Decision Processes in Agent-Based Automated Contracting

John Collins, Corey Bilot, Maria Gini
Department of Computer Science and Engineering
University of Minnesota
Bamshad Mobasher
School of Computer Science, Telecommunications, and Information Systems
DePaul University

Abstract

We analyze the criteria that a customer agent in agent-based automated contracting would use in making decisions during the bidding cycle. We use the University of Minnesota's MAGNET automated-contracting environment as a framework for this analysis. Two decisions must be made by a customer agent during this process: deciding the composition of the Request for Quotes, and evaluating and awarding bids. We show how principles from Expected Utility Theory can be applied and show how the market infrastructure can support agent decision-making by gathering and analyzing statistical data on activities in the market. We present a highly tunable anytime search algorithm for bid selection that incorporates cost, task coverage, temporal feasibility, and risk estimation into a simulated annealing framework. We report on an experimental evaluation using a set of increasingly informed search heuristics.

Keywords: automated negotiation, multiattribute combinatorial auctions, search algorithms.

1 Introduction

The business-to-business e-commerce market is expanding rapidly. 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 suppliers, while for sellers marketplaces provide access to much broader customer bases.

Finding potential suppliers is only a step in the process of producing goods. When component parts have to be assembled and there are time dependencies among the operations, scheduling becomes a major factor. A schedule with slack between tasks is less risky than a tight schedule, but in made-to-order products speed is the essence and taking extra time might prevent a supplier from getting a contract. After contracts have been signed, there is one more complication. A late delivery of a component part might produce a cascade of devastating effects on the rest of the contracted work. This has to be considered during contract negotiation. The supply web need not be static; it may be dynamically created as customer requests arrive and contracts are signed.

Yet, there are no existing mechanisms or frameworks to enable automated negotiation and contracting among manufacturers, part suppliers, and specialized subcontractors. Current e-commerce systems typically rely on either fixed-price catalogs or auctions, but companies usually work with prequalified suppliers, and buyer-supplier relationships depend on factors such as quality, delivery performance, and flexibility as opposed to just cost. These factors must be taken into account while negotiating contracts [3].

The major challenge in extending automated negotiations beyond what is currently available comes from the need to go beyond simple buying and selling, to incorporate and enforce time constraints or deadlines, to interact with a highly distributed community of suppliers, to interact over long periods of time through the completion of the contracted work, and to deal with failures in contract execution.

The University of Minnesota's MAGNET (Multi AGent NEgotiation Testbed) system [5] is a testbed for multi-agent contract negotiation for tasks that have temporal and precedence constraints. Agent interactions in MAGNET are mediated through an independent market infrastructure which, among other services, provides a domain ontology, a contracting protocol, authentication services, and tracks the requests, commitments, and progress towards task completion among the agents.

In this paper we focus in particular on two decision processes that take place during the bidding cycle:

1. The first decision is to determine the specific contents of a Request for Quotes (RFQ) at the start of a bidding cycle. This decision determines how much time suppliers are given to submit bids, and it determines an approximate schedule by setting limits on the start and end times for each individual task.
2. At the conclusion of the bidding cycle, the agent must decide whether to award bids, and which bids to award. We present an algorithm that operates on combinatorial problems with multiple tasks and multiple attributes, such as cost, supplier reputation and reliability, time constraints, and other risk factors.

2 Agent Interactions in MAGNET

In MAGNET agents find each other and communicate using the services provided by the market infrastructure [5].

An agent in MAGNET has three basic functions: planning, negotiation, and execution monitoring. Within the scope of a negotiation, we distinguish between two agent *roles*, the *customer* and the *supplier*. A customer is an agent who has a goal to satisfy, and needs resources outside its direct control in order to achieve its goal. The goal may have a *value* that varies over time. A supplier is an agent who has resources and who, in response to a *Request for Quotes* (RFQ), may offer to provide resources or services, for specified prices, over specified time periods. Figure 1 shows the relationships among customer agents, supplier agents, and the market.

