Harnessing the Search for Rational Bid Schedules with Stochastic Search and Domain-specific Heuristics

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

Abstract

In previous work we proposed an approach for computing an agent's preferences over different schedules of tasks, and for soliciting desirable bid combinations to cover the tasks. The proposed approach finds schedules that maximize the agent's Expected Utility.

The maximization problem is hard because the domain is piece-wise continuous, with the number of pieces and local maxima growing exponentially in the worst case scenario. For agents who are averse to taking risks, maximization algorithms tend to converge to degenerate maxima of no practical interest.

In this paper we demonstrate three different maximization methods based on domain-specific heuristics. We also present a new stochastic maximization approach and benchmark it in two substantially different problem setups.

1. Introduction

We describe and analyze the domain of agents who need to schedule tasks with temporal and precedence constraints. We are interested in situations where an agent does not have sufficient resources for all the tasks and thus has to recruit other agents to carry out those tasks. In our approach, an agent issues a *Request for Quotes* (RFQ), accepts bids by other agents, and evaluates the bids to determine the winners of the auction [3].

The agent needs to balance tradeoffs between allocating large time windows to give flexibility to suppliers and risking not being able to combine the bids into a feasible schedule. Additional tradeoffs have to be made between accomplishing tasks fast to receive the final reward and risking not having enough time to recover in case a task fails. To help the agent arrive at a compromise we designed a mechanism that generates schedules for the RFQ based on Expected Utility Theory [1].

Previously [2] we demonstrated the complexity of an agent's decision process in such domain, and the failure of direct approaches to find optimal expected profit solutions for agents who are averse to taking risks.

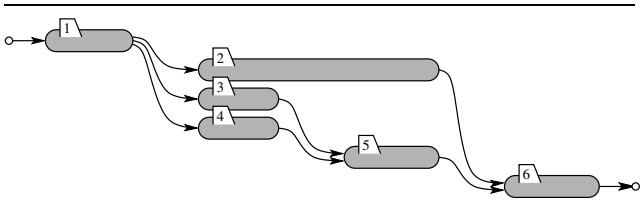

Figure 1. A reducible task network *A*.

In this work we develop heuristics to effectively overcome those problems. We perform a comprehensive study of the domain's properties using domain-specific heuristics and maximization algorithms. We further use the obtained results to design a fast stochastic method for solving the agent's problem, and to measure its performance.

2. Background

The work presented in this paper is part of the Multi-AGent NEgotiation Testbed (MAGNET) project [3]. MAGNET agents participate in first-price, sealed-bid, reverse combinatorial auctions over collections of tasks with precedence and time constraints. We assume that an agent specifies tasks in the form of a *task network* — a collection of tasks and transitive precedence relations between them.

Examples of task networks are shown in Figures 1 and 2. Both networks have the same number of tasks and the same number of precedence constraints, however they differ in a fundamental way. The first task network is *reducible* — it can be reduced to a single task by sequentially merging subsets of parallel and sequential tasks; the second network is *irreducible*. In our research we found that it is possible to study reducible networks analytically, while irreducible ones require an algorithmic approach [2]. For this reason we perform our further analysis on these two networks.

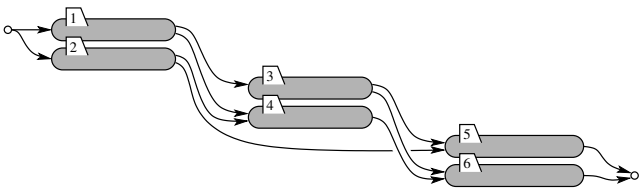

Figure 2. An irreducible task network *B*.

An agent's RFQ specifies preferred time windows for tasks in the task network. The placement of task i is characterized by the *task i start time* t_i^{s} and *task i finish time* t_i^{f}. Upon receiving bids, the agent uses the winner determination algorithm to select a feasible combination of bids that covers the entire collection of tasks and maximizes the agent's preferences.

Our preliminary results show that the specifics of how tasks are scheduled in a RFQ affect the quality and quantity of bids [1]. We proposed to use Expected Utility Theory [11, 10] as a criterion for comparing different RFQs. Expected utility provides a natural way of accounting for the risk posture of the person or organization on whose behalf the agent is acting, and for modeling the tradeoffs between risks and profit expectations. By maximizing the expected utility we can find rational RFQ compositions.

We represent the agent's preferences over payoffs by the *utility function* u:

$$u(z) = -e^{-rz} \text{ for } r \neq 0 \qquad u(z) = z \text{ for } r = 0.$$

Agents with positive r values are risk-averse, those with negative r values are risk-loving. We are mainly interested in risk-averse agents.

