Deciding Task Schedules for Temporal Planning via Auctions*

Alexander Babanov and Maria Gini
Department of Computer Science and Engineering, University of Minnesota
200 Union St SE, Minneapolis, MN 55455
gini@cs.umn.edu

Abstract

We present an auction-based approach to allocate tasks that have complex time constraints and interdependencies to a group of cooperative agents. The approach combines the advantages of planning and scheduling with the ability to make distributed decisions. We focus on methods for modeling and evaluating the decisions an agent must make when deciding how to schedule tasks. In particular, we discuss how to specify task schedules when requesting quotes from other agents. The way tasks are scheduled affects the number and types of bids received, and therefore the quality of the solutions that can be produced.

Introduction

Intelligent agents need the ability to schedule their activities under various temporal constraints. Such constraints might arise from interactions with other agents or directly from the specifics of the tasks an agent has to accomplish.

In this paper, we are interested in situations where the agents are cooperative and heterogeneous, such as a group of robots each with different sensors. The objective is to reduce task completion time by sharing the work, and to enable the robots to complete the tasks even if an individual robot is unable to do some of the tasks because of limitations in its capabilities.

In our approach, agents distribute tasks among themselves using a first-price, reverse combinatorial auction, where one agent acts as the auctioneer. The auction is a reverse auction, since the auctioneer is the buyer.

What is unique in our solution is that the auction includes time windows for each task, so in the process of allocating tasks a feasible schedule is constructed. The ability to deal with time is a distinctive feature of our system MAGNET (Multi-AGent NEgotiation Testbed) (Collins, Ketter, & Gini 2002).

In MAGNET, an agent, perhaps acting on behalf of its human user, acts as the auctioneer by (1) creating and distributing a *Request for Quotes (RFQ)* where tasks and time windows for them are specified, (2) receiving bids submitted by other agents, (3) evaluating the bids, and (4) selecting the best bids by a process known as winner determination (Sandholm 2002). Typically the agent selects the set of bids with minimal cost, but it has also to ensure that each task is covered exactly by one bid, and the task start and finish times specified in the selected bids are such that they can be composed into a feasible schedule. MAGNET agents are self-interested, but here we use the same algorithms for task allocation among cooperative agents.

Why auctions for task allocation

Framing the problem of task allocation in economic terms is attractive for several reasons. First, we are able to draw on a large body of results on auctions from economics (McAfee & McMillan 1987) as well as computer science. Second, since each agent has different resources and might have a different value for them, using auctions allows one to express in a natural way individual values and preferences. This is specially important when the agents are robots that have different sensors, different power consumption, different actuators, etc. In addition, it is well known that communications among robots can be unreliable and slow because of low bandwidth. One-shot auctions, as we propose, are an efficient and robust way of doing distributed task allocation. Any agent can act as the auctioneer, so an agent experiencing unexpected difficulties can start an auction to get help from others. Agents that are disabled can be left out of the decisions.

Auctions have been suggested for allocation of computational resources since the 60's (Sutherland 1968). The Contract Net (Smith 1980) is perhaps the most well known and most widely used protocol for distributed problem solving. Auction based methods for allocation of tasks have become popular in robotics (Gerkey & Matarić 2003; Goldberg *et al.* 2003) as an alternative to methods such as centralized scheduling (Chien *et al.* 2000) or application-specific methods, which do not easily generalize (Agassounon & Martinoli 2002).

A recent survey (Dias *et al.* 2005) covers in detail the state of the art in using auctions to coordinate robots. Most robot auctions either have limited scalability, because of the computational complexity of combinatorial auctions, or produce less than optimal allocations.

*Work supported by in part by NSF under grants EIA-0324864 and IIS-0414466.

When tasks are auctioned one at a time, as they arise, the group of robots can react rapidly to changes, which is important since robots might fail or the environment might change in unexpected ways, but at the cost of producing suboptimal solutions. Continuous auctions also tend to require more rebidding. A robot might decide it cannot complete some of the tasks it was awarded in previous auctions, either because it became incapacitated or because it is the only one capable of doing a task being auctioned. Fewer switches between tasks are desirable when the cost of switching is non trivial. When robots are involved, switching can have a high costs since a robot might have physically to move to a different location.

We take a different approach. To make the best use of the resources available to the group of agents, we combine bidding with scheduling. Bids have to include a time window, and winner determination accepts the best set of bids that fit the overall schedule. We use combinatorial auctions, since it is often more efficient for an agent to do a combination of tasks rather than a single task. This is specially true for physical agents, where the cost of traveling might be significant and combining multiple tasks saves time and energy.

