A testbed for multi-agent automated contracting

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

Abstract

We are interested in multi-agent contracting, in which customers must solicit the resources and capabilities of other, self-interested agents in order to accomplish their goals. Goals may involve the execution of multi-step plans, in which different steps are contracted out to different suppliers. We have developed a testbed that can generate sets of plans 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.

1 Introduction

Companies are increasingly outsourcing production processes to outside contractors, making supply chains longer and more convoluted. This increased complexity will be compounded by accelerated production schedules which demand tighter integration of all processes. Completing a task requires identifying suppliers with the capability to accomplish different parts of the task. In response to a demand from a customer, first a supply web has to be formed by the suppliers interested in doing one or more of the tasks. Suppliers might, in turn, decide to outsource some of their work. The supply web must then be monitored and coordinated during the time it takes the suppliers to actually execute the orders. Furthermore, it may need to be reconfigured if a supplier is late or defaults on an order. The supply web is not a static entity; it must be dynamically created as customer requests arrive and contracts are signed. The highly distributed nature of the supply web and the inherent uncertainties [Biswas, 1997] make automating the process challenging.

The business-to-business (B2B) 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 supply chains. 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. 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 [Kalin, 2000].

However, the ability to use e-commerce market infrastructures for business-to-business transactions is not yet fully realized. Although business-to-business e-commerce is gaining in popularity, few manufacturers are participating in it, according to a recent survey conducted by the Association of Manufacturers on 2,500 U.S. manufacturers. Only 32% use e-commerce, but only 12% of them procure intermediate parts or sub-assemblies online.

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 third-party value-added services such as security, match-making, and trusted intermediaries.

However, there are no existing frameworks to enable automated negotiation and contracting among manufacturers, part suppliers, and specialized subcontractors. The major challenge in automating negotiations beyond what is currently available comes from the necessity to go beyond simple buying and selling, to incorporate time constraints, to enforce deadlines, to interact with a highly distributed web of suppliers with different capabilities and resources, to interact over long periods of time through the completion of the contracted work, and to deal with failures in contract execution.

To help automate logistics planning among multiple entities such as manufacturers, part suppliers, shippers, and specialized subcontractors, we have proposed a market architecture. called MAGNET (Multi AGent NEgotiation Testbed) [Collins *et al.*, 1998]. MAGNET provides support for a variety of types of transactions, including complex multi-agent contract negotiations. We have implemented a prototype implementation of the bid-evaluation part of MAGNET, and we'll release it to the research community in 2001. Other parts of the MAGNET system will be released over the following two years.

This paper is organized as follows: Section 2 describes the MAGNET architecture and the basic activities and roles of agents in that environment. Section 3 describes our experimental implementation of a Customer agent that we are using to explore agent decision processes. Section 4 describes the implementation of abstract Supplier agents. Section 5 describes related work, and Section 6 concludes and outlines our future plans and open problems.

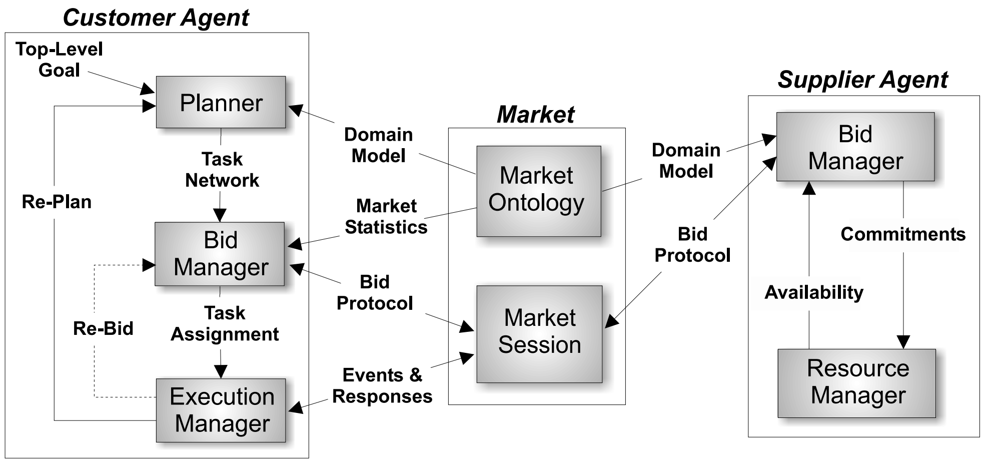

Figure 1: The MAGNET architecture

2 Agents and their Environment

We model the supply-chain formation and management problem using a community of self-interested agents with limited rationality, each representing a business entity or a decision-maker. Agents may act autonomously, or as decision-support tools for human decision-makers. The MAGNET model includes a market infrastructure that connects buyers and sellers, enforces protocol rules, and collects statistics for use in agent planning and evaluation processes.