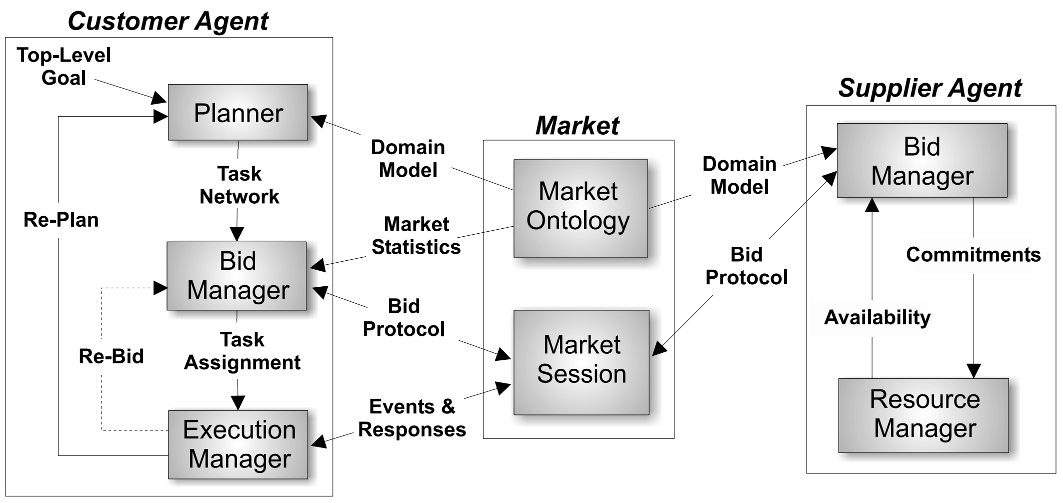

Figure 1: Interactions of customer and supplier agents with the market

The market contains an *Ontology* that describes the types of tasks or goods that the market deals in. Each description not only describes the item, but also contains statistics, including details like the number of suppliers that typically will bid on the item, and how long the task typically takes. The market also keeps a *Registry* of suppliers that have expressed an interest in participating in market activities, and maintains performance statistics that customers can use in their decision processes. A *market session* is the vehicle through which market services are delivered dynamically to participating agents. It serves as an encapsulation for a transaction in the market, as well as a persistent repository for the current state of the transaction, throughout the life of the contract.

The interaction between customer and supplier agents starts with a Request for Quotes (RFQ) issued by the customer, followed by a set of bids submitted by interested suppliers, and concludes with a set of bid awards issued by the customer. After contracts are awarded, the execution phase starts. For the purpose of this paper, we are primarily concerned with the decisions the customer must make during the bidding cycle, and we will not consider plan execution,

The exchange of messages between agents is designed to simplify negotiations without loss of generality. It is modeled after the leveled commitment protocol proposed by Sandohlm [16].

- The customer issues an RFQ which includes a specification of each task, and a set of precedence relations among tasks. For each task, a time window is specified giving the earliest time the task can start and the latest time the task can end.
- A supplier's bid includes a price for the task, a portion of the price required to be paid as a non-refundable deposit at the time the bid is awarded, an estimated duration for the task, and a time window within which the task can be started.
- When the customer awards a bid, it must pay to the supplier the deposit and specify the actual time, within the supplier's specified time window, at which it wishes to begin the task.
- When the supplier completes a task, the customer must pay the remainder of the price, beyond the deposit, as specified in the awarded bid.

- If the supplier fails to complete a task, the price is forfeit and the deposit must be returned to the customer. A penalty may also be levied for non-performance, but we ignore this complication at this point.

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

3 A Motivating Example

Assume Acme Widgets asks its agent to find the resources to prepare a display for a trade show in two weeks. Acme's sales department estimates they can book sales during the show that will result in $10 000 in profit (not including the cost of the display), if the display is ready in time.

There are three tasks to be done, and there is some uncertainty in the abilities of the suppliers to deliver on time. We ignore the uncertainty in the profit number. Figure 2 shows the financial situation of the customer agent as the plan progresses. Deposits on all tasks ($d_1 + d_2 + d_3$) are paid when bids are awarded at the conclusion of the bidding cycle, and payments for each of the tasks, i.e. the agreed cost minus the deposit, ($c_1 - d_1$ etc.) are made as the tasks are completed. Note that if a task n is not completed by the supplier, then the deposit d_n is returned to the customer. When the plan is complete, the value V of the goal accrues.

