Bid Selection Strategies for Multi-Agent Contracting in the presence of Scheduling Constraints

John Collins[1], Rashmi Sundareswara[1], Maria Gini[1], and Bamshad Mobasher[2]

[1] Department of Computer Science and Engineering,
University of Minnesota
[2] School of Computer Science, Telecommunications, and Information Systems,
DePaul University

Abstract. Bid evaluation in a multi-agent automated contracting environment presents a challenging search problem. We introduce a multi-criterion, anytime bid evaluation strategy 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 within simulated annealing. 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 most successful approach used a combination of random and focused bid selection methods, along with pruning and repeated restarts.

1 Introduction

The University of Minnesota's MAGNET (Multi-Agent Negotiation Testbed) system is an innovative agent-based approach to complex contracting and supply-chain management problems. The MAGNET system [7] comprises a set of agents who negotiate with each other through a market infrastructure using a finite, leveled-commitment protocol [22]. It is designed to support the execution of complex plans among a population of independent, autonomous, heterogeneous, self-interested agents. We call this activity *Plan Execution By Contracting.*

Plan Execution by Contracting is designed to extend the applicability of agent negotiation to new domains, where schedules for production and delivery affect the cost and feasibility of services and products, and where monitoring the performance of task execution is an essential part of the process. This is especially important for domains such as logistics, dynamic planning and scheduling, and coordination of supply-chain management with production scheduling, where what is important is flexibility, ease of use, quality, performance, as opposed to just cost [13].

In general, a MAGNET agent 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. Once the Customer agent receives bids, it must decide which bids to select by solving both bid-allocation and temporal feasibility constraints, while attempting to minimize cost and risk.

We have developed a highly tunable anytime search [2], based on a simulated annealing [18] framework with a set of modular selectors and evaluators. Given that the time allocated to search will seldom be sufficient to explore a significant fraction of the search space, we must find an appropriate tradeoff between systematic optimization and random exploratory behavior. We describe here a set of experiments using our bid selection algorithm and characterize its performance on a variety of problems.

This paper is organized as follows. We describe our study on bid selection in Section 2, and the experimental setup in Section 3. In Section 4 we present our empirical results on the performance of the various bid-selection strategies. Section 5 covers the background and related work. Section 6 describes our conclusions and suggestions for future work related to this problem.

2 Agent Interactions and Time Allocation

The negotiation portion of the MAGNET protocol is a finite 3-step process that begins when a Customer agent issues a RFQ. The RFQ specifies a set of tasks that must be performed, along with time and precedence constraints. Suppliers may reply with bids, and the Customer accepts the bids it chooses with bid-accept messages.

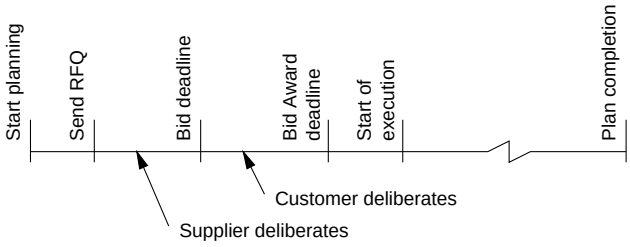

Fig. 1. Agent Interaction Timeline

The timeline in Figure 1 shows an abstract view of the progress of a single negotiation. At the beginning of the process, the Customer agent must allocate time to deliberation for its own planning, for supplier bid preparation, and for its own bid evaluation. In general, it is expected that bid prices will be lower if suppliers have more time to prepare bids, and more time and schedule flexibility

in the execution phase. On the other hand, the Customer's ability to find a good set of bids is dependent on the time allocated to bid evaluation, as we shall see. Since the Customer's goal is time-sensitive, the negotiation process requires Customer and Suppliers to agree on the times for execution of tasks, and time factors can affect the cost of execution.

For each task listed in the RFQ the following must be specified:

- a time window, consisting of an earliest start time $t_{es}(s)$ and a latest finish time $t_{lf}(s)$, for each task s,
- a set of precedence relationships for the task.

Our contracting protocol allows Supplier agents to bid on combinations of items (possibly including a discount or a premium for accepting the whole bid), and assumes exclusive OR bids, which means that when multiple bids on combinations of tasks are submitted by the same agent, at most one of them can be accepted. Each bid represents an offer to execute some subset of the tasks specified in the RFQ. Each bid b includes:

- the overall bid price,
- the set of tasks. For each task s it includes:
  - the time window, that consists of early start $t^b_{es}(s)$, and late finish $t^b_{lf}(s)$,
  - the duration $duration^b(s)$,
  - the individual price $price^b(s)$ for the task.