MAGNET gives an agent the ability to use market mechanisms (auctions, catalogs, timetables, etc.) to discover and commit resources needed to achieve its goals. We assume that agents are heterogeneous and self-interested, and may be acting on behalf of different individuals or commercial entities who have different goals. Although we use auction mechanisms, the problem a MAGNET agent must solve is a combination of a scheduling problem and a combinatorial auction problem.

We need to keep in mind that buyer-supplier relationships depend on factors such as quality, delivery performance, and flexibility as opposed to just cost [Helper, 1991], and these must be taken into account in automated negotiation. To model the uncertainties in the decision process, we use Expected Utility Theory [Biswas, 1997], which describes human economic decision-making. Expected Utility Theory models decision-making under uncertainty by using probabilities and a construct known as a utility curve, $U(W)$, a function that maps a level of "wealth" to a level of utility. Expected Utility Theory can guide the agent in situations in which there is a trade-off between the overall cost of a plan and the likelihood of the plan succeeding. For example, the agent may need to choose between suppliers, some of whom charge a higher price but are more likely to complete the task successfully, while others may be less likely to complete the task on time but may charge less. By computing the expected utility of the scenarios, the agent can choose from among them [Collins *et al.*, 2001].

Agents may fulfill one or both of two roles, as shown in Figure 1. Customer agents pursue their goals by contracting to Supplier agents tasks they themselves cannot or are unwilling to carry out. Supplier agents have resources or services to sell.

Customer agents formulate and present Requests for Quotations (RFQs) to Supplier agents through the market [Collins *et al.*, 1998]. The RFQ specifies a task network that includes task descriptions, a precedence network, and possibly other time constraints. Customer agents attempt to satisfy their goals for the least net cost, where cost factors can include not only bid prices, but also goal completion time and risk factors. More precisely, these agents are attempting to maximize the utility function of some user, as discussed in [Collins *et al.*, 2001].

Supplier agents attempt to maximize the value of their resources by submitting bids in response to those RFQs, specifying what tasks they are willing to undertake, when they are available to perform those tasks, and at what price. Bids may specify combinations of tasks with a single price, and may include prices on individual tasks. Prices for multiple tasks can include a discount or a premium.

A contract is a commitment the agent makes and has legal value [Singh, 1999]. We assume the market has rules to specify the conditions and penalties for breaking a contract that agents agree to when they register with the market.

When signing a contract, an agent takes multiple risks [Collins *et al.*, 2000a]. One or more suppliers might back out of the agreement, or fail to deliver on time, with potentially devastating cascade effects through the remaining parts of the plan. What happens if a part is not delivered on time, so the next step in the processing cannot be done? what if the agent does not succeed in renegotiating its contracts? should it decide to default on them, pay a penalty, and issue a new RFQ, or should it pay a premium for changing the terms of its current contracts? The agent's goal might have a time-dependent value, so at some point it might not be worth trying to accomplish it. An agent must also consider the effects of its actions on its reputation in the market.

3 A Customer Agent

We now focus on the structure and responsibilities of a Customer agent in the MAGNET environment. As indicated in Figure 1, the basic operations are planning, bidding, and plan execution. We have implemented a simple Planner that generates random plans with well-defined statistics, and a Bid Manager with a fairly rich implementation of tools for composing RFQs and selecting bids. The Execution Manager is not yet implemented.

3.1 Planner

The Planner's task is to turn goals into executable plans, represented as task networks. 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 task network is shown in Figure 2.

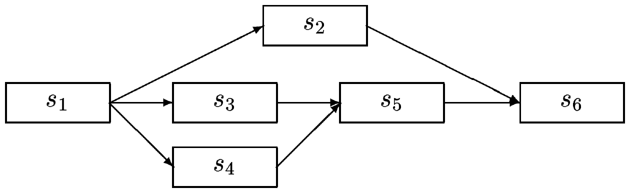

Figure 2: Example task network

The planner in the testbed generates tasks by selecting randomly from a library of task types, and then creates random precedence relations among them. It can also accept predefined plans. 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 definitions of tasks in the Domain Model are shared among the agents. This model includes not only the task definitions, but statistics (presumably collected by the market) about each task type. These statistics include expected duration and variability, expected price and variability, and resource availability data.

