Evolving Real-Time Local Agent Control for Large-Scale MAS

Thomas Wagner and Victor Lesser
Computer Science Department
University of Maine and University of Massachusetts
wagner@umcs.maine.edu

ABSTRACT

Control for agents situated in multi-agent systems is a complex problem. This is particularly true in hard, open, dynamic environments where resource, privacy, bandwidth, and computational limitations impose restrictions on the type of information that agents may share and the control problem solving options available to agents. The MQ or *motivational quantities* framework addresses these issues by evaluating candidate tasks based on the agent's organizational context and by framing control as a local agent optimization problem that approximates the global problem through the use of state and preference.

1. INTRODUCTION

Many researchers believe that one of the dominant future models of distributed computation will involve large networks of interacting heterogenous software agents. We, as a community, are showing significant progress in making this a reality but many research questions remain. Consider the requirements and characteristics of the problem space. The overall objective is to create *open*, *large-scale*, information and computational systems that are flexible, adaptable, robust, persistent, and autonomous. Now, consider the implications of this. Openness means that agents may interact freely, come and go from the network, and that the entire problem solving environment is dynamic. Openness thus often acts to thwart agent technologies that rely on detailed predictability or static properties of the problem space. In our work, we take the view that openness leads to a requirement for real-time agent control problem solving so that agents can respond to change and unexpected outcomes online.

Moving the scale of multi-agent systems from small groups to large groups, e.g., tens of thousands, throws two other problems into the mix: *increased interaction overhead* and *social complexity*. The term interaction overhead denotes the increase in communication between agents required to detect interactions in their problem solving and to coordinate their activities, i.e., it denotes the sheer volume of message traffic and problem solving required to evaluate the messages. This is being dealt with by imposing organizational structure on the agents so that they do not all communicate and by creating coordination and communication technologies that are adjustable [6, 10, 38, 11]. The other issue is social complexity and we do not mean social complexity in the human sense. Or rather, the goal of this research is not to study social complexity in human organizations [33] per se as our work in agent control has very specific task-centered properties. When agents are situated in a large open environment, and organizational structure is imposed upon them, they have different organizational objectives and they must reason about how their problem solving relates to satisfying their multiple, and possibly conflicting, organizational objectives.

This research focuses on exactly this problem – how agents in large-scale open environments reason about their organizational context and make appropriate choices about which actions to perform and how to go about performing them. It is important to emphasize that this research pertains to complex problem solving agents, e.g, the BIG Information Gathering Agent [28] and [23, 6, 19], where the agents are situated in an environment, able to sense and effect, and have explicit representations of candidate tasks and explicit representations of different ways to go about performing the tasks. Additionally, tasks are quantified or have different performance characteristics and, following in the thread of complex problem solving [14, 9, 31] there are relationships between the tasks. The implications are that tasks cannot be reasoned about independently and that the value or utility of particular tasks differs depending on the context. We call the process of reasoning about which tasks to perform, when, with what resources, how or in what fashion, and with whom to coordinate, the *local agent control* problem. The term "local" is used in this expression because agency, as we use it, denotes an autonomous distributed problem solving entity. In our work, there is no global picture[1] of all activities being carried out by all agents nor are the agents situated in specialized, tightly coupled environments like Tambe's teams [38] or robotic soccer [40].

[1]In multi-agent systems we take the position that it is not generally possible to compose a global picture of the activities happening at all the agents. This is due to the combinatorics involved in gathering the information and reasoning about it. It is also due to privacy, intellectual property, and autonomy issues. For example, an agent affiliated with Microsoft is unlikely to share its entire knowledge base with a Department of Justice agent even if it were computationally feasible.

We view local agent control in this context as a real-time action-selection-sequencing problem where an agent has n candidate tasks and alternative different ways to perform the tasks. Tasks have deadlines and other constraints as well as different performance properties, e.g., consuming different resources or producing results of varying quality. Control in this context is an optimization problem where different solutions are possible and they have different degrees of utility.

Historically in our work this class of control problem has been dealt with using the TÆMS task modeling framework [7, 27], GPGP coordination [7], and Design-to-Criteria (DTC) real-time agent scheduling [34, 42, 41, 45, 15]. Using these tools, an individual agent for use in a multi-agent environment is constructed by coupling a domain expert or planner with GPGP and DTC. In this model, the domain expert's function is to perform domain problem solving and to translate its internal representations into TÆMS for control problem solving by the coordination (GPGP) and trade-off/scheduling (DTC) experts. GPGP and DTC then work together to guide the actions of the individual agent and to coordinate the activities of the agent with the other agents in the network. This is the approach used in the BIG information gathering agent [29, 28], the Intelligent Home project (IHome) [26], the DARPA ANTS real-time agent sensor network for vehicle tracking [15, 19], and others [47]. Though in some of these applications GPGP is replaced by other communication machinery that forms commitments between agents about which tasks will be performed and when. In all of these applications, DTC or its predecessor, Design-to-Time [17], is the oracle that guides and constrains the communication and commitment formation processes.

TÆMS and DTC are mature research artifacts and have been successfully reused in many applications (DTC since 1995). However, TÆMS is not suited to addressing the situational complexity that arises when agents are deployed in larger groups or in open environments. One of the fundamental limitations of TÆMS is that it is a static representation of an agent's problem solving process at a given instant in time. It is, in essence, a snapshot of the options available to the agent and a snapshot of their characteristics. In our applications, generally, when the situation changes and the characteristics of tasks (used to determine utility) change, the problem solver must adjust the performance profiles and emit a new TÆMS task structure. Another limitation is that in TÆMS, action performance produces *quality* which then propagates throughout the entire graph-like structure in ways that is intended to model distributed problem solving as in a distributed interpretation problem [13]. The formal details of TÆMS are in [8]. While this view is appropriate for reasoning about interrelated domain problem solving activities at a detailed level, it is not readily used to model concepts like tasks that contribute to one organizational objective while being detrimental to another. TÆMS also does not adequately support concepts like the value of forming a commitment with another agent or the penalty for decommitting from an activity once a commitment is formed.