We assume that a future state of the world is described by the set S of all possible events. Each event $s \in S$ happens with a non-zero probability p_s and results in some monetary payoff z_s. We define a *lottery* L to be a set of *payoff-probability pairs*,

$$L = \{(z_s, p_s)\} \qquad \text{s.t.} \qquad p_s > 0 \quad \text{and} \quad \sum p_s = 1$$

The expectations of the utility values over a lottery L are captured by the von Neumann-Morgenstern *expected utility* function:

$$Eu[L] := \sum_{(z_i,p_i)\in L} p_i u(z_i).$$

The *certainty equivalent* (CE) of a lottery L is defined as the single payoff value whose utility matches the expected utility of the entire lottery L, i.e. $u(\mathrm{CE}[L]) := Eu[L]$. Hence, under our assumptions:

$$\mathrm{CE}[L] = \begin{cases} -\frac{1}{r}\log\sum_{(z_i,p_i)\in L} p_i e^{-rz_i} \text{ for } r \neq 0 \\ \\ \sum_{(z_i,p_i)\in L} p_i z_i \text{ for } r = 0 \end{cases}$$

The concept of certainty equivalent is crucial due to its properties. Unlike expected utility, CE values can be compared across different values of risk aversity r, since they represent certain monetary transfers measured in present terms. Naturally, an agent will not accept a lottery with a negative CE value, and higher values will correspond to more attractive lotteries.

Given a schedule, the agent needs to compute all the payoff-probability pairs that constitute the lottery, i.e. needs to compute how probable each outcome is and its payoff.

We represent the payoff-probability pairs in an *event tree*, where an event corresponds to a set of failed and completed tasks. We assume that once a task fails, the agent cancels all tasks that were not yet started. Each task in progress will proceed until its scheduled finish time, and is paid for on success. If all the tasks in the task network are completed successfully, the agent receives a final reward, which is paid as soon as all tasks are successfully completed.

In the event tree in Figure 3, we note in the framed boxes two payoffs and corresponding probabilities on two different branches of the tree. The branches represent respectively the situation when tasks 1 and 3 succeeded, task 4 and task 2 failed, and the situation when tasks 1 and 3 succeeded, task 4 failed, and task 2 succeeded. We $\tilde{c}_i$ to indicate the present payoff for task i.[1]

The unconditional probability that task i will be completed successfully is computed as

$$\tilde{p}_i^{\mathrm{c}} := \tilde{p}_i \times \prod_{j\in\tilde{P}(i)} \tilde{p}_j, \tag{1}$$

where $\tilde{P}(i)$ is a set of the *precursors* of task i — all tasks that finish before task i starts in the schedule.

Each way of scheduling the tasks produces a different lottery, and the CE values of each lottery might be different. Our objective is to find CE maxima. As we will discuss later, we do not need to find a global maximum, local maxima are often sufficient.

3. Properties of the Problem Domain

Our domain exhibits several properties that make it particularly hard to search for a global or, at least, predictably good local maxima.

The CE function is piece-wise continuous. It is easy to see that the certainty equivalent is a continuous function of task start and finish times as far as it is restricted to one particular event tree. We argue that in almost all generic problems there is a discontinuity of CE values between two arbitrarily close schedules with different corresponding event trees.

1 All the variables that depend on the current task schedule are "wiggled."

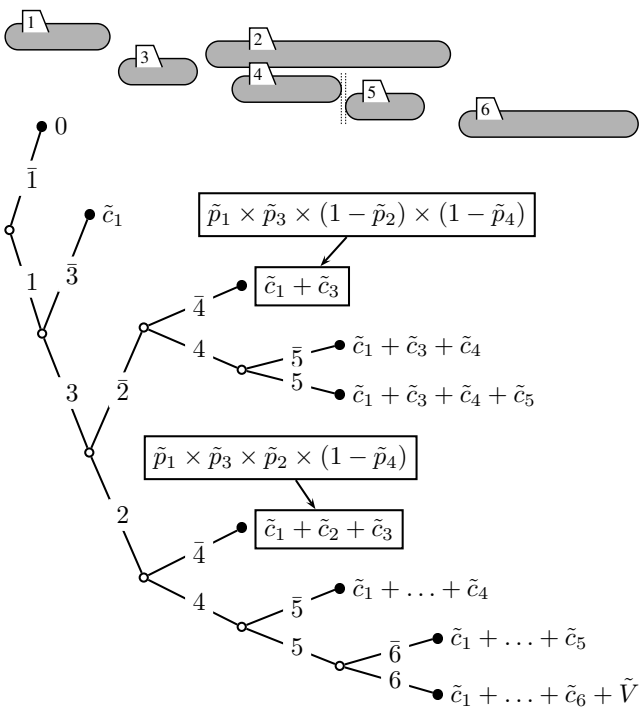

Figure 3. A schedule and the corresponding event tree.

