Plan Execution by Contracting in a Multi-Agent Environment

John Collins and Maria Gini
Department of Computer Science and Engineering,
University of Minnesota

Bamshad Mobasher
School of Computer Science, Telecommunications, and Information Systems,
DePaul University

We describe a computational model of planning, scheduling, and execution that is capable of supporting the interactions of a self-interested agent with other agents in a contracting environment over an extended period of time.

The model is appropriate for situations in which a customer agent, in order to fulfill its goals, must contract with other supplier agents for all or part of the necessary tasks. The agents are all assumed to be self-interested and with limited rationality. Supplier agents providing resources or services attempt to gain the greatest possible benefit, and customer agents requesting resources or services will attempt to pay the lowest price.

To evaluate this model, we have implemented a multi-agent distributed market and contracting environment and we are performing empirical studies on problem scenarios.

1 Overview

Plan Execution by Contracting is the type of activity in which a contractor agent, in order to fulfill its goals, must contract with other self-interested supplier agents for all or part of the necessary tasks. We are interested in agents that exhibit goal-directed behavior in a multi-agent environment. Specifically, we consider heterogeneous and self-interested agents that negotiate among themselves in order to carry out plans that require the resources of other agents. Furthermore, we assume the existence of an external market entity which facilitates agent interactions.

The contracting market framework we consider incorporates a three step protocol with a contractor agent issuing a call-for-bids, suppliers replying with bids, and the contractor accepting the bids it chooses. We avoid the need for open-ended negotiation by means of bid break-downs and time-based decommitment penalties [7]. Once the contractor agent receives the bids, it must evaluate the bids based on cost and time constraints, and select the optimal set of bids (or parts thereof) which can satisfy its goals. This *task assignment* forms the basis of an initial schedule for the execution of the tasks.

It is important to note that, in contrast to other planning/scheduling systems in which the agent is directly in charge of execution, in Plan Execution by Contracting, the contractor agent delegates the execution to supplier agents. As such, the system must provide a mechanism for monitoring task execution, throughout the life of the contract. This monitoring mechanism must interact with the planning and contracting components, since decommitments or under-performance by suppliers may require rebidding or replanning.

Within the market framework outlined above, these activities are encapsulated in a *market session* [9]. A market session (or simply a session) is the vehicle through which market services are delivered dynamically to participating agents. It serves as an encapsulation for a transaction in the market, as well as a persistent repository for the current state of the transaction. We have chosen the term "session" to emphasize the temporally extended nature of many of these interactions. For example, if an agent wishes to build a new house, it initiates a session and issues a call-for-bids. The session extends from the initial call-for-bids through the negotiation, awards, construction work, the paying of bills, and the final closing. In other

words, the session encloses the full life of a contract or a set of related contracts. The session mechanism ensures continuity of partially-completed transactions, and relieves the participating agents from having to keep track of detailed negotiation status themselves.

The interactions involved in the basic bidding and execution cycle among the contractor, supplier, and market session are illustrated in Figure 1. During execution, the interactions can be much more complex than indicated here, since either party may decommit from a contract (and pay a penalty), and the contractor is continuously monitoring and repairing its plan by replanning and rebidding when events fail to proceed according to expectations.

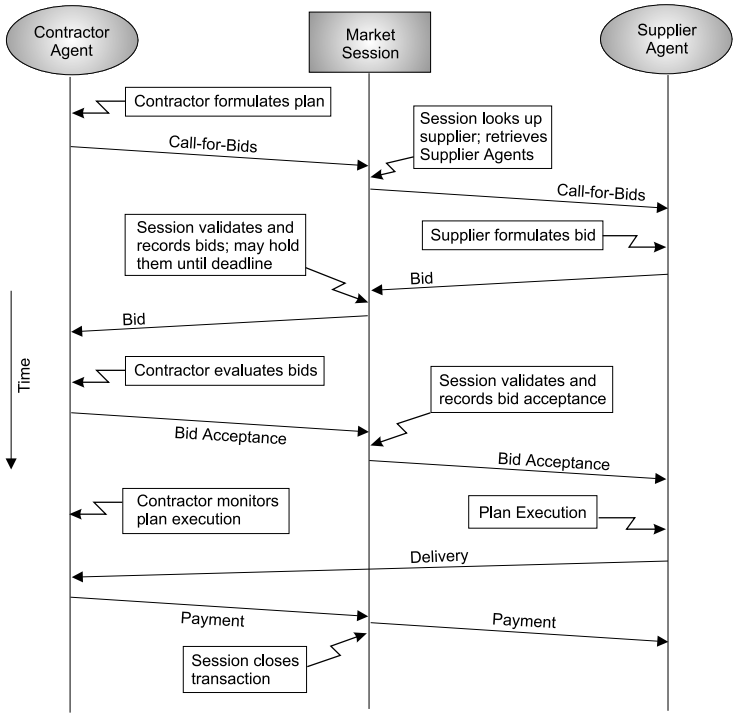

Figure 1: A Typical Session-Mediated Negotiation

At this point, we are mainly interested in the requirements of the contractor agent and its interaction with the Market Session. A contractor agent in our scheme contains three major functional elements, as shown in Figure 2. The *planner* converts a top-level goal into a partial-order plan, the *bid manager* acquires commitments from other agents to execute portions of the plan, and the *contract supervisor* oversees execution of the plan as contracted. We discuss each of these elements, as well as the relevant portions of the market infrastructure, in more detail in the following sections.