To address these limitations, we have developed a new framework for representing tasks and actions at a different level of abstraction. The framework, called the *motivational quantities* (MQ) [39, 44, 43, 46] framework, uses state to achieve "automatic" changes in task valuation or utility (unlike the static view taken in TÆMS). The MQ framework also describes tasks in many different attribute dimensions so that we can model tasks contributing to, or detracting from, different objectives to different degrees. While control at the TÆMS level pertains to detailed evaluation of domain problem solving activities of an agent, control at the MQ level pertains to high-level valuation of candidate tasks based on an understanding of the relationship between tasks and organizational objectives. In other words, in the MQ framework, task value is determined not only by the intrinsic properties of tasks, but by the benefits and costs of the intrinsic properties as determined by the agent's current organizational situation. From another view, there is an intermediate evaluation step in the control process whereas such processes typically focus on intrinsic value rather than contextually interpreted value. While we have ideas about how to combine and interface [39] the two levels, integration is clearly unnecessary for many applications.

A preliminary version of the MQ framework was presented in [44]. In this paper we refine the framework based on experiences gained by implementing and working with the model.

2. QUANTIFYING AND COMPARING MOTIVATIONS

In the MQ model we make the control restriction that for an agent to perform a task, or to consider a task, the task must produce value for the local agent. On the surface, this implies that the MQ model is only for controlling interacting self-interested agents. This is not the case. The restriction is to guarantee the ability to compare tasks from a unified perspective. Consider the issue of task value. When agents are isolated problem solving entities, task performance produces value that is entirely of local benefit. In multi-agent systems, value may be of local benefit and of benefit to other agents. The extremes are also possible; tasks may be only of local benefit and tasks may be only of benefit to agents other than the local agent. This latter case appears problematic for the control restriction above: *all tasks produce local value.* This case is problematic only on the surface. For the local agent to consider performing such a task, it must indeed have value, however, in this case the value is of a different type or class than the value of its other candidate tasks. The task, for example, may be performed to meet some organizational directive, e.g., *service requests from agent* β, or to reduce favors owed to the agent, to accumulate favors for future use with the agent, or because a different agent with which the local agent holds common goals requested it.

In the MQ framework, all tasks have value or a *motivation* for performing the task where the value is determined both by the value of the task and by the importance of the organizational objective with which the task is associated (and the current state of goal achievement). This enables the agent to compare and value tasks that are associated with different organizational goals, or tasks that are detrimental to one organizational goal while having positive benefit to a different organizational goal, or tasks associated with different organizations entirely, or tasks motivated by self-interested reasons to cooperative reasons. The MQ framework quantifies these different underlying motivational factors and provides the means to compare them via a multi-attributed utility function. Definitions:

Agents are autonomous, heterogenous, persistent, computing entities that have the ability to choose which tasks to perform and when to perform them. Agents are also rationally bounded, resource bounded, and have limited knowledge of other agents.[2] Agents can perform tasks locally if they have sufficient resources and they may interact with other agents. Additionally:

- Each agent has a set of MQs or *motivational quantities* that it tracks and accumulates. MQs represent progress toward organizational goals[3]. MQs are produced and consumed by task performance where the consumption or production properties are dependent on the context. For example, two agents interacting to achieve a shared organizational goal may both see an increase in the same local MQ levels as progress is made (this is not a zero sum game), whereas agents interacting to satisfy different goals may each obtain different types and quantities of MQs from the same interaction.
- Not all agents have the same MQ set. However, for two agents to form a commitment to a specific course of action, they must have at least one MQ in common (or have the means for forming an MQ dynamically). If they do not have an MQ in common, they lack any common goals or objectives and lack any common medium of exchange. (Proxy and reducibility are somewhat addressed in [44].)
- For each MQ_i belonging to an agent, it has a preference function or utility curve, U_{f_i}, that describes its preference for a particular quantity of the MQ, i.e., $\forall MQ_i,\ \exists U_{f_i}()$ such that $U_{f_i}(MQ_i) \mapsto U_i$ where U_i is the utility associated with MQ_i and is not directly interchangeable with U_j unless $i = j$. Different agents may have different preferences for the same MQ_i. Preferences in the framework are defined by the relation between task performance and organizational goals or directives.
- An agent's overall utility at any given moment in time is a function of its different utilities: $U_{agent} = \gamma(U_i,\ U_j,\ U_k,\ ..)$. We make no assumptions about the properties of $\gamma()$, only that it enables agents to determine preference or dominance between two different agent states with respect to MQs.
- For simplicity of presentation, let us assume that $\gamma()$ is not a multi-variate utility function and instead that for each U_i there is an associated function $\omega_i()$ [4] that translates MQ specific utility into the agent's general utility type, i.e., $\forall U_i,\ \exists \omega_i()$ such that $\omega_i(U_i) \mapsto U_{agent}$. Thus U_{agent} may take the form of $U_{agent} = \sum_{i=0}^{n} \omega_i(U_i)$.
- Change in agent utility, denoted ΔU_{agent}, is computed through changes to the individual utilities, U_i, U_j, etc. Let U_i denote the utility associated with MQ_i before the quantity of the MQ changes (e.g., as the result of task performance). Let U_i' denote the utility associated with the changed MQ quantity. The change in overall utility to the agent, in this simplified model, is expressed as $\Delta U_{agent} = |\sum_{i=0}^{n} \omega_i(U_i') - \omega_i(U_i)|$

[2] As agents are heterogenous, they may be associated with different corporate entities (privacy issues), and because the contextual valuation of tasks is generally an exponential problem we do not assume agents know each other's utility functions, plan libraries, etc.