There is a downside to combinatorial auctions, which is computational complexity. Winner determination for combinatorial auctions is known to be $\mathcal{NP}$-complete and inapproximable (Sandholm 2002). It scales exponentially with the number of tasks and at best polynomially with the number of bids. The addition of temporal constraints makes the problem scale exponentially in the number of bids as well (Collins 2002). However, in practice we believe this is not a problem for many robotic tasks. Computation times are often small compared to the time the robots need to execute the tasks, and many tasks have soft real-time requirements. We have shown that we can estimate the computing time needed for clearing the auction with a given level of confidence (Collins 2002), given the number of bids, of tasks, and the complexity of the precedence relations between tasks.

Why planning

There are many situations where planning is desirable, either because the tasks are part of a higher-level plan or simply because planning allows one to have a better use of resources. On the other side, planning requires effort from the agents, and there must be an advantage for them to plan in advance rather than react. When agents are self-interested by planning in advance what tasks to do and when, the agents are more likely to secure resources and at a better price. When agents are cooperative, as in the case, optimizing resource usage is still an important objective, so it is still rational for them to do planning.

Traditionally planning and auctions are not combined. A notable exception is the work reported in (Hunsberger & Grosz 2000), where combinatorial auctions are used for what they call "the initial commitment decision problem", which is the problem an agent has to solve when deciding whether to accept or refuse tasks that someone else has requested. No indication is given on how the request for tasks is made, and how the specifics of the request affect the bids received. These are the issues we address in this paper.

To sum this up, the ability to plan in advance and to bid for combinations of tasks permits a better use of resources. The ability of any agent to become an auctioneer and to resubmit for bids tasks it had accepted earlier makes our solution distributed and robust.

The specific problem we address in this paper is how to schedule tasks when requesting quotes from other agents. Our preliminary results show that the way tasks are scheduled in the RFQ affects the quality and quantity of bids (Babanov, Collins, & Gini 2003). There are tradeoffs between allocating large time windows to give flexibility to bidders but risking to get bids that cannot be combined into a feasible schedule, and between accomplishing tasks fast but not having enough time to recover in case a task fails.

Even though the problem we discuss in this paper might appear to be relevant only within the context of auctions, we are interested in solving the problems agents face when they have to decide task schedules.

In the rest of the paper, we describe how we model and evaluate the decisions the agent makes when scheduling tasks. The material presented has mostly been reported previously in different venues and is presented here to provide the necessary background.

Background on the Approach

In the following, we do not consider the problem of plan generation by assuming that an agent has a library of plans to choose from or is given a plan from a human supervisor.

Terminology and Definitions

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, and edges indicate precedence constraints. We use N to denote the set of individual tasks and the number of tasks where appropriate.

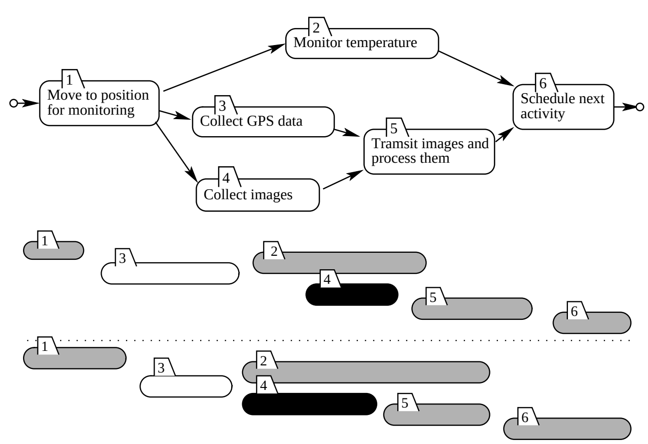

Figure 1: (top) A task network, where nodes indicate tasks and links precedence constraints, and (bottom) two different schedules for requesting quotes for the tasks in the task network.

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 task i in the schedule is characterized by the *task* i *start time* t_i^{s} and *task* i *finish time* t_i^{f}. A *time window* is the interval of time when a task can be scheduled. Note that the time windows in the RFQ do not need to obey the precedence constraints; the only requirement is that the accepted bids obey them.

We define $\tilde{P}(i)$ to be the set of *precursors* of task i, i.e. all tasks that finish before task i starts in a schedule, $\tilde{P}(n) := \{m \in N | t_m^{\mathrm{f}} \leq t_n^{\mathrm{s}}\}$.