Indeed, event trees change when the start time of some task is scheduled in a different order with the finish time of another task. Consider, for example, the schedule and its corresponding event tree in Figure 3. If the agent changes the order of task 4 finish time and task 5 start time, it is now liable for paying on task 5 success even if task 4 fails. Assuming that changes to the present values of payoffs and the probabilities of success of all tasks are negligible, this results in two payoff-probability pairs outlined in the event tree being replaced with the following pairs contingent on task 5 success:

$$\begin{aligned}
&(\tilde{c}_1 + \tilde{c}_3,\ \tilde{p}_1\tilde{p}_3(1-\tilde{p}_2)(1-\tilde{p}_4)(1-\tilde{p}_5))\\
&(\tilde{c}_1 + \tilde{c}_3 + \tilde{c}_5,\ \tilde{p}_1\tilde{p}_3(1-\tilde{p}_2)(1-\tilde{p}_4)\tilde{p}_5)\\
&(\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3,\ \tilde{p}_1\tilde{p}_3\tilde{p}_2(1-\tilde{p}_4)(1-\tilde{p}_5))\\
&(\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3 + \tilde{c}_5,\ \tilde{p}_1\tilde{p}_3\tilde{p}_2(1-\tilde{p}_4)\tilde{p}_5)
\end{aligned}$$

As long as $\tilde{c}_5 < 0$ and $\tilde{p}_5 > 0$ this results in different CE values of a schedule for a.e. value of the risk-aversity r.

The number of the continuity intervals of the CE function grows exponentially with the number of tasks in the worst case scenario (parallel tasks with no constraints). Precedence constraints in general will reduce this growth; in the case where all the tasks are in a single sequence, there is just one interval. However, even for simple task networks the number of the intervals of continuity (each corresponding to an event tree) is staggering.

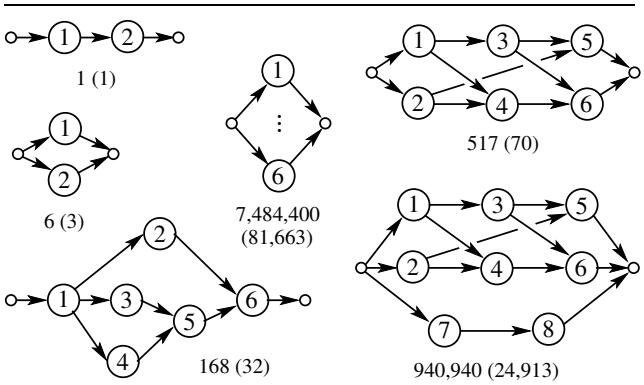

Figure 4. Number of start and finish times orderings (and number of event trees).

Figure 4 demonstrates several task networks, including our sample networks in Figures 1 and 2, together with the number of feasible strict orderings of start and finish times, and the number of corresponding event trees in parenthesis. There is a one to many correspondence between the event trees and the orderings. In our previous work we used the orderings as a base for the maximization algorithms; in this paper we use the event trees, which allows us to uniquely match CE maxima to the intervals of continuity.

Each interval of continuity has its own local maximum. Indeed, in all but special cases it has a degenerate maximum with zero CE value, which translates to abandoning the plan altogether. It almost always has other degenerated maxima with negative CE. And, sometimes, it also has one or more maxima with positive CE values. This property effectively prevents us from arranging intervals in order of clear preference, making the comprehensive exploration of the event trees the only reliable way to find a global maximum.

Local maximization algorithms are "lazy": given a starting point with negative CE value, they tend to converge to degenerate maximum with zero or negative value. Moreover, this "laziness" grows together with the risk-aversity.

Maximization algorithms are highly multidimensional. Our previous study [2] showed that not every maximization approach produces reliable results. Those which do, require special coordinate systems with up to $2N$ dimensions in addition to the original $2N$.

We do not know of any analytic formula for the expected utility computation. In addition to this, we cannot put any assumption on the probability of success distributions, since we assume they are derived from the real-world data. Thus we have to use a recursive algorithm for computing the payoff-probability pairs each time we need to find a CE

value. This imposes a computational tax on every prospective maximization method.

