Scheduling Tasks with Precedence Constraints to Solicit Desirable Bid Combinations

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 our previous research we suggested an approach to maximizing agents preferences over schedules of multiple tasks with temporal and precedence constraints. The proposed approach is based on Expected Utility Theory. In this paper we address two mutually dependent questions: (a) what are the properties of the problem domain that can facilitate efficient maximization algorithms, and (b) what criteria determine attractiveness of one or another potential solution to the agent. We discuss different ways of exploring the problem domain. We show that naive optimization approaches often fail to find solutions for risk-averse agents and propose ways of using properties of the domain to improve upon naive approaches.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: E-commerce; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; G.1.6 [**Numerical Analysis**]: Optimization

General Terms

Algorithms, Economics, Theory

Keywords

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

1. INTRODUCTION

We are interested in developing methods for soliciting desirable bids for collections of tasks with complex time constraints and interdependencies. By desirable we mean bids that can be feasibly combined in a low cost combination that covers the entire collection of tasks. We also want to receive the "right" number of bids. Too many bids will cause the winner determination algorithm to take excessive time, and are also undesirable for the suppliers because each bid represents a speculative resource commitment and reveals private information. Soliciting too few bids brings in a risk of not covering some tasks. To balance these extremes we need a way of managing schedule flexibility with respect to particular tasks.

We address aforementioned problem in the context of the MAGNET (Multi-AGent NEgotiation Testbed) research project. MAGNET agents participate in first-price, sealed-bid, reverse combinatorial auctions over collections of tasks with precedence relations and time constraints. An agent who wants to request services or resources from other agents does so by first preparing a *Request for Quotes* (RFQ) to solicit bids. The RFQ includes not only the tasks but also the time windows when the tasks should be executed.

Because of the dependencies among the tasks, a task not completed on time might have devastating effects on other tasks. Therefore, a major problem is to decide how to sequence the tasks and how to allocate time to each of them. Since the RFQ is issued before bids are received, we need to make these decisions using expected costs, probability of completion of tasks within a time window, and expected numbers of bidders. Since there is a probability of loss as well as a probability of gain, the decision process must also deal with the risk posture of the person or organization on whose behalf the agent is acting.

In previous work [2] we proposed an approach based on Expected Utility Theory to compute an agent's preferences over different schedules for the tasks in an RFQ, and we proposed an algorithm for computing the payoff-probability outcomes of the different schedules. In this paper we discuss how an agent may weigh its expectations of incoming bids' quality, i.e. benefits in terms of costs and risks they might offer, against the expected quantity of bids. To support the discussion, we explore methods for exploiting features of the problem domain to find schedules that maximize agent preferences among various classes of possible schedules.

2. TERMINOLOGY

We use a *task network* (see Figure 1) to represent tasks and the constraints between them. A task network is a connected directed acyclic graph, where nodes denote tasks with start and finish times, while edges indicate precedence constraints. More formally, a task network is a tuple $\langle N, \prec \rangle$ of a set N of individual tasks and strict partial ordering on

AAMAS'03, July 14-18, 2003, Melbourne, Australia.

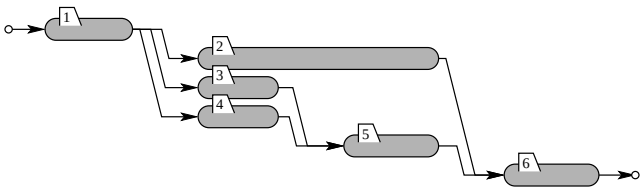

Figure 1: A task network example.

them. We also use N to denote the number of tasks.

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 determined by its *start time* t_n^{s} and *finish time* t_n^{f}, which are subject to the following constraints:

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

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 as the set of *immediate successors* of n.

The *probability of task n completion* by time t, conditional on the eventual completion of task n, is distributed according to a cumulative distribution function (CDF) $\Phi_n = \Phi_n(t_n^{\mathrm{s}}; t)$, $\lim_{t\to\infty} \Phi_n(t_n^{\mathrm{s}}; t) = 1$. Observe that Φ_n is defined to be explicitly dependent on the start time t_n^{s}. This enables us to represent situations such as weekends and seasonal market changes that may affect resource availability along with durations and completion probabilities.

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. Together with Φ_n it captures the distribution of successful task completion with no regard to whether the task succeeds or fails by the end of time (see Figure 2).

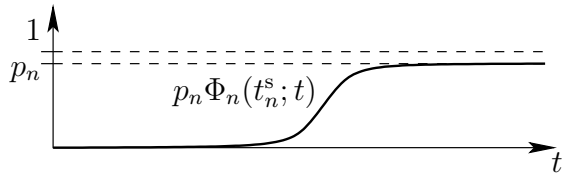

Figure 2: Unconditional distribution for successful completion probability.