We represent the agent's preferences over payoffs by the Bernoulli *utility function* u (Mas-Colell, Whinston, & Green 1995). We further assume that the *coefficient of absolute risk aversion* $r(x) := -u''(x)/u'(x)$ of u is defined and constant for any value of its argument x (Pratt 1964). We choose a function u satisfying the assumptions above as follows:[1]

$$u(x) = -\exp\{-rx\} \text{ for } r \neq 0 \qquad u(x) = x \text{ for } r = 0.$$

Agents with positive r values are risk-averse, those with negative r values are risk-loving. We are mainly interested in risk-neutral or 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 and results in some monetary payoff. We define a *lottery* to be a set of payoff-probability pairs G,

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

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

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

Cumulative Probabilities

To model the fact that there is no guarantee that a task will be completed in a given amount of time, we introduce the *probability of task* n *completion* by time t conditional on the eventual successful completion of task i. This probability is distributed according to the cumulative distribution function (CDF) $\Phi_i = \Phi_i(t_i^{\mathrm{s}}; t), \Phi_i(\cdot; \infty) = 1$.

There is an associated unconditional *probability of success* $p_i \in [0, 1]$ which characterizes the percentage of tasks that are successfully completed given infinite time. To simplify things, we assume we know the average cost of each task, its expected duration, and the probability of successful completion of each task as a function of time.

We assume the cost of task i, c_i, is due at task finish time t_i^{f} if the task is successfully completed. For each cost c_i there is an associated *rate of return* q_i which is used to calculate the *discounted present value* PV at time t as

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

[1]Although the shape of the utility function for negative values of r is counterintuitive, it must be noted that we do not consider expected utility values directly and use this formulation for simplicity.

There is a single *final reward* V paid as soon as all tasks in N are successfully completed, i.e. at time $t^{\mathrm{f}} = \max_{i \in N} t_i^{\mathrm{f}}$[2]. We associate the rate of return q with the final payment. We assume that the payoff c_i for task i is scheduled at t_i^{f}. is $\tilde{c}_i := c_i (1 + q_i)^{-t_i^{\mathrm{f}}}$. We use tilde to distinguish variables depending on the current task schedule.

The conditional probability that task i succeeds when given a chance to start is determined by its scheduled finish time: $\tilde{p}_i := p_i \Phi_i(t_i^{\mathrm{s}}; t_i^{\mathrm{f}})$. We consider the probability of successful completion of every precursor of task i and of task i itself as independent events. The unconditional probability that task i will be completed successfully is then

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

where $\tilde{P}(i)$ is a set of precursors of task i. The probability of receiving the final reward V is equal to the probability that all tasks in N are completed, which is $\tilde{p} = \prod_{n \in N} \tilde{p}_n$.

In general, we expect the probabilities of success and present values influences scheduling. The more time is given, the higher are the agent's odds of receiving the final reward. To the contrary, the difference between the present value of the reward and the present values of all payments is likely to decrease with the length of the schedule.

To estimate, for each task, the cumulative distribution of the probability the task will be completed given the amount of time allocated to it, the agent needs to build the probability model described above. The probability of success is relatively easy to observe. This is the reason for introducing the cumulative probability of success Φ_i and probability of success p_i, instead of the average project life span or probability of failure. It is rational for an agent to report a successful completion immediately in order to maximize the present value of a payment. It is also rational not to report a failure until the last moment due to the possibility of rescheduling, outsourcing, or fixing the problem in some other way.

Example. To illustrate the definitions above, let's return to the task network in Figure 1 and consider the two alternative schedules in Figure 2. The x-axis represents time, the y-axis shows both the task numbers and the cumulative distribution of the unconditional probability of completion. Circle markers show start times t_i^{s}. Crosses indicate both finish times t_i^{f} and success probabilities $\tilde{p}_i$ (numbers next to each point). Square markers denote that the corresponding task cannot span past this point due to precedence constraints. The thick part of each cumulative distribution shows the time allocated to each task.

