Risk and Expectations in a-priori Time Allocation in Multi-Agent Contracting

Alexander Babanov
Dept of Computer Science and Engineering
Dept of Economics
University of Minnesota
babanov@cs.umn.edu

John Collins
Dept of Computer Science and Engineering
University of Minnesota
jcollins@cs.umn.edu

Maria Gini
Dept of Computer Science and Engineering
University of Minnesota
gini@cs.umn.edu

ABSTRACT

In related research we have proposed a market architecture for multi-agent contracting and we have implemented prototypes of both the market architecture and the agents in a system called MAGNET. A customer agent in MAGNET solicits bids for the execution of multi-step plans, in which tasks have precedence and time constraints, by posting a Request for Quotes to the market. The Request for Quotes needs to include for each task its precedence constraints and a time window. In this paper, we study the problem of optimizing the time windows in the Requests for Quotes. Our approach is to use the Expected Utility Theory to reduce the likelihood of receiving unattractive bids, while maximizing the number of bids that are likely to be included in the winning bundle. We describe the model, illustrate its operation and properties, and discuss what assumptions are required for its successful integration into MAGNET or other multi-agent contracting systems.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: E-commerce; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Economics, Theory

Keywords

Automated auctions, multi-agent contracting, expected utility, risk estimation

1. INTRODUCTION

The MAGNET (Multi-AGent NEgotiation Testbed) [4] system is designed to support multiple agents in negotiating contracts for tasks with complex temporal and precedence constraints.

We distinguish between two agent *roles*, the *customer* and the *supplier*. A customer is an agent who needs resources outside its direct control in order to carry out his plans. It does so by soliciting the help of other self-interested agents through a Requests for Quotes (RFQ). A supplier is an agent who, in response to an RFQ, may offer to provide the requested resources or services for specified prices, over specified time periods. The objective of the agents is to maximize their profits while predicting and managing their financial risk exposure.

In this paper, we focus on the decision process a customer agent needs to go through in order to generate an RFQ. We study in particular the problem of how to specify the time windows for the different tasks in the RFQ. This decision determines an approximate schedule by setting limits on the start and finish times for each individual task, since the RFQ includes early start and late finish times for each task. Because there is a 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.

We show how to use the Expected Utility Theory to determine the time windows for tasks in the task network, so that bids that are close to these time windows form the most preferred risk-payoff combinations for the customer agent. We further examine to what extent the behavior of the model corresponds to our expectations, explain what market information needs to be collected in order to integrate the model in MAGNET system and, finally, discuss how to use the resulting time allocations to construct RFQs.

2. RELEVANCE OF THE PROBLEM

Before presenting our proposed solution, we need to understand the importance of selecting appropriate time windows for tasks in RFQs and how this choice affects the customer agent's ability to accomplish the tasks as economically and rapidly as possible.

Choosing appropriate time windows affects the number and price of the bids received, the ability to compose the bids into a feasible schedule, and the financial exposure of the customer agent.

There are two major decisions the agent has to take here: the relative allocation of time among the different tasks, and the extent to which the time windows of tasks connected by precedence relations are allowed to overlap.

AAMAS'02, July 15-19, 2002, Bologna, Italy.

We have shown [1] that the time constraints specified in the RFQ can affect the customer's outcome in two major ways:

1. by affecting the number, price, and time windows of the bids submitted. We assume that bids will reflect supplier resource commitments, and therefore larger time windows will result in more bids and better utilization of resources, in turn leading to lower prices [3]. However, an RFQ that features overlapping time windows makes the process of winner determination more complex [2]. Another less obvious problem is that every extra bid over the minimum needed to cover all tasks adds one more rejected bid. Ultimately, a large percentage of rejections will reduce the customer agent's credibility, which, in turn, will result in fewer bids and/or higher costs.
2. by affecting the financial exposure of the customer agent. We assume non-refundable deposits are paid to secure awarded bids, and payments for each task are made as the tasks are completed. The payment to the customer occurs only at the completion of all the tasks. Once a task starts and, in case it is successfully completed in the time period specified by the contract, the customer is liable for its full cost, 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.

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 to the customer of the work completed.

The uncertainty of whether the tasks will be completed on time as promised by suppliers further complicates the decision process. 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 fails to perform or decommits there are other suppliers willing to do the task and there is sufficient time to recover without invalidating the rest of the schedule.

If a task is not completed by the supplier, the customer agent is not liable for its cost, but this failure can have a devastating effect on other parts of the plan. Having slack in the schedule increases the probability that tasks will be completed successfully or that there will be enough time to recover if one of the tasks fails. However, slack extends the completion time and so reduces the reward. In made-to-order products speed is the essence and taking extra time might prevent a supplier from getting a contract. 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 also be reduced by consolidating tasks with fewer 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 of doing this is greater toward the end of the plan than near the beginning, since at that point the customer has already paid a significant part of the tasks. Having a greater financial exposure provides an additional incentive to reduce risk.