We assume the cost of task n, c_n, is due at some time after successful completion of n. There is a single *final payment* W scheduled at the time moment, by which all tasks have to be completed, i.e. $\max_{n\in N} t_n^{\mathrm{f}}$, and paid if each and every task is completed successfully.

Each payoff c_n has an associated *rate of return* q_n[1]. We associate the rate of return q with the final payment W. A monetary transfer x due at time t and its associated rate of return q_x are used to calculate the discounted *present value* (PV) $\tilde{x}$ as

$$\tilde{x} := x(1+q_x)^{-t}.$$

3. EXPECTED UTILITY

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* [6] $r := -u''/u'$ of u is constant for any value of its argument, hence u can be represented as follows[2]:

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

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

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

The concept of the certainty equivalent is crucial for our further consideration due its useful properties. One of these properties is that 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 be willing to accept gambles with less than positive CE values and higher CE values will correspond to more attractive gambles.

3.1 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

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

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.

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

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

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

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

[2]Although the shape of utility function for negative values of r may contradict intuition, it must be noted that we do not consider EU values directly and use this formulation for the uniformity of notation.

[3]Hereafter we use tilde to distinguish variables depending on the current task schedule.

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 W is

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

3.2 Tree Form of CE Calculation

In Figure 3 we show two alternate schedules for the task network of Figure 1, and in Figure 4 we see the gamble calculation corresponding to these schedules. Looking at the first task, 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). Before task 3 is completed, the agent starts task 4. If task 3 fails, the agent is liable for paying c_1 and c_4 (paths $1 \rightarrow \bar{3} \rightarrow 4$) or c_1 if task 4 also fails (path $1 \rightarrow \bar{3} \rightarrow \bar{4}$). In turn, if task 3 succeeds, the agent starts task 2 before task 4 is completed. If both tasks 4 and 2 fail, the resulting path in the tree is $1 \rightarrow 3 \rightarrow \bar{4} \rightarrow \bar{2}$ and the corresponding payoff-probability pair is framed in the figure.

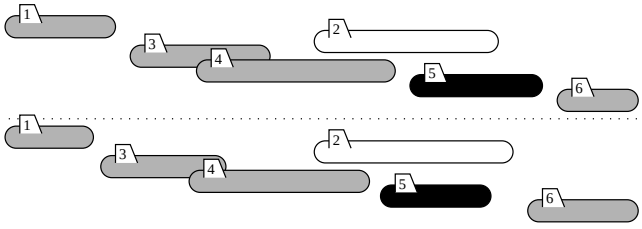

Figure 3: Two schedules for the task network of Figure 1.

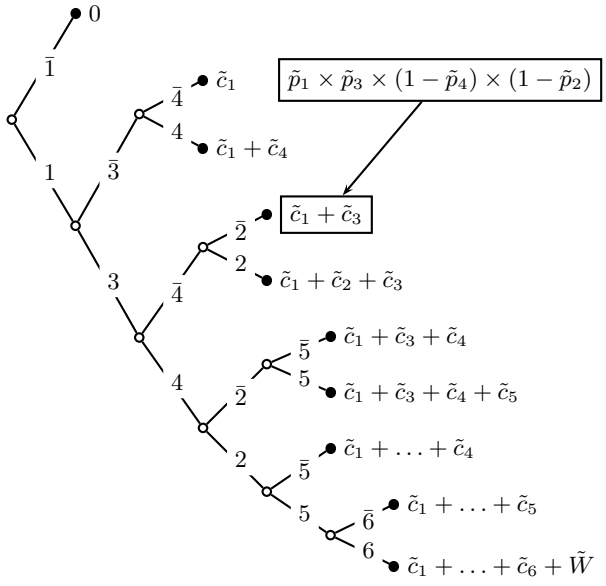

Figure 4: Payoff-probability tree corresponding to two schedules in Figure 3.

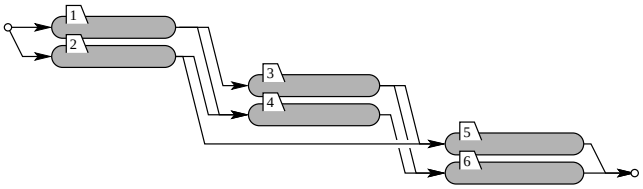

Figure 5: An irreducible task network example.