The difference between the overall bid price and the sum of the individual task prices is referred to as the *discount* or the *premium* of the combination.

The semantics of a bid is that the Supplier agent is willing to perform task s for the bid price $price^b(s)$ starting at any time $t_s(s)$ within the time window specified in the bid as long as the task is completed before the late finish, i.e. $t^b_{es}(s) \leq t_s(s) \leq t^b_{lf}(s) - duration^b(s)$, and finishing at time $t_s(s) + duration^b(s)$. It is a requirement of the protocol that the time parameters in a bid are within the time windows specified in the RFQ.

The Customer agent's objective is to maximize utility, which requires minimizing cost and risk of not accomplishing the goal. To determine which bids (or parts of bids) to accept, the Customer agent considers coverage (we assume that there is no point in accomplishing only a part of the goal), temporal feasibility (the time windows for tasks must allow to compose them in a feasible schedule), cost, and risk. Risk factors include elements such as availability of suppliers, supplier reliability, profit margin, expected cost of recovering from supplier decommitment or delay, loss of value if the end date is delayed, cost of plan failure. MAGNET agents have to assess their own preferences and risk tolerance, as described more in detail in [5].

Once the Customer has selected the optimal set of bids (or parts thereof) the resulting *task assignment* forms the basis of an initial schedule for the execution of the tasks. The execution may involve additional negotiations over schedule adjustments, and in some cases may require repeating the bidding cycle if a supplier decommits or if a delay in completing some task forces other tasks to

go outside the time windows agreed between Customer and Supplier. We will not cover the execution process here, we will limit our presentation to the selction of bids.

3 Experimental Setup

The experimental setup includes three main components: a MAGNET Server as described in [7], a Customer agent that requests and evaluates bids, and a Supplier agent that generates and submits bids. The bid evaluation process is instrumented to measure the rate of improvement for various search strategies. Random variable seeds are controlled to ensure that different search strategies are presented with exactly the same problems.

3.1 Customer Agent: Construct and Issue a Request for Quotes

The Customer agent starts with a set of tasks with their precedence relations, and constructs and issues a RFQ.

We assume the agent has general knowledge of normal durations of tasks, which can be obtained from the MAGNET market infrastructure [7]. The Customer agent schedules tasks using expected durations, and computes early start and late finish times using the Critical Path (CPM) algorithm [14].

The Critical Path algorithm walks the directed graph of tasks and precedence constraints, forward from the start time t_0 to compute the earliest start $t_{es}(s)$ and finish $t_{ef}(s)$ times for each task s, and then backward from time t_{goal} to compute the latest finish $t_{lf}(s)$ and start $t_{ls}(s)$ times for each task s. The minimum duration of the entire plan, defined as $max(t_{ef}(s)) - t_0$, is called the *makespan* of the plan.

The difference between t_{goal} and the latest early finish time is called the *total slack* of the plan. If t_{goal} is set equal to $t_0 + makespan$, then the total slack is 0, and all tasks for which $t_{ef}(s) = t_{lf}(s)$ are called *critical* tasks. Paths in the graph through critical tasks are called *critical paths.*

The tradeoff between minimizing plan duration and attracting usable bids from suppliers affects how slack should be set. Slack is important in determining the expected schedule risk, which is associated, among other things, with constraint-tightness.

In order to allow for variability in actual task durations, to allow suppliers some degree of flexibility in their resource scheduling, and to take advantage of lower-than-expected task durations, the specified start and finish times are relaxed from their expected values. For example, in the set of experiments described in Section 4.1, we relax the time windows by 40%. This amount has been shown to give a good balance between supplier flexibility and the need of the Customer agent to be able to compose feasible plans [6]. We do this by setting t_{goal} as discussed above, reducing the task durations to 60% of their expected values, and re-running the CPM algorithm on the task graph.

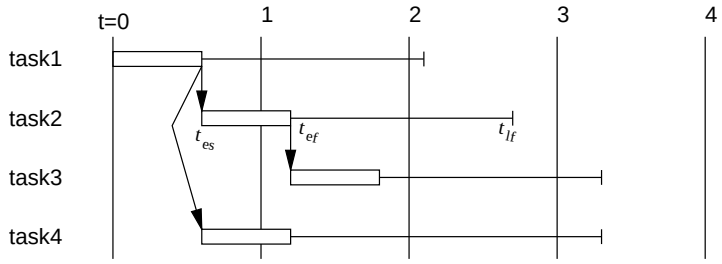

Fig. 2. Plan as specified by Customer

Figure 2 shows in Gantt chart form how such a plan would be specified by a Customer. In the example, each of the tasks has an expected duration of 1.0, and the plan has an overall slack of 10%, giving a target finish time of 3.3. The time windows in the RFQ are specified with task durations of 0.6. The arrows show precedence relationships, and the extension bars show per-task slack. Task 2 is labeled to show the early start, early finish, and late finish times.