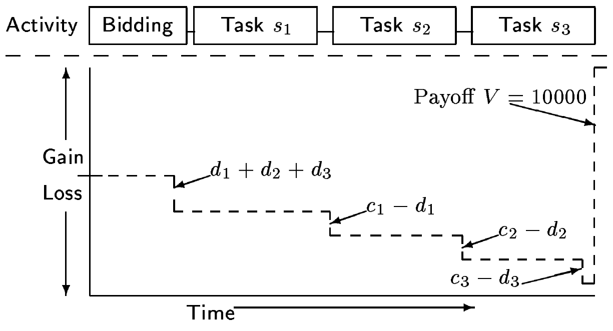

Figure 2: Financial position during project

Assuming that the customer has received multiple bids that specify different costs, deposits, and time parameters, the customer's decision process will attempt to award a combination of bids that maximizes its expected utility at the completion of the project. Because that expected utility involves some probability of loss as well as a probability of gain, we must deal with the risk posture of the person or organization on whose behalf the agent is acting. To do this, we will use the notion of *marginal expected utility*, which is the expected change in a decision maker's overall utility due to some decision.

We are interested in automating (completely or in part) the process of decision making, so we need to specify more precisely what we want to compute before explaining how we expect to perform the computation.

4 Expected Utility

If an automated agent is to produce plans that are acceptable to a human decision-maker, our agent must have the ability to handle decision-making in an environment of uncertainty. To model this concept, we will use Expected Utility Theory (EUT) [2], which describes human economic decision-making. EUT 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. According to EUT, a decision-maker who is faced with an opportunity consisting of a set of n wealth-based outcomes will calculate the expected utility over the set of outcomes:

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

where p_i is the probability of outcome i, and W_i is the resulting wealth of the decision-maker if outcome i is realized. Stated another way, the decision-maker weighs the utility of each outcome within the opportunity. To make her decision, the decision-maker compares this expected utility, $E(U)$, to her current utility, $U(W_0)$, where W_0 represents her current wealth. If the opportunity's expected utility, $E(U)$, exceeds her current utility, $U(W_0)$, she will pursue the opportunity. Similarly, a decision-maker faced with multiple opportunities can decide which (if any) she will pursue by comparing the expected utilities of the opportunities and her current level of utility.

We use EUT to 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; others of whom are less likely to complete the task but who will charge less. By computing the expected utility of the scenarios, the agent can choose from among them.

More specifically, in order to compute the customer's expected utility of a plan being executed by a set of 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 must pay some supplier. After completion of the last task, the customer gains the benefit of plan completion. If any task fails to complete, we assume the customer will abandon the plan, and forfeit the deposits paid to downstream suppliers (suppliers who have not yet begun processing their respective tasks). For n tasks, this gives

$$\begin{aligned} E(U) &= U(W_0) + \sum_{i=1}^{n} \left(M(-z_i)(1-p_i) \prod_{j=1}^{i-1} p_j \right) \\ &\quad + M(V) \prod_{j=1}^{n} p_j \end{aligned} \tag{2}$$

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 of 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. We introduce this notion to simplify thinking about situations where we are only concerned 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

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

It is important to understand that, for our purposes, $M(\Delta W)$ is really a qualitative concept, not a function we expect to be able to compute exactly. Many functions have been proposed [6], but little is known about how to elicit preferences from a human user or 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). This is fairly close to reality for small gambles in any case.

We define successful plan execution as "completed by the deadline," and we define successful completion of a task as "completed without violating temporal constraints in the plan." Note that a task can be completed successfully even if it is not finished within the duration promised by the bidder, as long as the schedule has sufficient slack to absorb the overrun. If a plan is completed after its deadline, it has failed, and we ignore any residual value of completed work to the customer. We plan on extending our analysis to more complicated cases, but we will use these definitions as a starting point.

In the example of Figure 2, the expected utility $E(U)$ for Acme Widgets resulting from the endeavor is:

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

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

To compute the expression in Equation 4, the customer agent needs to estimate, for each task and supplier, the probabilities that the tasks will be completed on time. It then must compute its marginal utility $M(x)$ for each possible outcome. Statistical information about tasks and suppliers is available from the market, but the utility function depends on the customer. A human decision-maker who trusts her agent to make autonomous decisions will specify an analytical form for her own utility function. A decision-maker who prefers to make decisions directly will use the agent to do some computations and present alternatives and will keep to herself the final decision.

If the set of tasks includes potentially parallel activities, the analysis becomes more complex. Different possible schedules may have different marginal utility values, depending on the relative costs and success probabilities of the individual tasks. Once a task starts, the customer is liable for its full cost at completion, regardless of whether in the meantime the plan as a whole has been abandoned due to a failure on some other branch of the plan.

As an example, consider the plan in Figure 3. In this plan, depending on expected task durations, it may be possible to complete task s_2 before starting s_5, or to delay the start of s_2 to after completion of either or both of s_3 and s_4. It may 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 value of $E(U)$. For example, if s_2 is expensive relative to tasks s_3 and s_4, then it should be delayed until after both s_3 and s_4 have been completed, if possible. This will reduce the number of terms in Equation 2 in which the cost of s_2 appears.