3.3 Maximization Issues

There are several important observations we have derived by closely studying the suggested form of the expected utility function. First, even for simple cases (e.g., a network of two parallel tasks) there exists a multitude of local maxima that may differ significantly in attained utility values. The large number of local maxima is caused mainly by two factors: major variations due to different ordering of tasks that significantly impact CE values[4] and small variations due to different scheduling of tasks that are not stressed by time.

For example, assume that it takes a full week to complete task 2 with high probability of success and it only takes 2 days for each of tasks 3, 4 and 5.Then, a maximizing schedule where tasks 3 and 4 are scheduled in parallel to each other and to task 2 will often have at least two corresponding maximizing schedules where tasks 3 and 4 are scheduled sequentially.

Our second observation is that the analytical form of the CE function is too complex to be of any practical use beyond a few special cases, such as a simple sequence of tasks. Although it is possible to write a relatively simple CE function expression for each way of ordering the tasks in a schedule (i.e. for each payoff-probability tree), the number of possible orderings grows exponentially with the complexity of the problem *(we address this in the next Section)*. At the same time it is easy to calculate the CE function numerically using the tree building algorithm from [2].

Because of these observations we have decided to study the domain by (a) using numerical maximization methods, and (b) thoroughly exploring different task orderings to ensure that no configuration escapes our consideration.

4. TASK ORDERING

In a given task network, there may be many different ways to order the individual tasks that are consistent with the precedence relationships and yet result in different payoff-probability trees. We divide task networks into two categories: reducible, which can be reduced to a single equivalent task by recursively merging sequential and parallel tasks, and irreducible. Examples of reducible and irreducible task networks are shown in Figure 1 and Figure 5 respectively.

To remove ambiguity in distinguishing between task orderings, we also assume that no two task start and/or finish times can be exactly equal. Under this assumption schedules where some start and finish times coincide will belong to two or more task orderings.

We start by considering simple cases of reducible networks and build up an apparatus to enumerate task orderings for arbitrary task networks, whether they are reducible or not.

[4]One may think of these tasks as if they are on the critical path, but this analogy is misleading since in our formulation the impact of a task on CE value depends on a schedule.

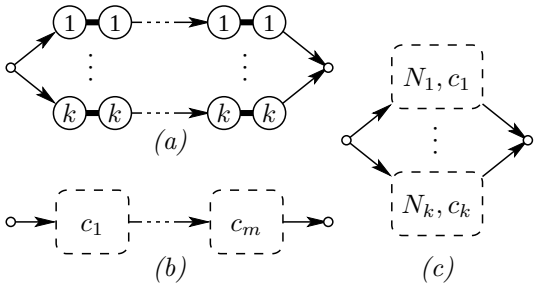

Figure 6: Counting scenarios.

4.1 Counting in Reducible Task Networks

Let's consider first a task network that consists of k parallel sequences of N_i, $i = 1, \ldots, k$ tasks each. We associate a set of $2N_i$ balls with each of k sequences, one ball to denote the start time of some task and another to denote its finish time. Since inside each sequence the order of task start and finish times is uniquely defined by precedence relations, we label all $2N_i$ balls in each sequence by the number i (see Figure 6(a)). The number of orderings $C^{(a)}$ of task start and finish times corresponds to the number of orderings of k sets of balls, i.e.

$$C^{(a)} = \frac{(2\sum_{i\in I} k_i)!}{\prod_{i\in I}(2k_i)!}$$

The number $C^{(b)}$ of orderings for m sequential task networks of an arbitrary (not necessarily reducible) configuration, where each network has c_j, $j = 1, \ldots, m$ possible orderings by itself (Figure 6(b)) is clearly

$$C^{(b)} = \prod_{j=1}^{m} c_j$$

Finally, we count the number $C^{(c)}$ of orderings in the case of k parallel task networks of arbitrary configuration, where each network i, $i = 1, \ldots, k$ consists of N_i tasks with c_i possible combinations on them (Figure 6(c)). To do this we first enumerate the possible orderings assuming that the order of task start and finish times is fixed inside each network i, and then account for the number of ways to fix the order independently in each task network i, i.e. c_i,

$$C^{(c)} = \frac{\left(2\sum_{i=1}^{k} n_i\right)!}{\prod_{i=1}^{k}(2n_i)!} \prod_{i=1}^{k} c_i$$

4.2 Counting in Irreducible Networks

For the cases where either it is impossible to reduce a task network or, in general, when one wants a universal way of enumerating orderings of task start and finish times in arbitrary networks, we propose the following algorithm. It starts by assuming that no task was started or finished yet and proceeds recursively by examining cases when one of tasks that can be started next is started or one of the tasks that have been started already is completed:

