Examples of Quantitative Modeling of Complex Computational Task Environments *

Keith Decker and Victor Lesser
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
Email: DECKER@CS.UMASS.EDU

Abstract

There are many formal approaches to specifying how the mental state of an agent entails that it perform particular actions. These approaches put the agent at the center of analysis. For some questions and purposes, it is more realistic and convenient for the center of analysis to be the task environment, domain, or society of which agents will be a part. This paper presents such a task environment-oriented modeling framework that can work hand-in-hand with more agent-centered approaches. Our approach features careful attention to the quantitative computational interrelationships between tasks, to what information is available (and when) to update an agent's mental state, and to the general structure of the task environment rather than single-instance examples. This framework avoids the methodological problems of relying solely on single-instance examples, and provides concrete, meaningful characterizations with which to state general theories. Task environment models built in our framework can be used for both analysis and simulation to answer questions about how agents should be organized, or the effect of various coordination algorithms on agent behavior. This paper is organized around an example model of cooperative problem solving in a distributed sensor network.

This workshop submission is a shortened version of a much longer technical report available from the authors; see also our paper in the main conference proceedings [Decker and Lesser, 1993b].

Introduction

This paper presents an overview of a framework, TÆMS (Task Analysis, Environment Modeling, and Simulation), with which to model complex computational task environments that is compatible with both formal agent-centered approaches [Cohen and Levesque, 1990; Shoham, 1991; Zlotkin and Rosenschein, 1990] and experimental approaches [Carver and Lesser, 1991; Cohen *et al.*, 1989; Durfee *et al.*, 1987; Pollack and Ringuette, 1990]. The framework allows us to both analyze and quantitatively simulate the behavior of single or multi-agent systems with respect to interesting characteristics of the computational task environments of which they are part. We believe that it provides the correct level of abstraction for meaningfully evaluating centralized, parallel, and distributed control algorithms, negotiation strategies, and organizational designs. No previous characterization formally captures the range of features, processes, and especially interrelationships that occur in computationally intensive task environments.

We use the term *computational task environment* to refer to a problem domain in which the primary resources to be scheduled or planned for are the computational processing resources of an agent or agents, as opposed to physical resources such as materials, machines, or men. Examples of such environments are distributed sensor networks, distributed design problems, complex distributed simulations, and the control processes for almost any distributed or parallel AI application. A job-shop scheduling application *is not* a computational task environment. However, the control[1] of multiple distributed or large-grain parallel processors that are jointly responsible for solving a job shop scheduling problem *is* a computational task environment. Distributed sensor networks use resources (such as sensors), but these resources are typically not the primary scheduling consideration. Computational task environments are the problem domain for control algorithms like many real-time and parallel local scheduling algorithms [Boddy and Dean, 1989; Garvey and Lesser, 1993; Russell and Zilberstein, 1991] and distributed coordination algorithms [Decker and Lesser, 1991; Durfee *et al.*, 1987].

The *reason* we have created the TÆMS framework is rooted in the desire to produce general theories in AI [Cohen, 1991]. Consider the difficulties facing an experimenter asking under what environmental conditions a particular local scheduler produces acceptable results, or when the overhead associated with a certain coordination algorithm or organizational structure is acceptable given the frequency of particular environmental conditions. At the very least, our framework provides a characterization of environmental features and a concrete,

*This work was supported by DARPA contract N00014-92-J-1698, Office of Naval Research contract N00014-92-J-1450, and NSF contract CDA 8922572. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

[1] Organization, planning, and/or scheduling of computation.

meaningful language with which to state correlations, causal explanations, and other forms of theories. The careful specification of the computational task environment also allows the use of very strong analytic or experimental methodologies, including paired-response studies, ablation experiments, and parameter optimization. TÆMS exists as both a language for stating general hypotheses or theories and as a system for simulation. The simulator supports the graphical display of generated subjective and objective task structures, agent actions, and statistical data collection in CLOS on the TI Explorer.