Certainty Equivalent

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. $u(\mathrm{CE}[L]) := Eu[L]$. The concept of CE is crucial due to its properties. Unlike expected utility, CE values can be compared across different

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

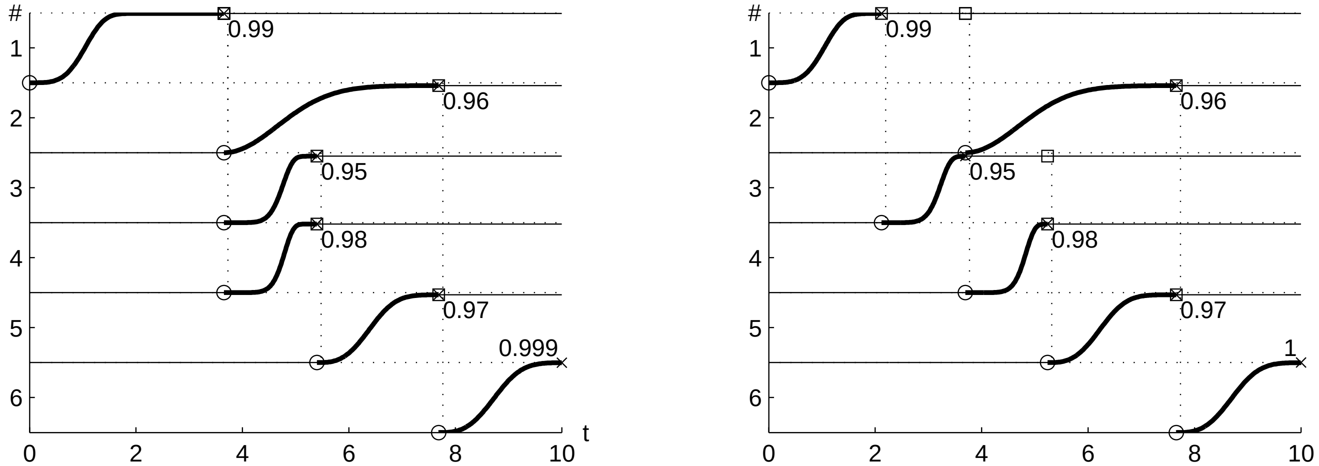

Figure 2: Optimal time allocations for the tasks in Figure 1 for $r = -0.01$ (left) and $r = 0.02$ (right).

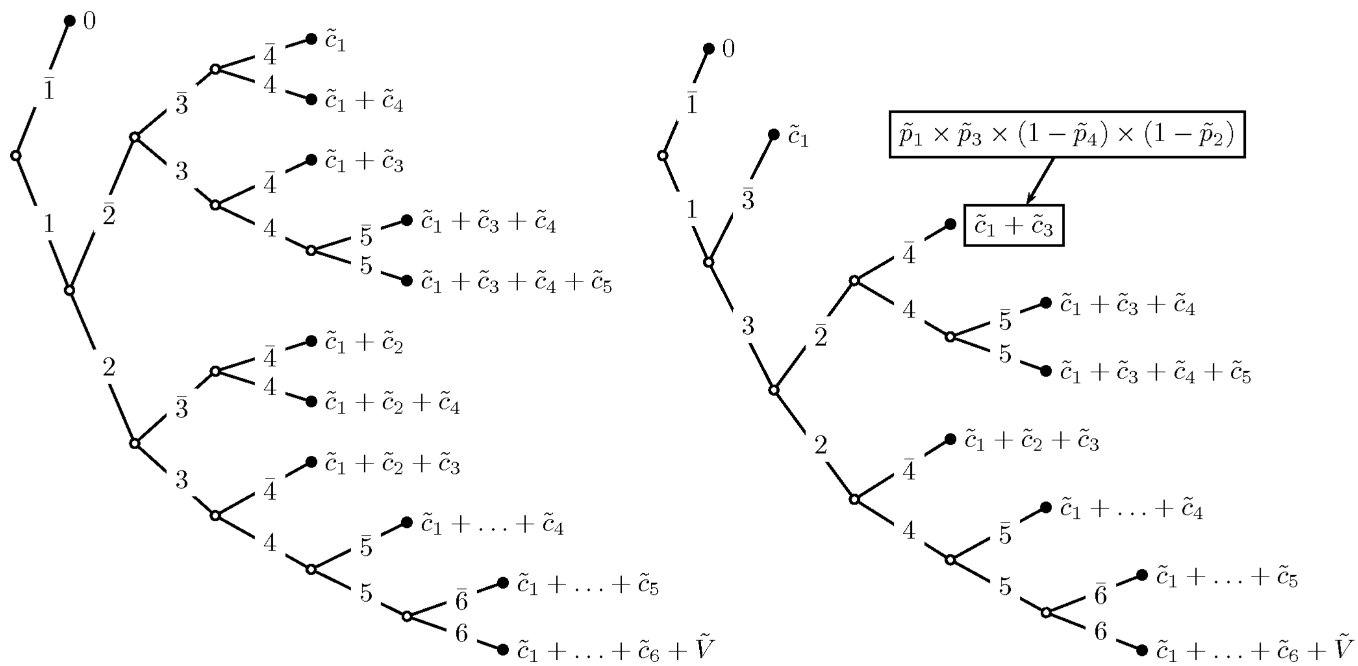

Figure 3: Event trees corresponding to the two schedules in Figure 2.

