–575.
5. Collins, J.; Bilot, C.; Gini, M.; and Mobasher, B. Decision processes in agent-based automated contracting. *IEEE Internet Computing*, 5, 2 (March/April 2001), 61–72.
6. Collins J., and Gini, M. An integer programming formulation of the bid evaluation problem for coordinated tasks. In B. Dietrich and R.V. Vohra (eds.), *Mathematics of the Internet: E-Auction and Markets*, vol. 127 of *IMA Volumes in Mathematics and Its Applications*. New York: Springer-Verlag, 2001, pp. 59–74.
7. Collins, J.; Gini, M.; and Mobasher, B. Multi-agent negotiation using

combinatorial auctions with precedence constraints. Technical Report 02-009. University of Minnesota, Minneapolis, Department of Computer Science and Engineering, February 2002.

8. Collins, J.; Jamison, S.; Gini, M.; and Mobasher, B. Temporal strategies in a multi-agent contracting protocol. In *AAAI-97 Workshop on AI in Electronic Commerce*. Providence, RI: ACM Press, 1997.

9. Collins, J.; Sundareswara, R.; Gini, M.; and Mobasher, B. Bid selection strategies for multi-agent contracting in the presence of scheduling constraints. In A. Moukas, C. Sierra, and F. Ygge (eds.), *Agent Mediated Electronic Commerce II*, vol. LNAI1788. New York: Springer-Verlag, 2000.

10. Collins, J.; Youngdahl, B.; Jamison, S.; Gini, M.; and Mobasher, B. A market architecture for multi-agent contracting. In *Proceedings of the Second International Conference on Autonomous Agents*. Minneapolis: ACM Press, 1998, pp. 285–292.

11. Faratin, P.; Sierra, C.; and Jennings, N.R. Negotiation decision functions for autonomous agents. *International Journal of Robotics and Autonomous Systems*, 24, 3–4 (1997), 159–182.

12. Flanagan, D.; Farley, J.; Crawford, W.; and Magnusson, K. *Java Enterprise in a Nutshell*. Sebastopol, CA: O'Reilly & Associates, 1999.

13. Fujishima, Y.; Leyton-Brown, K.; and Shoham, Y. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth Joint Conference on Artificial Intelligence*. Stockholm: Morgan Kaufmann, 1999, pp. 548–553.

14. Glass, A., and Grosz, B.J. Socially conscious decision-making. In *Proceedings of the Fourth International Conference on Autonomous Agents*. Barcelona: ACM Press, 2000, pp. 217–224.

15. Guttman, R.H.; Moukas, A.G.; and Maes, P. Agent-mediated electronic commerce: A survey. *Knowledge Engineering Review*, 13, 2 (June 1998), 143–152.

16. Hoos, H.H., and Stützle, T. Evaluating Las Vegas algorithms: Pitfalls and remedies. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. San Francisco: Morgan Kaufmann, 1998, pp. 238–245.

17. Hunsberger, L., and Grosz, B.J. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on Multi-Agent Systems*. Boston: IEEE, 2000, pp. 151–158.

18. Kalin, S. Trading places. *CIO Magazine*, 13, 9 (2000), 96.

19. Karacapilidis, N., and Moraitis, P. Intelligent agents for an artificial market system. In *Proceedings of the Fifth International Conference on Autonomous Agents*. Montreal: ACM Press, 2001, pp. 592–599.

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

21. Leyton-Brown, K.; Pearson, M.; and Shoham, Y. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of ACM Conference on Electronic Commerce*. Minneapolis: ACM Press, 2000, pp. 66–76.

22. Loritsch, B. *Developing with Apache Avalon*. Apache Software Foundation, 2001 (jakarta.apache.org/avalon/developing/index.html).