Algorithm: $C^{\forall} \leftarrow$ **calcOrderings**(T, D)
Requires: T "*started tasks*", D "*finished tasks*"
Returns: $C^{\forall}$ "*number of orderings*"

$X \leftarrow \{i \in N | P_1(i) \subset D, i \notin T \cup D\}$ "*tasks to start*"
if $X \cup T = \varnothing$ "*all tasks are done*"
  $C^{\forall} \leftarrow 1$
else "*can start of finish some tasks*"
  $C^{\forall} \leftarrow 0$
  foreach $i \in X$ "*start task i*"
    $C^{\forall} \leftarrow C^{\forall} +$ **calcOrderings**$(T \cup \{i\}, D)$
  endfor
  foreach $i \in T$ "*finish task i*"
    $C^{\forall} \leftarrow C^{\forall} +$ **calcOrderings**$(T \setminus \{i\}, D \cup \{i\})$
  endfor
endif
return $C^{\forall}$

4.3 Examples

Let's consider a few examples of using the counting methods described in this Section, starting with the task network in Figure 1. First we find the number of orderings in a network of parallel tasks 3 and 4 to be 6, in accordance with formula for $C^{(a)}$. Next, we consider a network of tasks 2 to 5 as consisting of two parallel networks, one with one task and one ordering, and the other with 3 tasks and 6 orderings. This, by a trivial application of rule for $C^{(b)}$, gives a total number of orderings equal to 168 (out of 7,484,400 orderings possible in a network of 6 tasks and no precedence constraints).

The number of orderings in the task network in Figure 5 has to be calculated using the `calcOrderings` algorithm. It is equal to 517 and thus is quite different from the one calculated before, although the number of tasks, 6, and the number of precedence constraints, 7, are equal for both cases.

Clearly, the number of orderings can grow exponentially with the number of tasks. However, precedence constraints may reduce this growth; in the case where all the tasks are in a single sequence, there is just one ordering. This observation emphasizes the need for understanding the problem domain in order to construct efficient maximization algorithms that will not require the exploration of all possible orderings to find desirable maxima[5].

5. MAXIMIZATION METHODS

Our initial attempt to explore the local maxima space was to use unconstrained maximization[6] with all precedence and order constraints internalized within the calculation of the certainty equivalent CE. This approach made it possible to find the global maximum of CE after a large number of restarts from random points in a $2N$–dimensional space of task start and finish times [2]. However, it also produced many "blind maxima" — points which wrongfully appeared to be maxima to the maximization algorithm because of constraints handling inside the calculations of CE. This approach was abandoned until we find a better way of internalizing constraints.

In this section we consider a variety of methods that we designed to tackle the problem of maximizing the CE function. The methods are illustrated using the experimental data from exploring the sample task network in Figure 1.

[5]Our favorite motivational example is that the addition of a sequence of just two tasks parallel to the task network in Figure 5 increases the number of orderings from 517 to 940,940.

[6]In particular, the `Matlab` implementation of Nelder-Mead direct search method in `fminsearch`.

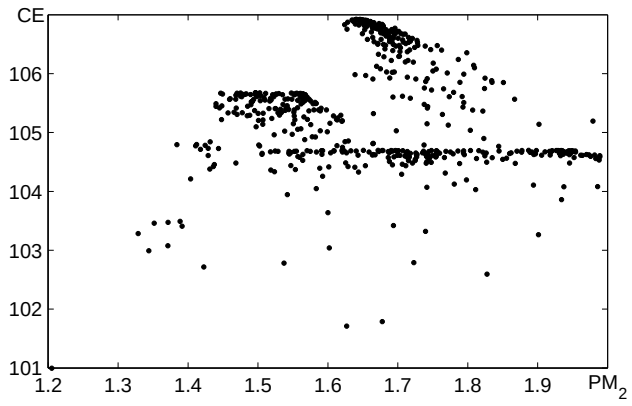

Figure 7: Local and pseudo-maxima found by the "loose " method, task network of Figure 1, and $r = 0$.

The general results were tested and remained applicable to several other networks including the irreducible task network in Figure 5.

5.1 Constrained Maximization Methods

We designed and tested three maximization approaches that utilize the ability to enumerate all possible orderings of task start and finish times. We call these methods *"loose," "strict"* and *"nice"* for the reasons that will become clear later. Each method is initialized by random schedules that satisfy the corresponding set of constraints.

"Loose" and "Strict" Maximizations

The "loose" method has its set of constraints defined by the precedence relations and time constraints as specified in Section 2.

The "strict" maximization method further restricts the "loose" method by adding task ordering constraints to the set, thus ensuring that every maximum found will have the same ordering of task times as in the initial point. The number of additional constraints introduced by this method depends on the selected ordering. For example, for our sample problem in Figure 1 it varies from 2 to 7.