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, like one of the two shown in Figure 2, 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. Each way of scheduling tasks produces a different lottery. This brings up the next problem, which is how to compute all the ways of scheduling the tasks.

Example. To illustrate the idea, let's consider the event trees in Figure 3 which are for the time allocations in Figure 2. Considering the more sequential schedule shown on the right, we note that with probability $1 - \tilde{p}_1$ task 1 fails, the agent does not pay or receive anything and stops the execution (path $\bar{1}$ in the right 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 work and paying a total of c_1 (path $1 \rightarrow \bar{3}$), or it is completed with the corresponding probability $\tilde{p}_3^c = \tilde{p}_1 \times \tilde{p}_3$. 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 is $1 \rightarrow 3 \rightarrow \bar{2} \rightarrow \bar{4}$ and the corresponding payoff-probability pair is framed in the figure.

Counting and Generating Task Orderings

Given a task network, there may be many different ways to order the individual tasks that are consistent with the precedence constraints and yet result in different event trees. So, the agent needs to explore all the task orderings and select the ordering that produces the largest CE.

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

We have shown (Babanov, Collins, & Gini 2003) that task orderings can be counted by examining recursively cases when one of the tasks that can be started next is started, or one of the tasks that were started is completed.

The number of orderings can grow exponentially with the number of tasks. Precedence constraints in general reduce this growth; in the case where all the tasks are in a single sequence, there is just one ordering.

Counting and Generating Event Trees

Instead of counting the number of task orderings, we can count unique event trees. Figure 4 illustrates the idea. Task starts, shown by vertical lines with the letter '*s*', and task finishes, shown with the letter '*f*'), alternate and always start or finish a non-empty set of tasks. We generate event trees, such as the ones shown in Figure 3, from this counting method. Notice that from Figure 4 we obtain the schedule equivalent to the right tree in Figure 3 and to both schedules in Figure 1.

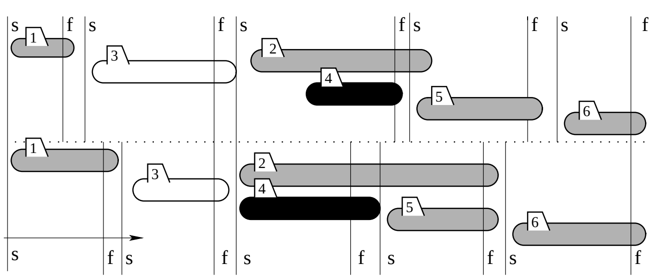

Figure 4: Method for counting event trees applied to the two schedules in Figure 1. Vertical lines show the steps from left to right. Despite the fact that the schedules are different, the event tree generated for both is the same, and it is the right tree in Figure 3.

The number of event trees is never greater and, in most cases, much lower than the number of task orderings. The numbers of task orderings and event trees for various examples are shown in Table 1. Our intuition is that event trees correspond to regions of continuity of CE, which should help us when maximizing CE.

Example. A task network with two parallel tasks has 6 different orderings, but only 3 event trees. Our task network from Figure 1 has 168 orderings (out of 7,484,400 orderings possible in a network of 6 tasks and no precedence constraints), but only 32 event trees.

Task network	# orderings	# event trees
Two sequential tasks	1	1
Two parallel tasks	6	3
Garage task network	168	32
Six parallel tasks	$7,484,400$	$81,663$

Table 1: Number of task orderings and event trees for sample problems.

Issues with Maximizing CE

Even for simple cases (e.g., a network of two parallel tasks) there are many local maxima for the CE that may differ significantly in utility values. Large variations of the CE values are caused mainly by different ordering of tasks that significantly impact the CE values. Different scheduling of tasks that are not stressed by time tend to produce small variations of the CE values. Examples are shown in Figure 5.

In our example in Figure 1, assume that it takes an hour to complete task 2 with high probability of success and it only takes 15 minutes for each of tasks 3, 4 and 5 to do the same. 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.

Although it is possible to write a CE expression for each payoff-probability tree, the number of possible orderings is too large for this to be practical. This suggests using instead a numerical approach, combined with heuristics to help choose "good" orderings for evaluation. We treat the problem as a constrained maximization, where the constraints are the precedence relations and time constraints.