3.2 Bid Manager

The Bid Manager is responsible for ensuring that resources are assigned to each of the tasks of a plan, 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.

The Bid Manager must construct and issue a RFQ, evaluate bids, and accept bids in order to carry out its responsibilities. Its high-level structure is shown in Figure 3.

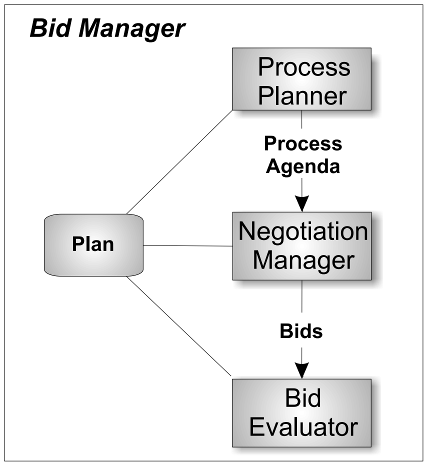

Figure 3: The Bid Manager

Process Planner

The Process Planner creates the high-level agenda for the Bid Manager. Its primary responsibility is to allocate time to negotiation and plan execution. It is 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. For example, it may decide to solicit bids on a high-value but risky approach, and if that fails to fall back on a lower-value but safer alternative. It could also decide to defer taking bids on later tasks until earlier tasks were underway or even completed.

Negotiation Manager

The Negotiation Manager handles the actual bidding process. Its overall job is to attach to the plan a feasible, minimum-cost set of resource assignments. It uses the Bid Evaluator to decide among alternative bid combinations.

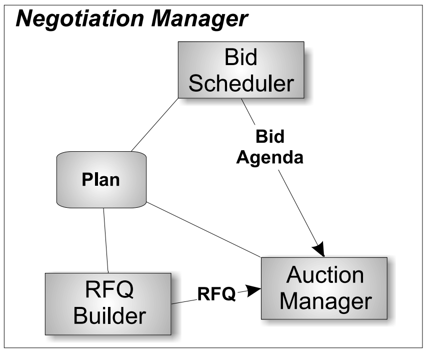

Figure 5: The Negotiation Manager

The Negotiation Manager is further broken down into a set of components, as shown in Figure 5. The Bid Scheduler assembles a schedule for the bidding process, possibly subdividing the time allocated by the Process Planner, and adds items to the agenda to drive the Auction Manager. Dividing