* * *

To summarize the above, to find a global maximum in our domain we have to explore every event tree using intrinsically ineffective and "lazy" methods. And even when we do so, we are not guaranteed to succeed.

Nevertheless, our domain has a number of important properties that we can use to create efficient domain-specific maximization methods.

We can use a variety of continuation techniques to overcome the "laziness" of the local maximization algorithms. For example, we can start with finding a maximizing schedule for a low risk-aversity value, and gradually increase the risk-aversity to its real value while tracking changes to the maximizing schedule. The same technique can be used with increasing the final reward or adding extra time slack to the plan.

We can decrease the impact of the high number of dimensions by maximizing along some projection before proceeding with a full-dimensional maximization.

It is possible to use one maximizing schedule to find others. We found that the large number of event trees and the corresponding local maxima are due to changes in scheduling tasks off the critical path. Not only such maxima are similar to each other in CE values, they can also be derived from each other [2] at a relatively small computational cost.

Most remarkably, we don't have to find the global maximum, a good set of local ones will fare better in most cases. The reason is that, when scheduling tasks for which there are few suppliers, the agent might need a lot of time slack to have any hope of receiving compatible bids [1]. The global maximum will often correspond to a schedule where there is no much slack left on the critical path and around it. Local maxima, especially those with more tasks scheduled in parallel, have more slack and hence might be preferable.

4. Direct Maximization Methods

In [2] we presented three constrained maximization approaches for studying the properties of the problem domain. The first one of these methods was based on the explicit enumeration of all precedence constraints, it produced a large number of pseudo-maxima where the algorithm got stuck on a plateau. Our attempt to improve this method by imposing extra task ordering constraints produced nearly the same results.

In our third and successful attempt we projected a set of ordering constraints in a space where they do not interact directly. To achieve this we fix the order of task start and finish times, and build a correspondence between points of a $(2N+1)$–dimensional unit cube and the ordered $2N$–dimensional vector of task times. This projection reflects proportions in which task start and finish times divide the $[t^{\mathrm{s}}, t^{\mathrm{f}}]$ interval.

Algorithm: $C \leftarrow$ **startTasks**(T, D)
Requires: T "*started tasks*", D "*finished tasks*"
Returns: C "*number of orderings*"

$C \leftarrow 0$
$X \leftarrow \{i \in N | P_1(i) \subset D, i \notin T \cup D\}$ "*tasks to start*"
foreach $Y \in 2^X \setminus \{\varnothing\}$
$C \leftarrow C +$ **finishTasks**$(T \cup Y, D)$
return C

Algorithm: $C \leftarrow$ **finishTasks**(T, D)
Requires: T "*started tasks*", D "*finished tasks*"
Returns: C "*number of orderings*"

if $X \cup T = \varnothing$ "*no tasks left to start*"
$C \leftarrow 1$
else "*can start some tasks*"
$C \leftarrow 0$
foreach $Y \in 2^T \setminus \{\varnothing\}$
$C \leftarrow C +$ **startTasks**$(T \setminus Y, D \cup Y)$
return C

Figure 5. Counting of event trees.

Two disadvantages of the last method are: (a) the large number of possible task orderings that must be investigated in the search for global maxima, (b) many-to-one correspondence between task orderings and event trees, resulting in finding multiple copies of the same maxima. In this paper we present methods derived from our successful effort and based on the enumeration of event trees.

Figure 5 shows the algorithm for counting unique event trees, Figure 6 illustrates its operation. One or more tasks start after each start strut **s**, shown by a solid vertical line. Similarly, a non-empty set of already started tasks finish after each finish strut **f**, shown by a dashed vertical line.

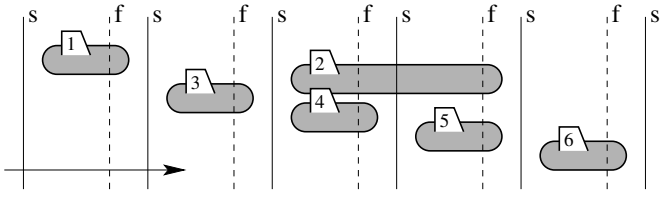

Figure 6. Execution path of the event tree enumeration algorithm corresponding to the event tree in Figure 3.