[3] In certain cases, MQs may also be used as a medium of exchange. Though little meaningful work has been done to explore this.

[4] $\omega_i()$ could be combined with $U_{f_i}()$. These are partitioned for mapping different organizational influences.

MQ Tasks are abstractions of the primitive actions that the agent may carry out. MQ tasks:

- May have deadlines, $deadline_i$, for task performance beyond which performance of said task yields no useful results. (This could also be defined via a function that describes a gradual decrease in utility as $deadline_i$ passes.) This temporal constraint, as with the one following, is particularly important for multi-agent coordination and temporal sequencing of activities over interactions.
- May have earliest start times, $start_i$, for task performance before which performance of said task yields no useful results. (This could also be defined via a function that describes a gradual increase in utility as $start_i$ approaches.)
- Each MQ task consists of one or more MQ alternatives, where one alternative corresponds to a different performance profile of the task. In many ways, this extension simplifies reasoning with the preliminary model presented in [44] while at the same time increasing the representational power of the framework by coupling different durations with the other performance characteristics. Each alternative:
  - Requires some time or duration to execute, denoted d_i. The durations for all the alternatives of the task may be the same as the different alternatives may differ in other ways (below). Deadlines and start time constraints remain at the task level – the idea being that tasks have constraints that apply to all of the alternatives.
  - Produces some quantity of one or more MQs, called an *MQ production set* ($MQPS$), which is denoted by: $MQPS_{i,j,k} = \{q_i, q_j, q_k, ..\}$, where $\forall i,\ q_i \geq 0$. These quantities are *positive* and reflect the benefit derived from performing the task, e.g., progress toward a goal or the production of an artifact that can be exchanged with other agents. In this model, the two are equivalent.
  - Akin to the $MQPS$, tasks may also consume quantities of MQs. The specification of the MQs consumed by a task is called an *MQ consumption set* and denoted $MQCS_{i,j,k} = \{q_i, q_j, q_k, ..\}$, where $\forall i,\ q_i \leq 0$. Consumption sets model tasks consuming resources, or being detrimental to an organizational objective, or agents contracting work out to other agents, e.g., paying another agent to produce some desired result or another agent accumulating favors or good will as the result of task performance. Consumption sets are the *negative* side of task performance.
  - All quantities, e.g., d_i, $MQPS$, $MQCS$, are currently viewed from an expected value standpoint.
- Note that the $MQPS_{alternative\_i} \cap MQPS_{alternative\_j}$ may $\neq\ \phi$ as different alternatives may have common

members. This is also true for the $MQCS$. The reason for this is that alternatives may represent producing different degrees of benefit (MQ levels) toward an objective as well as simply producing benefits toward different objectives (different MQs).

- Tasks whose performance at a given point in time, t, will miss their deadlines should not be performed. Likewise with tasks whose performance violates their start time constraint. If such tasks are executed: 1) all $MQCS$ will apply, 2) no $MQPS$ will apply, 3) tasks will take their full duration to execute. Conceptually, this models performing the task, and consuming the task's resources, but having the task fail to produce any benefit.
- In any given alternative, $MQPS$ and $MQCS$ must be disjoint. The reason for this restriction is that in order to reason about an alternative producing and consuming the same MQs, we must have a detailed model of the execution characteristics of the alternative. For example, we must know when it consumes (at the beginning, at the end, linearly across the execution, etc.) and when it produces. This is not consistent with the MQ task abstraction – for situations in which such detailed reasoning is desired, the MQ task must be broken into multiple different tasks.
- $MQCS$ defines quantities that are required for task performance. If a task lacks sufficient MQs for execution it is deemed un-executable and will not be performed in any fashion. This means it will have zero duration, consume zero MQs, and will produce zero MQs.
- If a task will both violate a deadline/start time constraint and lacks sufficient resources to execute, the $MQCS$-lacking semantics will apply. The rationale is that the task lacks the resources to begin execution and thus does not actually violate the temporal constraints.
- Tasks may be interrupted, however, when this occurs they consume all MQs in the $MQCS$ and produce none of the MQs in the $MQPS$. This restriction is to simplify the semantics for the reasoning process.

In this section we have presented a model for comparing tasks that are motivated by different factors. The model can support comparison between tasks that are performed for different organizational motivations to task that are performed for other agents in return for financial gain to tasks that are performed for other agents for cooperative reasons. Via the different preferences for the different quantities, agent control can be modulated and agents can reason about mixtures of different task types and different motivations. The use of state in the model also facilitates contextually dependent behaviors or adjustments to behaviors over time. Agent α performing cooperative work with a closely allied agent, β, for instance, may need to balance this work with cooperative work with others over time. As α accumulates progress toward goals held in common with β (represented as an MQ), its preference may shift to the accumulation of other MQs. The use of utility for this application is flexible and very general and there are many different ways to relate organizational goal importance to the process of task valuation.

The model relates to other recent work in the multi-agent community, such as agents interacting via obligations [1], or notions of social commitment [4], but it differs in its quantification of different concerns and its dynamic, contextual, relative, evaluation of these. The model resembles MarCon [32] as the different degrees-of-satisfaction afforded by the MQ model is akin to MarCon's constraint optimization approach, and MarCon too deals with utilities/motivations that cannot always be commingled. MarCon, however, views constraints as agents, assigning particular roles to particular agents, and the issue of which tasks to perform do not enter into the problem space.

In the sections that follow we discuss scheduling MQ tasks and present examples of using the framework for agent control.

