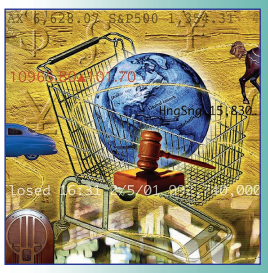

Decision Processes in Agent-Based Automated Contracting

The Magnet system meets many of the challenges of modeling decision making for customer agents in automated contract negotiation.

John Collins, Corey Bilot, and Maria Gini
University of Minnesota

Bamshad Mobasher
DePaul University

Business-to-business e-commerce is expanding rapidly, letting manufacturers both broaden their customer base and increase their pool of potential suppliers. However, negotiating supplier contracts for the multiple components that often make up a single product is a complicated process. Because component parts must be assembled and time dependencies often exist among operations, scheduling is a major challenge.

Currently, there are no existing mechanisms or frameworks for automated negotiation and contracting among manufacturers, part suppliers, and specialized subcontractors. As the sidebar, "Related Work on Market-Based Architectures," describes, current e-commerce systems typically rely instead on either fixed-price catalogs or auctions. However, such systems focus only on cost, which is just one factor in the complicated buyer-supplier relationship. For meaningful contract negotiations, systems must take into account other key factors, including schedule, quality, delivery performance, and flexibility.[1]

The University of Minnesota has developed the Multi-Agent Negotiation Testbed (Magnet) system,[2] which is designed to support multiple agents in negotiating contracts for tasks with temporal and precedence constraints. Magnet has been under development since early 1998. Currently, its distributed market infrastructure is in place, as are the principal elements of its customer agents, which we focus on here.

Magnet's customer agents have two key tasks. First, they must determine the specific contents of a Request for Quote (RFQ), which they send out at the start of negotiations to solicit supplier bids. The content of the RFQ determines how much time suppliers have to submit bids, and constrains the start and end times for the tasks. Next, once bids have been received, the customer agents review and

1089-7801/01/$10.00 ©2001 IEEE

Related Work on Market-Based Architectures

Markets play an essential role in the economy, and market-based architectures are a popular choice for multiple agents.[1-3] However, most existing market architectures limit agent interactions to manual negotiations, direct agent-to-agent negotiation,[4,5] or various types of auctions.[6]

Auctions are becoming the predominant mechanism for agent-mediated electronic commerce.[7] AuctionBot[6] and eMEDIATOR[8] are among the most well-known examples of multiagent auction systems. Determining the winners of combinatorial auctions is difficult. Dynamic programming[9] works for small sets of bids, but does not scale, and it imposes significant restrictions on the bids. Algorithms such as Bidtree[8] and CASS[10] have been proposed to reduce the search complexity, but price is their only criterion for bid selection. Our bids include a time window for each task, which ties bid selection to scheduling.

Also, existing architectures for multiagent virtual markets typically rely on the agents themselves to manage their interaction details 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.

Most work in supply-chain management is limited to strict hierarchical modeling of the decision-making process. This is inadequate for distributed supply-chains, since each organization is self-interested, not cooperative. A notable exception is the Mascot system.[11] Mascot agents coordinate scheduling with the user, but there is no explicit notion of payments or contracts, and the criteria for accepting or rejecting a bid are not explicitly stated. The system's major objective is to show the advantage of using lateral coordination policies that focus on optimizing schedules locally through temporal-constraint exchange.[12] Our objective is to negotiate contracts with suppliers that optimize the customer's utility.

References

1. K. Sycara and A.S. Pannu, "The Retsina Multiagent System: Toward Integrating Planning, Execution, and Information Gathering," *Proc. Second Int'l Conf. on Autonomous Agents*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 350-351.
2. M.P. Wellman and P.R. Wurman, "Market-Aware Agents for a Multiagent World," *Robotics and Autonomous Systems*, vol. 24, 1998, pp. 115-125.
3. M. Tsvetovatyy et al., "Magma: An Agent-Based Virtual Market for Electronic Commerce," *J. Applied Artificial Intelligence*, vol. 11, no. 6, 1997, pp. 501-524.
4. T.W. Sandholm, "Negotiation Among Self-Interested Computationally Limited Agents," doctoral thesis, Dept. of Computer Science, University of Massachusetts, Amherst, 1996.
5. P. Faratin, C. Sierra, and N.R. Jennings, "Negotiation Decision Functions for Autonomous Agents," *Int'l J. Robotics and Autonomous Systems*, vol. 24, no. 3-4, 1997, pp. 159-182.
6. P.R. Wurman, M.P. Wellman, and W.E. Walsh, "The Michigan Internet Auction Bot: A Configurable Auction Server for Human and Software Agents," *Second Int'l Conf. on Autonomous Agents*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 301-308.
7. R.H. Guttman, A.G. Moukas, and P. Maes, "Agent-Mediated Electronic Commerce: A Survey," *Knowledge Engineering Review*, vol. 13, no. 2, June 1998, pp. 143-152.
8. T. Sandholm, "Approaches to Winner Determination in Combinatorial Auctions," *Decision Support Systems*, vol. 28, no. 1-2, 2000, pp. 165-176.
9. M.H. Rothkopf, A. Pekeuc, and R.M. Harstad, "Computationally Manageable Combinatorial Auctions," *Management Science*, vol. 44, no. 8, 1998, pp. 1131-1147.
10. Y. Fujishjima, K. Leyton-Brown, and Y. Shoham, "Taming the Computational Complexity of Combinatorial Auctions: Optimal and Approximate Approaches," *Proc. 16th Joint Conf. on Artificial Intelligence*, Morgan Kaufman, San Francisco, 1999.
11. N.M. Sadeh et al., "Mascot: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling," *Workshop on Agent-Based Decision Support in Managing the Internet-Enabled Supply-Chain/Agents '99*.
12. D. Kjenstad, "Coordinated Supply Chain Scheduling," doctoral dissertation, Dept. of Production and Quality Engineering, Norwegian University of Science and Technology, Trondheim, Norway, 1998.

select bids. They do this using an algorithm we developed that operates on combinatorial problems with multiple tasks and attributes including cost, supplier reputation and reliability, time constraints, and other risk factors.