3.2 Supplier Agent: Generate Bids

The Supplier agent is a test agent that masquerades as an entire community of suppliers. Each time a new RFQ is announced, the Supplier agent attempts to generate some number of bids.

Bids are generated for random sets of contiguous tasks within the RFQ. Task durations are randomly distributed around the expected value used by the Customer, and the time windows are randomly set to be smaller than the time windows specified in the RFQ and larger than the computed durations. Because of this, arbitrary combinations of bids frequently generate infeasible schedules. Full details on the bid generation process are given in [6].

3.3 Customer Agent: Evaluate Bids

Once the bidding deadline is past, the Customer evaluates the set of bids in an attempt to find a combination that provides coverage of all tasks, allows for a feasible schedule, and minimizes a combination of cost and risk. Since bids are exclusive OR (when multiple bids are submitted by the same agent, only one of them can be accepted) for each bid the Customer has the option of selecting which parts of the bid (if any) to accept.

The evaluator uses a generalized simulated annealing [18] search as described below. Nodes are kept in an ordered queue of fixed maximum length (the *beam width*), 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 $t_{ls}i(s)$ for tasks already selected, the risk increases.

The simulated-annealing framework is characterized by two elements, the annealing temperature T which is periodically reduced by a factor ε, and the stochastic node-selection procedure. The algorithm is outlined here below.

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
     - V_i is the value of node i in the queue. Nodes in the queue are sorted by increasing values, V_{min} is the value of the first node.

     Choose node N as the last node in the queue such that $V_N \leq R$
   - 2.2 **Select a bid B**:
     Discard all bids that appear on the tabu list of N.
     Discard all bids that have already been used to expand 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 to bid B', remove bid B'.
     If B' was a singleton bid, 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, Random Bid Component : Choose a bid or a bid component at random, and attempt to add it to the node. The ratio of bids to bid components is adjustable. This method is fast ($O(1)$) and promotes general exploration of the search space.

Coverage Improvement : Choose a bid or bid component that covers a task that is not mapped in the node. The probability of choosing a bid component is equal to the coverage factor of the node. This method is also $O(1)$ if the set of unmapped tasks and the set of bids per task is stored.

Feasibility Improvement : The mapping is scanned to find tasks s that have negative slack, i.e. tasks such that $t^b_{es}(s) + duration(s) > t^b_{lf}(s)$. 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 or bid component 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. The complexity of this method is $O(xy)$, where x is the total number of tasks and y is the number of tasks in the mapping that meet the above improvement criteria, incurred once the first time a node is subjected to feasibility improvement, and $O(z)$, where z is the number of bids that could potentially be mapped to a task, each time a feasibility improvement is attempted on a node.

Cost Improvement : Choose the (untried) bid or bid component that is responsible for the maximum positive deviation from the average price, and replace it with a lower-priced bid that covers at least the task with the highest positive cost deviation. The first time this method is applied to a node, it has a complexity of $O(xy + z)$, where x is the number of bids mapped to a node, y is the number of tasks in a bid, and z is the number of potential bids per task. Subsequent expansions of the same node by this method incur a complexity of only $O(z)$.

These selectors can be composed together and used to generate focused improvement for a given node. The following 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*.

4 Experimental Results

We describe three different experiments that we have carried out in the process of developing and characterizing the bid evaluation search. The results give us confidence that the bid evaluation problem can be solved for reasonably large problems using the simulated annealing approach.

4.1 Comparing Systematic and Stochastic Search

Our first experimental goal was to determine how well the simulated annealing search performed with respect to a known optimal reference. For that purpose, we constructed an alternate search engine that generates all feasible combinations of bids and bid components in order to be guaranteed of finding optimal solutions. Because of bid overlap and feasibility issues, no more efficient method is known that will provide such a guarantee. Its structure is similar to the method reported in [23] with the addition of a feasibility test.

The test problem for this experiment is necessarily small, because of the long run times of the systematic search engine. We generated 20 random problems with 10 tasks and 11 bids each. The "branch factor" that controls the density of precedence relationships was 2.4, and the average bid size was 2.72 tasks. Overall schedule slack was set to 1.4, and task durations were set to 70% of expected values to open up the time windows in the RFQ.