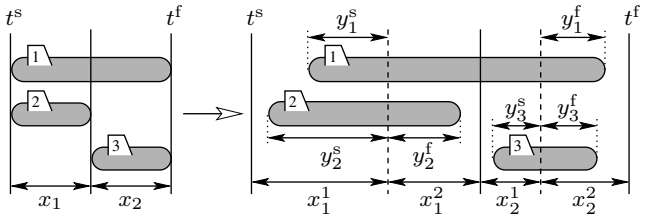

Figure 7. Conversion of a "tied" maximization coordinate system to full dimensions.

Each of the following three maximization methods explores one event tree at the time. They each maximize on a unit cube of some dimension, starting from a randomly selected point. The main difference between these methods is the projection methods they use to internalize the event tree structure.

Event Tree Maximization (ETM) is a direct full-dimensional method. For a given event tree we build a correspondence between $2N$ task start and finish times and a $(2N + 2M)$–dimensional unit cube, where M is the number of finish struts (refer to the right part of Figure 7). We use $2M$ variables x_j^1 and x_j^2 to reflect the ratios, in which start and finish struts break the $[t^{\mathrm{s}}, t^{\mathrm{f}}]$ interval. We use the other $2N$ variables y_i^{s} and y_i^{f} to show the proportions occupied by task starts and finishes in the intervals between adjacent struts.

In this coordinate system, the start and finish time of task 1 are expressed as

$$\begin{aligned} t_1^{\mathrm{s}} &= t^{\mathrm{s}} + \alpha x_1^1 (1 - y_1^{\mathrm{s}}) \\ t_1^{\mathrm{f}} &= t^{\mathrm{s}} + \alpha (x_1^1 + x_1^2 + x_2^1 + x_2^2 y_1^{\mathrm{f}}) \\ \alpha &= (t^{\mathrm{f}} - t^{\mathrm{s}}) / (x_1^1 + x_1^2 + x_2^1 + x_2^2) \end{aligned}$$

Tied Event Tree Maximization (TETM) addresses the issue of a high number of dimensions in ETM method via two-stage approach. First, we "tie" starts and finishes of tasks to the corresponding start struts and omit all finish struts, thus leaving only M dimensions. If the maximization in this smaller space is successful, we "untie" the task time windows and maximize once again. The last transition is defined as follows:

$$x_j^1 = x_j^2 = x_j, \qquad y_i^{\mathrm{s}} = y_i^{\mathrm{f}} = 1$$

Expanded Tied Event Tree Maximization (xTETM) is used when there is a lot of time slack in the plan, e.g., when the agent schedules a week-long plan to be performed some time during a full year. In such hypothetical case TETM is likely to fail, since its "tied" part always schedules the final reward at t^{f}. xTETM addresses this by adding two more intervals to the beginning and the end of the M–dimensional "tied" coordinate system.

Task network A (Figure 1)				
	ETM		TETM	
r	CE	%	CE	%
-0.05	32.44	3.36	32.44	39.25
-0.04	31.46	2.60	31.46	35.30
-0.03	30.18	1.72	30.18	32.17
-0.02	28.45	1.33	28.45	27.97
-0.01	26.03	0.82	26.03	24.37
0	22.70	0.46	22.70	20.00
0.01	18.29	0.19	18.29	15.83
0.02	13.30	0.11	13.30	12.34
0.03	7.84	0.03	7.86	7.87
0.04			2.51	2.92

Task network B (Figure 2)						
	ETM		TETM		xTETM	
r	CE	%	CE	%	CE	%
-0.05	150.3	53.7	150.3	84.7	144.8	11.7
-0.04	149.0	41.4	149.0	79.4	143.3	11.7
-0.03	146.9	29.1	146.9	71.5	141.3	10.0
-0.02	143.4	18.1	143.4	61.9	137.9	8.2
-0.01	136.0	9.2	134.7	49.9	131.2	6.1
0	119.2	4.2	118.9	37.7	115.6	4.6
0.01	87.9	1.6	87.9	26.4	81.5	3.0
0.02	43.9	0.5	43.9	16.8	36.4	1.4
0.03	11.2	0.1	11.2	5.7	5.1	0.2

Table 1. Results of maximization.

5. Comprehensive Maximization Results

We tested the three described event tree maximizations methods for the reducible task network A in Figure 1 and the irreducible task network B in Figure 2. In each of the experiments we performed 1000 maximizations for each event tree for values of risk-aversity r from -0.05 to 0.05 with 0.01 increment.

In the two parts of Table 1 we summarize the results of all runs, except for the xTETM experiment for the network A. In column 'CE' we list the maximum attained value of the certainty equivalent for each particular value of r for each maximization method. It may be regarded as a best-effort approximation of the global maximum. Column '%' shows the percentage of maximization attempts that resulted in a local maximum with a strictly positive CE value.