The basic form of the computational task environment framework—the execution of interrelated computational tasks—is taken from several domain environment simulators [Carver and Lesser, 1991; Cohen *et al.*, 1989; Durfee *et al.*, 1987]. If this were the only impetus, the result might have been a simulator like Tileworld [Pollack and Ringuette, 1990]. However, formal research into multi-agent problem solving has been productive in specifying formal properties, and sometimes algorithms, for the control process by which the mental state of agents (termed variously: beliefs, desires, goals, intentions, etc.) causes the agents to perform particular actions [Cohen and Levesque, 1990; Shoham, 1991; Zlotkin and Rosenschein, 1990]. This research has helped to circumscribe the behaviors or actions that agents can produce based on their knowledge or beliefs. The final influence on TÆMS was the desire to avoid the individualistic agent-centered approaches that characterize most AI (which may be fine) and DAI (which may not be so fine). The concept of agency in TÆMS is based on simple notions of *execution*, *communication*, and *information gathering*. An agent is a locus of belief (state) and action. By separating the notion of agency from the model of task environments, we do not have to subscribe to particular agent architectures (which one would assume will be adapted to the task environment at hand), and we may ask questions about the inherent social nature of the task environment at hand (allowing that the concept of society may arise before the concept of individual agents).

Section will discuss the general nature of the three modeling framework layers. Sections through discuss the details of the three levels, and are organized around a model built with this framework for the study of organizational design and coordination strategies in a multi-agent distributed sensor network environment.

General Framework

The principle purpose of a TÆMS model is to analyze, explain, or predict the performance of a system or some component. While TÆMS does not establish a particular performance criteria, it focuses on providing two kinds of performance information: the temporal intervals of task executions, and the *quality* of the execution or its result. *Quality* is an intentionally vaguely-defined term that must be instantiated for a particular environment and performance criteria. Examples of *quality* measures include the precision, belief, or completeness of a task result. We will assume that *quality* is a single numeric term with an absolute scale, although the algebra can be extended to vector terms. In a computationally intensive AI system, several quantities—the quality produced by executing a task, the time taken to perform that task, the time when a task can be started, its deadline, and whether the task is necessary at all—are affected by the execution of other tasks. In real-time problem solving, alternate task execution methods may be available that trade-off time for quality. Agents do not have unlimited access to the environment; what an agent believes and what is really there may be different.

The model of environmental and task characteristics proposed has three levels: *objective*, *subjective*, and *generative*. The *objective* level describes the essential, 'real' task structure of a particular problem-solving situation or instance over time. It is roughly equivalent to a formal description of a single problem-solving situation such as those presented in [Durfee and Lesser, 1991], without the information about particular agents. The *subjective* level describes how agents view and interact with the problem-solving situation over time (e.g., how much does an agent know about what is really going on, and how much does it cost to find out—where the uncertainties are from the agent's point of view). The subjective level is essential for evaluating control algorithms, because while individual behavior and system performance can be measured objectively, agents must make decisions with only subjective information.[2] Finally, the *generative* level describes the statistical characteristics required to generate the objective and subjective situations in a domain.

Objective Level

The *objective* level describes the essential structure of a particular problem-solving situation or instance over time. It focuses on how task interrelationships dynamically affect the *quality* and *duration* of each task. The basic model is that task groups appear in the environment at some frequency, and induce tasks T to be executed by the agents under study. Task groups are independent of one another, but tasks within a single task group have interrelationships. Task groups or tasks may have deadlines $D(T)$. The *quality* of the execution or result of each task influences the *quality* of the task group result $Q(T)$ in a precise way. These quantities can be used to evaluate the performance of a system.

An individual task that has no subtasks is called a method M and is the smallest schedulable chunk of work (though some scheduling algorithms will allow some methods to be preempted, and some schedulers will schedule at multiple levels of abstraction). There may be more than one method to accomplish a task, and each method will take some amount of time and produce a result of some *quality*. Quality of an agent's performance on an individual task is a function of the timing and choice of agent actions ('local effects'), and possibly previous task executions ('non-local effects').[3] The basic purpose of the objective model is to formally specify

[2] In organizational theoretic terms, subjective *perception* can be used to predict agent actions or *outputs*, but unperceived, objective environmental characteristics can still affect performance (or *outcomes*) [Scott, 1987].

[3] When local or non-local effects exist between tasks that are known by more than one agent, we call them *coordination relationships*[Decker and Lesser, 1991]

how the execution and timing of tasks affect this measure of quality.

A complete presentation of the mathematical details of an objective model can be found in our main conference paper [Decker and Lesser, 1993b]. We will concentrate here on examples.

