Plan Execution by Contracting in a Multi-Agent Environment

John Collins, Corey Bilot, Maxsim Tsvetovat, and Maria Gini
Dept of Computer Science and Engineering, University of Minnesota

Bamshad Mobasher
School of Computer Science, Telecomm., and Info. Systems, DePaul University

Abstract

We present 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 assumed to be self-interested and with limited rationality. Supplier agents attempt to gain the greatest possible benefit, and customer agents will attempt to pay the lowest price.

1 Introduction

The University of Minnesota's MAGNET (Multi-Agent Negotiation Testbed) system is an innovative agent-based approach to logistics and supply-chain management problems. It is designed to support the execution of complex plans among a population of independent, autonomous, heterogeneous, self-interested agents. We call this activity *Plan Execution By Contracting*. The MAGNET system is based on three elements:

1. The *agents*. Each agent is an independent entity, with its own structure, goals, and resources. In general, the resources under "control" of an individual agent are not sufficient to satisfy that agent's goals, and so the agent must negotiate with other agents in its environment in order to meet its goals.

2. The *market*. Agents find each other and carry out negotiations through a distributed infrastructure that enforces protocol rules, limits opportunities for fraud and counterspeculation, and tracks the requests, commitments, and progress toward goals among the agent population.

3. The *protocol*. Although the basic structure of the agents and the market will support a variety of negotiation protocols, we have developed a finite, leveled commitment protocol that limits the time and bandwidth required for the negotiation process without limiting the scope of possible agreements.

Because it is based on a market-based economic model, a MAGNET system acts to allocate resources to their highest-value uses over time in a completely distributed fashion. Because MAGNET agents are heterogeneous and self-interested, they may represent real-world entities who may tune their levels of cooperation and competitiveness to suit their own needs.

MAGNET is designed to extend the applicability of agent negotiation to domains where schedules for production and delivery affect the cost and feasibility of services and products, and where monitoring the performance of task execution is an essential part of the process. This is especially important for application domains such as logistics, dynamic planning and scheduling, and coordination of supply-chain management with production scheduling [21]. In these applications, factors such as flexibility, ease of use, quality, risk, and performance can be more important than cost [10].

In many of these domains, fully autonomous behavior is impractical or unacceptable due to risk factors that cannot easily be quantified [3], or simply because decision making is the responsibility of a person. In other domains, where specifications and business relationships are in place among the negotiating entities, fully autonomous behavior may be very acceptable.

We describe an approach to automated contracting that makes effective use of the capabilities of a community of automated agents, by using them to support human decision-making. The level of autonomy is adjustable; agents may act as communication conduits with some useful evaluation functions, or they may be given full authority to make commitments on behalf of their principals within a circumscribed domain. Agent interactions are mediated through an independent market infrastructure which, among other services, provides a domain ontology, the plan execution by contracting protocol, and authentication services.

2 A MAGNET Agent

In general, a MAGNET agent has four basic functions: planning, negotiation, execution monitoring, and resource management. Within the scope of a negotiation, we distinguish between two agent *roles*, the *Contractor* and the *Supplier*. A Contractor is an agent who has a plan to satisfy some goal, and needs resources outside its direct control in order to carry out that plan.

The plan may have a *value* that varies over time. In response to a call-for-bids, some set of Supplier agents may offer to provide the requested resources or services, for specified prices, over specified time periods. Figure 5 shows the general relationships among Contractor agents, Supplier agents, and the Market.

Once the Contractor agent receives bids, it must evaluate them based on cost, risk, and time constraints, and select the optimal set of bids (or parts thereof) which can satisfy its goals [4]. This *task assignment* forms the basis of an initial schedule for the execution of the tasks.

After a set of bids are selected, Suppliers are notified of their commitments, and the Contractor agent invokes its Execution Manager to oversee completion of the plan. Plan maintenance behaviors include re-negotiating existing commitments, re-bidding portions of the plan, re-planning for subgoals that are in jeopardy, and abandoning the goal.