2 Components of the Contractor Agent

2.0.1 Planner

We assume the existence of a nonlinear planner [42, 33, 5, 26] that converts a top-level goal into a partial-order plan according to a domain model. A plan is composed of a combination of operations to be executed by the agent itself and operations that are available from other agents, as expressed in the Market Ontology. The ontology is a domain-specific description of operations, constraints, and other information [17] about tasks that can be performed by supplier agents in a particular market. The planner need not linearize the plans it emits with respect to time, since multiple agents in the market environment are capable of executing operations in parallel. Instead, temporal constraints among plan operations are embedded in the plan and included in the call-for-bids as outlined in [7].

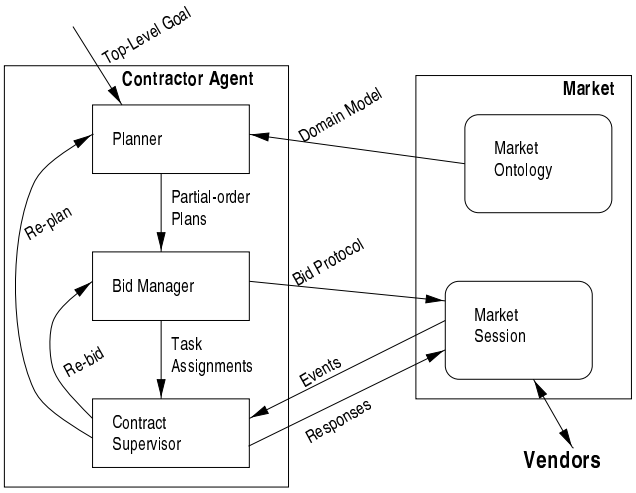

Figure 2: A Magnet Contractor Agent

The task network provided to the bid manager consists of a set of task descriptions of nonzero length, the temporal constraints among them, and possibly nonzero delays between tasks, to cover communication and transportation delays. The agent incorporates a temporal reasoning system similar to Dean and McDermott's Time Map Manager [11] to build and maintain this representation.

Our decision to use a traditional non-linear planner, as opposed to a formulation that integrates planning and scheduling [29], is based on the fact that the integrated approach operates from a resource-based time line, and in our system the set of resources is unknown at the time the planner operates. This, and the need for rational behavior on the part of the agent, introduces three complications to the standard non-linear planning problem. These issues must be addressed either by modifying the planner or by adding functionality to the agent.

1. For a feasible goal there are in general infinitely many plans that satisfy the goal. The planner, and indeed the agent itself, lacks the knowledge to determine at the time the plan is formulated which of these will be the lowest cost. Therefore, the ideal solution would be to have the planner produce a collection of possible plans and let the agent search over plan expansions to minimize cost. Unfortunately, this severely complicates the negotiation protocol, and we have elected not to attempt this. Instead, we propose using admittedly weak domain-specific heuristics to choose a plan that is likely to be among the lower-cost possible plans.

2. At the time the planner is invoked, some portion of a previous plan to achieve the goal may have already been executed. Finding a new plan that incorporates some or all of those elements might require modifications to the basic planning algorithm.

3. When the planner is invoked due to plan failure, it is possible that the plan has become infeasible due to lack of a vendor who is willing to bid on some particular element of the plan. Lack of bids could result either from no vendor willing to bid on a particular operation, or lack of resources to perform an operation in the time frame requested in the bid specification. This will require that the planner be capable of producing a new plan that does not incorporate the infeasible element.

2.1 Bid Manager

The bid manager is responsible for ensuring that resources are assigned to each of the tasks of a plan, that the assignments taken together form a feasible schedule, and that the cost of executing the plan is minimized. This cost must also be less than the value of the goal at the time the goal is reached. The bid manager may make assignments to local resources or acquire commitments from other agents to execute portions of the plan. Such commitments are gained through a negotiation protocol, such as the one described in [23] or the

one proposed in [36]. When the bid manager is invoked, the set of tasks in the plan may already be partially assigned. This is because the contract supervisor may use the bid manager to repair partially-completed plans in which previously determined assignments have failed.

The bid manager must perform these tasks in order to carry out its responsibilities: (1) constructing and issuing the call for bids, (2) receiving and evaluating bids, and (3) accepting a set of bids.

1. Construct and Issue Call For Bids.

For each element of the plan, the bid manager has two choices for making an assignment: (i) use a resource that is permanently available in the market at a fixed price, (see, for instance, how to create and use catalogues on the Web [24]), or (ii) assign a resource through the bidding process (possibly including local resources).