Objective Modeling Example

This example grows out of the set of single instance examples of distributed sensor network (DSN) problems presented in [Durfee *et al.*, 1987], which were solved using the Distributed Vehicle Monitoring Testbed (DVMT)[Lesser and Corkill, 1983]. The authors of that paper compared the performance of several different coordination algorithms on these examples, and concluded that no one algorithm was always the best. This is the classic type of result that the TÆMS framework was created to address—we wish to *explain* this result, and better yet, to *predict* which algorithm will do the best in each situation. The level of detail to which you build your model will depend on the question you wish to answer—we wish to identify the characteristics of the DSN environment, or the organization of the agents, that cause one algorithm to outperform another. Even more importantly (as will be addressed in Section) we are interested not just in single instance examples, but the general distribution of episodes in the environment to analyze the relative merits of different approaches.

In a DSN problem, the movements of several independent vehicles will be detected over a period of time by one or more distinct sensors, where each sensor is associated with an agent. The performance of agents in such an environment is based on how long it takes them to create complete vehicle tracks, including the cost of communication. The organizational structure of the agents will imply the portions of each vehicle track that are sensed by each agent.

In our model of DSN problems, each vehicle track is modeled as a task group. The simplest objective model is that each task group T_i is associated with a track of length l_i and has the following objective structure, based on a simplified version of the DVMT[Lesser and Corkill, 1983]: (l_i) vehicle location methods (VLM) that represent processing raw signal data at a single location resulting in a single vehicle location hypothesis; ($l_i - 1$) vehicle tracking methods (VTM) that represent short tracks connecting the results of the VLM at time t with the results of the VLM at time $t + 1$; (1) vehicle track completion method (VCM) that represents merging all the VTMs together into a complete vehicle track hypothesis. Non-local enablement effects exist as shown in Figure 1—two VLMs *enable* each VTM, and all VTMs *enable* the lone VCM.

We have used this model to develop expressions for the expected value of, and confidence intervals on, the time of termination of a set of agents in any arbitrary DSN environment that has a static organizational structure and coordination algorithm [Decker and Lesser, 1993a]. We have also used this model to analyze a dynamic, one-shot reorganization algorithm (and have shown when the extra overhead is worthwhile versus the static algorithm). In each case we can predict the effects of adding more agents, changing the relative cost of communication and computation, and changing how the agents are organized. These results were achieved by direct mathematical analysis of the model and verified through simulation in TÆMS. We will give a summary of these results later in the paper (Section), after discussing the subjective and generative levels.

Expanding the Model We will now add some complexity to the model. The length of a track l_i above is a *generative* level parameter. Given a set of these generative parameters, we can construct the objective model for a specific problem-solving instance, or *episode*. Figure 1 shows the general structure of episodes in our DSN environment model, rather than a particular episode. To display an actual objective model, let us assume a simple situation: there are two agents, A and B, and that there is one vehicle track of length 3 sensed once by A alone (T^1), once by both A and B (T^2), and once by B alone (T^3). We now proceed to model the standard features that have appeared in our DVMT work for the past several years. We will add the characteristic that each agent has two methods with which to deal with sensed data: a normal VLM and a 'level-hopping' (LH) VLM (the level-hopping VLM produces less quality than the full method but requires less time; see [Decker *et al.*, 1990; Decker *et al.*, 1993] for this and other approximate methods). Furthermore, only the agent that senses the data can execute the associated VLM; but any agent can execute VTMs and VCMs if the appropriate enablement conditions are met.

Figure 2 displays this particular problem-solving episode. To the description above, we have added the fact that agent B has a faulty sensor (the durations of the grayed methods will be longer than normal); we will explore the implications of this after we have discussed the subjective level of the framework in the next section. An assumption made in [Durfee *et al.*, 1987] is that redundant work is not generally useful; this is indicated by using *max* as the combination function for each agent's redundant methods. We could alter this assumption by simply changing this function (to *mean*, for example). Another characteristic that appeared often in the DVMT literature is the sharing of results between methods (at a single agent); we would indicate this by the presence of a sharing relationship (similar to *facilitates*) between each pair of normal and level-hopping VLMs. Sharing of results could be only one-way between methods.