Example. Figure 5 shows the schedules corresponding to some CE maxima for the task network in Figure 1. Examining this figure we see several distinct strategies of scheduling tasks 2 to 5. The most successful strategy, *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. Schedule *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 by scheduling it parallel to three (*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 in parallel to task 4 reveal that scheduling tasks 3 and 4 in parallel is

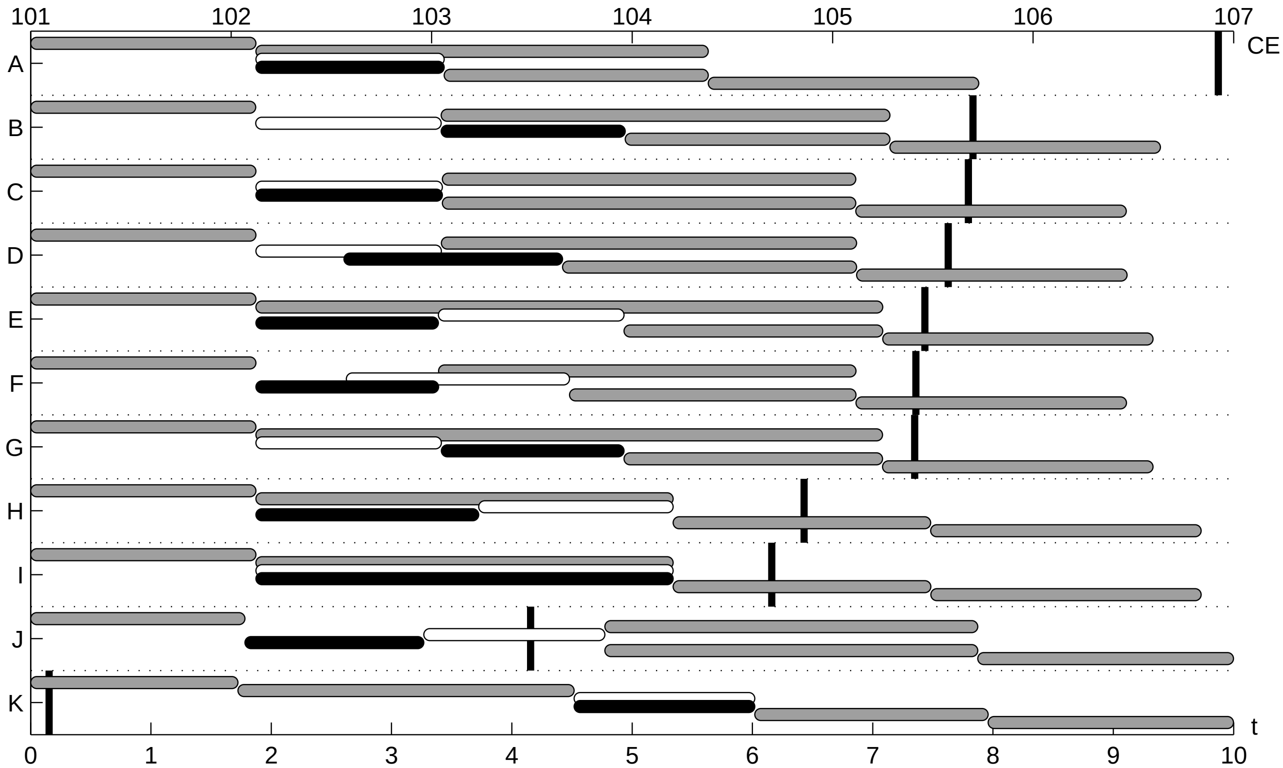

Figure 5: Maximizing schedules and CE values for the problem in Figure 1. The bottom x–axis shows time. Schedules are represented by rounded horizontal bars, and separated by dotted lines. Task 3 is in white and task 4 in black. Each schedule is indicated by a letter on the y–axis. The top x–axis represents CE, and vertical black bars show the CE values attained in the corresponding schedules by a risk neutral agent ($r = 0$).

important to ensure schedule flexibility (compare *J* to *C*).

The choice of one schedule over another must depend on the availability of bids for each task. For example, if there are many bidders for task 3 and 4, a risk-neutral agent may safely choose schedule *A* as a base for an RFQ. When one or both tasks are in rare supply, the agent might prefer schedule *I* and trade its expectations of large CE values for larger time windows.

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

Maximization Methods

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.

In our previous work (Babanov, Collins, & Gini 2003), we noticed that all constrained maximization methods we explored are "lazy," meaning that when initialized with a random schedule that results in a negative CE value, they often converge to a degenerate local maximum with one or more task durations equal to zero. This may be interpreted as a way of rejecting the plan. This dominates the results in cases with positive risk-aversity, which are the cases we are interested in.