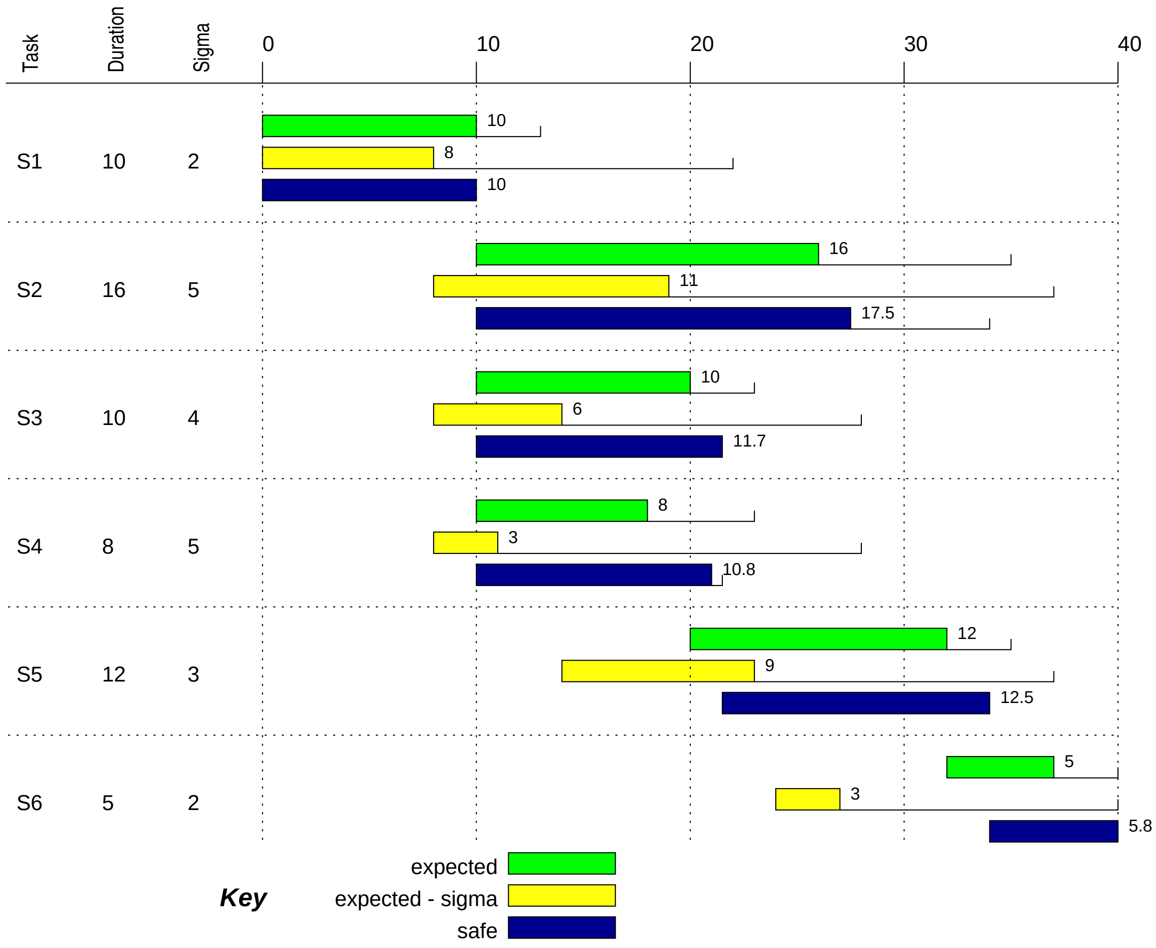

Figure 4: A timeline showing a set of tasks and alternative time allocations.

the bidding process into multiple phases can be an important strategy to reduce the level of uncertainty in the plan.

We have implemented several different versions of the Bid Scheduler to experiment with different strategies. Ultimately it will be up to the Process Planner to decide which strategy (or strategies) to use, and configure the Bid Scheduler accordingly through its agenda entries.

Before bids can be solicited in a market, an RFQ must be composed. The RFQ is a structure that contains some portion of the plan data (tasks and precedence relations) as determined by the Bid Scheduler, along with a set of scheduling constraints. The primary role of the RFQ Builder is to determine those scheduling constraints. Information comes from several sources:

- From the Planner, we have a set of tasks and their precedence constraints. This information is contained in the plan.
- From the Market, we have statistical information about duration and variability for the different task types. We also have information about resource availability and the number of vendors who are likely to bid on tasks of this type.
- From the Process Planner, we have the overall schedule for the execution of the plan.
- From the Bid Scheduler, we know which tasks are to be advertised for bid in the current RFQ.

The RFQ Builder has to produce an RFQ that will solicit the most advantageous set of bids possible. The approach we take is to find a balance between giving maximum flexibility to suppliers, ensuring that the resulting bids will combine feasibly, and ensuring that the job will be completed by the deadline. We do this by setting early-start and late-finish times in the RFQ for each task, as follows.

The RFQ Builder constructs an initial schedule using the expected duration data, and sets the initial time windows using the Critical Path algorithm [Hillier and Lieberman, 1990]. The Critical Path algorithm walks the directed graph of tasks and precedence constraints, forward to compute the earliest start times for each task, and then backward from the goal time to compute the latest finish and latest start times for each task. The minimum duration of the entire plan is called the *makespan* of the plan. The difference between the goal time and the latest early finish time is called the *total slack* of the plan.

In Figure 4 the gray bars show the expected durations of the tasks in the task network illustrated in Figure 2. The overall slack chosen in the schedule is 5 units for a 35 unit makespan, or about 14%. In the figure, the light bars show durations that are 1 standard deviation below the expected values, while the dark bars show an alternative formulation in which the total expected slack is apportioned among the tasks on the critical path to produce an RFQ that has no overlap among adjacent tasks. The former is likely to produce more bids, and potentially lower-cost bids, because of the additional scheduling flexibility offered to suppliers. The latter will reduce the effort the Customer agent must expend to compose a feasible plan, assuming bids are received to cover all the tasks. Specifying larger time windows without lengthening the overall schedule will increases the chance that bids from different suppliers will fail to satisfy the precedence relationships.