To model decision making in the uncertain environment of contract negotiation, we use the Expected Utility Theory. We describe that theory here, followed by more detailed descriptions of the customer agents' key decision-making processes and our search algorithm. First, we offer an overview of the Magnet environment.

The Magnet System

Figure 1 shows an overview of the Magnet system. The system's market infrastructure supplies customer agents with statistics to help with decision making, connects customer and supplier agents, and helps the agents communicate.

Market Infrastructure

Magnet mediates agent interactions through the market infrastructure,[2] which provides a domain ontology, a contracting protocol, and authentication services. It also tracks the requests, commitments, and progress toward task completion.

The market's *ontology* describes the types of tasks or goods available in the market. Along with item descriptions, the ontology contains statistics, including details such as how many suppliers typically bid on an item and how long a task typically takes. The market also keeps a *registry* of sup-

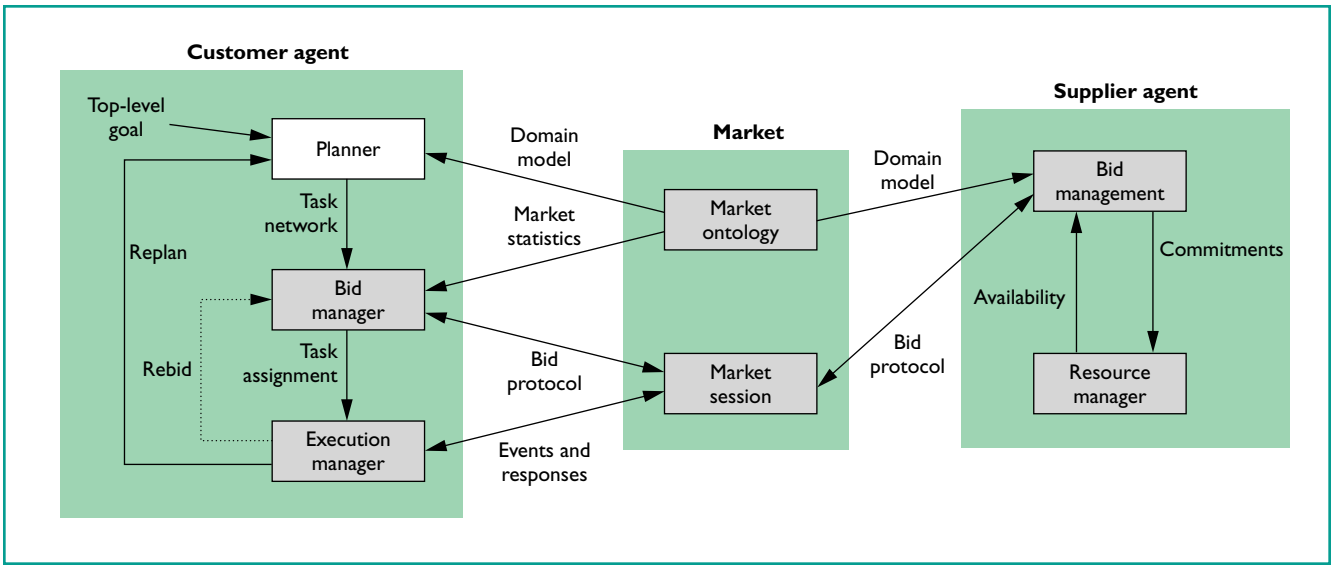

Figure 1. Overview of the Magnet system. Magnet mediates interactions among customer and supplier agents and provides customer agents with market statistics.

pliers who typically participate in market activities, and it maintains performance statistics to help customer agents (or the agent's human user) make decisions. Market services are dynamically delivered to participating agents through a *market session*. The session encapsulates a market transaction and also serves as a persistent repository for the current state of the transaction throughout the life of the contract.

Agent Interactions

In Magnet, agents function in three basic phases: planning, negotiation, and execution monitoring. Here, we focus on decisions made in the bidding cycle, which occurs in the negotiation phase. We distinguish between two agent roles: the customer and the supplier. A *customer* is an agent who has a goal and needs resources outside its direct control to achieve it. The goal's value can vary over time. A *supplier* is an agent who has resources or services and, in response to a customer agent's RFQ, might offer to provide them to the customer for a specified price over a specified time period.

The bidding cycle consists of several steps:

- The customer agent issues an RFQ, which includes a specification of each task and a set of precedence relations among tasks. For each task, the agent specifies a time window that gives the task's earliest start time and latest end time.
- Suppliers send bids, which include the total cost, estimates on how long each task will take, and a time window for startup on each task.
- When the customer agent awards a bid, it pays the supplier a deposit (refundable only if the supplier fails) and specifies a start time within the supplier's time window.
- When the supplier completes a task, the customer agent pays the outstanding balance, minus the deposit.
- If the supplier fails to complete a task, it forfeits the price and returns the deposit to the customer agent (the customer might also levy a penalty for nonperformance, though we ignore this possible complication here).

Once the customer agent awards bids, a secondary protocol lets agents negotiate schedule changes. This helps prevent outright failure and reduces risk for both parties, at the cost of complicating the agents' behavioral requirements during plan execution.

An Example

Assume Acme Widgets asks its agent to find the resources to prepare a trade-show display within two weeks. Acme's sales department has estimated that if the display is ready on time, it can book sales during the show that will earn the company US$10,000 in profit (minus the cost of the display).