Observe that the original ETM method exhibits a very poor performance for high risk-aversity values. It also performs better in the case of network B, which was intentionally set up to have extra time slack. The only area in which it performs better than its TETM extension is where there exist maxima with no clear path to them from "untied" schedules. For example, for $r = 0$ and $r = 0.01$ for the task net-

work B it finds higher maximum CE values while converging to local maxima with much smaller probability.

The overall performance of TETM method is remarkable, although it looses the percentage of found maxima as r increases, and does not find some specific maxima that ETM does. On the other hand, it also produces the results where TETM falls short, such as in the case of $r = 0.04$ for the network A.

The worst performance of all is exhibited by the xTETM method: it fails to produce any results for the network A. This behavior, however, is predictable, since none of two setups, especially the first one, has nearly enough time slack. Nevertheless, we felt it is important to list this method among others, since it complements TETM in the cases where there is a substantial time slack in the plan.

6. "Jumping" between Event Trees

Although we observe that direct maximization methods perform reasonably well for our sample networks, it must not be overlooked that they do so at a high computational cost. This cost grows together with the number of event trees, that is, exponentially in the worst case. There is also an issue of a relatively poor performance in the cases of high risk-aversity values — those most interesting for us.

The analysis of the problem domain, briefly summarized in Section 3, suggests that we don't need to explore all possible event trees. However, randomly selecting the trees from some list is not a reasonable solution: firstly, it might not be feasible to create this list for large problems; secondly, there is no reason to believe that the arbitrary sample of event trees has maxima close to the global, if any. An alternative way is to exploit the property that allows us to "jump" between sufficiently similar schedules by changing their underlying event trees and rearranging their task start and finish times.

Figure 8 demonstrates five simple rules for changing one schedule to another similar schedule, while possibly preserving the maximizing properties of the former. Each rule must obey the precedence relations.

The rules are as follows:

Rule 1: Change starts and finishes of a set of adjacent tasks simultaneously:

- **Rule 1(a):** Merge sequential tasks with no precedence relations between them to form a set of parallel tasks.
- **Rule 1(b):** Swap some start and finish times.
- **Rule 1(c):** Split a set of parallel tasks in two arbitrary nonempty subsets with randomly chosen shares of the original set's time.

Rule 2: Shuffle only start times.

Rule 3: Shuffle only finish times.

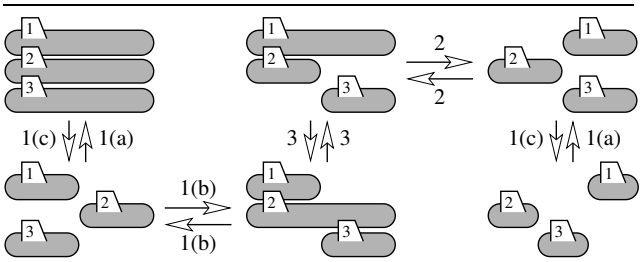

Figure 8. Jumping between schedules for 3 parallel tasks.

Jumping between trees is best coupled with the "tied" maximizations approach, otherwise it is too complex. Using these rules eliminates the need for building the complete list of event trees, since they might be applied to any single tree to obtain neighboring trees. In addition, we can guide the jumping process by refusing to move to lower CE values and thus directing it towards higher values.

7. Stochastic Maximization Method

We employ the idea of jumping between event trees and the continuation methods described earlier to create a new stochastic maximization method. We further augment it with a notion of a Simulated Annealing temperature [12] to encourage its moving towards event trees and local maxima with higher CE values.

The following list outlines the resulting algorithm:

1. First, for an arbitrary event tree we find some local maximum for a low risk-aversity r_{low}, and use continuation techniques to track it to a maximum for the target r_{high}. If the CE value of found maximum is not strictly positive, we repeat this step.
2. Next, we explore neighboring event trees to find a higher CE value. To do that we apply our set of 5 jumping rules to randomly select a neighbor. When it is found, we convert the current maximizing schedule for r_{low} to one compatible with the selected event tree.
3. With some probability we repeat the previous step. This helps us to get out of situation where all the new neighbors have much smaller local maxima and the annealing temperature T is low.
4. We maximize starting from the schedule for r_{low}, and use continuation techniques to obtain a maximum for r_{high}. If the CE value of found maximum is not strictly positive, we return to step 2.
5. If the found maximum CE value is higher than what we found before, we accept the new schedule and proceed to step 2. As the annealing temperature T drops,