The optimal setting of RFQ time windows requires detailed knowledge of other factors such as likely numbers of bidders and likely constraint tightness on the resources needed to carry out the tasks. Since the Customer agent cannot know this data precisely, we must use approximations. We assume the market will maintain statistics to support this, but there is clearly more work to be done in this area.

The Auction Manager interacts with the Market and/or other agents to solicit bids. Different versions of the Auction Manager can be implemented to interact with different market environments. We have a version that uses a MAGNET market to solicit bids, and one that uses a set of in-process simulated Supplier agents directly to generate bids for testing purposes. The latter version is useful for doing large statistical studies where throughput is a critical factor.

Bid Evaluator

A Bid Evaluator is a search engine that takes a plan and a set of bids, and attempts to find an optimal or near-optimal mapping of bids to tasks, respecting temporal constraints. It must do this within the period of time allocated by the Process Planner, which may have been subdivided by the Bid Scheduler.

If the agent is not able to process the bids fast enough, it might miss good deals and spend all its time processing bids instead of awarding contracts. A hostile agent could submit a huge number of bids to prevent another agent from accomplishing its task, although the market could be configured to limit this. Transaction or entry fees could be added to reduce the number of bids, but since they could discourage agents from submitting bids, they have to be designed carefully [Anderson *et al.*, 1999].

We have implemented two evaluators. One is based on Integer Programming [Collins and Gini, 2001], and the other is a highly modular Simulated Annealing (SA) search engine [Collins *et al.*, 2000b].

The Integer Programming (IP) solver operates in two phases. The first phase generates basic bid-compatibility constraints, and then walks all paths of length 2 or greater in the precedence network, across all compatible bid combinations, to discover feasibility constraints. These are then packaged up and sent off to an external IP solver.

The core part of the simulated-annealing engine is similar to the one described in [Reeves, 1993]. Starting with a plan and a set of bids, we generate and evaluate bid mappings until one of several stopping conditions holds. These include failure to find improvement for a configurable number of iterations, expiration of the deliberation time limit, and lack of mappings that have any untried expansions.

In our experiments, we have noticed that there is a significant variation in the amount of time taken by the IP method to terminate. We are exploring how to combine the IP optimizer with the simulated annealing engine to produce an any-time behavior. This is important because MAGNET agents might have a strictly limited amount of time to make bid-award decisions.

3.3 Execution Manager

The Execution Manager is responsible for overseeing execution of the plan as contracted, and making decisions on how to respond when events do not proceed as expected. It receives the task assignments from the Bid Manager and, through the market, receives updates on plan execution from contracted vendors. It maintains a time map with tasks, vendor commitments, and temporal constraints among tasks. For each event, it must decide whether to respond, and if so, whether to respond directly to a particular vendor, whether to re-bid a portion of the plan, or whether to re-plan and re-bid one or more subgoals of the plan.

All the activities of the Execution Manager revolve around the maintenance of the time map. The time map [Dean and McDermott, 1987] can be thought of visually as a Gantt chart, decorated with contract data and temporal constraints among tasks. For each task the time map records an early start, a late finish time, the committed start time and duration, and the set of precedence constraints.

As time passes and the execution of the plan proceeds, the Execution Manager works in conjunction with the market to drive the plan to completion. In general, the market session is responsible for releasing tasks to the suppliers when their prerequisites are satisfied, and for assessing decommitment penalties when the parties fail to satisfy their commitments. In the process, the session forwards to the Execution Manager notifications of task release and task completion events. The Execution Manager is then responsible for making decisions and taking appropriate action in response to those notifications.

The Execution Manager has not yet been implemented.

4 Supplier Agents

Since our primary interest has been in the workings of the Customer agent, our prototype Supplier agents are fairly simple-minded entities. They receive RFQs, and they respond by submitting bids. They do not maintain resource schedules, and they have no persistent identity.

The basic structure is shown in Figure 6. Each of these 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

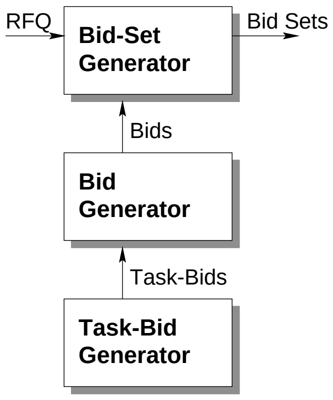

Figure 6: Simple Supplier Simulation

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. An obvious extension would be to generate role-based bids in the sense of [Hunsberger and Grosz, 2000].