The call-for-bids $\mathcal{C}$ contains a subset of the tasks in the plan $\mathcal{P}$ with their precedence relations. There might be elements of $\mathcal{P}$ that are not included in the call-for-bids $\mathcal{C}$. This could happen, for instance, if the contracting agent decides to use an advertised resource.

Let $\mathcal{C}.t_{es}(s)$ denote the early start time for task s as specified in a call-for-bids $\mathcal{C}$. Other contexts in which times and other factors can be specified include bids ($b.t_{es}(s)$) and the computed critical path ($\mathcal{A}.t_{es}(s)$). The notation $s' \prec s$ denotes the constraint $t_e(s') \leq t_s(s)$, meaning that task s' must be complete before task s can start. A necessary condition for this is $(t_{es}(s') + d_e(s')) \leq (t_{lf}(s) - d_e(s))$.

For each task s in the call-for-bids $\mathcal{C}$, the bid manager must specify:

- a time window, consisting of an earliest start time $\mathcal{C}.t_{es}(s)$ and a latest finish time $\mathcal{C}.t_{lf}(s)$,
- a set of precedence relationships $\mathcal{C}.Pred(s) = \{s' \in \mathcal{C} \mid s' \prec s\}$, the set of other tasks $s' \in \mathcal{C}$ whose completion must precede the start of s, and
- a decommitment penalty $dcom(s)$. This is the penalty the supplier has to pay to the contractor agent if the supplier decommits. (See [7] for further explanation of the decommitment penalties.)

There is no requirement that the bidding be driven through a single call-for-bids, and there is no requirement that all precedence relationships be specified in the call-for-bids. The only requirement is that all specified precedence relationships be among tasks in a single call-for-bids.

When specifying time windows for tasks, the bid manager must in general do some guessing. The agent may have general knowledge of normal durations of tasks, or it may not. In order to minimize bid prices, vendors need flexibility in making resource commitments. Two possible strategies are (a) to use time windows that start at time t_0 and end at time t_{goal}, where t_0 is the start time for the entire plan and t_{goal} is the deadline for achieving the top level goal, and (b) to schedule tasks ahead of time using approximate durations, computing early start and late finish times using the Critical Path algorithm [22]. The latter approach is preferable if adequate approximations are known, because it will reduce the likelihood of bids being rejected for failure to mesh with the overall schedule, and it will reduce the average bid prices to the extent that it reduces speculative resource commitments on the part of suppliers.

The Critical Path algorithm walks the directed graph of tasks and precedence constraints, forward from time t_0 to compute the earliest start t_{es} and finish t_{ef} times for each task, and then backward from time t_{goal} to compute the latest finish t_{lf} and start t_{ls} times for each task. The minimum duration of the entire plan is called the *makespan* of the plan. The difference between t_{goal} and the latest early finish time is called the *total slack* of the plan. If t_{goal} is set equal to $t_0 + makespan$, then the total slack is 0, and all tasks for which $t_{ef} = t_{lf}$ are called *critical* tasks. Paths in the graph through critical tasks are called *critical paths.*

2. Receive and Evaluate Bids.

When bids are returned, the bid manager must assemble them into a minimum-cost feasible schedule in order to determine which bids to accept.

Let $\mathcal{B}$ be the set of returned bids, each bid $b \in \mathcal{B}$ includes a price $b.price(\mathcal{P}')$ for some subset $\mathcal{P}'$ of the elements of the call-for-bids, prices $b.price(s)$ for individual elements of the bid subset $s \in \mathcal{P}'$, and timing information for the bid elements. Note that $(b.price(\mathcal{P}') \leq \sum_{s \in \mathcal{P}'} b.price(s)$ represents a *discount* associated with the bid b. Timing information for a bid b and a task s includes early start $b.t_{es}(s)$, late finish $b.t_{lf}(s)$, and expected duration $b.d_e(s)$ for each task in the bid. The semantics of a bid is that a supplier is willing to perform the task s for the bid price $b.price(s)$ starting at any time $t_s(s)$ such that $b.t_{es}(s) \leq t_s(s) \leq (b.t_{lf}(s) - b.d_e(s))$, and finishing at time $(t_s(s) + b.d_e(s))$.

The bid evaluation for a plan $\mathcal{P}$ consisting of tasks $s \in \mathcal{P}$, may be given as an objective function of the total *cost* and total expected schedule *risk*. A naive method for calculating the total cost would be to determine the sum of the minimum costs for each subtask over all the returned bids. In other words, given the subset $\mathcal{B}_s$ of all returned bids which contain the subtask s, the cost of s can be determined as:

$$cost(s) = \min_{b \in \mathcal{B}_s} \left(b.price(s) + b.dcom(s) \times b.Pr_{dcom(s)} \right)$$

where $b.price(s)$ is the bid price *price* for task s in the bid b containing task s, and the expected cost of the decommitment penalty for a task in a specific bid b is computed by multiplying the decommitment penalty of the bid $b.dcom(s)$ by the probability of decommitment $b.Pr_{dcom(s)}$.