3 The MAGNET Market Infrastructure

The MAGNET system incorporates an explicit market infrastructure. This is a distributed, hierarchical system that manages security, tracks commitments and performance of agents, and logs their interactions. The market specifies the terms of discourse among agents through an *Ontology*, and keeps historical data that may be used by agents in their risk evaluations. Different market segments may specify specialized ontologies. For example, one market segment may cover transportation services, and another construction services. Agents who wish to offer resources and services do so through one or more market segments whose ontologies describe their offerings. Each market includes a set of domain-specific services and facilities, as shown in Figure 1.

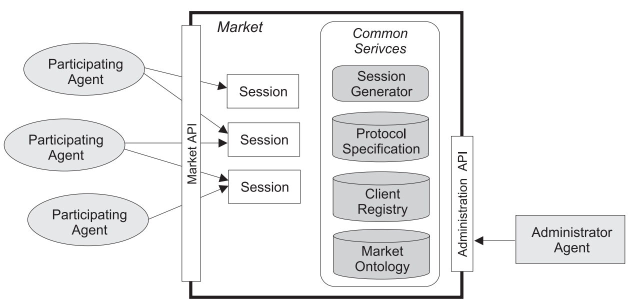

Figure 1: The Structure of a Market within the Exchange. ©1988 by ACM, Inc., appeared in [5]

An *Exchange* is A collection of domain-specific markets, along with generic services required by all markets, such as verifying identities of participants in a transaction, or a Better Business Bureau that can provide information about the reliability of agents based on past performance, is called an *Exchange*. Architecturally, an exchange is a network-accessible resource that supports a set of markets and common services, as depicted in Figure 2.

Within the MAGNET market framework, negotiation activities are encapsulated in a *market session* [5]. 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

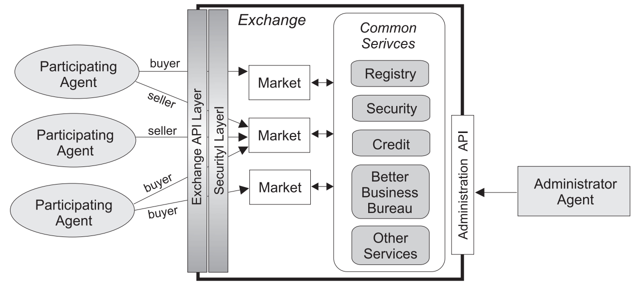

Figure 2: The Structure of an Exchange. ©1988 by ACM, Inc., appeared in [5]

a fuel depot, 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, and the final settlement. 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.

Figure 3 shows the structure of a session. Each session contains an Initiator Proxy that persistently stores the current state of the session from the standpoint of the agent that initiates the session. A Client Proxy is provided for each client that persistently stores the current state of the session from the standpoint of the client. Proxies are market entities that act on behalf of the agents and enforce market rules.

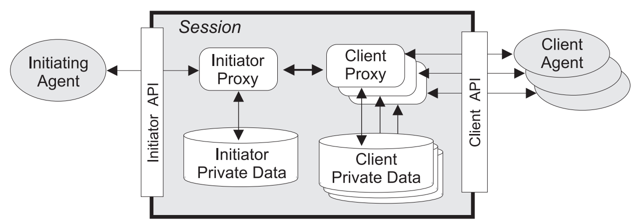

Figure 3: The Structure of a Market Session. ©1988 by ACM, Inc., appeared in [5]

There are two reasons for the existence of the proxy components. The first is related to security: client proxy components cannot see the private data of the initiator or of other clients. The second is that in a distributed system environment, the processing and persistent data elements of the initiator and clients would presumably be at different locations in the network to maximize performance.

4 The MAGNET Negotiation Protocol

The MAGNET negotiation protocol consists of a *contracting* phase and a *execution* phase. The contracting phase is a simple three-step process consisting of a call-for-bids, a Bidding cycle, and an Award cycle. We avoid the need for open-ended negotiation by means of bid break-downs and time-based decommitment penalties. The execution phase may involve negotiations over schedule adjustments, decommitments, and in some cases the bidding cycle might have to be repeated when it becomes necessary to re-allocate resources that had originally been committed.