3. THE MAGNET FRAMEWORK

3.1 General Terms

The *customer* is a human or artificial agent who wants to achieve some goal and needs resources or services beyond her direct control.

The *supplier* is a human or artificial agent who has direct control over some resources or services and may offer to provide those in response to external request, i.e., may submit and commit to bids.

The *mediator* is a MAGNET-assisted human agent who meets the needs of a customer by negotiating over multiple goods or services with one or more suppliers. We often refer to the artificial part of the duo as to the *customer agent*.

The *Request for Quotes* is a signal composed by the customer agent on the basis of the customer's needs and is sent to solicit suppliers' bids. MAGNET is a mixed initiative system, so between composing RFQs and sending them out, there is a stage where a human user can impose her preferences on the RFQ choices.

3.2 Task Network

The *task network* (see Figure 1) represents the structure of the customer's plan. In essence, it is a connected directed acyclic graph.

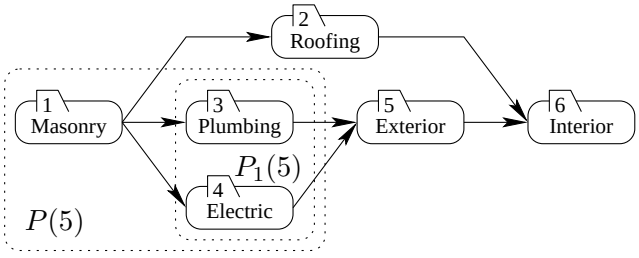

Figure 1: A task network example.

Mathematically speaking, a task network is a tuple $\langle N, \prec \rangle$ of a set N of individual tasks and strict partial ordering on them. We conveniently abuse N to also denote the number of tasks. A task network represents a *plan* to accomplish the agent's goal.

We define $P(n) := \{m \in N | m \prec n\}$ to be a set of *predecessors* of $n \in N$, where every predecessor m should be completed before task n might start. Note that, in general, $P(n)$ is not completely ordered by $\prec$.

Similarly, $S(n)$ is to denote a set of *successors* of n.

3.3 Time Allocation and Probabilities

A task network is characterized by a *start time* t^{s} and a *finish time* t^{f}, which delimit the interval of time when tasks can be scheduled.

The placement of an individual task n in the schedule is characterized by its *start time* t_n^{s} and *finish time* t_n^{f}, which are subject to the following constraints:

$$t^{\mathrm{s}} \leq t_m^{\mathrm{f}} \leq t_n^{\mathrm{s}}, \qquad \forall m \in P_1(n)$$
$$t_n^{\mathrm{f}} \leq t_m^{\mathrm{s}} \leq t^{\mathrm{f}}, \qquad \forall m \in S_1(n)$$

where $P_1(n)$ is the a set of *immediate predecessors* of n, $P_1(n) = \{m \in N | m \prec n, \nexists m' \in N, m \prec m' \prec n\}$. $S_1(n)$ is defined similarly to be the set of *immediate successors* of task n.

The *probability of task n completion* by the time t, conditional on the successful completion of task n, is distributed according to the cumulative distribution function (CDF) $\Phi_n = \Phi_n(t_n^{\mathrm{s}}; t), \Phi_n(\cdot; \infty) = 1$. Observe that Φ_n is defined to be explicitly dependent on the start time t_n^{s}. To see the rationale, consider the probability of successful mail delivery in x days for packages that were mailed on different days of a week.

There is an associated unconditional *probability of success* $p_n \in [0, 1]$ characterizing the percentage of tasks that are successfully completed given infinite time (see Figure 2).

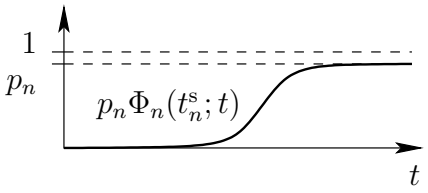

Figure 2: Unconditional distribution for successful completion probability.

3.4 Monetary Transfers

Task n bears an associated *cost*[1]. We assume the total cost of task n has two parts: a deposit, which is paid when the task starts, and a cost c_n which is due at some time after successful completion of n. Since we never compare plans with different deposits we assume without loss of generality the deposit to be 0.

There is a single *final payment* V scheduled at the plan finish time t^{f} and paid conditional on all tasks in n being successfully completed by that time.

There is an associated *rate of return* q_n[2] that is used to calculate the *discounted present value* (PV) for payoff c_n due at time t as

$$\mathrm{PV}(c_n; t) := c_n (1 + q_n)^{-t}.$$

We associate the return q with the final payment V.