The total cost can now be determined as $\sum_{s \in \mathcal{P}} cost(s)$. However, this naive evaluation does not take into account the *discounts* that may be available as part of each returned bid. We have proposed an efficient algorithm for finding a set of bids [8].

As noted above, another important component of bid evaluation is the determination of the expected schedule risk. The contractor agent would seek to minimize both the total cost and the total schedule risk. The expected cost associated with schedule risk can be given by a heuristic function *risk* applied to a constraint-tightness measurement, $k(path)$, which is determined according to the slack available along each partial path through the plan:

$$TotalRisk = \sum_{path \in \mathcal{P}} risk(b_{path}.k(path)).$$

Intuitively, the risk associated with constraint tightness is higher whenever accepting a particular set of bids will increase the probability of missing the goal deadline, or of missing the latest start time (which is $b.t_{lf}(s) - b.d_e(s)$) specified in the bid $b \in \mathcal{B}_s$ containing task s. Finding an effective and computationally inexpensive formula for this will be addressed empirically. This is a similar problem to that addressed by the notion of *textures* in [15]. Our approach is to measure constraint tightness by *path*, where a path is a sequence of tasks starting at the beginning of the plan and extending toward the goal along successor relations. We use partial paths as well as complete paths for this calculation, because tasks can be constrained both by their precedence relationships and by the time windows specified in the bids. For example, in the task network in Figure 3, there are 9 paths, all starting with s_1, as follows:

s_1	$s_1 \prec s_3$	$s_1 \prec s_4$
$s_1 \prec s_2$	$s_1 \prec s_3 \prec s_5$	$s_1 \prec s_4 \prec s_5$
$s_1 \prec s_2 \prec s_6$	$s_1 \prec s_3 \prec s_5 \prec s_6$	$s_1 \prec s_4 \prec s_5 \prec s_6$

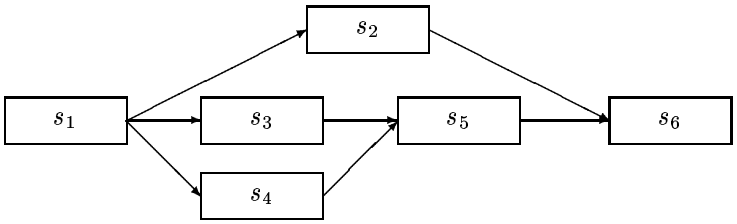

Figure 3: Example task network

Along each path, we measure constraint tightness as the ratio of slack to expected duration, as in

$$k(path) = \frac{t_{lf}(s_{last}) - t_0 - \sum_{s \in path} d_e(s)}{\sum_{s \in path} d_e(s)}$$

where s_{last} is the last task on *path* and t_0 is the start time. An example of a schedule risk computation is $risk(k(path)) = W_1 e^{-W_2 k(path)}$ where W_1 and W_2 are tuning parameters. Parameter W_2 determines the

slope of the curve, while W_1 scales the result. A path with no slack would then have a constraint tightness cost of W_1. Whether to allow slightly infeasible schedules to be contracted at all depends on the ability of the contract supervisor to recover from plan breakage, for example by offering bidders a reward for early completion.

We will consider additional factors to compute risk. These will be based on two sources of information:

1. Knowledge of the domain, or experience. This will be used to inform the agent that some kinds of work are inherently riskier than others. Examples might be tasks that are more complex or involve more creative content. We expect this knowledge to come from the ontology of the domain.
2. Experience with suppliers, either on the part of the agent or shared among agents (for example, through the Better Business Bureau described in [9]). This will lead the agent to ascribe a higher risk to some suppliers than others.

3. Accept Bids.

After building the schedule, the bid manager sends bid acceptance messages to the vendors of accepted bids, specifying which parts of which bids are accepted. In our contracting protocol, this completes the negotiation process.

2.2 Contract supervisor

The contract supervisor is responsible for overseeing execution of the plan as contracted, and making decisions about how to respond when events do not proceed as expected. It receives the task assignments from the bid manager. Through the market, it also receives updates on plan execution from contracted vendors. It maintains the time map with tasks, vendor commitments, and temporal constraints among tasks. For each event, it must decide whether to respond, and if so, whether to respond directly to a particular vendor, whether to re-bid a portion of the plan, or whether to re-plan and re-bid one or more subgoals of the plan.

The bid manager produces a set of task assignments, each of which includes one or more tasks from the plan, along with the contract data for execution of that task. Contract data includes the task, the resources committed to carrying out that task, an agreed-upon price, an agreed-upon time window and temporal constraints, and the supplier and contractor decommitment penalties. The temporal data from the accepted bids is added to the set of constraints in the time map.