The agent has three tasks to accomplish, and there is uncertainty as to whether the suppliers can deliver on time (we ignore the uncertainty in the profit number). Figure 2 (next page) shows the customer agent's financial situation as the plan progresses. The customer agent pays deposits on all tasks (d_1, d_2, and d_3) when it awards bids at the end of the bidding cycle. Once suppliers complete

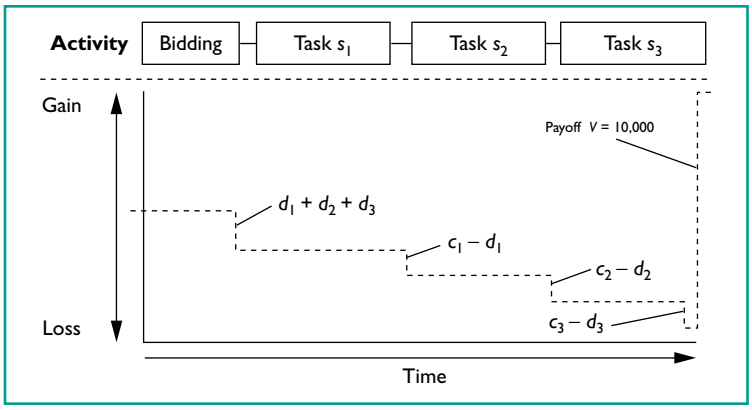

Figure 2. The financial position of the Acme Widget customer agent. The agent's financial position changes as it awards bids, makes deposits and payments, and accrues value for completed tasks.

the tasks, the customer agent makes payment on each task, minus the deposit ($c_1 - d_1$, for example). If a task, n, is not completed by the supplier, the supplier agent returns the deposit d_n to the customer agent. When the plan is complete, the value V of the goal accrues.

Assuming that the customer agent receives multiple bids that specify different costs, deposits, and time parameters, the goal of the decision process is to award a combination of bids that maximizes the expected utility—that is, the subjective value to the user—at the project's end. Because that utility involves some probability of both loss and gain, we must assess the risk posture of the person or organization on whose behalf the agent is acting. To do this, we model the agent's decision making based on *marginal expected utility*: the expected change in the decision maker's overall utility due to some decision.

Expected Utility Theory

To produce plans that are acceptable to users, an automated agent must be able to handle decision making in an uncertain environment. To model this, we use the Expected Utility Theory (EUT),[3] which models decision making under uncertainty using probabilities and a utility curve, $U(W)$, that maps a level of "wealth" to a level of utility.

According to EUT, decision makers faced with an opportunity consisting of a set of n wealth-based outcomes will calculate the expected utility over the set of outcomes:

$$E(U) = \sum_{i=1}^{n} U(W_i)p_i \qquad (1)$$

where p_i is the probability of outcome i, and W_i is the decision maker's resulting wealth if outcome i is realized. Basically, decision makers weigh the utility of each outcome within the opportunity. To make a decision, decision makers compare the expected utility, $E(U)$, to their current utility, $U(W_0)$, where W_0 represents their current wealth. If the opportunity's expected utility exceeds the current utility, the decision maker will pursue the opportunity. Similarly, when faced with multiple opportunities, decision makers can decide which (if any) they will pursue by comparing the expected utilities of each opportunity with their current utility.

Computing Utility

EUT guides our customer agents in situations where there is a trade-off between a plan's overall cost and its likelihood of success. For example, the agent might need to choose between suppliers who are very reliable but charge a high price, and others who charge less but are less likely to complete the task. By computing the expected utility of different scenarios, the agent can choose from among them.

To compute the expected utility for a plan being executed by several supplier agents, we treat the plan as a set of ordered task-completion events. Each event has a probability of succeeding, and, at the time of each event, the customer agent must pay some supplier agent. When the last task is complete, the customer agent gains the benefit of plan completion. If any task fails to complete, we assume the customer agent will abandon the plan and forfeit the deposits paid to downstream suppliers (suppliers who have yet to begin processing their respective tasks). For n tasks, this gives

$$E(U) = U(W_0) + \sum_{i=1}^{n}\left(M(-z_i)(1-p_i)\prod_{j=1}^{i-1} p_j \right) + M(V)\prod_{j=1}^{n} p_j \qquad (2)$$

where $M(x)$ is the change in utility due to a financial gain of x, the p_i are the success probabilities of the successive tasks, the z_i are the cumulative "debits" resulting from each task completion (the d_i and c_i in Figure 2), and V is the net "credit" that accrues on plan completion.

We call the function $M(x)$ the *marginal expected utility* of a gain of x. This notion lets us simplify thinking about situations in which we

are concerned only about changes in wealth due to some decision. Denoting with ΔW_i the change in wealth relative to W_0, for outcome i, Equation 1 becomes

$$E(U) = U(W_0) + \sum_{i=1}^{n} M(\Delta W_i) p_i \quad (3)$$

For our purposes, $M(\Delta W)$ is really a qualitative concept; we do not expect to compute it exactly. Although many functions have been proposed,[4] little is known about how to elicit preferences from users or an organization that will yield an accurate utility function. Instead, we recognize its existence and its general shape. To compute values, we will bound $M(\Delta W)$ with a linear function as an upper bound (risk neutral), which is fairly close to reality for small gambles.

We define successful plan execution as "completed by the deadline" and successful task completion as "completed without violating the plan's temporal constraints." A task can be completed successfully even if it is not finished within the bidder's proposed time frame, as long as the schedule has sufficient slack to absorb the overrun. A plan fails if it is completed after its deadline, and we ignore any of the completed work's residual value to the customer. These definitions form our starting point; we now extend our analysis to more complicated cases.

Application Example

Given the example of Figure 2, the expected utility $E(U)$ for Acme Widgets is:

$$\begin{aligned} E(U) \quad &= U(W_0) + M(-d_2 - d_3)(1 - p_1) \\ &+ M(-c_1 - d_3) p_1 (1 - p_2) \\ &+ M(-c_1 - c_2) p_1 p_2 (1 - p_3) \\ &+ M(-c_1 - c_2 - c_3) p_1 p_2 p_3 \\ &+ M(V) p_1 p_2 p_3 \end{aligned} \quad (4)$$

where V is the US$10,000 profit, d_n is the deposit paid when bid n is awarded, c_n is the price paid when task n is completed, and p_n is the probability that task n will be completed by the agreed-upon deadline.