Now we will add two final features that make this model more like the DVMT. First, low quality results tend to make things harder to process at higher levels. For example, the impact of using the level-hopping VLM is not just that its quality is lower, but also that it affects the quality and duration of the VTM it enables (because not enough possible solutions are eliminated). To model this, we will use the *precedence* relationship instead of *enables*: not only do the VLM methods enable the VTM, but they can also hinder its execution if the enabling results are of low quality. Secondly, the first VLM execution provides information that slightly shortens the executions of other VLMs in the same vehicle track (because the sensors have been properly configured with the correct signal processing algorithm parameters with which to sense that particular vehicle). A similar *facilitation* effect occurs at the tracking level. These effects occur both locally and when

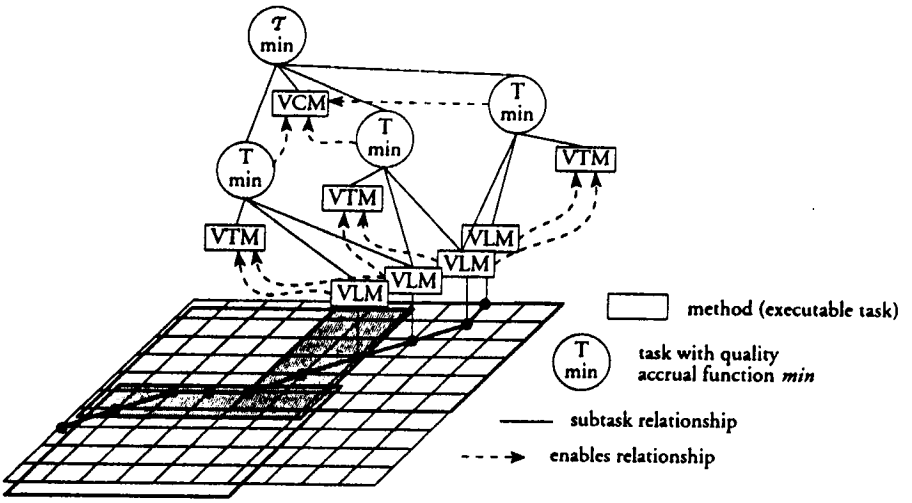

Figure 1: Objective task structure associated with a single vehicle track.

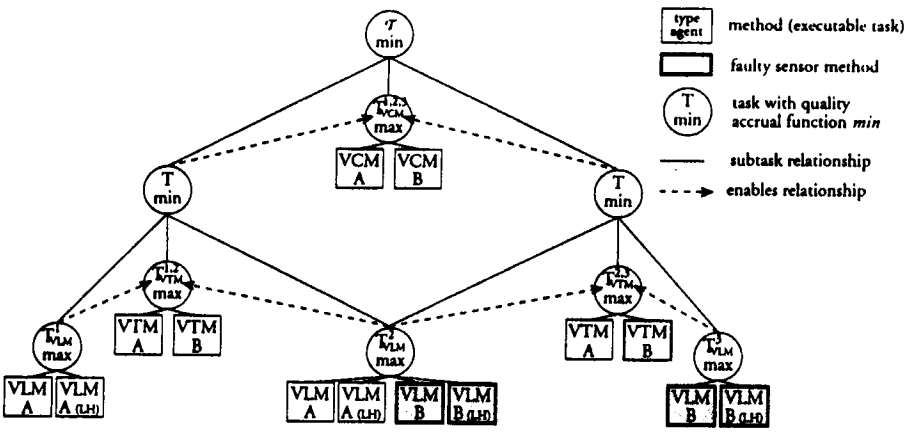

Figure 2: Objective task structure associated with two agents. $\mathcal{T}$ is the highest level task—to generate a single vehicle track. It has three subtasks: a task to generate a track from time 1 to time 2, a task to generate a track from time 2 to time 3, and the task $T_{VCM}^{1,2,3}$ to join these results into a full track. The subtasks each have a similar structure, again with three subtasks: process the data at time 1 (T_{VLM}^{1}), process the data at time 2 (T_{VLM}^{2}), and generate the track from time 1 to time 2 (T_{VTM}^{12}). Many of the low level tasks have methods that can be executed by either agent, and sometimes each agent has multiple methods (see text).