All the activities of the contract supervisor revolve around the maintenance of the time map. The time map can be thought of visually as a Gantt chart, decorated with contract data and temporal constraints among tasks. For each task s the time map records an early start time $t_{es}(s)$, a late finish time $t_{lf}(s)$, an expected duration $d_e(s)$, and the set of tasks $Succ(s) = \{s' \in \mathcal{P} | s \prec s'\}$ that are forced to be successors of s by precedence constraints.[1] Let b_s denote a returned bid that contains a subtask s. The early start time for a task s is

$$t_{es}(s) = max(\mathcal{A}.t_{es}(s), b_s.t_{es}(s))$$

where the notation $\mathcal{A}.t_{es}(s)$ refers to the earliest possible start time for task s as computed by the Critical Path algorithm. In other words, the derived early start time for a task s is the later of the early start time for s computed by the Critical Path algorithm and the beginning of the time window in the accepted bid for s. Similarly, the late finish time for a task s is

$$t_{lf}(s) = min(\mathcal{A}.t_{lf}(s), b_s.t_{lf}(s))$$

or the earlier of the late finish time for task s computed by the Critical Path algorithm and the end of the time window in the accepted bid for s. The *slack* for task s is

$$slack(s) = t_{lf}(s) - t_{es}(s) - b_s.d_e(s).$$

A task s for which $slack(s) = 0$ is said to be a *critical* task. This can occur for any one of three reasons:

[1] Note that the relationship $Succ(s)$ refers to the *successors* of s, while the relationship $Pred(s)$ in the call-for-bids refers to the *predecessors* of s.

1. If the completion time $t_{lf}(s)$ is constrained by $b_s.t_{lf}(s)$ in the accepted bid, then a delay in starting s would violate the bid contract.
2. If the completion time $t_{lf}(s)$ is constrained by the deadline for the overall goal t_{goal}, either directly or transitively through $Succ^{\star}(s)$, derived by computing the Critical Path, then a delay in completion will delay goal achievement.
3. If the completion time $t_{lf}(s)$ is transitively constrained by $b_{s'}.t_{lf}(s')$ for some $s' \in Succ^{\star}(s)$, then a delay in starting s would violate the bid contract for s'.

3 Interactions of the Contractor Agent with the Market Session

As time passes and the execution of the plan proceeds, the contract supervisor works in conjunction with the market session to drive the plan to completion. In general, the session is responsible for releasing tasks to the suppliers when their prerequisites are satisfied, and for assessing decommitment penalties when the parties fail to satisfy their commitments. In the process, the session forwards to the contract supervisor notifications of task release and task completion events. The contract supervisor is then responsible for making decisions and taking appropriate action in response to those notifications.

In order to carry out its role, the market session maintains a *performance monitoring table*, which is essentially a stripped-down version of the time map maintained by the contractor agent. It contains, for each contracted task s, the winning bidder, the early start and late finish times $t_{es}(s)$ and $t_{lf}(s)$, the contracted subset of successor tasks $\mathcal{C}.Succ(s)$ (it is possible that not all tasks $s' \in Succ(s)$ have been contracted out), and the decommitment penalty function $dcom(s)$.

The performance monitoring table is maintained by updating the expected release time $t_r(s')$ for each task $s' \in \mathcal{C}.Succ^{\star}(s)$ whenever the expected finish time $t_f(s)$ of a task s changes. Each time the expected release time $t_r(s)$ of a task s is changed, the market session updates

$$slack(s) = t_{lf}(s) - d_e(s) - t_r(s)$$

to determine whether task s has become critical ($slack(s) = 0$), or infeasible ($slack(s) < 0$). If so, the contractor is notified to enable it to take appropriate action.

Each supplier agent s is notified by the market session of task releases, and of changes to $t_r(s)$, for each task s for which a bid has been awarded to supplier s. Because of this limited visibility, the market session can be implemented as a fully distributed component which keeps the information related to supplier s local to that supplier, thus reducing network traffic and eliminating the possible bottleneck that would occur if all update processing were funneled through a single point.

The contract supervisor maintains the full version of the time map. It is responsible for updating expected release times whenever the market session does not have full precedence information, and for releasing each task $s' \in Succ(s)$ upon completion of any task s that is not known to the market or for which the precedence relationship $s \prec s'$ is not known to the market. The contract supervisor is also responsible for responding to situations where the plan becomes infeasible, and for notifying the market session of the actions it decides to take.

There are two types of events to which the contract supervisor must respond. Event notifications from the market session trigger updates to the time map, and possibly other actions as detailed below. The passage of time without notification of an expected event is also considered to be an event by the contract supervisor. In order to make rational decisions in response to events, the contract supervisor maintains a set of metrics on the plan and its representation in the time map. These metrics include:

- the time-dependent value of the goal V_{goal},[2] discounted by the likelihood of plan failure and expectation of paying contractor decommitment penalties,
- the total slack in the schedule, $slack_{total} = t_{goal} - d_m - t_0$, where d_m is the *makespan* derived from the the Critical Path calculation,

[2] In general, the value of the goal may vary arbitrarily with time, but initially we consider only the case where the value is a constant $V_{goal} = V$ until some time t_{goal}, after which $V_{goal} = 0$.

- the total cost of the plan, including the cost of all outstanding bids, the amount already paid on completed bids, the cost of using local resources to process any locally-assigned portions of the plan, minus the expected value of supplier decommitment penalties.

Following are the classes of events to which the contract supervisor must respond, and a brief outline of the response options.