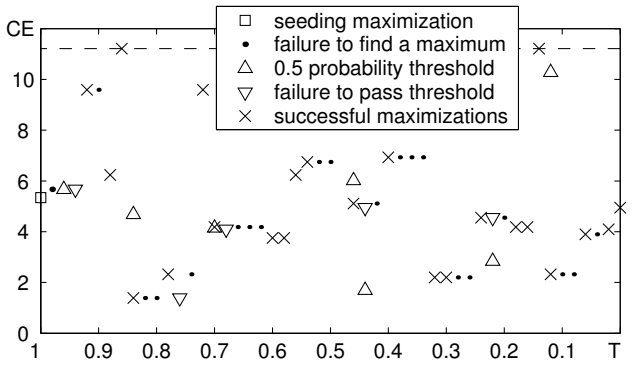

Figure 9. One run of the stochastic method for the task network in Figure 2, $dT = 0.04$, $r_{\text{low}} = -0.05$ and $r_{\text{high}} = 0.03$.

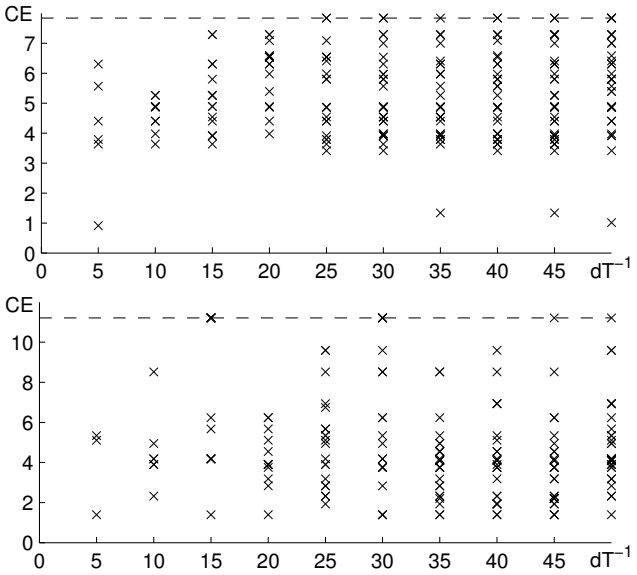

Figure 10.

we refuse to go to lower CE values with increasingly higher probability.

Figure 9 illustrates one run of the stochastic algorithm for the task network B and temperature decrement $dT = 0.04$, i.e. for a 25 steps long run. We use a square marker to label the result of maximization in step 1 and a cross for every subsequent successful maximization in step 4. We also put a dot at the level of the previous successful maximization whenever step 4 fails to produce a result. A downward triangle shows CE values that did not pass the annealing test, in step 5, while upward ones show CE values that should pass this test with a probability 0.5. Note that in this particular run the algorithm reached the maximum CE value found by the direct maximization algorithms (shown by a horizontal dashed line) almost instantly.

We examined the stochastic algorithm for both sample task networks for different values of dT, and summarized the findings in Figure 10. The x-axis on each plot shows the number of steps of the algorithm (i.e. dT^{-1}); the y-axis shows CE values. Cross markers correspond to found local maxima, and the dashed horizontal line shows the maximum CE value obtained from the direct maximization experiments.

Based on these test runs we might conclude that the stochastic maximization algorithm delivers a wide variety of local maxima, often reaching the best maxima found by teh more elaborate methods in a matter of a few steps.

8. Related Work

Many multi-agent systems use some form of auction [9] to allocate resources and arrive at coordinated decisions. The Contract Net [14] is perhaps the most well known and widely used protocol for distributed problem solving. Auctions are traditionally used for self-interested agents [19, 5], but they are being used also for cooperative agents [8, 7]. When auctions are used to distribute tasks [6] or to schedule a resource [18], items are typically auctioned one at a time. This reduces the opportunity for optimal allocations, and tends to make the systems reactive but myopic. In our prior work we have extended winner determination algorithms to include not only costs but also scheduling constraints. This enables the agents to combine the advantages of planning with the convenience of auctions.

Agents in MASCOT [13] coordinate scheduling with the user. Their major objective is to show policies that optimize schedules locally. Our objective is to optimize the expected utility before bids are submitted and schedules are finalized.

In [15] atemporal and temporal goods are considered. Temporal goods are collected into bundles that represent a good available over a time interval. This increases significantly the computational complexity. A protocol for combinatorial auctions for supply chain formation is proposed in [16]. Complex task networks are allowed, but they do not include time constraints. A protocol for decentralized scheduling is proposed in [19]. The study is limited to scheduling a single resource, while we are interested in multiple resources. In [18] agents bid for individual time slots on separate, simultaneous markets. Our agents use combinatorial bids.