The summary data for this experiment is in Table 1. Solutions (feasible mappings that covered all tasks) were found for 9 out of the 20 problems. The others either lacked coverage (in 7 of the 20 runs there was at least one task for which no bid was submitted) or no feasible combinations existed (4 cases). The node counts and solution evaluations are the mean for the cases where solutions were found. The data for the Stochastic 1 and Stochastic 3 trials are normalized to account for the missing solution. The four trials used identical plans and bid sets.

Table 1. Systematic and Stochastic Results

	Systematic	Stochastic 1	Stochastic 2	Stochastic 3
Nodes Generated	115480	2171	2205	1323
Covered & Feasible	46916	435	448	219
Best Solution Eval.	7960	8073	7998	8073
Solutions Found	9	8	9	8
Run Time (min.)	242	9.8	9.4	6.2

The search engine parameters were set up as follows:

Systematic: Systematic search with problem setup as described above.

Stochastic 1: Simulated-annealing search, initial temperature 0.35, reduced by 0.95 every 100 iterations, *patience factor* (number of iterations without improvement) 100. The stopping criterion for the stochastic search is either timeout (no timeout was used in this trial) or failure to improve for a number of iterations

$$n \geq patienceFactor \ln(taskCount) \ln(bidCount)$$

where *taskCount* is the number of tasks in the plan, and *bidCount* is the number of bids being considered. The bid selector is the Combined selector, as described in the previous section.

Stochastic 2: Setup as in Stochastic 1, except that a uniqueness test is added to prevent identical nodes from being evaluated and added to the search queue.

Stochastic 3: Setup as in Stochastic 2, except that the patience factor was reduced to 50.

The conclusion is that the simulated-annealing search engine performs well, with solution quality within 2% of the systematic search at radically reduced run times. It is also clear that it needs to be tuned to avoid missing solutions.

4.2 Comparing Bid Selectors

In the second study, we are comparing 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. The results show that the more focused methods alone are ineffective, even if used with high annealing temperatures. The best results were obtained by combining random and focused methods. The *Combined* selector outperforms all the others. It seems 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 constraints.

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. Both of them contain 50 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. In the *small-bid* problem, the average bid size (number of tasks included in a discounted bid) is 5, and in the *large-bid* problem, the average bid size is 15. Earlier work [23] has shown that this difference has a significant impact on the search difficulty due to the greater probability of overlap among bids.

Figure 3 shows the improvement curves for the four bid selectors on the *small-bid* problem, and Figure 4 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

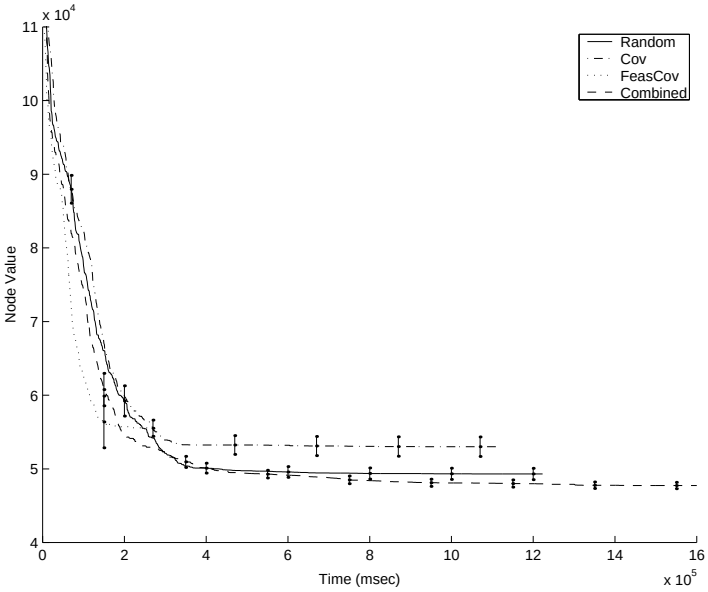

Fig. 3. Improvement curves for the *small-bid* problem. Averages are shown for 20 runs.