4. EXPECTED UTILITY

4.1 General Terms

We represent the customer agent's preferences over payoffs by the von Neumann-Morgenstern utility function u. We further assume that the *absolute risk-aversion coefficient* [14] $r := -u''/u'$ of u is constant for any value of its argument, hence u can be represented as follows:

$$u(x) = \begin{cases} -\exp\{-rx\} & \text{for } r \neq 0 \\ x & \text{for } r = 0 \end{cases}$$

A *gamble* is a set of payoff-probability pairs $G = \{(x_i, p_i)_i\}$ s.t. $p_i > 0, \forall i$ and $\sum_i p_i = 1$. The expectation of the utility function over a gamble G is the *expected utility* (EU):

$$Eu[G] := \sum_{(x_i, p_i) \in G} p_i u(x_i)$$

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

$$\mathrm{CE}(G) = \begin{cases} -\frac{1}{r} \log \sum_{(x_i, p_i) \in G} p_i \exp\{-rx_i\} & \text{for } r \neq 0 \\ \sum_{(x_i, p_i) \in G} p_i x_i & \text{for } r = 0 \end{cases}$$

Naturally, the agent will not be willing to accept gambles with less than positive certainty equivalent and the higher values of the certainty equivalent will correspond to more attractive gambles.

To illustrate the concept, Figure 3 shows how the certainty equivalent depends on the risk-aversity of an agent. In this figure we consider a gamble that brings the agent either 100 or nothing with equal probabilities. Agents with positive r's are risk-averse, those with negative r's are risk-loving. Agents with zero risk-aversity zero, i.e. risk-neutral, have a CE equal to the gamble's weighted mean 50.

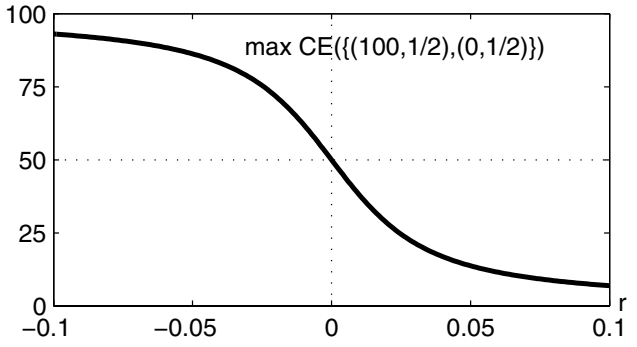

Figure 3: Certainty equivalent of a simple gamble as a function of the risk-aversity.

4.2 Cumulative Probabilities

To compute the certainty equivalent of a gamble we need to determine a schedule for the tasks and compute the payoff-probability pairs.

We assume that a payoff c_n for task n is scheduled at t_n^{f}, so its present value $\tilde{c}_n$[3] is

$$\tilde{c}_n := c_n (1 + q_n)^{-t_n^{\mathrm{f}}}$$

[1]Hereafter we use words "cost" and "reward" to denote some monetary value, while referring the same value as "payoff" or "payment" whenever it is scheduled at some time t.

[2]The reason for having multiple q_n's is that individual tasks can be financed from different sources, thus affecting task scheduling.

[3]Hereafter we "wiggle" variables that depend on the current task schedule, while omitting all corresponding indices for the sake of simplicity.

We define the conditional probability of task n success as

$$\tilde{p}_n := p_n \Phi_n \left(t_n^{\mathrm{s}}; t_n^{\mathrm{f}} \right).$$

We also define the *precursors* of task n as a set of tasks that finish before task n starts in a schedule, i.e.

$$\tilde{P}(n) := \left\{ m \in N | t_m^{\mathrm{f}} \leq t_n^{\mathrm{s}} \right\}.$$

The unconditional probability that the task n will be completed successfully is

$$\tilde{p}_n^{\mathrm{c}} = \tilde{p}_n \times \prod_{m \in \tilde{P}(n)} \tilde{p}_m.$$

That is, the probability of successful completion of every precursor and of the task n itself are considered independent events. The reason this is calculated in such form is because, if any task in $\tilde{P}(n)$ fails to be completed, there is no need to execute task n.

The probability of receiving the final payment V is therefore

$$\tilde{p} = \prod_{n \in N} \tilde{p}_n.$$

4.3 Example and Discussion

To illustrate the definitions and assumptions above, let's return to the task network in Figure 1 and consider a sample task schedule in Figure 4. In this figure the x-axis is time, the y-axis shows both the task numbers and for each individual task it also shows the cumulative distribution of the unconditional probability of completion (compare to Figure 2). Circle markers show start times t_n^{s}. Crosses indicate both finish times t_n^{f} and success probabilities $\tilde{p}_n$ (numbers next to each point). Square markers denote that the corresponding task cannot span past this point due to precedence constraints. Finally, the thick part of each CDF shows the time allocation for the task.

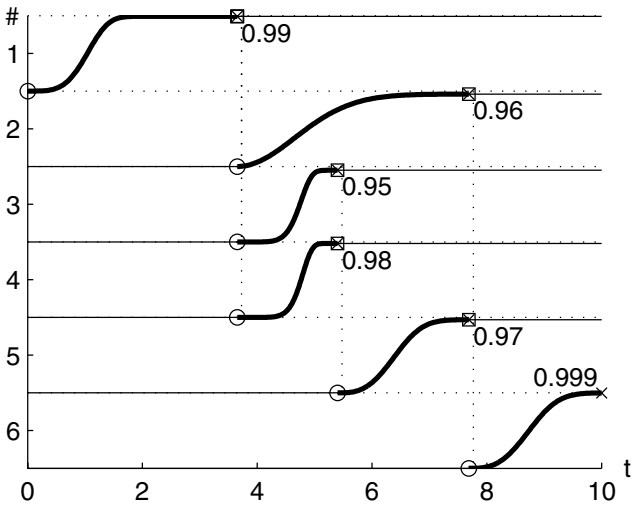

Figure 4: CE maximizing time allocations for the plan in Figure 1 for $r = -0.01$.