To compute the expression in Equation 4, the customer agent estimates, for each task and supplier, the probabilities that the tasks will be completed on time. The agent then computes its marginal utility $M(x)$ for each possible outcome. Although statistical information about tasks and suppliers is available from the market, the utility function depends on the user. Users who trust their agents to make autonomous decisions can specify an analytical form for their own utility function. Other users might let the agent do some computations and present alternatives, and then make the final decision themselves.

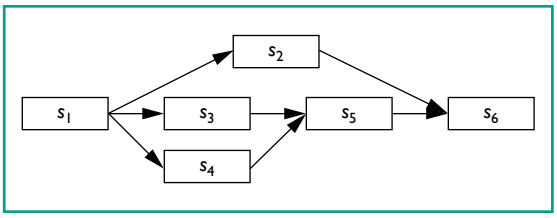

Figure 3. A branching task network. Tasks can be scheduled in qualitatively different ways, yielding different marginal utility values.

When a set of tasks includes potentially parallel activities, the analysis is more complex. Different possible schedules might have different marginal utility values, depending on the relative costs and success probabilities of the individual tasks. Once a task starts, the user is liable for its full cost at completion, regardless of whether the plan is abandoned due to a failure on some other branch of the plan.

Consider the plan in Figure 3, for example. Depending on expected task durations, it might be possible to complete task s_2 before starting s_5, or to delay s_2's start until either or both of s_3 and s_4 are complete. It might even be possible to serialize s_3 and s_4 in either order if the plan has sufficient slack. Each of these orderings will yield a different $E(U)$ value. For example, if s_2 is expensive relative to tasks s_3 and s_4, it should be delayed until after both s_3 and s_4 have completed, if possible. This reduces the number of times the cost of s_2 appears in Equation 2.

How sensitive is a risk estimate to uncertainty about supplier reliability? One way to answer this is to look at the partial derivatives of $E(U)$ with respect to p_k in Equation 2:

$$\begin{aligned} \frac{\partial E(U)}{\partial p_k} \quad &= -M(-d_k) \prod_{j=1}^{k-1} p_j \\ &+ \sum_{i=k+1}^{n} \left(M(-z_i)(1 - p_i) \frac{\prod_{j=1}^{i-1} p_j}{p_k} \right) \\ &+ M(V) \frac{\prod_{i=1}^{n} p_i}{p_k}. \end{aligned} \quad (5)$$

If the upper bound of any derivative is negative,

we are better off abandoning the plan because a higher success probability for that element will lead to a worse overall outcome. If any of the derivatives are especially large and the confidence in the corresponding probability estimates are low, the customer agent should warn the user about the uncertainty and its potential impact, and consider adding slack to the schedule to allow for recovery.

Because of the temporal constraints among tasks, one supplier's failure to accomplish a task does not necessarily mean the overall goal will fail; recovery is possible if another supplier assumes the task without invalidating the schedule. However, permitting such flexibility complicates bid selection. Selecting the cheapest bids and the tightest possible schedule is likely to create a brittle combination, and is thus not always the best choice.

We can reduce risk by consolidating tasks with single suppliers who can bid on "packages" composed of task subsets from the RFQ. In general, we're better off accepting such packages from a risk standpoint, assuming that the supplier will let us pay for the whole package when it's complete. In some cases, we might be willing to pay a premium over the individual task prices to reduce risk. This is most advantageous toward the end of the plan. To see why, we restructure Equation 4 to consolidate tasks 2 and 3, without changing the costs and deposit amounts:

$$\begin{aligned} E(U) &= U(W_0) + M(-d_2 - d_3)(1 - p_1) \\ &\quad + M(-c_1)p_1(-p_{23}) \\ &\quad + M(-c_1 - c_2 - c_3 + V)p_1 p_{23}. \end{aligned} \tag{6}$$

The fourth term from Equation 4 is now missing, and the third term represents a smaller outlay. Also, the last term's probability factor might be larger if the supplier's probability of delivering on the consolidated task (p_{23}) is nearly the same as delivering on a single task (p_2 or p_3).

Risk and Utility

For agents to make appropriate autonomous decisions, we must estimate how risks affect the marginal expected utility. First, however, we must determine which elements contribute to risk, and how to estimate and decrease risk.

To support risk evaluation, Magnet's market maintains several types of data on each supplier and task type.

- *Performance to commitment* (P_c): the ratio of successes to attempts, where the task was completed within the promised duration. It does not include bid awards that the customer agent abandoned before starting the task.
- *Performance with overruns* (P_l): the percentage of attempts completed successfully but late.
- *Overrun duration* (t_l): the lateness of completions with respect to bid durations.

For each factor, the market maintains the sample mean, sample size, and variance. This lets agents compute a confidence interval and issue risk reports to the user in a standard form: "There is an n percent probability that your risk is less than x."

To estimate risk, we compute a lower bound of the risk R. This is the absolute value of the negative part of the expected value computation, not including the payoff for plan completion:

$$R = \left| \sum_{i=1}^{n} \left(-z_i (1 - p_i) \prod_{j=1}^{i-1} p_j \right) \right|. \tag{7}$$

A risk estimate must be made in the context of a particular schedule for two reasons. First, specific start times determine the schedule slack available for recovery if a supplier misses a deadline. Second, scheduling decisions affect the ordering of parallel tasks.