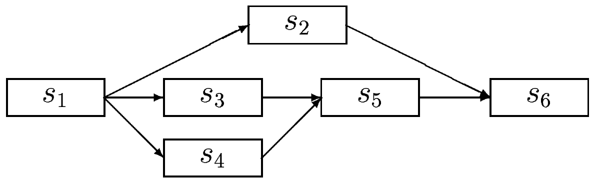

Figure 3: Branching task network

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

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

If the upper bounds of any of the derivatives are negative, then the message is that we are better off abandoning the plan, because a higher probability of success for that element leads to a worse overall outcome. If any of the derivatives are especially large, and the confidence in the corresponding probability estimates are low, then the decision maker should be warned of the uncertainty and its potential impact, and the agent should consider adding slack to the schedule in order to allow for recovery.

Because of the temporal constraints between tasks, failure to accomplish a task does not necessarily mean failure of the goal. Recovery might be possible, provided that whenever a supplier decommits there are other suppliers willing to do the task and that there is sufficient time to recover without invalidating the rest of the schedule. This complicates the selection of which bids to accept. The lowest cost combination of bids and the tightest schedule achievable is not necessarily the preferable schedule because it is more likely to be brittle.

Risk can be reduced by consolidating tasks with single suppliers. Suppliers can bid on "packages" composed of subsets of tasks from the RFQ. In general, the customer is better off from a risk standpoint if it takes these packages, assuming that the supplier is willing to be paid for the whole package at the time of its completion. In some cases, the customer may be willing to pay a premium over the individual task prices in order to reduce risk. The advantage is greater toward the end of the plan than near the beginning. To see why this is so, we restructure Equation 4 to consolidate tasks 2 and 3, without changing the costs and deposit amounts:

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

Notice that the fourth term from Equation 4 is missing, and the third term represents a smaller outlay. Also, the probability factor in the last term may be larger, assuming that the probability of the supplier delivering on the consolidated task (p_{23}) is nearly the same as delivering on a single task (p_2 or p_3).

To allow agents to make appropriate autonomous decisions, we need a way of estimating how risks affect the marginal expected utility. Before doing that, we need to examine what elements contribute to risk, what strategies are available to decrease risk, and how risk should be estimated.

The MAGNET market maintains several types of data on each supplier and on each task type in support of risk evaluation.

- *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 were abandoned by the customer before the task was started.
- *Performance with overruns* P_l – The proportion of attempts that were completed late.
- *Overrun duration* t_l – The lateness of late completions, with respect to bid durations.

For each of these factors, the market maintains the sample mean, sample size, and variance. This permits agents to compute a confidence interval, in order to be able to make reports to the user of the form "there is a $n\%$ probability that your risk is less than x."

To estimate the 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:

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

These risk estimates cannot be produced without a fixed schedule. This is because specific start times determine the schedule slack available for recovery if a supplier misses a deadline, and because they affect the ordering of parallel tasks.

Clearly, minimizing risk by adjusting the start times of tasks is a non-linear combinatorial optimization problem, since the individual completion probabilities can be influenced by the amount of slack available to recover from failure. Different approximations can be used, such as:

1. Estimate marginal completion probabilities given additional time. We can use the performance with overrun and overrun duration data, and assume that the improvement in completion probability is linear in time.
2. Initialize the start times of each task to be as early as possible, consistent with precedence constraints and bid specifications. This is a heuristic driven by the observation that later tasks tend to be riskier because of the larger outlays later in the plan.
3. Present the user with the choices, the risk data, and the sensitivities from Equation 5 and let her decide.

5 Decision Processes of the Customer Agent

As described earlier, there are two points in the bidding cycle where the customer agent needs to take utility and risk into account. One is during the composition of the RFQ, where the tasks and time windows are specified. The other is during evaluation of bids, when decisions need to be made regarding which bids to accept. Factors that must be considered include the price, the time window specified in the bid, the reliability of the supplier, and the confidence we have in the reliability data.

5.1 Composing the Request For Quotes

When the customer composes the RFQ, the goal is to maximize the expected marginal utility of the plan at completion time. The customer can't schedule tasks directly; instead, it must issue a RFQ that is likely to garner a set of bids from potential suppliers, that will then be composed into a schedule.