In practice, the customer agent needs a way of collecting the market information necessary to build and use the model. The probability of success is relatively easy to observe in the market. This is the reason for introducing the cumulative probability of success Φ_n and the probability of success p_n, instead of the average project life span or probability of failure or hazard function. Indeed, it is rational for the supplier to report a successful completion immediately in order to maximize the present value of a payment. Also it is rational not to report a failure until the last possible moment due to a possibility of earning the payment by rescheduling, outsourcing or somehow else fixing the problem.

To be specific, the information that the agent needs to collect is the empirical distribution of how long does it take from the point of starting some task to the point its completion is reported. This data, unlike the data on failures or actual positions in the supplier's schedule is less likely to be private or unobservable.

5. MAXIMIZATION

5.1 Gamble Calculation Algorithm

Given a schedule, like the one shown in Figure 4, we need to compute the payoff probability and then maximize the CE for the gamble. Writing an explicit description of the expected utility as a function of gambles is overly complicated and relies on the order of task completions. Instead we propose a simple recursive algorithm that creates these gambles. We then maximize the CE over the space of gambles. The proposed algorithm does not depend on the structure of the task network, but on the number of tasks scheduled in parallel.

Algorithm: $G \leftarrow$ **calcGamble**(T, D)
  Requires: T "*tasks to process*", D "*processed tasks*"
  Returns: G "*subtree gamble*"

$M \leftarrow \{m \in T | \tilde{P}(m) \subset D\}$
if $M \neq \varnothing$ "*it's a branch*"
  $n \leftarrow$ ***first***$\{M\}$ "*according to some ordering*"
  $T \leftarrow T \setminus \{n\}$
  $G \leftarrow \varnothing$
  $E \leftarrow$ **calcGamble**(T, D) "*follow* $\ldots \rightarrow \bar{n}$ *path*"
  forall $(x, p) \in E$
    $G \leftarrow G \cup \{(x, p \times (1 - \tilde{p}_n))\}$
  endfor
  $I \leftarrow$ **calcGamble**$(T, D \cup \{n\})$ "*follow* $\ldots \rightarrow n$ *path*"
  forall $(x, p) \in I$
    $G \leftarrow G \cup \{(x + \tilde{c}_n, p \times \tilde{p}_n)\}$
  endfor
  return G "*subtree is processed*"
else "*it's a leaf*"
  if $N = D$ "*all tasks are done*"
    return $\{(V, 1)\}$
  else "*some task failed*"
    return $\{(0, 1)\}$
  endif
endif

In the first call the algorithm receives a "todo" task list $T = N$ and a "done" task list $D = \varnothing$, all the subsequent calls are recursive. To illustrate the idea behind this algorithm, we refer to the payoff-probability tree in Figure 6. This tree was built for the time allocations in Figure 5 and reflects the precursor relations for this case.

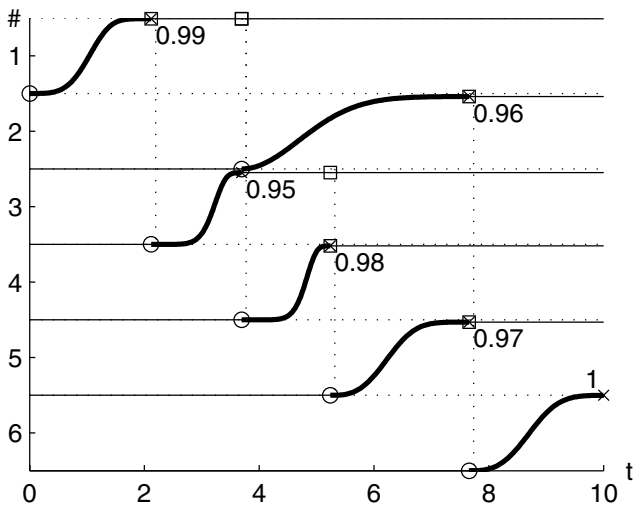

Figure 5: CE maximizing time allocations for the plan in Figure 1 for $r = 0.02$.