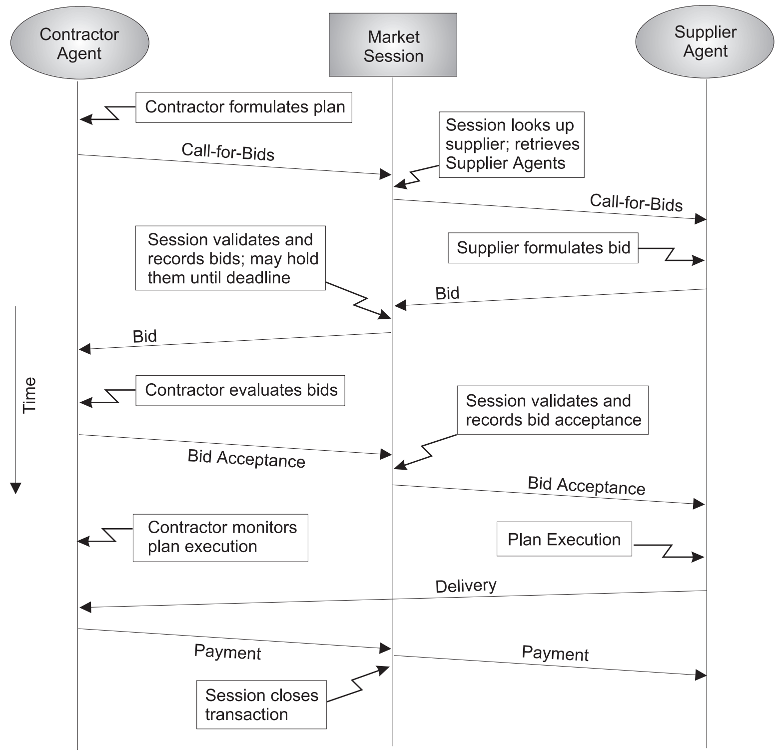

Figure 4: A Typical Session-Mediated Negotiation

The interactions involved in the basic bidding and execution cycle among the contractor, supplier, and market session are illustrated in Figure 4. 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.

5 The Market Ontology

To facilitate the negotiation process and provide a common representation of objects and concepts among agents, a MAGNET market provides a *market ontology* as a unified contract specification system. The roles of the contract specification systems are:

- *Descriptive:* to provide a description of a process that has already occurred;
- *Prescriptive:* to provide a recipe describing how a process can occur;
- *Semantic:* to provide a semantic model, define objects and activities and establish the scope of the system;
- *Communicative:* to enable interoperability between different agents in the marketplace.

The contract specification system is composed of several classes of elements: *Atoms* are the basic units of trade in the system. An atom can represent a physical object, a unit of a resource, or a unit of work done. A market ontology contains a fixed set of atoms which define the terms of discourse in a particular market. There are several main types of atoms, characterized by their behavior in the system, such as abstract, concrete, consumable, capital, whole, fractional, etc.

Actions represent all activities that are possible in the system. An action is defined as $A = \{P, [T], [S]\}$ where P is a set of input and output ports that define the required materials and tools as well as the result of the action, T is an optional set of time constraints, and S is the state.

A port specifies the type of atom that is required to commence the action or the type of atom produced as the result of this action. There are several types of ports that define what happens to an atom that has been passed through a given port.

Consume is an input port that signifies a resource to be used up in process of completing an action. The atom passing through a Consume port will not exist after the action has started. *Produce* is an output port that creates an atom which did not exist before the performance of the action. *Use* represents a resource that is exclusively used for performance of the action, and released once the action has concluded. Two other types of ports, *Specify* and *ToSpec*, allow for dynamic creation of atoms in the system. *Specify* produces an abstract atom specifying the action to occur at some later time, and *ToSpec* performs an action specified by such abstract atom. Within a particular market, actions defined in the ontology represent the production capabilities of actual or potential suppliers in that market.