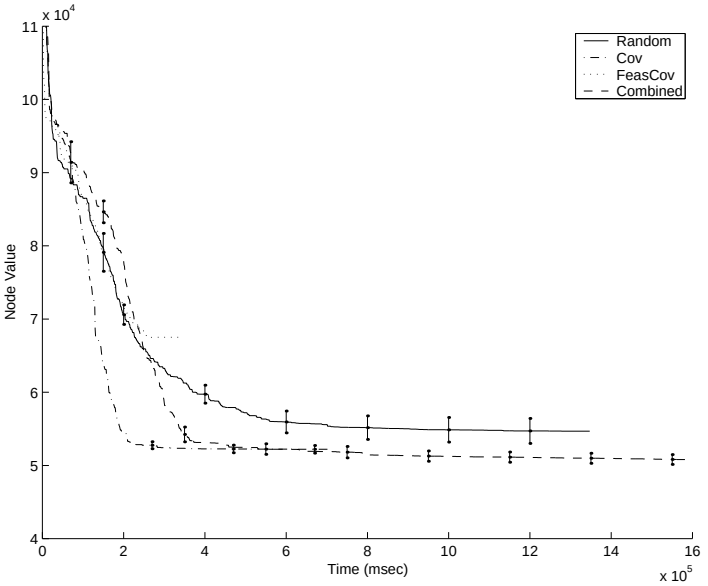

Fig. 4. Improvement curves for the *large-bid* problem. Averages are shown for 20 runs.

the best overall performance, both in terms of solution quality and in terms of consistency.

Table 2 shows the number of acceptable assignments found for the *small-bid* and *large-bid* problems. The table shows how effective the four selectors were at finding solutions that satisfied all constraints. The actual number of such solutions is not known. Again, we see the advantage of the *Combined* selector, which uses random selection to generate sets of candidates, and then switches to more focused selectors to clean up.

Table 2. Solutions Found

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

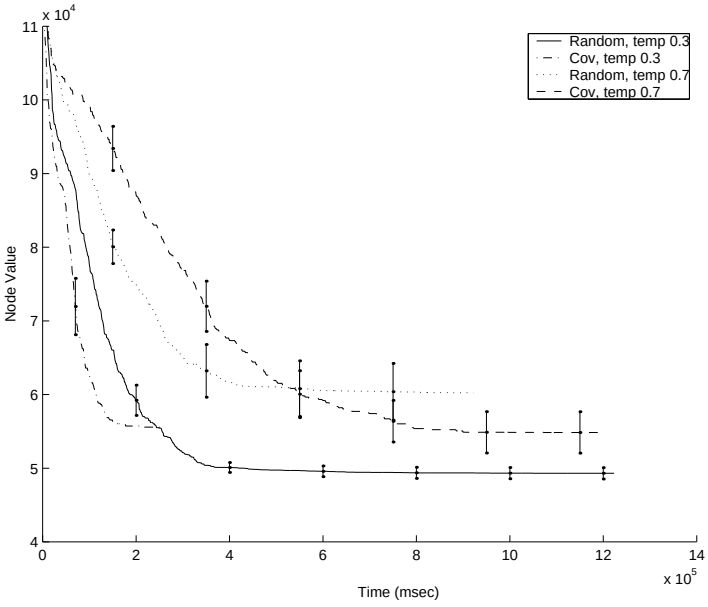

Fig. 5. Improvement curves for two different annealing temperatures.

In Figure 5, we explore the effect of raising the annealing temperature on the performance of the selectors. The experiments described earlier were all run with an initial annealing temperature of 0.3. We see that raising the annealing

temperature does not improve performance, and the focused selectors do not perform any better at higher temperatures.

4.3 Improving Search Results

The third series of experiments is an attempt to improve the performance of the Simulated Annealing search in terms of its ability to find solutions. In the previous experiment we were concerned with the rate of improvement, but only a few of the problems were actually solved. This happened because the problems generated by the given test conditions had few solutions, but could also indicate that the search method was ineffective. In this experiment, we modify the problem parameters to increase the number of problems that have solutions, and we try several modifications to the search procedure in an attempt to find a combination that performs well across a range of problem characteristics.

In this experiment we use four problem sets that vary in two dimensions. The *branch factor* is the average number of precedence relationships per task in the plan. The *bid size* is the average number of tasks per bid. In the problem setup, the branch factor is controlled directly, while the bid size is controlled indirectly through the probability that the bidder will follow a precedence link when composing a bid. This behavior produces bids that are primarily composed of tasks that are *contiguous*, connected by precedence relations. Each problem set consists of 50 problems, 40 tasks/problem, and 90 bidders. Overall schedule slack is set to 40%, and task durations are set to 80% of their expected values.