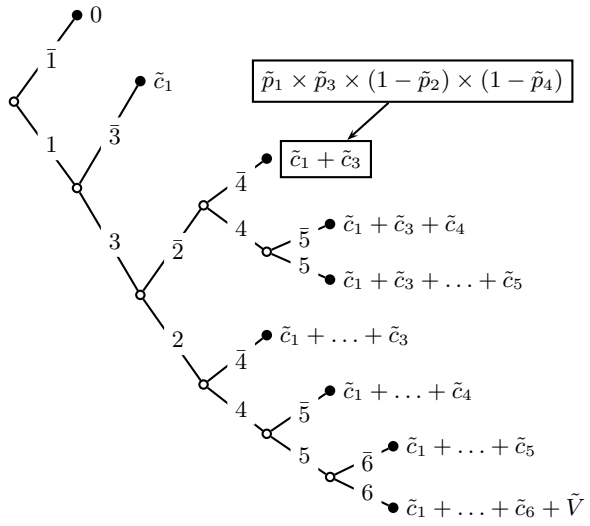

Figure 6: Payoff-probability tree for the time allocations in Figure 5.

Looking at the time allocation, we note that with probability $1 - \tilde{p}_1$ task 1 fails, the customer agent does not pay or receive anything and stops the execution (path $\bar{1}$ in the tree). With probability $\tilde{p}_1^{\mathrm{c}} = \tilde{p}_1$ the agent proceeds with task 3 (path 1 in the tree). In turn, task 3 either fails with probability $\tilde{p}_1 \times (1 - \tilde{p}_3)$, in which case the agent ends up stopping the plan and paying a total of c_1 (path $1 \rightarrow \bar{3}$), or it is completed with the corresponding probability $\tilde{p}_3^{\mathrm{c}} = \tilde{p}_1 \times \tilde{p}_3$.

In the case where both 1 and 3 are completed, the agent starts both 2 and 4 in parallel and becomes liable for paying c_2 and c_4 respectively even if the other task fails (paths $1 \rightarrow 3 \rightarrow 2 \rightarrow \bar{4}$ and $1 \rightarrow 3 \rightarrow \bar{2} \rightarrow 4$). If both 2 and 4 fail, the resulting path in the tree is $1 \rightarrow 3 \rightarrow \bar{2} \rightarrow \bar{4}$ and the corresponding payoff-probability pair is framed in the figure.

5.2 Computational Complexity

The computational complexity of the maximization procedure is determined by two parameters: first, the procedure itself is a non-linear maximization over $2N$ choice variables with internal precedence constraints. Second, to calculate a certainty equivalent value for every time schedule, the maximization procedure should be able to build a corresponding gamble and compute its expected utility.

For maximization we use the Nelder-Mead simplex (direct search) method from the `Matlab` optimization toolbox.

The complexity of the **calcGamble** algorithm shown before is $O\left(2^{K-1} \times N\right)$, where K is the maximum number of tasks that are scheduled to be executed in parallel.

The complexity estimate is based on the observation that the depth of the payoff-probability tree is N and that any subtree following an unsuccessful task execution has a depth of no more than $K - 1$. The last statement follows from the assumption that there are no more than $K - 1$ tasks running in parallel to the one that failed and therefore no other tasks will start after the failure was reported. Whether it is possible to create an algorithm with significantly lower computational costs is one of the questions we plan to address in future research.

In commercial projects the ratio K/N is usually low, since not many of these exhibit a high degree of the parallelism. Our preliminary experiments, reported in Section 5.3, allow us to conclude that K/N ratio is likely to be lower for risk-averse agents (presumably, businessmen) than for risk-lovers (gamblers).

5.3 Preliminary Experimental Results

We have conducted a series of experiments on the CE maximization. Some of the results are summarized in Figure 7. In this figure, the y-axis shows 11 different risk-aversity r settings, the bottom x-axis — time t in the plan, and the top x-axis — maximum CE value for each r setting. The rounded horizontal bars in each of 11 sections denote time allocations for each of six tasks with task 1 being on top. Sections $r = -0.01$ and $r = 0.02$ correspond to Figure 4 and Figure 5 respectively. Finally, the vertical bars show the maximum CE values.

Let's examine the relative placement of time allocations as a function of r. For this purpose we highlighted task 3 (black bars) and task 4 (white bars). Here task 3 has higher variance of CDF and lower probability of success than task 4 (0.032 and 0.95 vs. 0.026 and 0.98), also task 3 is more expensive (-15 vs. -7). There are four different cases in the experimental data:

1. Risk-loving agents tend to schedule tasks in parallel and late in time in order to maximize the present value of expected difference between reward and payoffs to suppliers. This confirms the intuition from Figure 3 – risk-lovers lean toward receiving high risky payments rather than low certain payments.

2. Risk neutral and low risk-averse agents place risky task 3 first to make sure that the failure doesn't happen too far in the project. Note, that they still keep task 2 running in parallel, so, in case 2 fails, they are liable for paying the supplier of task 4 on success. One can consider those agents as somewhat optimistic.