Nominal Completion. Task s was completed (as expected) at time $t_f(s) = t_s(s) + d_e(s)$, where $d_e(s)$ is the duration of the task as specified in the bid. No action is required.

Early Completion. Task s was completed at time $t_f(s) < t_s(s) + d_e(s)$. For $s' \in Succ^\star(s)$, update $t_r(s')$ if necessary (it is not necessary if $t_r(s') = t_{es}(s')$). If any update to a task $s' \in \mathcal{C}.Succ(s)$ is not known to the market session, the contract supervisor must notify the session of the change. If a critical path is affected, and if the value of the plan could be improved by changing $b.t_{es}(s)$ for some task s, then the contract supervisor could request the vendor of s whether the schedule can be moved up, and for what cost. After evaluating the cost/benefit tradeoff, the contract supervisor will request schedule changes accordingly.

Vendor Decommitment. When a supplier decommits, the contract supervisor has three choices: (1) contractor decommitment, (2) attempt to re-bid decommitted task(s), (3) attempt to re-plan and re-bid unsatisfied subgoal(s). The choice that is expected to maximize the profit is the one that will be made.

Missing Event. Completion events are considered missing if their failure to arrive triggers violation of a temporal constraint. This is considered non-performance on the part of the vendor. The contract supervisor responds by notifying the market session of vendor non-performance.

Late Completion. A late completion event is one that occurs later than promised but does not violate temporal constraints. It is a configuration option whether to treat this as a missing event. If not, the contract supervisor responds by re-evaluating the critical path and notifying vendors of affected tasks of changed time windows. All such time window changes will result in tighter windows.

Notice of Late Completion. If a vendor wishes to extend a deadline, it must initiate a negotiation with the contractor using a Notice of Late Completion, giving a new expected completion time and an updated bid. The contract supervisor must then choose whether to accept the updated bid, with the new time commitment, or whether to treat this event as a contractor decommitment.

4 Related Work

Planning, Scheduling, and Execution

Interleaving planning with execution has been proposed by many (for instance, [28]) but it is rarely used in practice. Wilkins [47] interleaves planning with execution, but uses two different systems, one for planning (SIPE-2) and one for reactive execution (PRS-CL).

Projects such as O-Plan [10] and the Multiagent Planning architecture (MPA) [48] aim at developing an architecture for large planning problems and mixed initiative planning, where coordination of experts and adaptation to changes in tasks, requirements, and environment are needed. Our goal is to automate the planning/scheduling/execution cycle of a single autonomous agent that needs the services of other agents to accomplish its task. Pollack's DIPART system [30] assumes multiple agents that operate independently but all work towards the achievement of a global goal. Our agents are trying to achieve their own goal and to maximize their profit, there is no global goal.

Various classes of scheduling problems have been considered in the Operations Research literature [22, 25, 16]. Many interesting scheduling problems are computationally intractable, and numerous heuristic approaches have been described [6, 21, 31]. Much of the recent work in scheduling has focused on the problem of maintaining or updating an existing schedule in the face of changes [50, 40]. All of these systems are concerned with determining schedules for individual resources; the assumption is that there is some set of tasks to be done, and some set of resources available to do those tasks, and the problem is to find an

optimal or near-optimal assignment of tasks to resources over time. The contractor agent we describe is not resource-limited; these scheduling approaches will be needed for the supplier agents, but they are not well suited for the problem that the contractor agent must solve.

The notion of using an explicit time-map to support reasoning about events, actions, states, and causality over time was first described in [11] and further elaborated in [4]. Hanks [20] has dealt with the problem of projecting the effects of actions into the future under uncertainty. Schwalb, Kask, and Dechter [38] describe a formal system for reasoning about time and events, including another form of a time map called a Conditional Temporal Network.

The problem faced by the contractor agent in our design is that of monitoring a plan and its schedule, and finding ways to repair it when it is broken. Muscettola [29] and Tate et al [43] advocate combining the planning and scheduling problems to deal with this issue. Muscettola's approach is based on maintaining state information over time for a relatively fixed set of resources, and so is not directly applicable to our work. Doyle [13] argues for an economic approach to plan maintenance, an approach that makes sense in the context of a market-based agent.

Market Architectures for Agents

Markets play an essential role in the economy, by facilitating the exchange of information, goods, and services [1], and there is growing evidence that software agents will play an increasing role as mediators in electronic markets [19, 41]. Mediators typically provide services such as searching for a product or supplier, negotiating the terms of a deal, providing payment services, and ensuring delivery of goods.

The architecture we propose improves on current proposals for standardizing an open architecture for electronic commerce [44, 27, 3] by adding support for complex negotiations during the contracting phase, for monitoring the execution of contracts, and for curtailing unproductive value-based [32] or time-based [7] counterspeculation by participating agents.

Of the essential functions of a market identified by [1], existing software agents mostly satisfy the need to search for product and price information (see, for instance, [12], but there is a growing need for agents capable of more sophisticated automated negotiations [2, 18].