The RFQ is generated from a task network, which consists of a set of tasks, the temporal constraints among them, and possibly nonzero delays between tasks, to cover communication and transportation delays. An example task network was shown earlier in Figure 3. The operations in the task network do not need to be linearized with respect to time, since operations can be executed in parallel by multiple agents.

There are three time-related factors in the RFQ that can affect the successful outcome:

1. the allocation of time between bidding and execution,
2. the allocation of time within the bidding cycle between suppliers and the customer,
3. the time constraints on each task.

Suppliers need time to evaluate their resource availability and compose bids, and the customer needs time to evaluate bids. If more time is allocated to the bidding process, then the time available for execution will be reduced, and the risk of plan failure increased. If less time is allocated to the bidding process, then either the suppliers, the customer, or both will have less time to consider their options.

The customer's principal strategy in allocating time between itself and suppliers is to allocate just enough time to itself to make a decision, and no more. This is because 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 spent in customer decision-making reduces the time available for plan execution. We have attempted to characterize our bid-evaluation process [4] in order to provide guidance for this time allocation problem.

The RFQ includes early start and late finish times for each task. Setting these "time windows" is the second major decision the customer needs to make prior to soliciting bids. At the conclusion of the bidding cycle, the agent will need to compose the bids into a feasible schedule. The ability to do that depends on suppliers returning bids that satisfy precedence constraints in the plan. There are two decisions here: the relative allocation of time among the tasks, and the extent to which the time windows of adjacent tasks (connected by precedence relations) are allowed to overlap.

The MAGNET market provides three kinds of data about each task type in its ontology that can be used to make these decisions: (1) the number of bidders that are likely to submit bids, (2) the expected duration, and (3) the amount of variability in the duration data.