3. Moderately risk-averse agents try to dodge the situation above by scheduling task 3 after task 2 is finished.

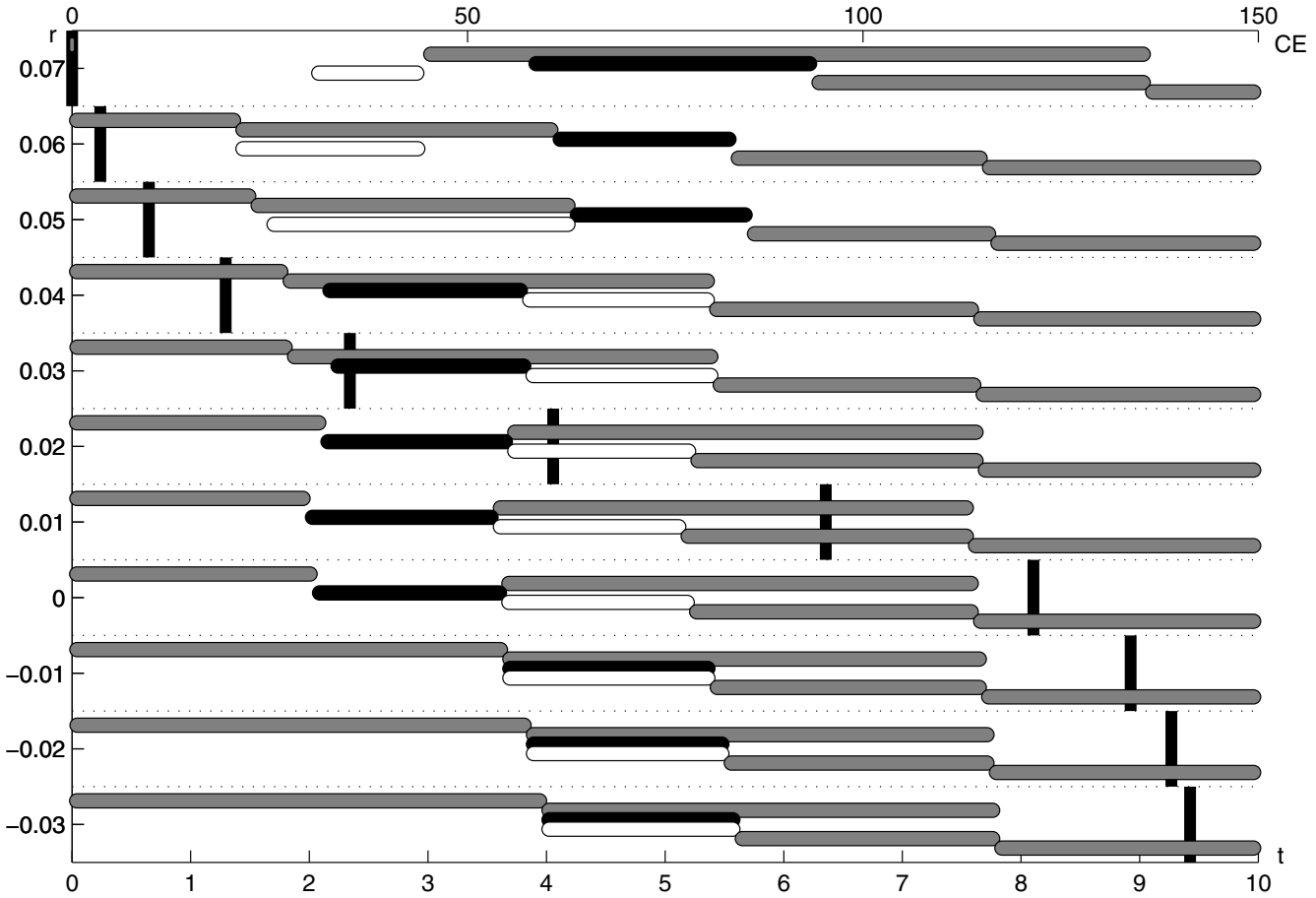

Figure 7: CE maximizing schedules and CE values for the plan in Figure 1 and $r \in [-0.03, 0.07]$.

These agents are willing to accept the plan, but their expectations are quite pessimistic.

4. Highly risk-averse agents shrink task 1 interval to zero, thus "cheating" to avoid covering any costs. One may interpret this as a way of signaling a refusal of accepting the plan.

6. ISSUES AND FUTURE RESEARCH

6.1 Multiple Local Maxima

One of the open issues is the existence of multiple local maxima of CE, even in cases where task networks are fairly simple. The reason for this is that the relative task placement has two preferred configurations: independent individual tasks can be either performed in parallel (thus increasing the probability of successful completion) or they can be scheduled in sequence to minimize overall payoffs, in case one of tasks fails.

To illustrate the issue, we constructed a sample task network with two parallel tasks. Task 1 has a higher variance of completion time probability and lower probability of success than task 2, everything else is the same. The resulting graph of CE is shown in Figure 8. There are 3 local maxima in this figure: one in the left side that corresponds to task 2 being scheduled first in sequential order, another on the right side corresponding to task 1 being first, and yet another one in the furthermost corner of the graph representing both tasks being scheduled at time 0 and executed in parallel.