There are a number of issues that have to be resolved in order for the ontology to be useful in the real world. *Authentication issues* arise when atoms in an electronic commerce system must represent real physical goods. Every physical object is unique in the sense that one cannot make a perfect atom-by-atom copy of it (and even if that was possible, it won't be made out of the same atoms). An electronic

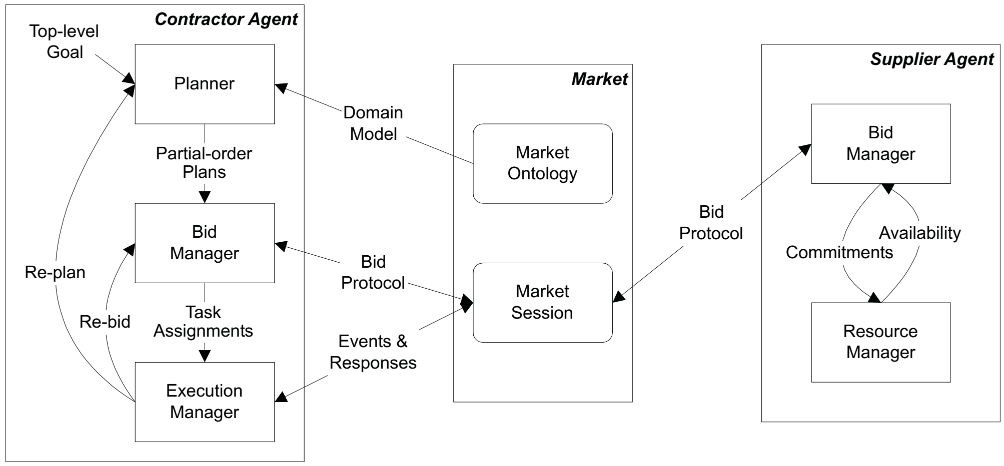

Figure 5: MAGNET Agents

representation of such physical object must also be unique, even though data objects can be freely copied. One way to achieve that uniqueness is to produce physical objects that carry a serial number, and include an encrypted version of the same serial number in its electronic representation.

Money is just one type of a fractional atom. Thus, nothing prevents some market from declaring its universal medium of exchange as something else (shells, colored beads or crude oil). In fact, nothing prevents the economy from developing multiple and parallel mediums of exchange, or from building a barter-based economy. This makes bidding and evaluation of bids a much more complex process, and therefore agents or markets may need to specify the types of "currency" allowed in exchanges.

6 Components of the Contractor Agent

6.1 Planner

A plan is 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 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.

The plan 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 [6] 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 [14], 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.

6.2 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. When the bid manager is invoked, some tasks in the plan may already be assigned. This is because the contract supervisor may use the bid manager to repair a partially-completed plan in which previously determined assignments have failed.

The bid manager must perform these tasks in order to carry out its responsibilities:

Construct and Issue Call For Bids. 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 that are not included in the call-for-bids. This could happen, for instance, if the contracting agent decides to use an advertised resource that is permanently available in the market.

Let $\mathcal{C}.\sqcup_{es}(\int)$ 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 $(\mathcal{A}.\sqcup_{es}(\int))$. 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 the bid manager must specify

- a time window, consisting of an earliest start time $\mathcal{C}.\sqcup_{es}(\int)$ and a latest finish time $\mathcal{C}.\sqcup_{lf}(\int)$,
- a set of precedence relationships $\mathcal{C}.Pred(s) = \{s' \in \mathcal{C} \mid \int' \prec \int\}$ 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.

Receive and Evaluate Bids. When bids are returned, the bid manager has to determine which bids to accept. The bid evaluation process is more complex than just finding the lowest cost solution, since bids include several attributes, such as delivery time, quality of the goods, decommitment costs, that have to be considered in addition to just price, and since bids might include various combinations of goods.

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. The requirement to include prices for individual elements reduces the need for open-ended negotiation, since the customer will know how the supplier will break down the costs of individual tasks.

Note that $(b.price(\mathcal{P}') \leq \sum_{\int \in \mathcal{P}'} \lfloor.price(s)$ 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 $(t_s(s) + b.d_e(s))$.

For each bid, the bid manager has the option of selecting the entire bid and paying an overall discounted price, or selecting a subset of the task bids from the bids, and has to consider coverage, feasibility, cost, and risk. In general, risk factors include decommitment cost, recovery cost, loss of value as the end date is delayed, cost of plan failure, and other factors.