3. SCHEDULING AND ANALYSIS

If the agent's objective is to simply select which task to perform next, and tasks do not have associated deadlines or earliest start times, and the present and future value of MQs are equivalent, then it can reason using the maximum expected utility principle and select the task at each point that maximizes immediate utility. However, this simple choose-between-available-tasks model does not map well to situations in which tasks have individual earliest start times and/or deadlines. Note that in general, to coordinate the activities of multiple agents, temporal constraints such as these are needed to sequence activities over inter-agent interactions. In situations with such temporal interactions, it is difficult to produce optimal or even "good" results without considering sequences of activities and thus for most applications, scheduling of MQ tasks is required.

The implemented MQ task scheduler employs a generative state space search process where states record the agent's current MQ levels, utility functions, organizational roles and objectives, completed task set, candidate task set, and some estimation of the agent's future candidate task set. Transitions correspond to the performance of an MQ task. During the duration required for task performance (a transition), new tasks may arrive or the agent's organizational situation may change thus transitions can also be viewed as marking these changes as well. The MQ model was designed specifically to lend itself to this form of representation to ease reproducibility and to leverage the large body of research pertaining to state-based search. For example, the MQ model can be scheduled using standard real-time search technologies like SMA* [35], RTA* [25], and others [21], enabling the model to address the real-time requirement of online control for agents in open environments. This is particularly important because the search space is exponential in the number of actions being scheduled. Implementationally, the scheduler is unable to schedule problem instances of nine or more activities using exhaustive search[5] and other techniques are required. Depending on the characteristics of the problem instance, standard A* may produce results in a reasonable amount of time on a larger problem instance, though, approximate rather than admissible heuristics are sometimes required to avoid exhaustive generation by A*. Scheduling issues beyond the scope of this paper include approaches for constructing good search heuristics in the face

[5]On Pentium III class machine with 256 megabytes of RAM running Redhat Linux 6.0 and IBM's 1.1.6 jdk.

of non-linear utility curves and considering the future value of MQs when estimating state potentials.

Opportunity cost also factors-into the scheduling process and into the estimation of the potential value of a given state. There are many different uses of opportunity cost in control and many different ways to compute or predict the future value of a unit of time. In the examples presented in this paper, the scheduler is configured so that opportunity cost is used to determine the value of a given activity relative to the time it requires to perform and it is computed using a running average of the amount of utility produced by the agent per time unit. Opportunity cost is tracked and expressed as a pair $< OCM, Window >$ where: 1) the OCM or *opportunity cost metric* expresses the current value of a unit of the agent's time, i.e., a utility-per-time-unit factor, and 2) the $Window$ denotes the time over which the OCM was produced. The $Window$ is necessary as time moves forward so the agent can reweight and adjust the OCM based on recent changes. As defined, opportunity cost is computed from the agent's initialization forward and will gradually be prone to little movement as weight ($Window$) is accumulated. There are obviously many different variations for the running average computation. Once the value of a unit of time is computed, the issue then becomes how to employ it when considering a given MQ task or estimating the potential of MQ tasks for use in search heuristics. In this paper the opportunity cost of an MQ task will have the same weight as the utility produced by the task. If t_i is a task being evaluated: $adjusted\_utility_{t_i} = utility_{t_i} - opportunity\_cost_{t_i}$. Note that initially the agent's OCM is 0 and its $Window$ is also 0, thus, in many cases the pair is "seeded" with initial estimations of appropriate values.

4. DEMONSTRATING CONTROL VIA MQS

In this section we demonstrate the use of the MQ model and the MQ task scheduler in an organizational control problem. The agent's objective in this example is to maximize its total utility over the set of candidate tasks. The agent in question is an *Information Gathering* agent for the Merrill Lynch corporation, IG_{ML}, and is situated in a network of financial information agents. The agent network is patterned after the the WARREN [6] style and the space is populated by three types of agents 1) *Database Manager* agents, 2) *Information Gathering* (IG) agents that are experts in particular domains and whose task is to plan, gather information, assimilate it, and produce a report, possibly accompanied by a recommendation to the client about a particular action to take based on the gathered information. 3) *Personal Agents* (PA) that interface directly with the human client, perhaps modeling the client's needs. These agents also decide with which information specialists to interact to solve a client's information need. In this paper, we focus on the interactions between IG_{ML} and personal agents for Merrill Lynch, PA_{ML1} and PA_{ML2}, that are associated with different Merrill Lynch employees. PA_{ML1} represents a mutual fund manager and PA_{ML2} represents an individual broker, thus, their requests must be evaluated differently by IG_{ML}.