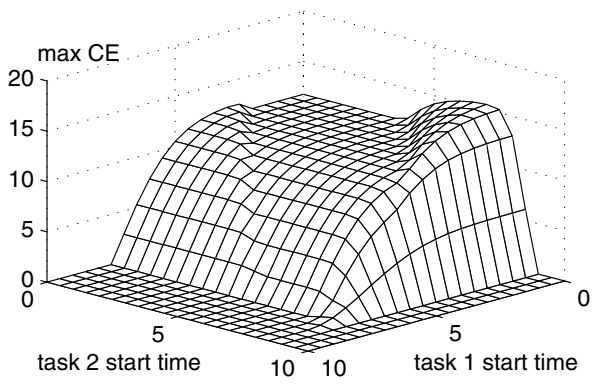

Figure 8: Local maxima for two parallel tasks.

In the course of the research, we were able to get around the issue of multiple local maxima by starting the maximization procedure from different points. However, one may note that the number of possible start points grows considerably with the complexity of the task network and the algorithm that checks each and every one of them is not scalable. A solution we are considering is to use Simulated Annealing [20],

where each node in the search queue represents a local maximum for some particular ordering of tasks.

6.2 Slack Allocation in RFQ

The last issue we want to address in this paper is how to use the CE maximization procedure to construct RFQs in the MAGNET framework. The CE maximizing schedule contains information on what is the most desirable task scheduling for the customer agent. However, it is hard to imagine that there will always be bids that cover exactly the same time intervals as in the maximizing schedule.

We suggest the following approach: first, specify what percentile α of the maximum CE value is considered acceptable by the agent. then define the start time for the task n as the set of values of t_n^{s}, such that the CE of a schedule that differs from the maximizing one only in the start time of task n is no less than α of the maximum.

Graphically, this process is represented by building the projection of the CE α-percentile graph (see Figure 9) on the task n time axis. Assuming there is only one continuous interval of t_n^{s} values for every $n \in N$, denote it as $\left[t_n^{\mathrm{s}-}, t_n^{\mathrm{s}+}\right]$. Finally, submit the interval $\left[t_n^{\mathrm{s}-}, t_n^{\mathrm{f}} + \left(t_n^{\mathrm{s}+} - t_n^{\mathrm{s}}\right)\right]$, where t_n^{s} and t_n^{f} are times from the maximizing schedule, as a part of the RFQ.

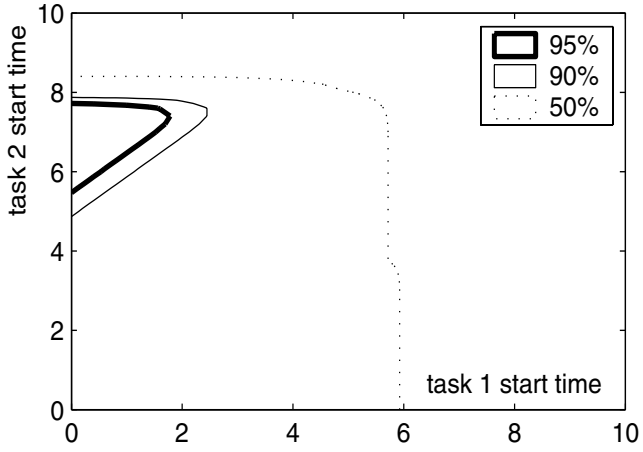

Figure 9: Contours of some α-percentile graphs for the CE graph in Figure 8.

This leaves several open questions for further study. For instance, there could be more than one interval $\left[t_n^{\mathrm{s}-}, t_n^{\mathrm{s}+}\right]$ for some tasks, so we need to distinguish them in the RFQ composition. Also, it might be appropriate to decrease the acceptable CE percentile for tasks involving goods and services that are rare and do not attract many bids, and, at the same time, increase the percentile for those tasks that receive overly many bids. In addition, the RFQ might have to be split in two or more parts, so that the requests for rare goods and services are submitted first and the rest of RFQ is composed after the bids for those rare products are received. Deciding how and when to split the RFQs is still an open question.

Although we do not specifically address the above mentioned and related issues in the current paper, the CE maximization approach promises to be powerful and flexible enough to help us resolve those in our future research.

7. RELATED WORK

Expected Utility Theory [19] is a mature field of Economics, that has attracted many supportive as well as critical studies, both theoretical [12, 13] and empirical [23, 10]. We believe that expected utility will play an increasing role in automated auctions, since it provides a practical way of describing risk estimations and temporal preferences.

In our previous work on Expected Utility [1] we were mostly concerned with computing the marginal expected utility of completing successfully all the tasks within the duration promised.

Our long term objective is to automate the scheduling and execution cycle of an autonomous agent that needs the services of other agents to accomplish its tasks. Pollack's DIPART system [18] and SharedPlans [7] assume multiple agents that operate independently but all work towards the achievement of a global goal. Our agents are trying to achieve their own goals and to maximize their profits; there is no global goal.

Combinatorial auctions are becoming an important mechanism not just for agent-mediated electronic commerce [8, 26, 22] but also for allocation of tasks to cooperative agents (see, for instance, [9, 5]).