Unfortunately, both of these methods have a major drawback which prevents us from using them to study the domain reliably. The issue is that they produce a large number of pseudo-maxima in cases where the maximization algorithm[7] is stuck on a plateau with a very small and irregular slope.

The results produced by the "loose" method for our sample problem and a risk-neutral agent are shown in Figure 7, where each dot corresponds to some local or pseudo-maximum. In this figure the y–axis shows CE values and x–axis represents a quadratic measure of schedule $\{t^{\mathrm{s}}_n, t^{\mathrm{f}}_n\}_{n\in N}$ parallelism, which attempts to capture all 12 dimensions of task start and finish times in one measure.

We define the measure of parallelism of degree x as

$$\mathrm{PM}_x(\cdot) = \left\{ \frac{1}{t^{\mathrm{f}} - t^{\mathrm{s}}} \int_{t^{\mathrm{s}}}^{t^{\mathrm{f}}} \left[\sum_{n\in N} 1_{t\in[t^{\mathrm{s}}_n, t^{\mathrm{f}}_n]} \right]^x dt \right\}^{\frac{1}{x}}$$

The use of a quadratic ($x = 2$) measure is motivated by our expectation that schedules with higher number of parallel tasks will be less attractive to risk-averse agents due increasingly higher risk of payoffs after one of the parallel task fails.

[7]We use `fmincon` from `Matlab` that implements a sequential quadratic programming method with a numerical updating of the Hessian on every step.

"Nice" Maximization

The "nice" method was created in the attempt to fix the pseudo-maxima problem by projecting a set of precedence and order 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 $\bar{x}$ of a $(2N+1)$–dimensional unit cube and the ordered vector of $2N$ task times. This many-to-one correspondence reflects proportions in which task start and finish times divide the $[t^{\mathrm{s}}, t^{\mathrm{f}}]$ interval. Figure 8 illustrates the suggested variable transformation for the network consisting of two sequential tasks.

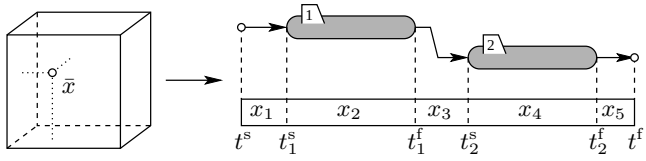

Figure 8: Transformation of a point in 5–dimensional unit cube to a 2-task schedule.

This approach works remarkably well and produces nice, hence its name, local maxima without artifacts like "blind maxima" or pseudo-maxima described above. The transition from zero of the unit cube is not defined. However, in our experience this never caused a problem, presumably because the CE function is always flat in the direction of zero.

The results produced by the "nice" maximization for the sample problem and $r = 0$ are shown in Figure 9 in the same scale as the results for the "loose" method in Figure 7. The comparison of Figures 7 and 9 reveals that the bulk of points in the former are, indeed, pseudo-maxima corresponding to unsuccessful attempts to reach clusters of maxima distinctly displayed in the latter.

5.2 Domain-Specific Methods

One important observation is derived from our experiments that renders all three methods above nearly useless for the cases we are interested in, i.e. for risk-averse (business people, not gamblers) agents. All three methods are "lazy," meaning that, being initialized with a random schedule that results in a negative CE value, they converge to a degenerate local maximum with one or more task durations equal to zero. This may be interpreted as a way of signaling rejection of the plan, and although it is useful in general, it dominates maximization results in cases with positive risk-aversity.

We approached the problem of poor maximization performance by guessing and verifying the properties of the CE function in the task network environment. These efforts led to the creation of two approaches that we refer to as *"boost"* and *"copycat"* maximization methods.

"Boosted" Maximization

This method arises from the intuition that lowering the final payoff might change a schedule somewhat, but will not lead

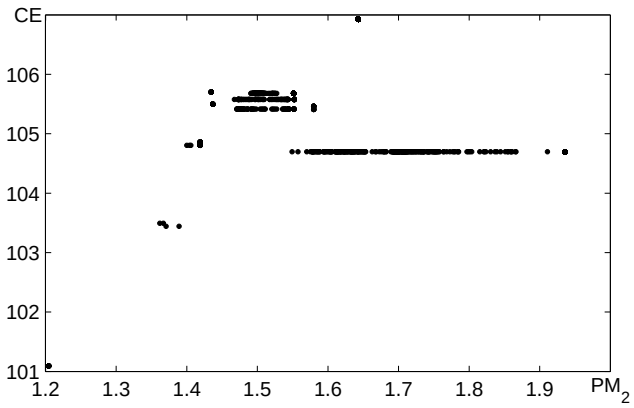

Figure 9: Local maxima found by the "nice" approach in the same setup as in Figure 7.

to abandoning it completely unless its CE value becomes negative. Guessing this, we "float" the search space by increasing the final payoff (in these experiments we doubled it) to get a maximizing schedule. We then restart the maximization procedure starting with this maximizing schedule, using the actual payoff to get an adjusted schedule.