results are shared between agents—in fact, this effect is very important in motivating the agent behavior where one agent sends preliminary results to another agent with bad sensor data to help the receiving agent in disambiguating that data. Figure 3 repeats the objective task structure from the previous figure, but omits the methods for clarity. Two new tasks have been added to model facilitation at the vehicle location and vehicle track level.[4] T_{VL} indicates the highest quality initial work that has been done at the vehicle level, and thus uses the quality accrual function *maximum*. T_{VT} indicates the progress on the full track; it uses *summation* as its quality accrual function. The more tracking methods are executed, the easier the remaining ones become. The implications of this model are that in a multi-agent episode, then, the question becomes when to communicate partial results to another agent: the later an agent delays communication, the more the potential impact on the other agent, but the more the other agent must delay. We examined this question somewhat in [Decker and Lesser, 1991].

Other effects, such as how early results *facilitate* later ones, can also be modeled. At the end of the next section, we will return to this example and add to it subjective features: what information is available to agents, when, and at what cost.

Subjective Level

The purpose of a subjective level model of an environment is to describe what portions of the objective model of the situation are available to 'agents'. It answers questions such as "when is a piece of information available," "to whom is it available," and "what is the cost to the agent of that piece of information". This is a description of how agents might interact with their environment—what options are available to them.

To build such a description we must introduce the concept of *agency* into the model. Ours is one of the few comprehensive descriptions of computational task environments, but there are many formal and informal descriptions of the concept of agency (see [Gasser, 1991; Hewitt, 1991]). Rather than add our own description, we notice that these formulations define the notion of *computation* at one or more agents, not the environment that the agents are part of. Most formulations contain a notion of *belief* that can be applied to our concept of "what an agent believes about its environment". Our view is that an "agent" is a locus of belief and action (such as computation).

A subjective mapping of an objective problem solving situation is a function φ from an agent and objective assertions to the beliefs of an agent. For example, we could define a mapping φ where each agent has a probability p of believing that the maximum quality of a method is the objective value, and a probability $1 - p$ of believing the maximum quality is twice the objective value. Any objective assertion has some subjective mapping, including $\mathbf{q}$ (maximum quality of a method), $\mathbf{d}$ (duration of a method), deadlines, and relationships like subtask or non-local effects.

The beliefs of an agent affect its actions through some control mechanism. Since this is the focus of most of our and others' research on local scheduling, coordination, and other control issues, we will not discuss this further . The agent's control mechanism uses the agent's current set of beliefs to update three special subsets of these beliefs (alternatively, *commitments*[Shoham, 1991]) identified as the sets of information gathering, communication, and method execution actions to be computed. We provide a meta-structure for the agent's state-transition function that is divided into the following 4 parts: control, information gathering, communication, and method execution. First the control mechanisms assert (commit to) information-gathering, communication, and method execution actions and then these actions are computed one at a time, after which the cycle of meta-states repeats. Details about the state changes associated with these actions can be found in [Decker and Lesser, 1993b].

Subjective Modeling Example

Let's return to the example we began in Section to demonstrate how adding a subjective level to the model allows us to represent the effects of faulty sensors in the DVMT. We will define the default subjective mapping to simply return the objective value, i.e., agents will believe the true objective quality and duration of methods and their local and non-local effects. We then alter this default for the case of faulty (i.e., noisy) sensors—an agent with a faulty sensor will not initially realize it ($\mathbf{d}_0$(faulty-VLM) = $2\mathbf{d}_0$(VLM), but $\varphi(A, \mathbf{d}_0(\text{faulty-VLM})) = \mathbf{d}_0(\text{VLM})$).[5] Other subjective level artifacts that are seen in [Durfee *et al.*, 1987] and other DVMT work can also be modeled easily in our framework. For example, 'noise' can be viewed as VLM methods that are subjectively believed to have a non-zero maximum quality ($\varphi(A, \mathbf{q}_0(\text{noise-VLM})) > 0$) but in fact have 0 objective maximum quality, which the agent does not discover until after the method is executed. The strength with which initial data is sensed can be modeled by lowering the subjectively perceived value of the maximum quality $\mathbf{q}$ for weakly sensed data. The infamous 'ghost track' is a subjectively complete task group appearing to an agent as an actual vehicle track, which *subjectively* accrues quality until the hapless agent executes the VCM method, at which point the true (zero) quality becomes known. If the track (subjectively) spans multiple agents' sensor regions, the agent can potentially identify the chimeric track through communication with the other agents, which may have no belief in such a track (but sometimes more than one agent suffers the same delusion).