Automated contracting protocols generally assume direct agent-to-agent negotiation. For example, Smith [39] pioneered research in communication among cooperating distributed agents with the Contract Net protocol. The Contract Net has been extended by Sandholm and Lesser [34] to self-interested agents. In these systems, agents communicate and negotiate directly with each other. In our proposed work, agents interact with each other through a market.

Mechanisms to reduce counterspeculation, such as the Clarke tax mechanism [14] or the Vickrey auction [45] have been proposed for automated negotiation of self-interested agents. In Sandholm's TRACONET system [36] both the bidding and contract execution mechanisms are complicated by the need to operate in an environment where agents cannot trust each other. He does not assume or take advantage of an independent market infrastructure. The architecture we present can support, among other things, the Vickrey auction, and eliminates one of its limitations by providing a structure that can act as a trusted auctioneer [35].

To the extent that we require the existence of an external market mechanism as an intermediary, our proposed framework is similar to that of Wellman's market-oriented programming used in AuctionBot [49] and The University of Michigan Digital Library [46]. AuctionBot, for instance, supports a variety of auction types each imposing a set of market rules on the agent interactions. Hence, the auctions, themselves, become the intermediaries. In contrast, our framework provides explicit market mechanisms which can not only specify and enforce auction parameters, but also support more complex interactions such as automated contracting. Furthermore, these market mechanisms also enforce general market rules and "social laws", such as government regulations, by which all participants must abide.

Rosenschein and Zlotkin [32] showed how the behavior of the agents can be influenced by the set of rules that the system designers choose for the agents' environment. In their study the agents are homogeneous and there are no side payments. In other words, the goal is to share the work, not to pay for work. Sandholm's agents [34, 37, 36] redistribute work among themselves by a contracting mechanism. Unlike Rosenschein and Zlotkin, Sandholm considers agreements involving explicit payments, but he also assumes that the agents are homogeneous – they have equivalent capabilities, and any agent can handle any task. We do not make

any of these assumptions. Our agents are heterogeneous, and decide on their own what tasks to handle by responding to a call for bids that requires specific tasks or products within a specified time window.

5 Conclusions and Future Work

The MAGNET system offers a framework and capability for multiple agents to reach agreement on complex contracts in a fully autonomous manner.

This paper has presented a work in progress. The basic MAGNET infrastructure and the evaluator have been implemented and tested, and much of the functionality needed for mixed-initiative behavior is in place. A user interface that allows user interaction with the evaluator has not been built, and effectively communicating such multidimensional evaluation results will be a challenge.

References

[1] Yannis Bakos. The emerging role of electronic marketplaces on the Internet. *Comm. of the ACM*, 41(8):33–42, August 1998.

[2] Carrie Beam and Arie Segev. Automated negotiations: A survey of the state of the art. Technical Report CITM Working Paper 96-WP-1022, Walter A. Haas School of Business, 1997.

[3] Martin Bichler and Arie Segev. A brokerage framework for internet commerce. Technical Report CITM Working Paper 98-WP-1031, Walter A. Haas Schhol of Business, 1998.

[4] Mark S. Boddy. Temporal reasoning for planning and scheduling in complex domains: Lessons learned. In Austin Tate, editor, *Advanced Planning Technology*, pages 77–83. AAAI Press, Menlo Park, CA, 1996.

[5] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[6] Anne Collinot, Claude Le Pape, and Gerard Pinoteau. Sonia: A knowledge-based scheduling system. *Artificial Intelligence in Engineering*, 3:86–94, 1988.

[7] John Collins, Scott Jamison, Maria Gini, and Bamshad Mobasher. Temporal strategies in a multi-agent contracting protocol. In *AAAI-97 Workshop on AI in Electronic Commerce*, July 1997.

[8] John Collins, Makesim Tsvetovat, Rashmi Sundareswara, Joshua Van Tonder, Maria Gini, and Bamshad Mobasher. Evaluating risk: Flexibility and feasibility in multi-agent contracting. Technical Report 99-001, University of Minnesota Department of Computer Science and Engineering, Minneapolis, Minnesota, February 1999.

[9] John Collins, Ben Youngdahl, Scott Jamison, Bamshad Mobasher, and Maria Gini. A market architecture for multi-agent contracting. In *Proc. of the Second Int'l Conf. on Autonomous Agents*, pages 285–292, May 1998.

[10] Ken Currie and Austin Tate. O-plan: The open planning architecture. *Artificial Intelligence*, 52:49–86, 1991.

[11] Thomas L. Dean and Drew V. McDermott. Temporal data base management. *Artificial Intelligence*, 32:1–55, 1987.

[12] B. Doorenbos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proc. of the First Int'l Conf. on Autonomous Agents*, pages 39–48, February 1997.

[13] Jon Doyle. Toward rational planning and replanning. In Austin Tate, editor, *Advanced Planning Technology*, pages 130–135. AAAI Press, Menlo Park, CA, 1996.

[14] E. Ephrati and J. Rosenschein. Deriving consensus in multiagent systems. *Artificial Intelligence*, 87:21–74, 1996.