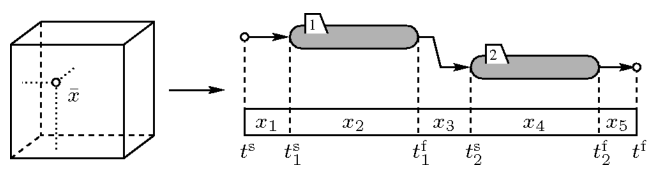

Figure 6: Transformation of a point in 5–dimensional unit cube to a 2-task schedule.

To avoid getting stuck in local pseudo-maxima, in (Babanov, Collins, & Gini 2004) we have proposed the following method. We project a set of precedence and order constraints in a space where they do not interact directly. We fix the order of start and finish times for each task, 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 $[t^s, t^f]$ interval. Figure 6 illustrates the variable transformation for a network consisting of two sequential tasks. More details on this and other maximization methods we tried can be found in (Babanov, Collins, & Gini 2004).

The number of local maxima of the CE function might grow exponentially with the number of tasks. We believe that the performance of maximization algorithms can be improved by using heuristics derived from domain properties. One heuristic is based on observing that there are obvious transitions between maximizing schedules, such as rescheduling two tasks from sequential to parallel or vice-versa. Such heuristic should reduce the problem from considering all task orderings to finding a few maxima and exploring near-optimal schedules suggested by the heuristic.

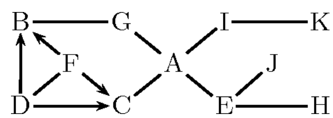

Figure 7: Some simple transitions between the schedules shown in Figure 5.

Generation of bids

The methods we have presented 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.

The same methods can be used to solve the decision problem of an agent when generating its own bids for tasks that have complex time constraints and interdependencies. The problem is similar, in that the agent has to decide if a new task fits into its existing schedule, and needs to find an optimal placement for the tasks it wants to bid on. To generate a bid, the agent needs also to estimate its own cost.

The factors involved in the agent determinations are multiple and application dependent. A major component is the time the agent needs to accomplish the task. This has to be evaluated in the context of its current and expected commitments. For a robot its distance from the expected position at the time the task will start is an important factor; for an active sensor node it might be the time to reorient itself. Energy consumption is another important factor, mainly when using miniature robots or sensors (Byers & Nasser 2000; Rybski *et al.* 2002; Sibley, Rahimi, & Sukhatme 2002) The degree of match of the agent resources to the task is another factor, given that robots have different sensors and might have varying level of success at doing the tasks.

Related Work

Many multi-agent systems use some form of auction (Krishna 2002) to allocate resources and arrive at coordinated decisions. When auctions are used to distribute tasks (Gerkey & Matarić 2002) or to schedule a resource (Wellman *et al.* 2003), items are typically auctioned one at a time. This reduces the opportunity for optimal allocations, and tends to limit the systems to a single market. A protocol to exchange information between several markets along one supply chain to achieve global efficiency is developed in (Babaioff & Nisan 2001). Unlike in our work, agents' decisions are only influenced by costs and not subject to scheduling constraints or risks.

A protocol for decentralized scheduling is proposed in (Wellman *et al.* 2001). The study is limited to scheduling a single resource, while we are interested in multiple resources. In (Wellman *et al.* 2003) agents bid for individual time slots on separate, simultaneous markets. Our agents use combinatorial bids.

The problem of deciding whether and how to fit a new task in a schedule of current commitments has been studied in (Hunsberger & Grosz 2000; Hunsberger 2002). These studies provide algorithms but do not address how to request bids and how the way bids are requested affects the resulting bids. The problem of how agents should coordinate the execution of a set of temporally-dependent tasks is discussed in (Hunsberger 2002) and a family of sound and complete algorithms is given.

Hadad et al. (Hadad & Kraus 2001; 2002) propose a method for cooperative agents to interleave planning and execution. The method enables each agent to plan and schedule its work independently of other agents, but to coordinate with them. Each agent has to keep track of the entire plan and to know all the steps in the plan, which makes the system not scalable. Complex steps are expanded at execution time, which limits the opportunities for optimization.

Negotiation strategies for concurrent tasks using complex constraints are proposed in (Zhang & Lesser 2002). We avoid this complexity by using bids and winner determination algorithms that include time. The task allocation problem is posed as an Optimal Assignment Problem in (Gerkey & Matarić 2002), with the assumption that a robot can do only a task at a time and each task can be done by a single robot. They assign tasks to robots to maximize overall expected performance, iterating the process over time. We assign tasks with combinatorial bids.