In [9] combinatorial auctions are used for the initial commitment decision problem, which is the problem an agent has to solve when deciding whether to join a proposed collaboration. Their agents have precedence and hard temporal constraints. However, to reduce search effort, they use domain-specific *roles*, a shorthand notation for collections of tasks. In their formulation, each task type can be associated with only a single role. MAGNET agents are self-interested, and there are no limits to the types of tasks they can decide to do. In [6] scheduling decisions are made not by the agents, but instead by a central authority. The central authority has insight to the states and schedules of participating agents, and agents rely on the authority for supporting their decisions. Nisan's bidding language [16] allows bidders to express certain types of constraints, but in MAGNET both the bidder and the bid-taker (the customer) need to communicate constraints.

Inspite of the abundance of work in auctions [15], limited attention has been devoted to auctions over tasks with complex time constraints and interdependencies. In [17], a method is proposed to auction a shared track line for train scheduling. The problem is formulated with mixed integer programming, with many domain-specific optimizations. Bids are expressed by specifying a price to enter a line and a time window. The bidding language, which is similar to what we use in MAGNET, avoids use of discrete time slots. Time slots are used in [25], where a protocol for decentralized scheduling is proposed. The study is limited to scheduling a single resource. MAGNET agents deal with multiple resources.

Most work in supply-chain management is limited to hierarchical modeling of the decision making process, which is inadequate for distributed supply-chains, where each organization is self-interested, not cooperative. Walsh et al [24] propose a protocol for combinatorial auctions for supply chain formation, using a game-theoretical perspective. They allow complex task networks, but do not include time constraints. MAGNET agents have also to ensure the scheduling feasibility of the bids they accept, and must evaluate risk as well. Agents in MASCOT [21] coordinate scheduling with

the user, but there is no explicit notion of money transfers or contracts, and the criteria for accepting/rejecting a bid are not explicitly stated. Their major objective is to show policies that optimize schedules locally [11]. Our objective is to optimize the customer's utility.

8. ACKNOWLEDGMENTS

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

9. REFERENCES

[1] J. Collins, C. Bilot, M. Gini, and B. Mobasher. Decision processes in agent-based automated contracting. *IEEE Internet Computing*, pages 61–72, March 2001.

[2] J. Collins, M. Gini, and B. Mobasher. Multi-agent negotiation using combinatorial auctions with precedence constraints. Technical Report 02-009, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, Minnesota, February 2002.

[3] J. Collins, M. Tsvetovat, R. Sundareswara, J. V. Tonder, M. Gini, and B. Mobasher. Evaluating risk: Flexibility and feasibility in multi-agent contracting. In *Proc. of the Third Int'l Conf. on Autonomous Agents*, May 1999.

[4] J. Collins, B. Youngdahl, S. Jamison, B. Mobasher, and M. Gini. A market architecture for multi-agent contracting. In *Proc. of the Second Int'l Conf. on Autonomous Agents*, pages 285–292, May 1998.

[5] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In *Sixth Int'l Conf. on Intelligent Autonomous Systems*, pages 115–122, Venice, Italy, July 2000.

[6] A. Glass and B. J. Grosz. Socially conscious decision-making. In *Proc. of the Fourth Int'l Conf. on Autonomous Agents*, pages 217–224, June 2000.

[7] B. J. Grosz, L. Hunsberger, and S. Kraus. Planning and acting together. *AI Magazine*, 20(4):23–34, 1999.

[8] R. H. Guttman, A. G. Moukas, and P. Maes. Agent-mediated electronic commerce: a survey. *Knowledge Engineering Review*, 13(2):143–152, June 1998.

[9] 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.

[10] B. Jullien and B. Salanie. Estimating preferences under risk: The case of racetrack bettors. *The Journal of Political Economy*, 108(3):503–530, June 2000.

[11] D. Kjenstad. *Coordinated Supply Chain Scheduling*. PhD thesis, Dept of Production and Quality Engineering, Norvegian University of Science and Technology, Trondheim, Norway, 1998.

[12] M. J. Machina. Choice under uncertainty: Problems solved and unsolved. *The Journal of Economic Perspectives*, 1(1):121–154, 1987.

[13] M. J. Machina. Dynamic consistency and non-expected utility models of choice und er uncertainty. *The Journal of Economic Literature*, 27(4):1622–1668, December 1989.

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

[15] R. McAfee and P. J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[16] N. 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.

[17] D. C. Parkes and L. H. Ungar. An auction-based method for decentralized train scheduling. In *Proc. of the Fifth Int'l Conf. on Autonomous Agents*, pages 43–50, Montreal, Quebec, May 2001. ACM Press.

[18] M. E. Pollack. Planning in dynamic environments: The DIPART system. In A. Tate, editor, *Advanced Planning Technology*. AAAI Press, 1996.

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

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

[21] 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, May 1999.

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

[23] V. K. Smith and W. H. Desvousges. An empirical analysis of the economic value of risk changes. *The Journal of Political Economy*, 95(1):89–114, February 1987.

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

[25] 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.

[26] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second Int'l Conf. on Autonomous Agents*, pages 301–308, May 1998.