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

24. Nisan, N. Bidding and allocation in combinatorial auctions. In *1999 NWU Microeconomics Workshop*. 1999 (www.cs.huji.ac.il/~noam/mkts.html).
25. Parkes, D.C., and Ungar, L.H. An auction-based method for decentralized train scheduling. In *Proceedings of the Fifth International Conference on Autonomous Agents*. Montreal: ACM Press, 2001, pp. 43–50.
26. Phillips, C., and Meeker, M. The B2B Internet report: Collaborative commerce. *Morgan Stanley Dean Witter*, April 2000.
27. Rodriguez, J.A; Noriega, P; Sierra, C.; and Padget. J. FM 96.5: A Java-based electronic auction house. In *Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*. London: Practical Applications Company, 1997, pp. 207–224.
28. Rosenschein, J.S., and Zlotkin, G. *Rules of Encounter*. Cambridge, MA: MIT Press, 1994.
29. Sandholm, T. An algorithm for winner determination in combinatorial auctions. In *Proceedings of the Sixteenth Joint Conference on Artificial Intelligence*. Stockholm: Morgan Kaufmann, 1999, pp. 524–547.
30. Sandholm, T. Negotiation among self-interested computationally limited agents. Ph.D. dissertation, University of Massachusetts, 1996.
31. Sandholm, T. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135 (2002), 1–54.
32. Sandholm, T., and Lesser, V. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *First International Conference on Multi-Agent Systems*. San Francisco: ACM Press, 1995, pp. 328–335.
33. Sycara, K.; Decker, K.; and Williamson, M. Middle-agents for the Internet. In *Proceedings of the Fifteenth Joint Conference on Artificial Intelligence*. Nagoya, Japan: Morgan Kaufmann, 1997, pp. 578–583.
34. Sycara, K., and Pannu, A.S. The RETSINA multi-agent system: Towards integrating planning, execution, and information gathering. In *Proceedings of the Second International Conference on Autonomous Agents*. Minneapolis: ACM Press, 1998, pp. 350–351.
35. Tsvetovaty, M.; Gini, M; Mobasher, B; and Wieckowski, Z. MAGMA: An agent-based virtual market for electronic commerce. *Journal of Applied Artificial Intelligence*, 11, 6 (1997), 501–524.
36. Varian, H. R. Economic mechanism design for computerized agents. In *USENIX Workshop on Electronic Commerce*, New York, July 1995 (www.usenix.org/publications/library/proceedings/ec95/).
37. Vickrey, W. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16 (1961), 8–37.
38. Wellman, M.P, and Wurman, P.R. Market-aware agents for a multi-agent world. *Robotics and Autonomous Systems*, 24 (1998), 115–125.
39. Wurman, P.R.; Wellman, M.P.; and Walsh, W.E. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second International Conference on Autonomous Agents*. Minneapolis: ACM Press, 1998, pp. 301–308.
40. Zwass, V. Electronic commerce: Structures and issues. *International Journal of Electronic Commerce*, 1, 1 (fall 1996), 3–23.

JOHN COLLINS (jcollins@cs.umn.edu) has returned to academia after a 30-year career in industry. He received a Ph.D. in computer science at the University of Minnesota in June 2002 and is now an assistant professor at the University of Minnesota. His research interests focus on artificial intelligence, especially the decision problems of autonomous agents.

WOLFGANG KETTER (ketter@cs.umn.edu) is pursuing a Ph.D. in computer science at the University of Minnesota and has master's degrees in electrical engineering from the University of Applied Sciences in Trier, Germany, and in software engineering from the University of St. Thomas in St. Paul, Minnesota. His research interests focus on artificial intelligence, machine learning, intelligent agents, software architecture, and software patterns. He is a member of the ACM and AAAI and a founding member of the Twin Cities Patterns Group.

MARIA GINI (gini@cs.umn.edu) is a professor in the department of computer science and engineering at the University of Minnesota. Before that she was a research associate at the Polytechnic of Milan, Italy, and a visiting research associate at Stanford University. Her research interests focus on the use of artificial intelligence to create autonomous entities, such as robots and intelligent software agents. Dr. Gini's major contributions include negotiation strategies for agents, planning with incomplete information, and coordination of multiple robots. She has co-authored more than 100 technical papers and is on the editorial board of *Autonomous Robots* and *Integrated Computer-Aided Engineering*. She is the general chair for the First International Conference on Autonomous Agents and Multi-Agent Systems, 2002, and for the Seventh International Conference on Intelligent Autonomous Systems, 2002.