In this scenario, IG_{ML} has the organizational roles and objectives shown in Figure 1. To track contributions toward each of its organizational roles, and thus to monitor the state of its requirement to perform a certain amount of maintenance prior to servicing data requests, IG_{ML} will use two different MQs: $MQ_{service}$ and $MQ_{maintenance}$. All tasks will produce some quantity of each of these MQs in addition to producing an MQ related to the task itself. IG_{ML} monitors and tracks the following MQs: MQ_{update}, $MQ_{explore}$, MQ_{PAML1}, MQ_{PAML2}, $MQ_{service}$, and $MQ_{maintenance}$. In accordance with the organizational objectives, the curve used for $MQ_{PA_{ML1}}$is $U = 2x + 2$ and for $MQ_{PA_{ML2}}$is $U = x + 1$. MQ_{update} and $MQ_{explore}$ are both assigned a curve of $U = 2x + 2$, though exploration tasks produce more units of $MQ_{maintenance}$ per the objectives. We model the maintenance versus service objective by using a curve with two segments for $MQ_{maintenance}$: $U = 2 * x \ when \ x <= 5$ and $U = x/2 \ when \ x > 5$ (the specified quantity of maintenance MQs to produce before servicing requests is 5 in this example). For $MQ_{service}$, we use a constant curve of $U = x/2$. In all cases, $\omega(U_i) = 1$. In this scenario, tasks do not produce unit quantities of MQs but instead produce different volumes of MQs.

```
Task: Update1
    Alternative: Update1.0
      Duration: 3.0
      MQPS u MQCS: {(mq_maintenance 2.5),
                    (mq_update 1.5)}
      MQPS: {(mq_maintenance 2.5),
             (mq_update 1.5)}
      MQCS: {}

Task: Explore1
    Alternative: Explore1.0
      Duration: 10.0
      MQPS u MQCS: {(mq_maintenance 5.0),
                    (mq_explore 2.0)}
      MQPS: {(mq_maintenance 5.0),
             (mq_explore 2.0)}
      MQCS: {}

Task: Request1_PAML1
    Alternative: Request1_PAML1.0
      Duration: 4.5
      MQPS u MQCS: {(mq_service 1.0),
                    (mq_paml1 4.0)}
      MQPS: {(mq_service 1.0),
             (mq_paml1 4.0)}
      MQCS: {}

Task: Request1_PAML2
    Alternative: Request1_PAML2.0
      Duration: 2.0
      MQPS u MQCS: {(mq_service 1.0),
                    (mq_paml2 5.0)}
      MQPS: {(mq_service 1.0),
             (mq_paml2 5.0)}
      MQCS: {}
```

Figure 2: Subset of Service Requests and Maintenance Tasks

We will explore multiple variations using this model. First, we demonstrate appropriate agent control behavior given the organizational objectives. Consider the subset of IG_{ML}'s tasks and requests pictured in Figure 2. The tasks in the figure represent one half of the tasks assigned to the agent; there are actually two identical tasks of each task instance assigned to IG_{ML}, i.e., two requests from PA_{ML1}, two requests from PA_{ML2}, two exploration maintenance tasks and two updating maintenance tasks. Note that each of the tasks

Organizational Membership IG_{ML} belongs to a single organization (Merrill Lynch).

Roles IG_{ML} has two different organizational roles and all tasks performed by IG_{ML} pertain to one role or the other:

1. IG_{ML} has a *service role* that requires servicing requests and queries from other agents seeking information. In this example, IG_{ML} will service requests from PA_{ML1} and PA_{ML2}.
2. IG_{ML} also has a *maintenance role* that entails keeping its data repository up to date so that it may effectively service requests. In this role, IG_{ML} performs update tasks and exploration tasks. Update tasks entail verifying the integrity of its database by confirming that it has valid and current information on the known database management agents. Exploration tasks entail seeking out new database management agents and building profiles of such sources for inclusion into its database.

Organizational Goals Produce profit by servicing requests. Maintain quality of existing repository to ensure long-term revenue stability. Grow repository to improve coverage and to keep pace with the growth of networked information sources. Repository growth is considered particularly valuable at this time.

Organizational Objectives The organizational goals translate into several organizational objectives for IG_{ML}.

1. During any period (we will explore one period), IG_{ML} is to give preference to maintenance tasks until it has performed a specified amount and then it is to balance request service with maintenance on the basis of returns.
2. Requests from PA_{ML1} are *preferred* to PA_{ML2} as they have a higher potential to produce more revenues for the company. IG_{ML} is thus advised to regard one unit of MQ from PA_{ML1} as having approximately twice the value of one unit of $MQ_{PA_{ML2}}$. (In this scenario there is no strict power relationship between the agents.)
3. Exploration tasks contribute more to the maintenance requirement than update tasks as it is perceived that growth is currently necessary. However, exploration tasks require more time to perform.

Figure 1: IG_{ML}'s Organizational Context

Attributes & State	Task or Request Scheduled: Init	Explore2	Request2 PAML1	Request1 PAML1	Explore1	Request2 PAML2	Request1 PAML2	Update2	Update1
# in Sequence	n/a	1	2	3	4	5	6	7	8
Start Time	n/a	0	10	14.5	19	29	31	33	36
End Time	n/a	10	14.5	19	29	31	33	36	39
Utility After	7	21	29.5	38	44.5	50	55.5	59.75	64
Level MQ_{PAML1}	0	0	4	8	8	8	8	8	8
Level MQ_{PAML2}	0	0	0	0	0	5	10	10	10
Level MQ_{update}	0	0	0	0	0	0	0	1.5	3
Level $MQ_{explore}$	0	2	2	2	4	4	4	4	4
Level $MQ_{service}$	0	0	1	2	2	3	4	4	4
Level $MQ_{maintenance}$	0	5	5	5	10	10	10	12.5	15

Figure 3: Optimal Schedule and Control State Changes for IG_{ML} That Balances Maintenance and Service

produce an MQ unique to the task type as well as an MQ relating to its broader organizational categorization (service or maintenance). The optimal schedule for IG_{ML} and the associated changes to the agent's state after each task is performed is shown in Figure 3 (the alternative identifier is omitted because each task has a single alternative).

In accordance with the specified objectives, the agent first elects to perform one of the exploration tasks. This produces the required five units of $MQ_{maintenance}$ as well as two units of $MQ_{explore}$. Because the utility curve for $MQ_{maintenance}$ changes after five units are produced, the agent then elects to service requests for PA_{ML1} rather than performing the remaining exploration task. Subsequent effort returns to the remaining exploration maintenance task, then the service of requests for PA_{ML2}, and finally the update tasks are performed. After the initial exploration task, and the change in $U_{maintenance}$, the choice between the exploration task, the two update tasks, and the requests for $PA_{ML1}$is determined by the quantity of MQs produced by each as $U_{maintenance}$ $==$ $U_{service}$ and $U_{explore}$ $==$ U_{PAML1}. The requests for PA_{ML2} are competitive with the update tasks for this same reason – though U_{PAML2} is dominated by U_{update} the requests for PA_{ML2} produce larger quantities of MQs than the update tasks.