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 Related Work

Markets play an essential role in the economy, and market-based architectures are a popular choice for multiple agents (see, for instance, [Chavez and Maes, 1996; Sycara and Pannu, 1998; Wellman and Wurman, 1998]. Most market architectures limit the interactions of agents to manual negotiations, direct agent-to-agent negotiation [Sandholm, 1996; Faratin *et al.*, 1997], or various types of auctions [Wurman *et al.*, 1998].

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 the predominant mechanism for agent-mediated electronic commerce [Wurman *et al.*, 1998; Sandholm, 1999]. Methods for improving the efficiency of combinatorial auctions have been developed, among others, by Sandholm [Sandholm, 1999] and Fujishima [Fujishjima *et al.*, 1999]. Mixed integer programming has been demonstrated to work extremely well even on large problems by Andersson [Andersson *et al.*, 2000]. Walsh et al [Walsh *et al.*, 2000] study combinatorial auctions for problems in supply chain, but ignore time constraints. However, none of those algorithms has been applied to situations with time constraints of the type and complexity we are concerned with. MAGNET agents have to deal with multiple resources. Customer agents use the bidding process as a way of obtaining the use of resources of supplier agents. Customer agents have also to ensure the scheduling feasibility of the bids they accept, and must evaluate risk as well as simple schedule feasibility.

MAGNET agents are similar to the agents used for collaborative planning by [Hunsberger and Grosz, 2000], where 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 agents 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.

6 Conclusions and Future Work

The MAGNET testbed is a prototype implementation of a Customer agent, along with simulated 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 to be very useful for the 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.

Some domains, notably the International Shipping domain we are currently studying in collaboration with an Import-Export firm, will require an enhanced plan representation to deal with the fact that alternate routes or shipping modalities may be acceptable.

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 [Leyton-Brown *et al.*, 2000] 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 needed in the MAGNET environment.

Acknowledgments

Partial support for this research is gratefully acknowledged from the National Science Foundation under award NSF/IIS-0084202.

References

[Anderson *et al.*, 1999] Axel Anderson, Ionel Birgean, and Jeffrey K. McKie-Mason. Bilateral negotiation with fees. In *First IAC Workshop on Internet Based Negotiation Technologies*, March 1999.

[Andersson *et al.*, 2000] 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. IEEE Computer Society Press, July 2000.

[Biswas, 1997] Tapan Biswas. *Decision-Making Under Uncertainty*. St. Martin's Press, Inc., 1997.

[Chavez and Maes, 1996] 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.

[Collins and Gini, 2001] John Collins and Maria Gini. Bid evaluation for coordinated tasks: an integer programming formulation. In *IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms*, August 2001.

[Collins *et al.*, 1998] John Collins, Maxim Tsvetovat, Bamshad Mobasher, and Maria Gini. Magnet: A multi-agent contracting system for plan execution. In *Proc. of SIGMAN*, pages 63–68. AAAI Press, August 1998.

[Collins *et al.*, 2000a] John Collins, Corey Bilot, Maria Gini, and Bamshad Mobasher. Mixed-initiative decision support in agent-based automated contracting. In *Proc. of the Fourth Int'l Conf. on Autonomous Agents*, pages 247–254, June 2000.

[Collins *et al.*, 2000b] 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.

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

[Dean and McDermott, 1987] Thomas L. Dean and Drew V. McDermott. Temporal data base management. *Artificial Intelligence*, 32:1–55, 1987.

[Faratin *et al.*, 1997] 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.

[Fujishjima *et al.*, 1999] 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.

[Helper, 1991] S. Helper. How much has really changed between us manufacturers and their suppliers. *Sloan Management Review*, 32(4):15–28, 1991.

[Hillier and Lieberman, 1990] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, 1990.

[Hunsberger and Grosz, 2000] 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.

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

[Leyton-Brown *et al.*, 2000] 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.

[Reeves, 1993] Colin R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, New York, NY, 1993.

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

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

[Singh, 1999] Munindar P. Singh. An ontology for commitments in multiagent systems. towards a unification of normative concepts. *AI and Law*, 7(1):97–113, 1999.

[Sycara and Pannu, 1998] 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.

[Walsh *et al.*, 2000] William E. Walsh, Michael Wellman, and Fredrik Ygge. Combinatorial auctions for supply chain formation. In *Proc. of ACM Conf on Electronic Commerce (EC'00)*, October 2000.

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

[Wurman *et al.*, 1998] 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.