Clearly, minimizing risk by adjusting task start times is a nonlinear combinatorial optimization problem, since the amount of slack available for failure recovery influences individual completion probabilities. We have several options, such as:

- Estimate marginal completion probabilities given additional time. We can estimate performance with overrun and overrun duration data, and assume that the improvement in completion probability is linear in time.
- Initialize each task's start time to be as early as possible, consistent with precedence constraints and bid specifications. This is a heuristic based on our observation that later tasks tend to be riskier because of the larger outlays later in the plan.

Another option is to present users with the choices, risk data, and sensitivities from Equation 5, and let them decide.

Decision Process in Customer Agents

The customer agent takes utility and risk into

account at two points during the bidding cycle: during RFQ composition, when it specifies the tasks and time windows; and during bid evaluation, when it makes decisions about which bids to accept based on such things as price, timing, and supplier and data reliability.

Composing the RFQ

The customer agent's goal in composing the RFQ is to maximize the plan's expected marginal utility at completion time. The agent can't schedule tasks directly; instead, it issues an RFQ to garner a set of bids from potential suppliers. It then composes a schedule based on these bids.

The agent generates the RFQ from a task network that consists of a set of tasks, the temporal constraints among them, and possibly nonzero delays between tasks to cover communication and transportation delays. Operations in task networks need not be linearized with respect to time, since multiple agents can execute operations in parallel. Figure 3 shows an example task network.

Time allocation. When composing an RFQ, the agent must decide how to allocate time in three ways:

- between bidding and execution of the plan;
- within the bidding cycle, between suppliers and the customer agent; and
- among the tasks in the plan.

Supplier agents need time to evaluate their resource availability and compose bids, and the customer agent needs time to evaluate bids. When we allocate additional time to the bidding process, we reduce the available execution time and increase the risk of plan failure. If we decrease time for the bidding process, supplier and customer agents have less time to consider their options.

Our principal strategy in time allocation between the customer agent and supplier agents is to allocate just enough time for the customer agent to make a decision. The reason is that suppliers will likely either not bid or will raise prices if they have to reserve resources while speculating on outstanding bids. Also, any extra time the customer agent spends on decision making reduces the available execution time. For further guidance on this time-allocation problem, see our detailed characterization of the bid-evaluation process.[5]

Scheduling. The RFQ includes early start and late finish times for each task. The customer agent must set these "time windows" prior to soliciting bids. Once the bidding cycle concludes, the agent composes the bids into a feasible schedule. But first, suppliers must return bids that satisfy the plan's precedence constraints. The customer agent attempts to influence the availability of attractive bids by setting the relative time allocation among the tasks and the extent to which the time windows of adjacent tasks—those connected by precedence relations—can overlap.

To make these decisions, the customer agent uses three kinds of task data from the Magnet market's ontology:

- the number of bidders likely to submit bids,
- the expected duration, and
- the amount of variability in the duration data.

The customer agent builds an initial schedule using the expected duration data. It then sets the initial time windows using the Critical Path algorithm.[6] This algorithm first 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 possible finish and start times for each task. The entire plan's minimum duration is called the *makespan;* the difference between the goal time and the latest early finish time is called the plan's *total slack*.

In Figure 4 (next page), the green bars show the expected duration of the tasks in Figure 3's task network. The customer agent selected a total slack of 5 units for a 35-unit makespan, or about 14 percent. The yellow bars' durations are one standard deviation below the expected values, while the blue bars show an alternative formulation in which the total slack is apportioned among the tasks on the critical path to produce an RFQ with no overlap among adjacent tasks. The yellow-bar option is likely to produce more bids and potentially lower-cost bids because it offers suppliers additional scheduling flexibility. The blue option will make it easier for the customer to compose a feasible plan, assuming bids are received to cover all the tasks. In this case, the bid evaluation problem is reduced to a version of the combinatorial auction-winner-determination problem, for which there are reasonably efficient optimal solution methods.[7-9]

Discussion. We can make an additional adjustment using bidder population and variability data. We're still investigating the detailed relationships between the bidder count and variability data, and the optimal adjustments that must be made to the