The following table shows the results of "boosting" each of the three constrained maximization methods described before in the case of the sample problem and a risk-neutral agent. Each number in the table shows the number of CE function maxima with positive CE values found after 100 restarts for each of 168 possible orderings (Section 4.3).

Method	# max	"boosted"
"loose"	404	660
"strict"	366	687
"nice"	410	1562

"Copycat" Maximization

The "copycat" method arises from the intuition that, although the exact location and relative attractiveness of different maxima may change as the agent's risk aversity changes, these changes won't likely be large. To verify this guess we used maxima for risk-loving agents as initial points for maximizing CE functions of agents with higher risk-aversity.

The table below shows an example of applying the "copycat" improvement to our "nice" method in the sample problem. The set of maxima for $r = 0.03$ was used to improve the "nice" method's performance for higher risk-aversity cases. The results show that not only the number of maxima found rises dramatically, but the maximum CE value among all maxima increases as well.

	"nice"		"copycat nice"	
r	# max	max CE	# max	max CE
0.06	0	0.00	149	3.66
0.05	0	0.00	6194	9.70
0.04	0	0.00	6195	19.32
0.03	0	0.00	6198	34.35
0.02	1	54.50	6198	57.46
0.01	36	84.77	6198	85.12
0.00	410	106.92	6198	106.92
-0.01	1602	118.13	6198	118.13
-0.02	3736	123.16	6198	123.16
-0.03	6198	125.64	—	—.—

6. PROPERTIES OF THE DOMAIN

We have employed the "nice" method with "boost" and "copycat" extensions to thoroughly study the properties of task scheduling using CE function maximization. In this section we illustrate our findings using the sample network in Figure 1 and assumption of the risk-neutrality. The reason we choose the risk-neutral agent's case is due to our empirical results showing that CE maximizing schedules will not change much with r, although their value relative to each other may and will change.

6.1 Quality and Quantity

Figure 10 exposes the structure of schedules behind some of CE maxima shown in Figure 9. In this figure the bottom x–axis represents time. There are 11 schedules represented by rounded horizontal bars in the same manner as in the previous figures. Each schedule is assigned a letter on the y–axis and in each schedule task 3 is highlighted in white color and task 4 in black. Finally, the top x–axis represents the certainty equivalent and vertical heavy black bars show the CE values attained in the corresponding schedules by the risk-neutral agent.

By carefully examining this figure we conclude that CE maximization discovered several distinct strategies of scheduling tasks 2 to 5. The most successful strategy in this setup, A, schedules as many tasks in parallel as possible to get a final payment as soon as possible, hence increasing its present value. Indeed, for $r = 0$ a random schedule has almost 80% probability of converging to schedules A, C or I, in which tasks 3 and 4 are parallel. K, though, has too many tasks scheduled sequentially, thus running against the time limit.

One alternative is to schedule tasks 3 and 4 sequentially, giving task 2 more time to run by scheduling it parallel to three (as in E and G), two (B, H) or, at least, one and a half (D, F) of other tasks. Attempts to give task 2 less time by scheduling it parallel to task 4 reveal that scheduling tasks 3 and 4 in parallel is important to ensure necessary schedule flexibility (compare J to C).

However, a schedule having highest CE value is not necessary the most preferable one. In fact, by examining Figure 11 we see that schedule A has hardly any flexibility, which is indicated by its very small support on the parallelism measure axis. In contrast, strategy I allows for a great flexibility in choosing time windows for tasks 3 and 4, hence its large cluster of maxima points in CE–PM_2 space.

The particular choice of one schedule over another must depend on the availability of bids for each task. For example, if a market has an abundance of task 3 and 4 suppliers, a risk-neutral agent may safely choose schedule A as a base for an RFQ [1] that will solicit desirable bid combinations. In the case when one or both of these tasks are in rare supply, the agent might prefer schedule I and trade its expectations of incoming bid quality, as expressed by the certainty equivalent, for larger RFQ time windows and, thus, higher expected number of bids.

We want to stress that an agent's preferences over CE maximizing schedules vary greatly with risk aversity. For example, in Figure 12, schedule A turns to be less attractive in the eye of a more risk-averse agent than schedules H, I and even K. One cause of such change is that risk-averse agents generally prefer "fail fast" approach that distributes more time to the tasks that are late in the schedule.