Because of the way resource availability is simulated, not all bidders will bid on each problem. Because of our random approach to bid generation, there is no guarantee that bids will be generated for all tasks in a particular plan. Problems for which a full set of bids is generated are called *covered*. Among the covered problems, it is not known how many actually have feasible solutions. We call the problems sets *p-b-*, *p-b+*, *p+b-*, and *p+b+*, to indicate low-precedence – small-bid, low-precedence – large-bid, etc. Their statistics are given in Table 3.

Table 3. Problem Sets

Problem type	p-b-	p-b+	p+b-	p+b+
Branch factor	2.4	2.4	4.0	4.0
Mean bid size	11.79	20.93	11.95	19.62
Number of bids	88.68	88.62	87.72	87.92
Number covered	29	37	45	49

Preliminary experiments showed that four specific modifications to the search procedure had the potential to improve the ability to find solutions. We will show that, under the stated experimental conditions, the combination of all four performed better than any three of them. The modifications are:

1. A "variable-temperature" bid selector uses the current annealing temperature to adjust the amount of randomness in bid selection. At the initial temperature T_0, bid selection is completely random. As the temperature changes, coverage improvement is attempted with a probability of $(1 - \frac{T}{T_0})$.
2. Infeasible nodes are pruned from the search space rather than being left in for improvement. The intent is to improve the probability that a given node can be extended into a solution. Since feasibility checking is part of the node evaluation process, there is essentially no extra cost for this test.
3. Non-unique nodes are pruned from the search space. Because of the exclusive-or nature of bid combination, non-unique nodes can be produced by many paths, and the standard method of using tabu lists and preventing repeated identical expansions to a given node are not strong enough to prevent non-unique nodes from being produced. The cost for uniqueness checking is linear in the number of tasks in the plan.
4. The search is repeated multiple times, each time clearing out the queue and increasing the "patience factor" by some amount (10% in these tests). A maximum of 9 restarts is allowed. Once a solution is found, one additional restart is permitted in an attempt to further optimize the result.

We ran 5 tests on each of four problem sets, 50 runs per test, for a total of 2000 runs. In each test, we are interested in the following:

- significant differences in search effort as evidenced by the number of nodes generated,
- significant differences in the number of problems solved, and
- significant differences in the evaluation scores of the solutions found.

The criterion for significant difference will be a probability of the null hypothesis (no difference, or in some cases no improvement) of less than 5% using a paired-sample t-test. Note that the normal-distribution assumption of the t-test may not be satisfied. We will evaluate the criteria on each problem type, and on the composite of all four types. When looking for difference, we use the 2-tail criterion, and when looking for specific improvement we use the 1-tail criterion

The tests conditions are:

1. **Baseline**: The "variable-temp, adaptive restart" baseline, containing all the elements described above.
2. **Random**: As 1, but replace the variable-temp selector with a purely random selector. We expect to see higher search effort.
3. **Infeasible**: As 1, but do not filter out infeasible nodes. We expect to find fewer solutions.
4. **Non-unique**: As 1, but do not filter out non-unique nodes. We expect to find fewer solutions.
5. **Early stop**: As 1, but do not run the extra restart after a solution is found. We expect to see lower search effort and poorer optimization performance.

The next three tables give the mean results for search effort, problem-solving ability, and optimization performance for each of the five test conditions and for

each of the problem types. We also combine the results from the four problem types into a "composite" problem type. Where a significant difference is found, we give the probability of the null hypothesis and indicate whether it is a one-tail or two-tail probability.

Table 4. Search Effort (mean number of nodes generated)

	p-b-	p-b+	p+b-	p+b+	composite
baseline	4803	4827	7195	6600	6036
random	8237	6850	10172	9779	8933
	$p_{1T} < .01$	$p_{1T} < .01$	$p_{1T} < .01$	$p_{1T} < .01$	$p_{1T} < .01$
infeasible	8751	6697	4898	4899	5999
	$p_{2T} < .01$	$p_{2T} < .01$	$p_{2T} < .01$	$p_{2T} < .01$	–
non-unique	5974	3510	7822	6466	6491
	–	–	–	–	–
early stop	4864	3713	5794	5385	5019
	–	$p_{1T} < .05$	$p_{1T} < .05$	$p_{1T} < .05$	$p_{1T} < .01$

We see from Table 4 that the use of the random selector drives up search effort, while the elimination of the extra restart drives search effort down. This is what we expected.

In Table 5 the first row is simply the number of problems for which bid coverage is achieved. There is no guarantee that all covered problems actually have feasible solutions, although only one problem, in the *p-b-* set, was not solved by any method. This data set shows that the pruning of infeasible nodes has a dramatic impact on the problem-solving capability of the search, while the pruning of non-unique nodes has a small but significant impact. The use of the purely random bid selector also had a negative impact on problem-solving, but it is only significant when the results of all four problem sets is combined.