Consider what happens if the organizational objective is changed and the maintenance objective is relaxed. In this case, the curve for the maintenance MQs is the same as that used for the service MQs, namely $U = x/2$ at all times. The optimal schedule and the state changes for the agent are shown in Figure 4. In this situation, the utility produced by requests for PA_{ML1} outweighs the benefits of the exploration tasks and they are performed first, rather than second. The exploration tasks are performed second, and the requests for PA_{ML2} follow, and finally the update tasks are performed. When the organizational objective is removed, the resulting task / utility curve set produces a non-interleaved ordering for the tasks (as each task of each type has the same MQ levels). This underscores the role of the two-segment utility curve modeling technique of the previous example in mapping the organizational objective into utility for the first 5 units of $MQ_{maintenance}$.

Consider a different scenario. Figure 5 shows the optimal schedule produced if we reinstate the organizational objective to produce 5 units of $MQ_{maintenance}$ before servicing requests, and, if we instruct the agent to factor-in the opportunity cost of the associated tasks. The agent is given an initial opportunity cost pair of $< OCM = 1.0, Window = 100.0 >$ In this case, the agent elects to satisfy the maintenance requirement by performing both of the update tasks, each of which produces 2.5 units of $MQ_{maintenance}$, rather than performing a single exploration task (5 units of $MQ_{maintenance}$ are produced by a single exploration task). This is because the exploration tasks require ten time units to execute and the update tasks require only three time units to execute. The utility state after each task without considering the opportunity cost of the task is given by *Utility After* whereas the utility adjusted to reflect opportunity cost is indicated by *Utility$_{oc}$ After*. Note that the exploration tasks are the least appealing options to the agent and are scheduled last in this scenario. Note also that the agent's *Utility$_{oc}$* actually decreases when the exploration tasks are performed. This is because the tasks' opportunity costs are greater than the utility they produce. Depending on how the agent is configured, it may elect to wait-and-see while adjusting its opportunity cost OCM and $Window$ (as time moves forward) rather than performing the exploration tasks. In this case, the agent is configured to keep performing requests regardless. The first exploration task causes a net change in utility of -6.6 while the second exploration task incurs a lesser change of -5.2. This is because after performing the first exploration action, the agent's OCM is lowered by the negative utility produced by the task so that the opportunity cost of the second exploration task is less.

5. CONCLUSION, LIMITATIONS AND FUTURE WORK

We have presented and extended the MQ model for local agent control of organized agents and shown its use in different applications. The strength of the model is its use of state to obtain appropriate local control – the model does not require common social-level control assumptions like shared or visible utility functions, which are useful in applications where shared and static knowledge are possible or as a theoretical foundation, e.g., [18].

Inherent in the model's state-based design are the assumptions that: 1) agents have imperfect knowledge of the problem solving taking place at other agents; 2) the utility function of a given agent cannot generally be shared and computed by other agents because it is dependent on the agent's problem solving state; 3) globally optimal behavior can be approximated through local reasoning. In this latter case, the precision of the approximation is dependent on the degree to which agents can *communicate* or *observe* progress toward organizational objectives. Consider IG_{ML}'s maintenance requirement from the previous section. If the maintenance requirement could be met by another IG agent of Merrill Lynch, e.g., IG_{ML_B}, and IG_{ML_B} decided to perform the required maintenance operations, the $MQ_{maintenance}$ requirement of both IG_{ML} and IG_{ML_B} would be met by IG_{ML_B}'s operation. However, IG_{ML}'s recognition of this depends on communication between the agents, observation, default reasoning, plan inference, or a similar mechanism. The communications required are beyond the scope of the MQ framework though the framework is designed explicitly to support such activities. Using the MQ model, notions of social utility are decomposed and distiled into the control regime of local agents – a feature we believe is important for application in large-scale MAS.

Intellectually, one of the contributions of the model is the attempt to address complexity in MAS – complexity that is even more important as we move to large-scale MAS and persistent agents. Agents in complex environments, having multiple organizational objectives and different relationships with other agents, require a certain level of complexity in their objective functions and in their action and situation models.

Another important characteristic of the framework is its support of local approximation of the global optimization problem of a large group of interacting agents. Aspects of this include the idea that different activities contribute to different aspects of the global objectives, the need to consider of history or state in decision making, and that in certain situations there are interactions between the local utility computations of different agents.

To summarize the model's positive attributes: 1) it en-

Attributes & State	Task or Request Scheduled Init	Request2 PAML1	Request1 PAML1	Explore2	Explore1	Request2 PAML2	Request 1 PAML2	Update2	Update1
# in Sequence	n/a	1	2	3	4	5	6	7	8
Start Time	n/a	0	4.5	9	19	29	31	33	36
End Time	n/a	4.5	9	19	29	31	33	36	39
Utility After	7	15.5	24	30.5	37	42	48	52.25	56.5
Level MQ_{PAML1}	0	4	8	8	8	8	8	8	8
Level MQ_{PAML2}	0	0	0	0	0	5	10	10	10
Level MQ_{update}	0	0	0	0	0	0	0	1.5	3
Level $MQ_{explore}$	0	0	0	2	4	4	4	4	4
Level $MQ_{service}$	0	1	2	2	2	3	4	4	4
Level $MQ_{maintenance}$	0	0	0	5	10	10	10	12.5	15