Generative Level

By using the objective and subjective levels of TÆMS we can model any *individual* situation; adding a *generative* level to the model allows us to go beyond that and determine what the expected performance of an algorithm is over a long period of time and many individual problem solving episodes. Our previous work has created generative models of task interarrival times (exponential distribution), amount of work

[4] Note that these tasks were added to make the model more expressive; they are not associated with new methods.

[5] At this point, one should be imagining an agent controller for this environment that notices when a VLM method takes unusually long, and realizes that the sensor is faulty and replans accordingly.

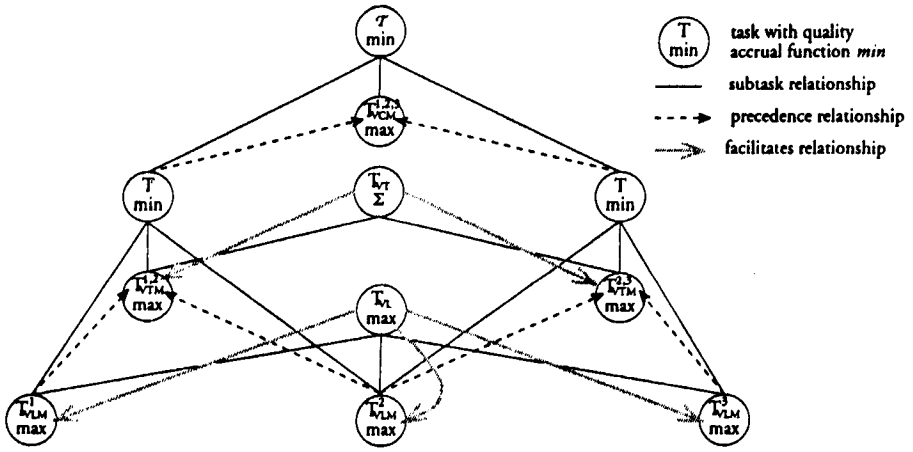

Figure 3: Non-local effects in the objective task structure. This is an expansion of the previous figure with the methods removed for clarity. Two new tasks have been added: one to represent the amount of work that has been done on individual tracks (T_{VT}), and one to represent the best work done on the initial data (T_{VL}). These new abstract tasks are used to indicate facilitation effects: information from the first VL method can be used to speed up other VL methods (by providing constraints on vehicle type, for example), and the more tracking information is available, the easier tracking becomes as the vehicle's path is constrained.

in a task cluster (Poisson), task durations (exponential), and the likelihood of a particular non-local effect between two tasks [Decker and Lesser, 1993a; Decker and Lesser, 1991; Garvey and Lesser, 1993]. Generative level statistical parameters can also be used by agents in their subjective reasoning, for example, an agent may make control decisions based on the knowledge of the expected duration of methods.

A generative level model can be constructed by careful analysis of the real environment being modeled, or by observing the statistical properties of real episodes (if that is possible). Even when certain parameters of the real world are unknown, they can be made variables in the model and then you can ask questions about how much they affect the things you care about. Our approach so far has been to verify our assumptions about the environment with simple statistical approaches [Kleijnen, 1987]. Detailed model verification will be more important when using our framework to optimize parameters in a real application, as opposed to learning the general effects of parameters on a coordination or negotiation algorithm (see Section).

Analysis Summary

Organizational theorists have long held that the organization of a set of agents cannot be analyzed separately from the agents' task environment, that there is no single best organization for all environments, and that different organizations are not equally effective in a given environment [Galbraith, 1977]. Most of these theorists view the uncertainties present in the environment as a key characteristic, though they differ in the mechanisms that link environmental uncertainty to effective organization. In particular, the *transaction cost economics* approach [Moe, 1984] focuses on the *relative efficiencies* of various organizations given an uncertain environment, while the updated versions of the *contingency theory* approach [Stinchcombe, 1990] focus on the need for an organization to expand toward the earliest available *information that resolves uncertainties* in the current environment.