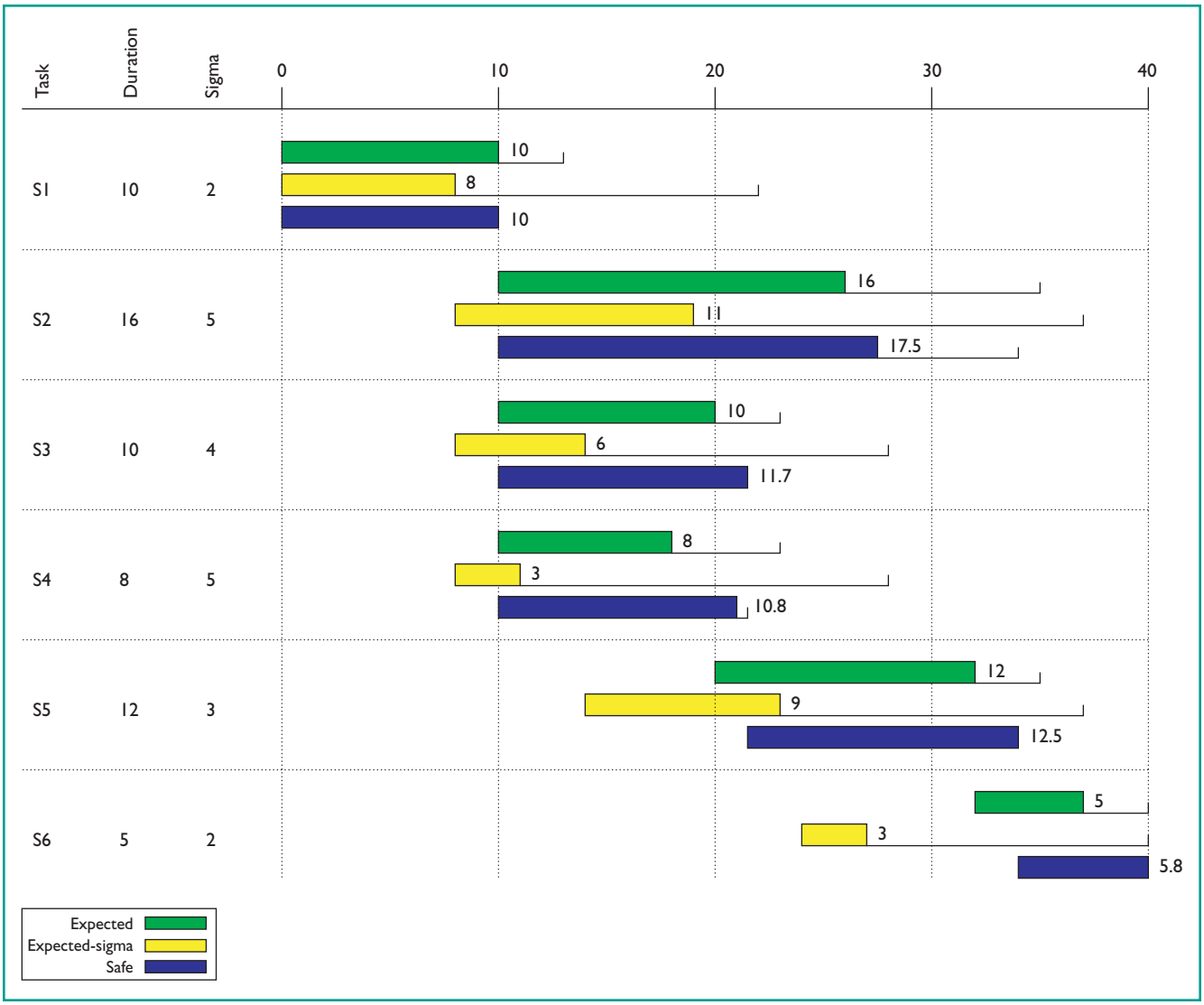

Figure 4. A timeline showing a set of tasks and alternative time allocations. The colored bars show how the critical path algorithm can be used to reduce task durations and increase time windows.

RFQ schedule. Our current approach is to increase the relative time allocation when duration data is more variable, and increase the overlap when there are many bidders.

There is a tension between issuing an RFQ that guarantees the feasibility of any plan constructed with the resulting bids and issuing one that will solicit the maximum number of bids. We assume that supplier bids are based on an evaluation of their current resource commitments, and therefore that larger time windows will result in more bids. Suppliers will know that more time flexibility in their bids will give them a competitive advantage.

Evaluating and Awarding Bids

Once suppliers send their bids, the customer agent's goal is to find and schedule a bid combination that maximizes $E(U)$, then award the bids. The challenge is to find a good mapping of bids to tasks, then schedule the bids so that there is a low risk of unrecoverable failure.[5] A "good" mapping covers all tasks (each task is mapped to exactly one bid), satisfies temporal constraints, and is relatively low-cost. Also, in our system, bids are exclusive: When a single supplier agent submits multiple bids, only one can be accepted (though other bid semantics are possible).[8]

Search algorithm. To decide on bids, our customer agents use our adaptive, anytime search algorithm shown in Figure 5. We based the algorithm on a simulated annealing[10] framework with modular selec-

1. **Initialize search**:
   - 1.1 **Pre-process bids**: For each task, generate a list of the bids that include the task and its average bid price.
   - 1.2 **Coverage test**: If any task lacks a bid, exit (coverage cannot be achieved).
   - 1.3 **Single bid test**: If any task has a single bid, the bid must be part of any solution. The bid might contain one or more tasks. Create node(s) that map the bid, compute their value V, and add them to the queue.
   - 1.4 **Initialize queue**: If there are no singletons, create a node mapping all tasks to no bids and add it to the queue.
   - 1.5 **Set initial annealing temperature**.

2. **While not** time out **and** improving **do**:
   - 2.1 **Select a node N for expansion**: Select a random number $R = V_{min} - T \ln(1-r)(V_{max} - V_{min})$, where
     - r is a random number uniformly distributed between 0 and 1,
     - T is the current annealing temperature, and
     - nodes in the queue are sorted by increasing values; V_{min} is the value of the first node, V_{max} the value of the last node.

     Choose node N as the last node in the queue for which $V_N \leq R$.
   - 2.2 **Select a bid B**:
     Discard all bids that appear on the tabu list of N.
     Discard all bids that appear on the tried list of N.
     Choose a bid according to the current bid selection policy.
   - 2.3 **Expand node N with bid B, producing node N'**: For each task mapped by bid B that is already mapped in N to some other bid B', remove bid B'.
     If B' is a singleton bid (see 1.3 above), abandon the expansion.
     Add B to the expansions-tried list of node N.
     Copy the tabu list of node N into node N' and add bid B in front.
     Truncate the tabu list in node N' to the tabu size limit.
   - 2.4 **Evaluate node N'**:
     $V_N = Cost_N + Risk_N + Feas_N + Cov_N$ where
     - $Cost_N$ is the sum of bid prices (uncovered tasks are assigned an average price),
     - $Risk_N$ is the expected cost of recovering from plan failure times a weighting factor,
     - $Feas_N$ is the weighted sum of schedule overlaps, and
     - Cov_N is the number of tasks that are not mapped to a bid, times a weighting factor.
   - 2.5 Update best-node statistics.
   - 2.6 Adjust the annealing temperature T.

3. **Return** the best node found.

Figure 5. The search algorithm for bid selection. We based the adaptive, anytime search algorithm on a simulated annealing framework.

tors and evaluators. Simulated-annealing is characterized by two elements: the annealing temperature T, which is periodically reduced by a factor ε; and the stochastic node-selection procedure. Nodes represent sets of bids mapped to their respective tasks; we keep them in an ordered queue of fixed maximum length, sorted by their value. Step 2.4 in the algorithm shows how we compute the node value, which is a combination of factors and includes a risk component. Intuitively, risk increases whenever an agent accepts a bid that increases either (a) the probability of missing the goal deadline or (b) the probability of missing the latest possible start time for some subsequent task.