Figure 4: Optimal Schedule and Control State Changes for IG_{ML} When Maintenance Objective is Relaxed

Attributes & State	Task or Request Scheduled Init	Update2	Update1	Request2 PAML1	Request1 PAML1	Request2 PAML2	Request1 PAML2	Explore2	Explore 1
# in Sequence	n/a	1	2	3	4	5	6	7	8
Start Time	n/a	0	3	6	10.5	15	17	19	29
End Time	n/a	3	6	10.5	15	17	19	29	39
Utility After	7	15	23	31.5	40	45.5	51	57.5	64
Utility_{oc} After	7	12	˜ 16.8	˜ 20.4	˜ 23.8	˜ 27	˜ **31.2**	˜ **24.6**	˜ **19.4**
Level MQ_{PAML1}	0	0	0	4	8	8	8	8	8
Level MQ_{PAML2}	0	0	0	0	0	5	10	10	10
Level MQ_{update}	0	1.5	3	3	3	3	3	3	3
Level $MQ_{explore}$	0	0	0	0	0	0	0	2	4
Level $MQ_{service}$	0	0	0	1	2	3	4	4	4
Level $MQ_{maintenance}$	0	2.5	5	5	5	5	5	10	15

Figure 5: Optimal Schedule and Control State Changes for IG_{ML} When Opportunity Cost is Considered

ables organizationally appropriate behavior through local agent reasoning, 2) it is amenable to real-time control problem solving and this problem solving is fairly straightforward to reproduce as a state-based search, 3) the model is well suited to adjustable degrees of approximation – it can be optimal in a non-local sense when complete information is available and it can be very coarse when agent's have little information about the activities of other agents, 4) the model provides a unified evaluation framework for agent activities, 5) it represents aspects of the complexity inherent in large MAS.

In terms of problems and limitations, one attribute of the model that has not been explored in any meaningful fashion is use of MQs as a medium of exchange. It appears that such exchanges are synonymous with an agent performing a task that is of benefit to one of its organizational objectives while detrimental to another, however, further research in this area is required.

The most significant criticism of the model is that the translation of organizational structure into MQs, utility curves, initial MQ assignments, etc., as presented in this paper is ad-hoc. Though it required little knowledge engineering to produce the desired control behavior, experiments with a randomized assignment of MQ quantities to tasks illustrate a potential problem. Without design principles to guide the mapping, or appropriate corresponding verification techniques, it is difficult to be confident that a given mapping will result in the desired control behavior in the local agents. Ideally, the mapping of organizational objectives to organizational roles, the assignment of roles to agents, and the decomposition into MQs should be automated or performed by an organizational design component. Given the complexity of the problem, design principles and an approach for verification are the natural next step.

6. REFERENCES

[1] Mihai Barbuceanu. Agents that work in harmony by knowing and fulfiling their obligations. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 89–96, 1998.

[2] Sviatoslav Brainov. The role and the impact of preferences on multiagent interaction. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI*, Lecture Notes in AI. Springer-Verlag, Berlin, 2000.

[3] K.M. Carley and M.J. Prietula, editors. *Computational Organization Theory*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

[4] Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS95)*, pages 41–48, 1995.

[5] Phillip R. Cohen, Adam Cheyer, Michelle Wang, and Soon Cheol Baeg. An open agent architecture. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 197–204. Morgan Kaufmann, 1998.

[6] K. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Proceedings of the 1st Intl. Conf. on Autonomous Agents*, pages 404–413, Marina del Rey, February 1997.

[7] Keith Decker and Jinjiang Li. Coordinated hospital patient scheduling. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, pages 104–111, 1998.

[8] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms.* PhD thesis, University of Massachusetts, 1995.

[9] Keith S. Decker, Edmund H. Durfee, and Victor R. Lesser. Evaluating research in cooperative distributed problem solving. In L. Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence, Vol. II*, pages 485–519. Pitman Publishing Ltd., 1989. Also COINS Technical Report 88-89, University of Massachusetts, 1988.

[10] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the Thirteenth International Workshop on Distributed AI*, pages 65–84, Seattle, WA, July 1994. AAAI Press Technical Report WS-94-02. Also UMass CS-TR-94-14. To appear, Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, AAAI Press, 1995.

[11] C. Dellarocas and M. Klein. An experimental evaluation of domain-independent fault handling services in open multi-agent systems. In *Proceedings of the Fifth International Conference on Multi-Agent Systems (ICMAS2000)*, 2000.

[12] Robert Doorenbos, Oren Etzioni, and Daniel Weld. A scalable comparision-shopping agent for the world-wide-web. In *Proceedings of the First International Conference on Autonomous Agents*, pages 39–48, Marina del Rey, California, February 1997.

[13] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, August 1987.

[14] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11):1275–1291, November 1987.

[15] Regis Vincent et al. Distributed Sensor Network for Real Time Tracking. In *Proceedings of Autonomous Agents (Agents-2001)*, 2001. To appear.

[16] P. Faratin, C. Sierra, and N. Jennings. Negotiation Decision Functions for Autonomous Agents. *International Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1997.

[17] Alan J. Garvey. *Design-to-Time Real-Time Scheduling.* PhD thesis, University of Massachusetts at Amherst, Amherst, Massachusetts, February 1996.

[18] Lisa Hogg and Nick Jennings. Variable sociability in agent-based decision making. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI*, Lecture Notes in AI. Springer-Verlag, Berlin, 2000.

[19] Bryan Horling, Regis Vincent, Roger Mailler, Jiaying Shen, Raphen Becker, Kyle Rawlins, and Victor Lesser. Distributed sensor network for real-time tracking. In *Proceedings of Autonomous Agent 2001*, 2001. To appear.