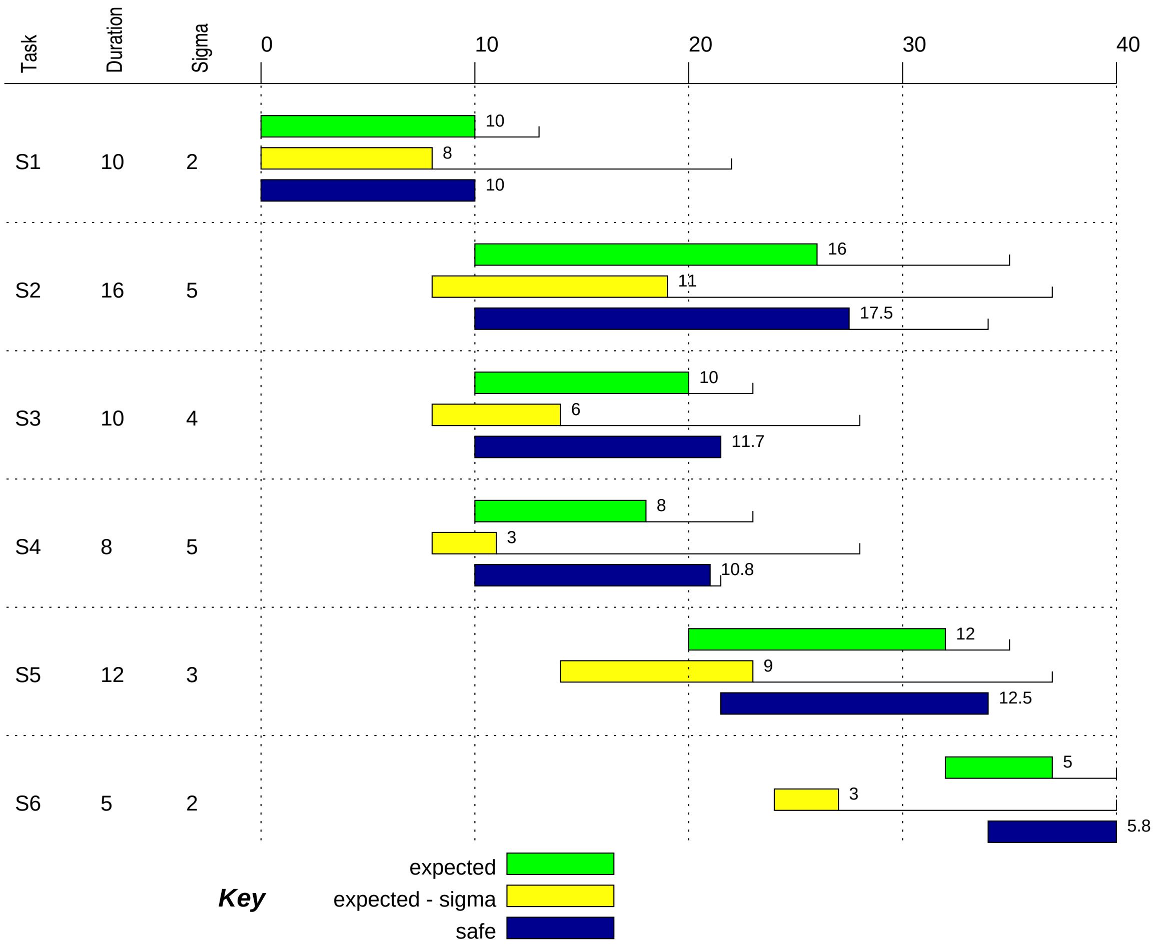

Figure 4: Reducing task durations to increase time windows

To construct the time windows, the customer agent constructs an initial schedule using the expected duration data, and sets the initial time windows using the Critical Path algorithm. 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 green bars show the expected durations of the tasks in the task network illustrated in Figure 3. The overall slack chosen by the customer in the schedule is 5 units for a 35 unit makespan, or about 14%. In the figure, the yellow bars show durations that are 1 standard deviation below the expected values, while the blue 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 must expend 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 reasonably efficient optimal solution methods are known [1, 10, 15].

An additional adjustment can be made using bidder population and variability data. The detailed relationships between the bidder count and variability data, and the optimal adjustments that need to be made in the RFQ schedule, are still under active investigation. Our current approach is to increase the relative time allocation when duration data is more variable, and to increase the overlap when the number of bidders is higher.

There is a tension between issuing a RFQ that will guarantee 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's bids result from an evaluation of their current resource commitments, and therefore larger time windows will result in more bids. Suppliers know that more time flexibility in their bids will give them a competitive advantage.

5.2 Evaluating and Awarding Bids

Once bids have been received, the customer's goal is to find and schedule the combination of bids that maximizes $E(U)$, and award them. The problem is to find a "good" mapping of bids to tasks and then find a schedule for those bids that has a low risk of unrecoverable failure [4]. A "good" mapping is one that *covers* all the tasks (each task being mapped to exactly one bid), is *feasible* in terms of satisfying the temporal constraints, and is relatively low-cost. Bids are exclusive OR, so when multiple bids are submitted by the same agent, only one of them can be accepted, although other bid semantics are possible [10].

The customer agents makes decisions using an adaptive anytime search algorithm we have developed. The algorithm is based on a simulated annealing [11] framework with a set of modular selectors 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, representing sets of bids mapped to their respective tasks, are kept in an ordered queue of fixed maximum length, sorted by the node's value. The node value is computed as shown in step 2.4 in the algorithm. The value is a combination of factors and includes a risk component. Intuitively, whenever accepting a task from a bid will increase the probability of missing the goal deadline, or of missing the latest start time for tasks already selected, the risk increases.

We also maintain a *tabu list* and a *tried list* on each node. The tabu list is a list of bids that have been used “recently” in the history of node expansions leading to the node in question. By prohibiting expansion by bids on the tabu list, we prevent 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.

Bid Selection Algorithm

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 has no bids, exit (coverage cannot be achieved).
   - 1.3 **Single bid test**: If any task has a single bid, then 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 were no singletons then create a node mapping all tasks to no bids, and add it to the queue.
   - 1.5 **Set initial annealing temperature**.
2. **while not** timeout **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 was already mapped in N to some other bid B', remove bid B'.
     If B' was 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.
- 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.

The following bid-selection methods, used in Step 2.2 of the algorithm, have been implemented and tested. Note that the feasibility and cost improvement methods have significant complexity costs associated with them.

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 : The mapping is scanned to find tasks that have negative slack. Of those, the tasks that are constrained by their bids rather than by predecessors or successors could be moved in a direction that would relieve the negative slack. They are sorted by their potential to reduce infeasibility, and saved. The untried bid with the highest potential to reduce infeasibility is chosen. Note that when a bid is chosen, there is no guarantee that it will not introduce 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-priced bid that covers at least the task with the highest positive cost deviation. This is only useful if average or expected cost data is available on a per-task basis.

These selectors can be composed together and used to generate focused improvement for a given node. The following combined selectors were used in our experiments:

Random : The random selector 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 covered portion of the node is above average, attempt to reduce its cost; otherwise use the *FeasCov* selector.