Table 5. Problem Solving (number of problems solved)

	p-b-	p-b+	p+b-	p+b+	composite
maximum	29	37	45	49	160
baseline	28	37	44	49	158
random	26	37	39	47	149
	–	–	–	–	$p_{2T} < .01$
infeasible	15	24	23	22	84
	$p_{1T} < .01$	$p_{1T} < .01$	$p_{1T} < .01$	$p_{1T} < .01$	$p_{1T} < .01$
non-unique	27	37	39	48	151
	–	–	$p_{1T} < .05$	–	$p_{1T} < .01$
early stop	28	37	43	49	157
	–	–	–	–	–

Finally, in Table 6 we see the results for the optimization performance of the various approaches. The improvement afforded by running an extra round of search is surprisingly small; the reduction in solution quality for the "early stop" case is only significant when data from all problem types is combined. The use of the purely random selector yields a slight improvement in quality, and the pruning of non-unique nodes is shown to improve quality slightly.

Table 6. Solution Quality (mean evaluation score)

	p-b-	p-b+	p+b-	p+b+	composite
baseline	35236	31886	27691	25902	29539
random	32683	30729	26524	24569	28032
	$p_{2T} < .05$	–	–	–	$p_{2T} < .01$
infeasible	32326	31318	29318	24740	29280
	–	–	–	$p_{2T} < .05$	–
non-unique	34878	32564	30237	26982	30593
	–	–	$p_{2T} < .05$	–	$p_{2T} < .05$
early stop	34758	32654	29510	27299	30503
	–	–	–	–	$p_{2T} < .05$

All hypotheses are confirmed, at least in the composite case. Note that the evaluation score data in Table 6 are only approximately comparable, since for each comparison the analysis must compare scores only for problems that both methods solved. Some interesting results can be observed:

- The random method is clearly more costly, and is a poorer solver, but when it succeeds it produces slightly better solutions.
- The extra restart appears to have a rather high cost in terms of search effort for a small but significant improvement in solution quality.
- The filtering of infeasible nodes appears to have a positive impact on search effort when the density of precedence relations is low, but a negative impact when the density is high. On the other hand, this is the factor that appears to have the largest impact on problem-solving ability.

5 Related Work

Markets play an essential role in the economy [1], and market-based architectures are a popular choice for multiple agents (see, for instance, [4, 19, 24, 26] and our own MAGMA architecture [25]). Most market architectures limit the interactions of agents to manual negotiations, direct agent-to-agent negotiation [22, 8], or some form of auction [27].

Auctions are becoming the predominant mechanism for agent-mediated electronic commerce [11]. AuctionBot [27] and eMEDIATOR [21] are among the most well known examples of multi-agent auction systems. They use economics

principles to model the interactions of multiple agents. Auctions are not the most appropriate mechanism for the business-to-business transactions we are interested in, where scheduling plays a major role, and where reputation and maintaining long term business relations are often more important than cost.

The determination of winners of combinatorial auctions [16] is hard. Dynamic programming [20] works well for small sets of bids, but does not scale and imposes significant restrictions on the bids. Shoham [9] produces optimal allocations for OR-bids with dummy items by cleverly pruning the search space. Sandholm [21] uses an anytime algorithm to produce optimal allocations for a more general class of bids, that includes XOR and OR-XOR bids. The major difference is that in the cases studied for combinatorial auctions bid allocation is determined solely by cost. Our setting is more general, MAGNET agents have the ensure the scheduling feasibility of the bids they accept. Finding a set of bids that has minimal cost but is not feasible (i.e. no feasible schedule for the bids exists) is of no use.

We have chosen to use a simulated annealing framework for bid evaluation. Since the introduction of iterative sampling [15], a strategy that randomly explores different paths in a search tree, there have been numerous attempts to improve search performance by using randomization. Randomization has been shown to be useful in reducing the unpredictability in the running time of complete search algorithms [10].

A variety of methods that combine randomization with heuristics have been proposed, such as Least Discrepancy Search [12], heuristic-biased stochastic sampling [3], and stochastic procedures for generating feasible schedules [17], just to name a few. The algorithm we presented is based on simulated annealing, and as such combines the advantages of heuristically guided search with some random search. Our experimental results show that additional benefit can be obtained by using domain-specific heuristics when deciding how to expand a node. This combined with the basic simulated annealing framework produces good results in a short time frame.