We have used these general concepts, in conjunction with our modeling framework, to analyze potential organizational structures for the environment that has been our example in this paper, i.e., the class of naturally distributed, homogeneous, cooperative problem solving environments where tasks arrive at multiple locations, exemplified by distributed sensor networks. Our approach was to first construct the task environment model we have been using as an example in this paper, and then to develop expressions for the *expected efficiencies* of static and dynamic organizational structures, in terms of performance—the cost of communication and time to complete a given set of tasks. Finally, we validated these mathematical models by using simulations.

We briefly showed at the end of Section how we can predict the performance of a system given the objective and subjective models of a particular episode. This is very useful for explaining or predicting agent behavior in a particular episode or scenario, but not over many episodes in a real environment. To do this, we build probabilistic models of the relevant objective and subjective parameters (now viewed as random variables) that are based on generative level parameters. Another paper, [Decker and Lesser, 1993a], details this process, and shows how the distributions of objective parameters such as "the number of VLM methods seen by the maximally loaded agent" ($\hat{S}$) and "the max number of task groups seen by the same agent" ($\hat{N}$) can be defined from just the generative parameters $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$.

For example, the total time until termination for an agent receiving an initial data set of size $\hat{S}$ is the time to do local work, combine results from other agents, and build the completed results, plus two communication and information gathering

actions:

$$T_{\text{static}} = \hat{S}\mathbf{d}_0(\text{VLM}) + (\hat{S} - \hat{N})\mathbf{d}_0(\text{VTM}) + (a-1)\hat{N}\mathbf{d}_0(\text{VTM}) + \hat{N}\mathbf{d}_0(\text{VCM}) + 2\mathbf{d}_0(I) + 2\mathbf{d}_0(C) \quad (1)$$

We can use Eq. 1 as a predictor by combining it with the probabilities for the values of $\hat{S}$ and $\hat{N}$. Again, we refer the interested reader to [Decker and Lesser, 1993a] for derivations, verification, and applications of these results. Note that if the assumptions behind our generative model change (for instance, if we assume all agents initially line up side-by-side, instead of in a square, or if vehicles made a loop before exiting the sensed area) the probability distributions for $\hat{S}$ and $\hat{N}$ might change, but that the form of Eqn. 1 does not. If the agent's coordination algorithm changes, then Eqn. 1 will change (see [Decker and Lesser, 1993a]).

We have looked at both static organzations and dynamic organizations (in which the responsibilities of agents can be reassigned based on a developing view of the problem at hand). Due to the uncertainties explicitly represented in the task environment model, there may not be a clear performance tradeoff between static and dynamic organizational structures. Agents that have a dynamic organization have the option of meta-level communication—communicating about the current state of problem solving as opposed to communicating about solving the problem itself. In this way, *information that resolves uncertainties* about the current environment becomes available to the agents, allowing the agents to then create the most efficient organization for the situation. In [Decker and Lesser, 1993a] we present equations similar to Eq. 1 that show the potential benefits of dynamic reorganization in an arbitrary environment $\mathcal{D}$, and discuss when the overhead of meta-level communication is worthwhile.

Conclusions

This paper has presented an overview of TÆMS, a framework for modeling computationally intensive task environments. TÆMS exists as both a language for stating general hypotheses or theories and as a system for simulation. The important features of TÆMS include its layered description of environments (*objective* reality, *subjective* mapping to agent beliefs, *generative* description of the other levels across single instances); its acceptance of any performance criteria (based on temporal location and *quality* of task executions); and its non-agent-centered point of view that can be used by researchers working in either formal systems of mental-state-induced behavior or experimental methodologies. TÆMS provides environmental and behavioral structures and features with which to state and test theories about the control of agents in complex computational domains, such as how decisions made in scheduling one task will affect the utility and performance characteristics of other tasks.

TÆMS is not only a mathematical framework, but also a simulation language for executing and experimenting with models directly. The TÆMS simulator supports the graphical display of generated subjective and objective task structures, agent actions, and statistical data collection in CLOS on the TI Explorer. These features help in both the model-building stage and the verification stage. The TÆMS simulator is being used not only for research into the organization and coordination of distributed problem solvers[Decker and Lesser, 1993a; Decker and Lesser, 1991; Decker and Lesser, 1992], but also for research into real-time scheduling of a single agent[Garvey and Lesser, 1993], scheduling at an agent with parallel processing resources available, and soon, learning coordination algorithm parameters.