As Figure 5 shows, we maintain a tabu list and a tried list on each node. The *tabu list* is a list of bids recently used in the history of node expansions leading to the target node. By prohibiting expansion by bids on the tabu list, we prevent

Table 1. Solutions found.

Selector	Small-bid problem	Large-bid problem
Random	2	2
Cov	3	0
FeasCov	2	0
Combined	6	1

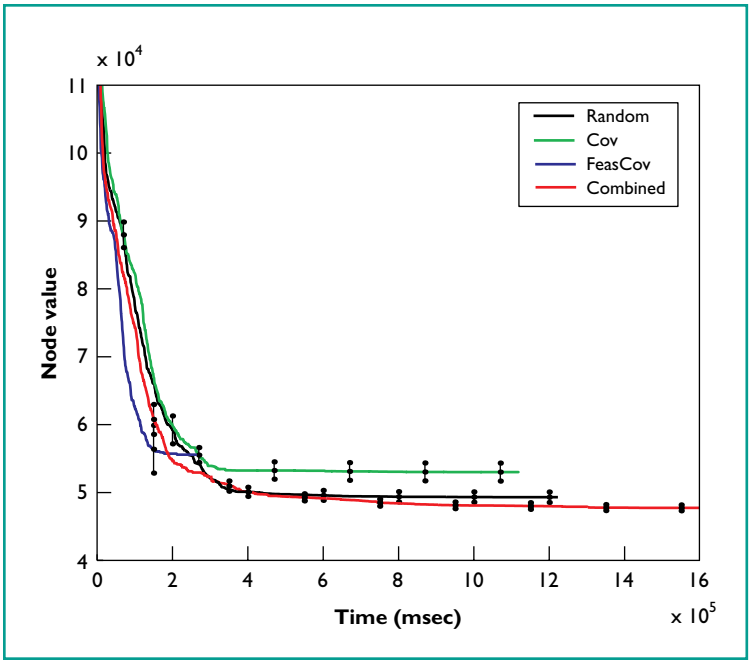

Figure 6. Improvement curves for the small-bid problem. Averages are shown for 20 runs.

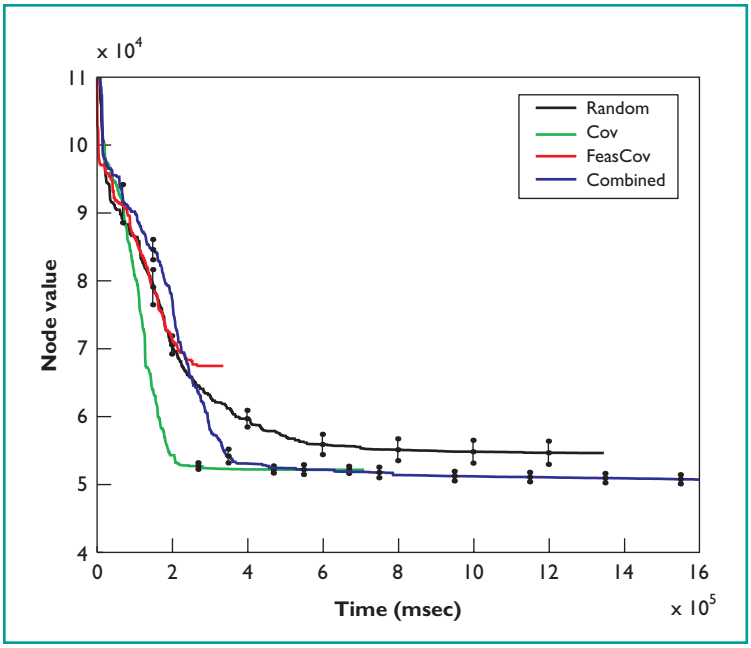

Figure 7. Improvement curves for the large-bid problem. Averages are shown for 20 runs.

short cycles in the search space and thereby encourage exploratory behavior. The *tried list* is simply the list of bids that have already been used to expand the node in question; we don't try the same expansion twice.

We have implemented and tested several bid-selection methods (see Step 2.2 in Figure 5):

- *Random Bid:* Choose a bid at random, and attempt to add it to the node.
- *Coverage Improvement:* Choose a bid that covers a task that is not mapped in the node.
- *Feasibility Improvement:* Scan mapping to find tasks with negative slack. Of those, move tasks constrained by their bids (rather than by predecessors or successors) to relieve negative slack. Sort tasks by their potential to reduce infeasibility and save them. Choose the untried bid with the highest potential to reduce infeasibility. Note that with this selector, it's possible to choose a bid that introduces other infeasibilities.
- *Cost Improvement:* Choose the (untried) bid that is responsible for the maximum positive deviation from the average (or expected) price, and replace it with a lower bid that covers at least the task with the highest positive cost deviation. This selector is useful only if you have average or expected cost data on each task.

Clearly, significant complexity costs are associated with both the *Feasibility Improvement* and *Cost Improvement* methods.

To generate focused improvements for a given node, you can compose these selectors into different configurations. In our experiments, we used the following selectors:

- *Random* (described above).
- *FeasCov:* If the node is infeasible, use the *Feasibility Improvement* selector; otherwise, if it is not fully covered, use the *Coverage Improvement* selector; otherwise, use the *Random* selector.
- *CostFeasCov:* If the cost of the node's covered portion is above average, attempt to reduce its cost; otherwise, use the *FeasCov* selector.
- *Combined:* Run the *Random* selector until it stops producing improvement, then switch to *Feasibility Improvement* until it stops producing improvement, then switch back to *Random*, then to *Coverage Improvement*, then back to *Random*, then to *CostFeasCov*, and finally back to *Random*.