The complexity of job-shop scheduling [17] is similar to the complexity our agents face. Our problem is not job-shop scheduling; we are not scheduling resources the agent has. Instead we are producing a schedule of tasks that other

agents will carry out. Our objective is to schedule tasks in a way that optimizes the expected utility of the agent.

Our results show that the specifics of how tasks are scheduled in a RFQ affect the quality and quantity of bids [1]. Elicitation of preferences is known to play a significant role in reducing the number of bids submitted to combinatorial auctions [4] and in auction mechanism design [20]. However, we are not aware of any research that has studied the dependencies between task schedules, bids, and costs of accomplishing the tasks.

9. Conclusions

We presented three new methods for maximizing an agent's Expected Utility over different schedules of tasks. The maximization problem is hard mostly because the domain is piece-wise continuous, with an exponentially large number of pieces, and because the maximization algorithms tend to converge to degenerate maxima.

The methods we presented take advantage of the fact that the function to be maximized is continuous over a single event tree. We use domain-specific heuristics to guide the search. We also create the effective stochastic maximization method and illustrate how all the methods work for two substantially different problem setups.

The methods have been developed in the context of supporting the decisions an agent has to make when generating a RFQ. This is an important problem, since the ability of agents to bid for tasks depends on their previous commitments, so different settings of time windows in a RFQ will end up soliciting different bids with different costs. The same methods can be used to solve the decision problem an agent has when submitting bids for tasks that have complex time constraints and interdependencies.

10. Acknowledgments

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

References

[1] A. Babanov, J. Collins, and M. Gini. Asking the right question: Risk and expectation in multi-agent contracting. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 17(4):173–186, September 2003.

[2] A. Babanov, J. Collins, and M. Gini. Scheduling tasks with precedence constraints to solicit desirable bid combinations. In *Proc. of the Second Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 345–352, Melbourne, Australia, July 2003.

[3] J. Collins, W. Ketter, and M. Gini. A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints. *Int'l Journal of Electronic Commerce*, 7(1):35–57, 2002.

[4] W. Conen and T. Sandholm. Preference elicitation in combinatorial auctions. In *Proc. of the First Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, volume 1, pages 168–169, Bologna, Italy, July 2002.

[5] P. S. Dutta, S. Sen, and R. Mukherjee. Scheduling to be competitive in supply chains. In *IJCAI workshop on E-Business and the Intelligent Web*, August 2001.

[6] B. P. Gerkey and M. J. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Trans. Robotics and Automation*, 18(5):758–786, October 2002.

[7] L. Hunsberger. Generating bids for group-related actions in the context of prior commitments. In *Intelligent Agents VIII*, volume 2333 of LNAI. Springer-Verlag, 2002.

[8] L. Hunsberger and B. J. Grosz. A combinatorial auction for collaborative planning. In *Proc. of 4th Int'l Conf on Multi-Agent Systems*, pages 151–158, Boston, MA, 2000. IEEE Computer Society Press.

[9] V. Krishna. *Auction Theory*. Academic Press, London, UK, 2002.

[10] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, January 1995.

[11] J. W. Pratt. Risk aversion in the small and in the large. *Econometrica*, 32:122–136, 1964.

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

[13] N. M. Sadeh, D. W. Hildum, D. Kjenstad, and A. 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, 1999.

[14] R. G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Trans. Computers*, 29(12):1104–1113, December 1980.

[15] W. Walsh and M. Wellman. A market protocol for decentralized task allocation and scheduling with hierarchical dependencies. In *Proc. of 3th Int'l Conf on Multi-Agent Systems*, 1998.

[16] W. E. Walsh, M. Wellman, and F. Ygge. Combinatorial auctions for supply chain formation. In *Proc. of ACM Conf on Electronic Commerce (EC'00)*, pages 260–269, October 2000.

[17] J. P. Watson, J. C. Beck, A. Howe, and L. D. Whitley. Problem difficulty for tabu search in job-shop scheduling. *Artificial Intelligence*, pages 189–217, 2002.

[18] M. Wellman, J. MacKie-Mason, D. Reeves, and S. Swaminathan. Exploring bidding strategies for market-based scheduling. In *Proc. of Fourth ACM Conf on Electronic Commerce*, 2003.

[19] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.

[20] P. R. Wurman, M. P. Wellman, and W. E. Walsh. Specifying rules for electronic auctions. *AI Magazine*, 23(3):15–24, 2002.