In (Tovey *et al.* 2005) exploration tasks are auctioned to robots using a sequential single-item auction. The allocation produced has a total cost which is at most twice the optimal cost. The method works well for robot exploration tasks, but has high communication costs and will fail if communication is lost.

Conclusions and Future Work

We have discussed how an agent should request quotes from other agents for tasks that have complex precedences and time constraints. To construct schedules for requesting quotes, the agent computes the schedule's equivalent in certain current money. This requires the agent to decide how to sequence the tasks, how much time to allocate to each, and to find the schedule for which the certainty equivalent is maximal. The same method can be used by agents to schedule their own activities or to decide for which activities to request bids from other agents.

Much work remains to be done to apply the methods proposed to groups of real robots to do a real set of tasks. Future work will include improving the algorithms for maximizing the certainty equivalent, developing domain heuristics for maximization, applying the methods we presented to agents when they generate their own bids, and experimenting with ways for measuring costs and rewards for cooperative agents.

References

Agassounon, W., and Martinoli, A. 2002. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proc. of the First Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, 1090–1097.

Babaioff, M., and Nisan, N. 2001. Concurrent auctions across the supply chain. In *Proc. of ACM Conf on Electronic Commerce (EC'01)*, 1–10. ACM Press.

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

Babanov, A.; Collins, J.; and Gini, M. 2004. Harnessing the search for rational bid schedules with stochastic search and domain-specific heuristics. In *Proc. of the Third Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, 269–276. New York: AAAI Press.

Byers, J., and Nasser, G. 2000. Utility-based decision making in wireless sensor networks. In *Proc. of IEEE MobiHoc 2000*.

Chien, S.; Barrett, A.; Estlin, T.; and Rabideau, G. 2000. A comparison of coordinated planning methods for cooperating rovers. In *Proc. of the Fourth Int'l Conf. on Autonomous Agents*, 100–101. ACM Press.

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

Collins, J. 2002. *Solving Combinatorial Auctions with Temporal Constraints in Economic Agents*. Ph.D. Dissertation, University of Minnesota, Minneapolis, Minnesota.

Dias, M. B.; Zlot, R. M.; Kalra, N.; and Stentz, A. T. 2005. Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

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

Gerkey, B. P., and Matarić, M. J. 2003. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proc. Int'l Conf. on Robotics and Automation*.

Goldberg, D.; Cicirello, V.; Dias, M. B.; Simmons, R.; Smith, S.; and Stentz, A. 2003. Market-based multi-robot planning in a distributed layered architecture. In *Multi-Robot Systems. Proc. 2003 Int'l Workshop on Multi-Robot Systems*, volume 2.

Hadad, M., and Kraus, S. 2001. A mechanism for temporal reasoning by collaborative agents. In Klusch, M., ed., *Cooperative Information Agents*, volume LNAI 2182. Springer-Verlag, Heidelberg, Germany. 229–234.

Hadad, M., and Kraus, S. 2002. Exchanging and combining temporal information in a cooperative environment. In Klusch, M., ed., *Cooperative Information Agents*, volume LNAI 2446. Springer-Verlag, Heidelberg, Germany.

Hunsberger, L., and Grosz, B. J. 2000. A combinatorial auction for collaborative planning. In *Proc. of 4th Int'l Conf on Multi-Agent Systems*, 151–158.

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

Krishna, V. 2002. *Auction Theory*. London, UK: Academic Press.

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

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

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

Rybski, P. E.; Stoeter, S. A.; Gini, M.; Hougen, D. F.; and Papanikolopoulos, N. 2002. Performance of a distributed robotic system using shared communications channels. *IEEE Trans. on Robotics and Automation* 22(5):713–727.

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

Sibley, G. T.; Rahimi, M. H.; and Sukhatme, G. S. 2002. Robomote: A tiny mobile robot platform for large-scale sensor networks. In *Proc. Int'l Conf. on Robotics and Automation*.

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

Sutherland, L. E. 1968. A futures market in computer time. *Comm. of the ACM* 11(6):449–451.

Tovey, C.; Lagoudakis, M.; Jain, S.; and Koenig, S. 2005. The generation of bidding rules for auction-based robot coordination. In *Multi-Robot Systems Workshop*.

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

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

Zhang, X., and Lesser, V. 2002. Multi-linked negotiation in multi-agent systems. In *Proc. of the First Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, volume 3, 1207–1214.