We performed numerous studies of the algorithm's performance.[5] In particular, we compared the *Random* bid selector with the more focused bid selectors. Our interests are in both the ability to find solutions and the rate of improvement. The

latter is important for setting time limits in an anytime search.

To probe a range of problem complexity factors, we ran the *Random*, *Coverage*, *FeasCov*, and *Combined* selectors against two problem types of the same size, but with different complexity levels. Both sets had 50 tasks and 100 bidders, all of which are generated with the same random number sequences. Total slack was 10 percent, and we set task durations to 60 percent of their expected values to relax the RFQ time windows. The average bid size (the number of tasks included in a bid) was 5 in the small-bid problem and 15 in the large-bid problem. Table 1 shows the number of solutions that the different selectors found for each problem. The actual number of such solutions is not known.

Figures 6 and 7 show the improvement curves for the four bid selectors on the small- and large-bid problems, respectively. Error bars show $\sigma/\sqrt{n}$ where σ is the standard deviation across runs, and n is the number of runs. The *Combined* selector clearly gives the best overall performance, both in terms of consistency and solution quality.

Conclusions and Future Work

Many business-to-business interactions are moving to the Internet and to auction-based pricing mechanisms. Although some businesses have achieved success trading individual goods, many problems remain in trading bundles of goods or coordinated tasks. If an agent is to act on behalf of a human decision-maker in such a market, it must be able to evaluate risk factors in ways that the person will find reasonable. We believe Expected Utility Theory offers a good framework for doing this.

A major problem in markets with coordinated tasks is determining the winning bids. It is not sufficient to pick the lowest prices. Our stochastic approach, based on simulated annealing, offers good, anytime performance. Our experimental results show that excessive focus on improvement leads to faster improvement early on, at the price of a lowered likelihood of eventually finding a solution that satisfies all constraints. We've reported on additional experiments elsewhere.[5] Among our other studies, we compared our simulated annealing search method to a purely systematic search. As expected, the systematic approach reliably finds the optimum bid assignment, at the cost of exponential scalability and anytime controllability. We are currently studying the possibility of using integer programming for bid evaluation.

Many challenging problems remain before a fully automated agent can function effectively in this environment. A vocabulary must be developed to allow unambiguous specification of tasks and bids. Many domains require more complex relationships among tasks than we currently support, such as staged delivery over time, or task splitting across multiple suppliers. On the supplier side, we must develop an understanding of how to use the bidding process to maximize the value of supplier resources.

Our bid-evaluation test framework will be released to the research community in 2001. Other parts of the Magnet system will be released over the following two years. Further information is available at http://www.cs.umn.edu/Research/airvl/magnet.

Acknowledgments

We gratefully acknowledge partial support for our research from the National Science Foundation, award NSF/IIS-0084202.

References

1. J. Collins et al., "Mixed-Initiative Decision Support in Agent-Based Automated Contracting," *Proc. Fourth Int'l Conf. on Autonomous Agents*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 247-254.
2. J. Collins et al., "A Market Architecture for Multi-Agent Contracting," *Proc. Second Int'l Conf. on Autonomous Agents*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 285-292.
3. T. Biswas, *Decision-Making Under Uncertainty*, St. Martin's Press, New York, 1997.
4. L. Eeckhoudt and C. Gollier, *Risk Evaluation, Management, and Sharing*, Harvester Wheatsheaf, Hemel Hempstead, Hertfordshire, UK, 1995.
5. J. Collins et al., "Bid Selection Strategies for Multi-Agent Contracting in the Presence of Scheduling Constraints," in *Agent Mediated Electronic Commerce*, Vol. LNAI1788, Springer Verlag, New York, 2000.
6. F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research*, McGraw Hill, New York, 1990.
7. A. Andersson, M. Tenhunen, and F. Ygge, "Integer Programming for Combinatorial Auction Winner Determination," *Proc. 4th Int'l Conf on Multi-Agent Systems*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 39-46.
8. N. Nisan, "Bidding and Allocation in Combinatorial Auctions," *Proc. ACM Conf. on Electronic Commerce* (EC'00), ACM Press, New York, 2000, pp. 1-12.
9. T. Sandholm, "Approaches to Winner Determination in Combinatorial Auctions," *Decision Support Systems*, vol. 28, no. 1-2, 2000, pp. 165-176.

10. C.R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley & Sons, New York, 1993.

John Collins is currently finishing a PhD in computer science at the University of Minnesota. His research interests are in decision-theoretic areas of artificial intelligence, including planning and scheduling, resource management, and decision support. He is the principal developer of the Magnet system. Collins is a member of the IEEE Computer Society, ACM, AAAI, and INFORMS.

Corey Bilot currently works in the corporate database systems department at 3M and is finishing a masters' degree in computer science at the University of Minnesota. He received a bachelors degree in economics from the University of Wisconsin-Eau Claire in 1991. Bilot was a casualty actuary for four years and received his associateship in the Casualty Actuarial Society in 1995 before leaving the profession to pursue computer science.

Maria Gini is a professor in the Department of Computer Science and Engineering at the University of Minnesota. Her research focuses on methods to distribute intelligence among robots or software agents. Her major contributions include multiple agents, motion planning, and navigation. Gini has coauthored more than 100 technical papers. She is on the editorial boards of the journals *Autonomous Robots* and *Integrated Computer-Aided Engineering*. She is the general co-chair for the International Conference on Autonomous Agents, 2002.

Bamshad Mobasher is an assistant professor of computer science at DePaul University, where he also serves as the director of the Center for Web Intelligence. Previously, he was on the faculty at the University of Minnesota. He received his PhD from Iowa State University in 1994. He is active in several research areas, including Web data mining, autonomous software agents, and multiagent systems for e-commerce. He has authored more than 30 papers and book chapters, and has served on program committees of numerous conferences and workshops.

Readers can contact the authors at gini@cs.umn.edu.