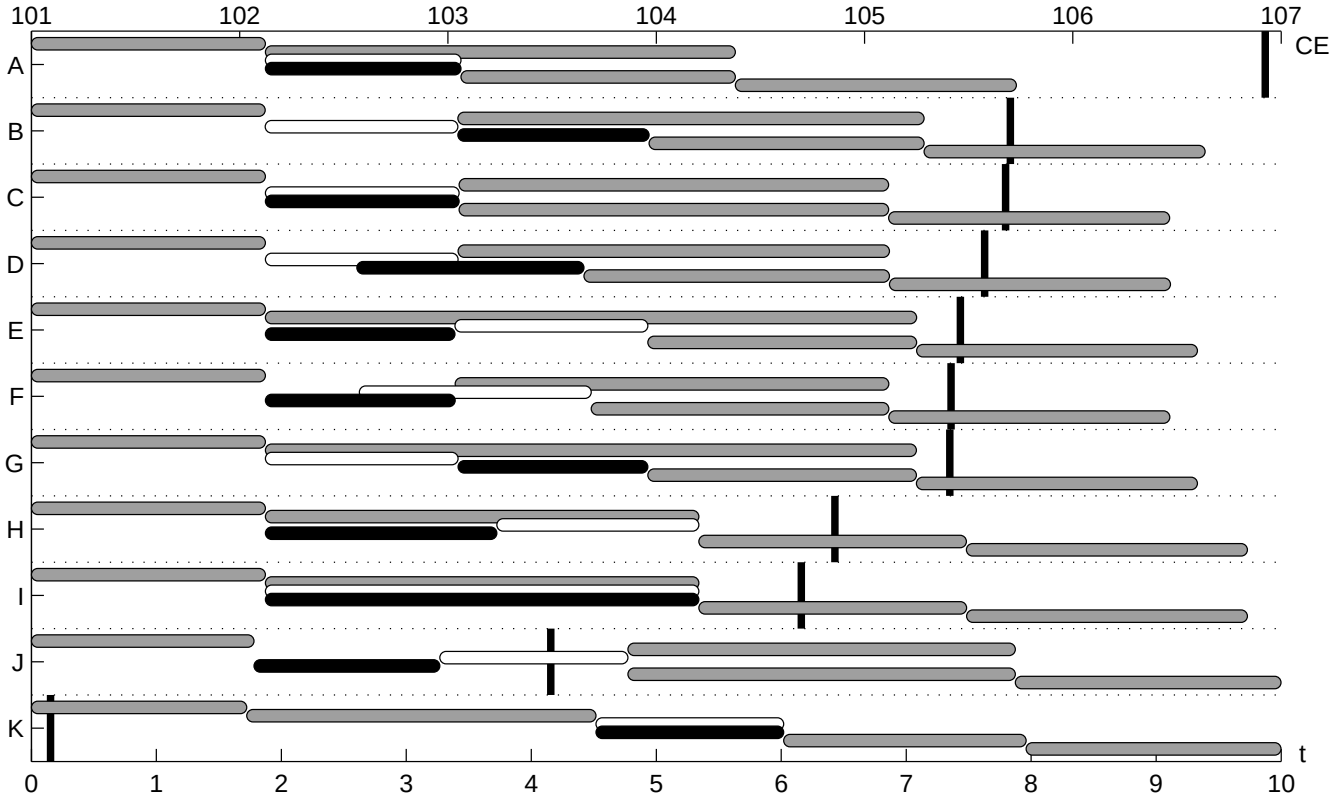

Figure 10: Structure and CE values for different clusters of CE maximizing schedules corresponding to maxima in Figure 9.

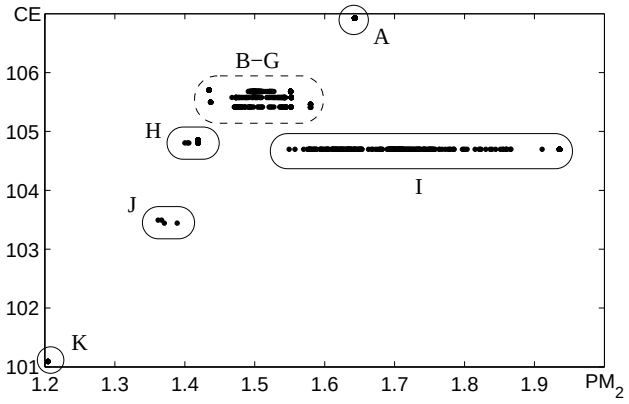

Figure 11: Local maxima in Figure 9 annotated with schedule labels from Figure 10.