Combined : Run the *Random* selector as long as it produces improvement, then switch to *Feasibility Improvement* until that fails to produce improvement, then switch back to *Random*, then to *Coverage Improvement*, then back to *Random*, then to *CostFeasCov*, and finally back to *Random*.

We have performed numerous experimental studies on the performance of the algorithm, which are reported in [4]. In particular, we have compared the random bid selector with the more "focused" bid selectors. We are interested in both the ability to find solutions, and in the rate of improvement. The latter attribute is important for setting time limits in an anytime search.

In order to probe a range of problem complexity factors, we ran the *Random*, *Cov*, *FeasCov*, and *Combined* selectors against two different problem types of the same size but different levels of complexity. The features of the problems sets are given in Table 1. Both sets have the same number of tasks and 100 bidders, and all are generated with the same random number sequences. Total slack is 10%, and task durations are set to 60% of their expected values to relax time windows in the RFQ. The average bid size, which is number of tasks included in a bid, is different in the two sets. In the *small-bid* problem, the average bid size is 5, and in the *large-bid* problem it is 15.

Table 1: Problem Sets

Problem type	small-bid	large-bid
Number of Tasks	50	50
Number of Bidders	100	100
Mean Bid Size	5	15

Table 2 shows the number of solutions found by the different selectors for the *small-bid* and *large-bid* problems. The actual number of such solutions is not known.

Figure 5 shows the improvement curves for the four bid selectors on the *small-bid* problem, and Figure 6 shows improvement curves for the same selectors on the *large-bid* problem. Error bars show $\frac{\sigma}{\sqrt{n}}$ 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 solution quality and in terms of consistency.

Additional experimental results are reported in [4], where we have compared the solutions produced by our search algorithm with the solutions obtained by systematic search, and where we have analyzed several modifications to the search procedure, in an attempt to find a combination that performs well across a range of problem characteristics.

Table 2: Solutions Found

Selector	small-bid problem	large-bid problem
Random	2	2
Cov	3	0
FeasCov	2	0
Combined	6	1

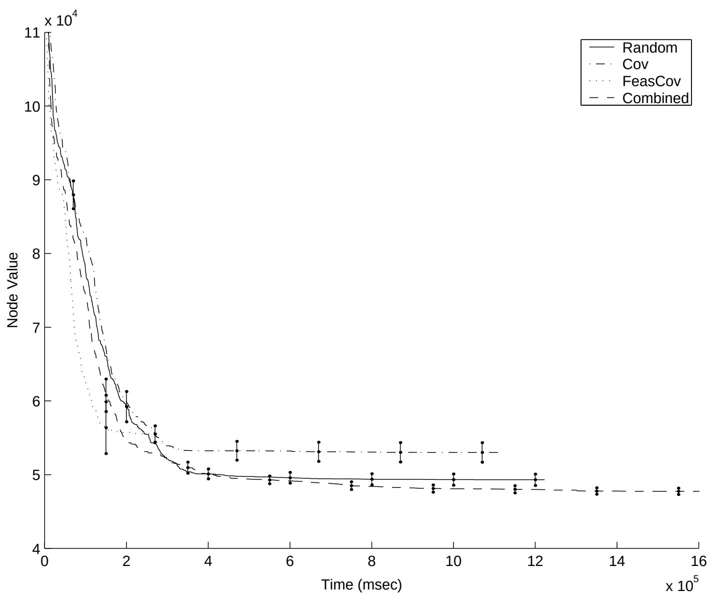

Figure 5: Improvement curves for the *small-bid* problem. Averages are shown for 20 runs.

6 Related Work

Markets play an essential role in the economy, and market-based architectures are a popular choice for multiple agents (see, for instance, [17, 19] and our own MAGMA architecture [18]). Most market architectures limit the interactions of agents to manual negotiations, direct agent-to-agent negotiation [16, 7], or various types of auctions [20].

Auctions are becoming the predominant mechanism for agent-mediated electronic commerce [9]. AuctionBot [20] and eMEDIATOR [15] are among the most well known examples of multi-agent auction systems. The determination of winners of combinatorial auctions is hard. Dynamic programming [12] works for small sets of bids, but does not scale and imposes significant restrictions on the bids. Algorithms such as Bidtree [15] and CASS [8] have been proposed to reduce the search complexity, but their criterion to select bids is just price. Our bids include a time window for each task, and so bid selection cannot be separated from scheduling.

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.