[20] Michael N. Huhns and Munindar P. Singh. Agents and multiagent systems: Themes, approaches, and challenges. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 1–23. Morgan Kaufmann, 1998.

[21] Toru Ishida. Real-time search for autonomous agents and multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 1(2):139–167, October 1998.

[22] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):8–38, 1998.

[23] N.R. Jennings, J.M.Corera, L. Laresgoiti, E.H. Mamdani, F. Perriollat, P. Skarek, and L.Z. Varga. Using ARCHON to develop real-world dai applications for electricity transportation management and particle accelerator control. *IEEE Expert*, 1995. Special issue on real world applications of DAI systems.

[24] Henry Kautz, Bart Selman, Michael Coeh, Steven Ketchpel, and Chris Ramming. An experiment in the design of software agents. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 125–130. Morgan Kaufmann, 1998.

[25] Richard E. Korf. Depth-limited search for real-time problem solving. *The Journal of Real-Time Systems*, 2(1/2):7–24, 1990.

[26] Victor Lesser, Michael Atighetchi, Bryan Horling, Brett Benyo, Anita Raja, Regis Vincent, Thomas Wagner, Ping Xuan, and Shelley XQ. Zhang. A Multi-Agent System for Intelligent Environment Control. In *Proceedings of the Third International Conference on Autonomous Agents (Agents99)*, 1999.

[27] Victor Lesser, Bryan Horling, and et al. The TÆMS whitepaper / evolving specification. http://mas.cs.umass.edu/research/taems/white.

[28] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. BIG: An agent for resource-bounded information gathering and decision making. *Artificial Intelligence*, 118(1-2):197–244, May 2000. Elsevier Science Publishing.

[29] Victor Lesser, Bryan Horling, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. Sophisticated Information Gathering in a Marketplace of Information Providers. *IEEE Internet Computing*, 4(2):49–58, Mar/Apr 2000.

[30] Victor R. Lesser. Reflections on the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems*, 1(1):89–111, 1998.

[31] Victor R. Lesser and Daniel D. Corkill. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):81–96, January 1981.

[32] H. Van Dyke Parunak, Allen Ward, and John Sauter. A Systematic Market Approach to Distributed Constraint Problems. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, 1998.

[33] Michael J. Prietula, Kathleen M. Carley, and Les Gasser. A Computational Approach to Oganizations and Organizing. In Michael J. Prietula, Kathleen M. Carley, and Les Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*, pages xiv–xix. AAAI Press / MIT Press, 1998.

[34] Anita Raja, Victor Lesser, and Thomas Wagner. Toward Robust Agent Control in Open Environments.

In *Proceedings of the Fourth International Conference on Autonomous Agents (Agents2000)*, 2000.

[35] S.J. Russell. Efficient memory-bounded search methods. In *ECAI 92: 10th European Conference on Artifical Intelligence*, pages 1–5, 1992.

[36] Paul A. Samuelson and William D. Nordhaus. *Economics*. McGraw-Hill Book Company, 1989. 13th Edition.

[37] Sandip Sen and Anish Biswas. Effects of misconception on reciprocative agents. In *Proceedings of the Second International Conference on Autonomous Agents (Agents98)*, pages 430–435, 1998.

[38] Milind Tambe. Agent Architectures for Flexible, Practical Teamwork. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 22–28, July 1997.

[39] Thomas A. Wagner. *Toward Quantified Control for Organizationally Situated Agents*. PhD thesis, University of Massachusetts at Amherst, Amherst, Massachusetts, February 2000.

[40] Manuela Veloso, Peter Stone, and Kwun Han. The CMUnited-97 robotic soccer team: Perception and multiagent control. In *Proceedings of the Second International Conference on Autonomous Agents (Agents98)*, pages 78–85, 1998.

[41] Thomas Wagner, Brett Benyo, Victor Lesser, and Ping Xuan. Investigating Interactions Between Agent Conversations and Agent Control Components. In Frank Dignum and Mark Greaves, editors, *Issues in Agent Communication*, Lecture Notes in Artificial Intelligence, pages 314–331. Springer-Verlag, Berlin, 2000.

[42] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, 19(1-2):91–118, 1998. A version also available as UMASS CS TR-97-59.

[43] Thomas Wagner and Victor Lesser. Motivational Quantities: State-based Control for Organizationally Situated Agents. Computer Science Technical Report TR-99-68, University of Massachusetts at Amherst, November 1999. Research abstract appears in the proceedings of the International Conference on Multi-Agent Systems (ICMAS) 2000.

[44] Thomas Wagner and Victor Lesser. Relating quantified motivations for organizationally situated agents. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI (Proceedings of ATAL-99)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2000.

[45] Thomas Wagner and Victor Lesser. Design-to-Criteria Scheduling: Real-Time Agent Control. In Thomas Wagner and Omer Rana, editors, *To appear in Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, LNCS. Springer-Verlag, 2001. Also appears in the 2000 AAAI Spring Symposium on Real-Time Systems and a version is available as University of Massachusetts Computer Science Technical Report TR-99-58.

[46] Thomas Wagner and Victor Lesser. Organizational level control for real-time agents. In *To Appear in the Proceedings of Autonomous Agents (Agents-2001)*, 2001.

[47] Thomas Wagner, John Phelps, Yuhui Qian, Erik Albert, and Glen Beane. A modified architecture for constructing real-time information gathering agents. In *Proceedings of Agent Oriented Information Systems*, 2001. To appear.

[48] M.P. Wellmen, E.H. Durfee, and W.P. Birmingham. The digital library as community of information agents. *IEEE Expert*, June 1996.

[49] Mary Zey. *Rational Choice Theory and Organizational Theory: A Critique*. Sage Publications, Thousand Oaks, CA 91320, 1998.