Because of the wide expected variability in problem size, and because of the need to be able to limit the time spent evaluating bids, the evaluator uses an adaptive strategy. This is a form of multi-attribute combinatorial auction [13], for which no efficient algorithms are known. If the problem complexity is low, a systematic search is used. Bids that are incompatible with each other, either due to overlapping coverage or temporal infeasibility, are marked. If task coverage remains possible after this step, then combinations that meet task coverage and temporal feasibility constraints are evaluated for cost and risk. If the combination of plan size, number of bids, and number of tasks per bid is beyond the systematic-search threshold, then a heuristically-guided simulated-annealing search [16] is used to generate bid combinations for evaluation. More details are reported in [4].

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. This completes the negotiation process.

6.3 Contract supervisor

The contract supervisor is responsible for overseeing execution of the plan as contracted, and making decisions on how to respond when events do not proceed as expected. It receives the task assignments from the bid manager and, through the market, 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 $Succ(s) = \{s' \in \mathcal{P} | \int \prec \int'\}$ 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

$$t_{es}(s) = max(\mathcal{A}.\sqcup_{es}(\int), \lfloor_{\int}.\sqcup_{es}(\int))$$

where the notation $\mathcal{A}.\sqcup_{es}(\int)$ refers to the earliest possible start time for task s as computed by the Critical Path algorithm [11]. 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. 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

$$t_{lf}(s) = min(\mathcal{A}.\sqcup_{lf}(\int), \lfloor_{\int}.\sqcup_{lf}(\int))$$

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 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.

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. 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'.

[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.

7 Interactions of the Contractor Agent with the Market Session

In order to carry out its role, the market session maintains a *performance monitoring table*, which 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 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,
- 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 possible response options.

[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 $V_{goal} = 0$.

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.

8 Limitations of Automatic Risk Evaluation Strategies

If MAGNET is to produce plans that are acceptable to a human decision-maker, then MAGNET must consider not only the expected value of a plan of action, but also the risk associated with the plan. At first glance, it may seem intuitive that MAGNET should select the plan that maximizes expected value. Paradoxically, maximizing expected value may lead to plans that few human decision maker's would find acceptable. This observation is hardly new to the field of economics. Early economic ideas by Nicholas Bernoulli showed the weakness of maximizing expected value as a sole criterion for choosing a gamble. Bernoulli constructed a gamble, dubbed the St. Petersburg paradox, that very few people would accept, even though the expected value of the gamble approaches positive infinity. Bernoulli thus argued that people choose a gamble on some basis other than expected value [3].

To produce plans that are more acceptable to a human decision maker, MAGNET will incorporate concepts from expected utility theory. Expected utility theory states that a gamble will be acceptable to a decision maker based, not on the expected value of the gamble, but on the expected utility of the gamble. The expected utility of a gamble is the sum, for each possible outcome, of the probability of the outcome and the utility of the outcome. The utility of an outcome is based on a von Neumann-Morgenstern utility curve, a curve that is a subjective mapping of a level of wealth to a level of utility. Thus the shape of the decision maker's utility curve will guide MAGNET in its choice of a plan that is acceptable to the decision maker. By contrast, the plans produced by an expected value approach implicitly assume that a decision maker derives equal utility from each successive dollar gained or lost. This is the implicit assumption that Bernoulli argued was incorrect [3].

On a more intuitive level, expected utility theory will allow MAGNET to model risk aversion. Risk aversion results when the shape of a decision maker's von Neumann-Morgenstern utility curve is increasing, but concave. This corresponds to a decision maker who always prefers more wealth to less wealth, but who derives less additional utility for each successive increment in wealth. A decision maker with a utility curve of this shape will often reject a fair gamble. Such a decision maker is termed "risk averse". The risk-averse decision maker will reject a fair gamble because the decision maker will derive less utility from the wealth gained than from the wealth that could potentially be lost, even though the amount of wealth gained or lost may be equal [15].

Borrowing concepts from expected utility theory, while resolving some issues, raises others. Is a utility curve something that a decision maker can intuitively understand and input into the MAGNET system? If not, can MAGNET present a series of questions to the decision maker, using the answers to derive a curve? Will MAGNET derive a utility curve by observing and learning from the choices of the decision maker? Or will it be some combination of the above?