Most work in supply-chain management is limited to strict hierarchical modeling of the decision making process, which is inadequate for distributed supply-chains, since each organization is self-interested, not cooperative. A notable exception is the MASCOT [13] system. Agents in MASCOT coordinate scheduling with the user, but there is no explicit notion of payments or contracts, and the criteria for accepting/rejecting a bid are not explicitly stated. Their major objective is to show the advantage of using lateral coordination policies that fo-

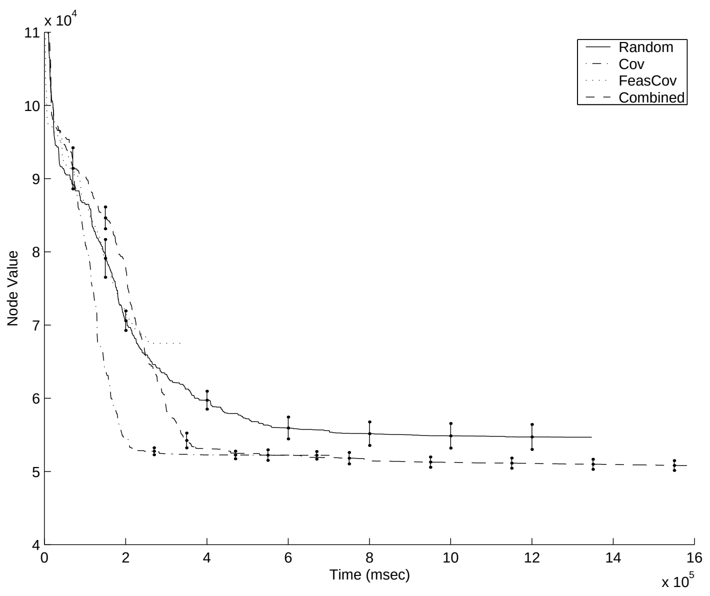

Figure 6: Improvement curves for the *large-bid* problem. Averages are shown for 20 runs.

cus on optimizing schedules locally through exchange of temporal constraints. Our objective is to negotiate contracts with suppliers that optimize customer's utility.

7 Conclusions and Future Work

The MAGNET automated contracting environment is designed to support negotiation among multiple, heterogeneous, self-interested agents over the distributed execution of complex tasks. We have shown how a marginal-utility approach can be used to support the two primary decisions that must be made by a customer agent during the bidding cycle. We have discussed the types of statistical data the market infrastructure needs to build and maintain in order to provide the information the agents will need for these decision processes. Finally, we have presented a multi-criterion, anytime bid selection algorithm, and reported on experiments using a set of increasingly informed search heuristics. The results show that excess focus on improvement leads to faster improvement early on, at the cost of a lower likelihood of finding a solution that satisfies all the constraints.

The MAGNET system has been under development since early 1998. The distributed market infrastructure, including much of the data-collecting capability described here, is in place. The principal elements of a customer agent are complete, including the highly adaptable search engine for bid evaluation.

Acknowledgments

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

References

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

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

[3] 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.

[4] John Collins, Rashmi Sundareswara, Maria Gini, and Bamshad Mobasher. Bid selection strategies for multi-agent contracting in the presence of scheduling constraints. In *Agent Mediated Electronic Commerce*, volume LNAI1788. Springer-Verlag, 2000.

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

[6] Louis Eeckhoudt and Christian Gollier. *Risk Evaluation, Management, and Sharing.* Harvester Wheatsheaf, Hemel Hempstead, Hertfordshire, UK, 1995.

[7] 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.

[8] 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.

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

[10] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proc. of ACM Conf on Electronic Commerce (EC'00)*, pages 1–12, Minneapolis, Minnesota, October 2000. ACM SIGecom, ACM Press.

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

[12] Michael H. Rothkopf, Alexander Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[13] Norman M. Sadeh, David W. Hildum, Dag Kjenstad, and Allen Tseng. MASCOT: an agent-based architecture for coordinated mixed-initiative supply chain planning and scheduling. In *Workshop on Agent-Based Decision Support in Managing the Internet-Enabled Supply-Chain, at Agents '99*, pages 133–138, May 1999.

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

[15] Tuomas Sandholm. Approaches to winner determination in combinatorial auctions. *Decision Support Systems*, 28(1-2):165–176, 2000.

[16] Tuomas W. Sandholm. *Negotiation Among Self-Interested Computationally Limited Agents.* PhD thesis, University of Massachusetts, 1996.

[17] 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.

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

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

[20] 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*, May 1998.