6 Conclusions and Future Work

Bid evaluation in the MAGNET automated contracting environment is a difficult optimizing search problem that must be performed within a predetermined amount of time. Ignoring the use of individual bid components, in our experiment there are approximately 10^{14} bid combinations that could be tested.

We have chosen a simulated annealing framework to drive broad exploration of the search space. For this to be effective, it is necessary to set a number of parameters, including the beam width, annealing temperature and rate, and penalty factors, to avoid placing large potential barriers around the solutions. Eventually, we hope to be able to use problem metrics to set these parameters on the fly.

We have presented a bid evaluation process for automated contracting that incorporates cost, task coverage, temporal feasibility, and risk estimation, and we

have provided, using this evaluation process, an empirical study of the tradeoffs between focus and ultimate success on this large, multicriterion search problem. Time constraints dictate that only a tiny fraction of the search space can be explored, and local minima abound. The best results were obtained with a combination of random and informed bid selection methods, along with pruning of infeasible and duplicate nodes and repeated restarts of the search.

This work raises several interesting questions for future research. Work is required to develop a clear understanding of how the various tuning parameters should be adjusted in accordance with problem parameters. An ideal tuning would reduce the incidence and size of local minima, and minimize potential barriers around usable solutions. At a higher level, the scheduling of planning time, bid preparation time, bid evaluation time, and execution time needs to be driven by knowledge of the relative contributions of each of those components to overall solution quality.

References

1. Yannis Bakos. The emerging role of electronic marketplaces on the Internet. *Comm. of the ACM*, 41(8):33–42, August 1998.
2. Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proc. of the 11th Joint Conf. on Artificial Intelligence*, volume 2, pages 979–984, Detroit, MI USA, August 1989.
3. John L. Bresina. Heuristic-biased stochastic sampling. In *Proc. of the Thirteenth Nat'l Conf. on Artificial Intelligence*, 1996.
4. 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.
5. 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*, June 2000. (to appear).
6. John Collins, Maksim Tsvetovat, Rashmi Sundareswara, Joshua Van Tonder, Maria Gini, and Bamshad Mobasher. Evaluating risk: Flexibility and feasibility in multi-agent contracting. Technical Report 99-001, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, Minnesota, February 1999.
7. 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.
8. 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.
9. Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions. In *Proc. of the 16th Joint Conf. on Artificial Intelligence*, 1999.
10. Carla Gomes, Bart Selman, and Henry Kautz. Boosting combinatorial search through randomization. In *Proc. of the Fifteen Nat'l Conf. on Artificial Intelligence*, pages 431–437, 1998.
11. Robert H. Guttman, Alexandros G. Moukas, and Pattie Maes. Agent-mediated electronic commerce: a survey. *Knowledge Engineering Review*, 13(2):143–152, June 1998.

12. William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *Proc. of the 14th Joint Conf. on Artificial Intelligence*, pages 607–613, 1995.
13. S. Helper. How much has really changed between us manufacturers and their suppliers. *Sloan Management Review*, 32(4):15–28, 1991.
14. Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research.* McGraw-Hill, 1990.
15. Pat Langley. Systematic and nonsystematic search strategies. In *Proc. Int'l Conf. on AI Planning Systems*, pages 145–152, College Park, Md, 1992.
16. R. McAfee and P. J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.
17. Angelo Oddi and Stephen F. Smith. Stochastic procedures for generating feasible schedules. In *Proc. of the Fourteenth Nat'l Conf. on Artificial Intelligence*, pages 308–314, 1997.
18. Colin R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems.* John Wiley & Sons, New York, NY, 1993.
19. J. A. Rodriguez, P. Noriega, C. Sierra, and J. Padget. FM96.5 - a Java-based electronic auction house. In *Second Int'l Conf on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97)*, London, April 1997.
20. Michael H. Rothkopf, Alexander Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.
21. Tuomas Sandholm. An algorithm for winner determination in combinatorial auctions. In *Proc. of the 16th Joint Conf. on Artificial Intelligence*, pages 524–547, 1999.
22. Tuomas W. Sandholm. *Negotiation Among Self-Interested Computationally Limited Agents.* PhD thesis, University of Massachusetts, 1996.
23. Erik Steinmetz, John Collins, Maria Gini, and Bamshad Mobasher. An efficient algorithm for multiple-component bid selection in automated contracting. In *Agent Mediated Electronic Trading*, volume LNAI1571, pages 105–125. Springer-Verlag, 1998.
24. 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.
25. 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.
26. Michael P. Wellman and Peter R. Wurman. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24:115–125, 1998.
27. 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.