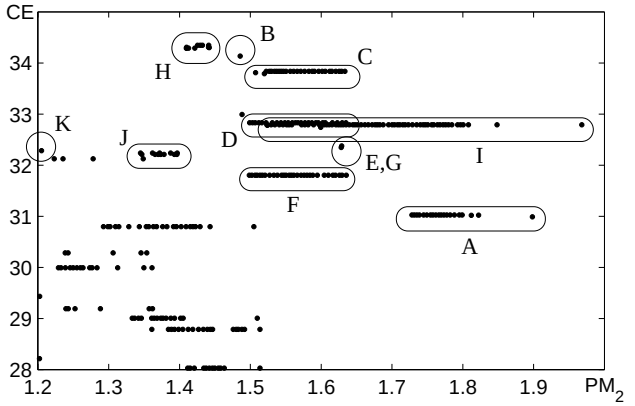

Figure 12: Change of local maxima structure in Figure 11 for $r = 0.03$.

6.2 Design of Domain-Specific Heuristics

Our findings showed that CE maximizing schedules have different properties with regard to quality and quantity of bids that are potentially solicited by the corresponding RFQs. We also demonstrated that naive CE maximization approaches do not perform well in interesting cases, but their performance can be significantly improved by using domain properties. Finally, we concluded that the maximization methods that we use to explore the problem domain are hardly useful in practice, as they rely on the task ordering enumeration and therefore are exponentially hard in a worst case.

As a result, we plan on designing heuristics that will let agents effectively explore maximizing schedules of different nature and choose desired quality-quantity pairs. One possible heuristic might be based on the observation that there exist fairly obvious transitions between many maximizing schedules (see Figure 13), such as rescheduling two tasks from sequential ordering to parallel or other way around. When designed, such heuristic will reduce the search problem from considering all task orderings and random sched-

ules to finding a few maxima and exploring near-optimal schedules suggested by the heuristic.

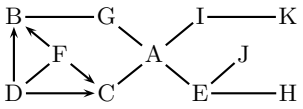

Figure 13: Some simple transitions between schedules in Figure 10.

Another potential approach would be to discover the exact mechanism behind the changes in preferences over maximizing schedules for different degrees of risk-aversity. This heuristic would allow us to concentrate search efforts on specific classes of schedules that are expected to be of interest for agents with known risk-aversity.

7. RELATED WORK

Auctions are becoming a widely used mechanism not just for agent-mediated electronic commerce [4, 10], but also for allocation of tasks to cooperative agents [5, 3].

Despite the abundance of work in auctions [7], limited attention has been devoted to auctions over tasks with complex time constraints and interdependencies, as we do in MAGNET. In [8], 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 [13], where a protocol for decentralized scheduling is proposed. The study is limited to scheduling a single resource. MAGNET agents deal with multiple resources. Walsh et al [11] 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.

Agents in MASCOT [9] coordinate scheduling with the user. Their major objective is to show policies that optimize schedules locally. Our objective is to optimize the expected customer's utility before bids are submitted and schedules are finalized.

The results reported in [12] on the problem difficulty in job-shop scheduling show many similarities to the problems we encountered in maximizing the certainty equivalent. Our problem is not job-shop scheduling; we are not scheduling resources the agent has, but we are producing a schedule of tasks that other agents will do. Our objective is to schedule tasks for the RFQ is a way that optimizes the expected utility of the agent. Knowledge of the market, in terms of expected number of bidders and probability of success, play a role in our formulation, as well as knowledge of the risk aversity of the agent.

8. CONCLUSIONS

We have presented a variety of methods for generating schedules of tasks that maximize the expected utility of an agent. In our future work we plan to study two domain-specific heuristics suggested here as well as other possibilities using the proposed maximization techniques. Our goals include using created heuristics to design an efficient mechanism for direct exploration of the CE maximizing schedules. Such mechanism will be useful in studying the construction of well-balanced RFQs, which solicit reasonable number of bids with good risk-payoff characteristics.

9. ACKNOWLEDGMENTS

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

10. REFERENCES

[1] A. Babanov, J. Collins, and M. Gini. Asking the right question: Risk and expectations in multi-agent contracting. Technical Report 02-034, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN, 2002.

[2] A. Babanov, J. Collins, and M. Gini. Risk and expectations in a-priori time allocation in multi-agent contracting. In *Proc. of the First Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, volume 1, pages 53–60, Bologna, Italy, July 2002.

[3] 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.

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

[5] 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.

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

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

[8] 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.

[9] 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.

[10] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[11] 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.

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

[13] 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.