[15] Mark S. Fox, Norman Sadeh, and Can Baykan. Constrained heuristic search. In *IJCAI89*, volume 1, pages 309–315, Detroit, MI USA, August 1989.

[16] M. R. Garey, R. L. Graham, and D. S. Johnson. Performance guarantees for scheduling algorithms. *Operations Research*, 26(1):3–21, January 1978.

[17] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[18] Robert H. Guttman and Pattie Maes. Cooperative vs competitive multi-agent negotiations in retail electronic commerce. In *2nd Int'l Workshop on Cooperative Information Agents (CIA'98)*, July 1998.

[19] Robert H. Guttman, Alexandros G. Moukas, and Pattie Maes. Agent-mediated electronic commerce: a survey. *Knowledge Engineering Review*, June 1998.

[20] Steve Hanks. Practical temporal projection. In *AAAI90*, volume 1, pages 158–163, Menlo Park, CA, July 1990. AAAI, AAAI Press.

[21] Othar Hansson and Andrew Mayer. Dts: A decision-theoretic scheduler. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, pages 371–388. Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[22] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research.* McGraw-Hill, 1990.

[23] Scott Jamison. A negotiation protocol and market agent model for complex transactions in electronic commerce. Technical report, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN, 1997.

[24] A. M. Keller. Smart catalogs and virtual catalogs. In R. Kalakota and A. Whinston, editors, *Readings in Electronic Commerce.* Addison-Wesley, Reading, Mass., 1996.

[25] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, January 1978.

[26] David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proc. of the Ninth Nat'l Conf. on Artificial Intelligence*, pages 634–639, Menlo Park, CA, July 1991. AAAI, AAAI Press.

[27] S. McConnell, M. Merz, L. Maesano, and M. Witthaut. An open architecture for electronic commerce. Technical report, Object Management Group, Cambridge, MA, 1997.

[28] Drew McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[29] Nicola Muscettola. HSTS: Integrating planning and scheduling. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, chapter 6, pages 169–212. Morgan Kaufmann, San Francisco, CA, 1994.

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

[31] Y. Ronen, D. Mosse', and M. E. Pollack. Value-density algorithms for the deliberation scheduling problem. *SIGART Bulletin*, 7(2), 1996.

[32] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter.* MIT Press, Cambridge, MA, 1994.

[33] E. D. Sacerdoti. *A Structure for Plans and Behavior.* American Elsevier, New York, NY, 1977.

[34] Tuomas Sandholm and Victor Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *1st Int'l Conf. on Multiagent Systems*, pages 328–335, San Francisco, 1995.

[35] Tuomas W. Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In *Second Int'l Conf on Multi-Agent Systems)*, Kyoto, Japan, December 1996.

[36] Tuomas W. Sandholm. *Negotiation Among Self-Interested Computationally Limited Agents.* PhD thesis, University of Massachusetts, 1996.

[37] Tuomas W. Sandholm and Victor Lesser. On automated contracting in multi-enterprise manufacturing. In *Distributed Enterprise: Advanced Systems and Tools*, pages 33–42, Edinburgh, Scotland, 1995.

[38] Eddie Schwalb, Kalev Kask, and Rina Dechter. Temporal reasoning with constraints on fluents and events. In *AAAI94*, Seattle, WA, August 1994. AAAI.

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

[40] Stephen F. Smith. OPIS: A methodology and architecture for reactive scheduling. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, chapter 8, pages 29–66. Morgan Kaufmann, San Francisco, CA, 1994.

[41] K. Sycara, K. Decker, and M. Williamson. Middle-agents for the Internet. In *Proc. of the 15th Joint Conf. on Artificial Intelligence*, 1997.

[42] Austin Tate. Generating project networks. In *Proceedings of the Joint Conference on Artificial Intelligence (IJCAI-77)*, Cambridge, MA, 1977.

[43] Austin Tate, Brian Drabble, and Richard Kirby. O-plan2: An open architecture for command, planning, and control. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, chapter 7, pages 213–240. Morgan Kaufmann, San Francisco, CA, 1994.

[44] J. M. Tennenbaum, T. S. Chowdhry, and K. Hughes. eCo System: CommerceNet's architectural framework for internet commerce. Technical report, Object Management Group, Cambridge, MA, 1997.

[45] H. R. Varian. Economic mechanism design for computerized agents. In *USENIX Workshop on Electronic Commerce*, New York, NY, July 1995.

[46] M.P. Wellman, W.P. Birmingham, and E.H. Durfee. The digital library as a community of information agents. *IEEE Expert*, 11(3):10–11, June 1996.

[47] D. E Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:197–227, 1994.

[48] D.E. Wilkins, K. L. Myers, M. desJardins, and P. M. Berry. Summary of multiagent planning architecture. Technical Report SRI Project 7150, SRI International, 1997.

[49] 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*, May 1998.

[50] Monte Zweben, Brian Daun, Eugene Davis, and Michael Deale. Scheduling and rescheduling with iterative repair. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, chapter 8, pages 241–256. Morgan Kaufmann, San Francisco, CA, 1994.