In its attempt to weigh plans based on expected utility, MAGNET must overcome yet another obstacle: uncertainty. Thus far, we have assumed that the probabilities of various outcomes are known. MAGNET, however, must work in an environment in which these probabilities are unknown.

Certainly, the fields of economics and statistics have tools to aid with the estimation of unknown probabilities, e.g., entropy, the Central Limit Theorem, and Bayes Theorem. The MAGNET architecture also provides a market mechanism to collect data on suppliers and their past performance. These will certainly act as starting points for estimating probabilities connected with the future performance of individual suppliers.

Still, larger challenges remain. These challenges will tend to drive the MAGNET implementation away from a purely autonomous agent that calculates many-parametered complex probability distributions — and more towards a mixed-initiative system which requires the seasoned guidance of a human decision maker. First, MAGNET may be called upon to operate in markets for which there is little prior data on supplier performance. In the absence of data, MAGNET will need guidance from the decision maker in the form of subjective probabilities. Furthermore, MAGNET is designed to operate in an environment of heterogeneous self-interested agents. These agents may change their behaviors over time (perhaps abruptly) so that past data, particularly old data, may be less relevant or even irrelevant to estimating the future performance of suppliers. Thus the problem becomes less about estimating unknown probabilities and more about estimating probabilities that are largely unknowable. Furthermore, while some market ontologies may be well defined in that all relevant data can be represented in the ontology, in real-life situations a decision maker may need to intervene in MAGNET's decisions, providing the type of subjective input and seasoned judgments that an ontology cannot reasonably represent.

9 Related Work

Markets play an essential role in the economy, by facilitating the exchange of information, goods, and services [1], and there is a growing need for agents capable of more sophisticated automated negotiations [2, 8]. Automated contracting protocols generally assume direct agent-to-agent negotiation [18]. In our work, agents interact with each other through a market.

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 [25, 24]. Our framework provides market mechanisms which support more complex interactions among the agents. Furthermore, the market mechanisms enforce general market rules and "social laws", such as government regulations, by which all participants must abide.

Sandholm's agents [18, 19] redistribute work among themselves by a contracting mechanism. 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.

Various classes of scheduling problems have been considered in the Operations Research literature [11, 12, 7]. Many interesting scheduling problems are computationally intractable, and numerous heuristic approaches have been described [9, 17]. Much of the recent work in scheduling has focused on the problem of maintaining or updating an existing schedule in the face of changes [26, 20]. 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 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 [14] and Tate et al [23] 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.

10 Status of the MAGNET Project

The MAGNET market infrastructure has been implemented as a distributed server written in Java, and significant portions of the Bid Manager portion of a Contractor agent have been implemented and tested [4]. We expect to adapt an existing non-linear planner to handle the planning tasks, and we are studying several options for the market Ontology [22]. The resource management portion of an agent in the Supplier role could be met by existing scheduling methods [20, 23, 14]. Our current research activities are concentrated on the negotiation mechanisms and decision procedures in the Bid Manager and Execution Manager portions of the Contractor role.

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] Tapan Biswas. *Decision-Making Under Uncertainty*. St. Martin's Press, Inc., 1997.

[4] John Collins, Makesim Tsvetovat, Rashmi Sundareswara, Joshua Van Tonder, Maria Gini, and Bamshad Mobasher. Evaluating risk: Flexibility and feasibility in multi-agent contracting. In *Proc. of the Third Int'l Conf. on Autonomous Agents*, May 1999.

[5] 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.

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

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

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

[9] 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.

[10] S. Helper. How much has really changed between us manufacturers and their suppliers. *Sloan Management Review*, 32(4):15–28, 1991.

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

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

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

[14] 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.

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

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

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

[18] 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.

[19] 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.

[20] 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.

[21] J. M. Swaminathan, S.F. Smith, and N.M. Sadeh. Modeling the dynamics of supply chains: A multi-agent approach. *Decision Sciences*, 1997.

[22] Austin Tate. Towards a plan ontology. *Journal of the Italian Artificial Intelligence Association*, January 1996.

[23] 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.

[24] P. R. Wurman, W. E. Walsh, and M. P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24(1):17–27, 1998.

[25] 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.

[26] 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.