TÆMS does not at this time automatically learn models or automatically verify them. While we have taken initial steps at designing a methodology for verification (see [Decker and Lesser, 1993a]), this is still an open area of research [Cohen, 1991]. Our future work will include building new models of different environments that may include physical resource constraints, such as airport resource scheduling. The existing framework may have to be extended somewhat to handle consumable resources. Other extensions we envision include specifying dynamic objective models that change structure as the result of agent actions. We also wish to expand our analyses beyond the questions of scheduling and coordination to questions about negotiation strategies, emergent agent/society behavior, and organizational self-design.

References

Boddy, Mark and Dean, Thomas 1989. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.*

Carver, Norman and Lesser, Victor 1991. A new framework for sensor interpretation: Planning to resolve sources of uncertainty. In *Proceedings of the Ninth National Conference on Artificial Intelligence.* 724–731.

Cohen, Philip R. and Levesque, Hector J. 1990. Intention is choice with commitment. *Artificial Intelligence* 42(3).

Cohen, Paul; Greenberg, Michael; Hart, David; and Howe, Adele 1989. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine* 10(3):33–48. Also COINS-TR-89-61.

Cohen, Paul R. 1991. A survey of the eighth national conference on artificial intelligence: Pulling together or pulling apart? *AI Magazine* 12(1):16–41.

Decker, Keith S. and Lesser, Victor R. 1991. Analyzing a quantitative coordination relationship. COINS Technical Report 91–83, University of Massachusetts. To appear in the journal *Group Decision and Negotiation*, 1993.

Decker, Keith S. and Lesser, Victor R. 1992. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems* 1(2).

Decker, Keith S. and Lesser, Victor R. 1993a. An approach to analyzing the need for meta-level communication. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambéry.

Decker, Keith S. and Lesser, Victor R. 1993b. Quantitative modeling of complex computational task environments. In

Proceedings of the Eleventh National Conference on Artificial Intelligence, Washington.

Decker, Keith S.; Lesser, Victor R.; and Whitehair, Robert C. 1990. Extending a blackboard architecture for approximate processing. *The Journal of Real-Time Systems* 2(1/2):47–79.

Decker, Keith S.; Garvey, Alan J.; Humphrey, Marty A.; and Lesser, Victor R. 1993. A real-time control architecture for an approximate processing blackboard system. *International Journal of Pattern Recognition and Artificial Intelligence* 7(2).

Durfee, E.H. and Lesser, V.R. 1991. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics* 21(5):1167–1183.

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

Galbraith, J. 1977. *Organizational Design.* Addison-Wesley, Reading, MA.

Garvey, Alan and Lesser, Victor 1993. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man, and Cybernetics* 23(6). Special Issue on Scheduling, Planning, and Control.

Gasser, Les 1991. Social conceptions of knowledge and action. *Artificial Intelligence* 47(1):107–138.

Hewitt, Carl 1991. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence* 47(1):79–106.

Kleijnen, Jack P. C. 1987. *Statistical Tools for Simulation Practitioners.* Marcel Dekker, New York.

Lesser, Victor R. and Corkill, Daniel D. 1983. The distributed vehicle monitoring testbed. *AI Magazine* 4(3):63–109.

Moe, Terry M. 1984. The new economics of organization. *American Journal of Political Science* 28(4):739–777.

Pollack, Martha E. and Ringuette, Marc 1990. Introducing Tileworld: Experimentally evaluating agent architectures. In *Proceedings of the Eighth National Conference on Artificial Intelligence.* 183–189.

Russell, Stuart J. and Zilberstein, Shlomo 1991. Composing real-time systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia. 212–217.

Scott, W. Richard 1987. *Organizations: Rational, Natural, and Open Systems.* Prentice-Hall, Inc., Englewood Cliffs, NJ.

Shoham, Yoav 1991. AGENT0: A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim. 704–709.

Stinchcombe, Arthur L. 1990. *Information and Organizations.* University of California Press, Berkeley, CA.

Zlotkin, Gilad and Rosenschein, Jeffrey S. 1990. Blocks, lies, and postal freight: The nature of deception in negotiation. In *Proceedings of the Tenth International Workshop on Distributed AI*, Texas.