Analyzing the Need for Meta-Level Communication [1]

Keith S. Decker and Victor R. Lesser
Technical Report 93–22 (IJCAI-93 + AAAI-93)

UMass Computer Science Technical Report 93–22
May 1, 1995

Abstract

In naturally distributed, homogeneous, cooperative problem solving environments where well-defined tasks arrive at multiple locations, decisions must be made about the extent of, and overlap between, each agent's area of responsibility—the agents' *organization*. The organization may be constructed statically by a system designer, or dynamically by the agents during problem solving. No one organization is optimal across environments or even specific problem solving instances [6, 7, 8].

This paper presents an analysis of static and dynamic organizational structures for this class of environments, exemplified by distributed sensor networks. We first show how the performance of any static organization can be statistically described, and then show under what conditions dynamic organizations do better and worse than static ones. Finally, we show how the variance in the agents' performance leads to uncertainty about whether a dynamic organization will perform better than a static one given only agent *a priori* expectations. In these cases, we show when meta-level communication about the actual state of problem solving will be useful to agents in constructing a dynamic organizational structure that outperforms a static one. Viewed in its entirety, this paper also presents a methodology for answering questions about the design of distributed problem solving systems by analysis and simulation of the characteristics of a complex environment rather than by relying on single-instance examples.

[1] Portions of this technical report appeared in the proceedings of IJCAI-93 'An approach to analyzing the need for meta-level communication', and AAAI-93 'A one-shot dynamic coordination algorithm for distributed sensor networks'. 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 Introduction

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 [8]. 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 [11] focuses on the *relative efficiencies* of various organizations given an uncertain environment, while the modern *contingency theory* approach [14] focuses on the need for an organization to expand toward the earliest available *information that resolves uncertainties* in the current environment.

In this paper we use both of these concepts to analyze potential organizational structures for a class of naturally distributed, homogeneous, cooperative problem solving environments where tasks arrive at multiple locations, exemplified by distributed sensor networks [10]. Previous approaches to analyzing organizations in distributed sensor networks have either not focused on the effectiveness of the organization [2, 12], or have only analyzed organizational effectiveness in particular, single-instance examples [7]. Our approach is to model the task environment mathematically, using a formalism developed specifically to study distributed coordination and scheduling [5]. We then develop expressions for the *expected efficiencies* of static and dynamic organizational structures, in terms of the cost of communication and time to complete a given set of tasks. Finally, we validate these mathematical models by using simulations.

A dynamic organization is one 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.

Section 2 describes the task environment model, the assumptions behind it, and analyzes the uncertainties present. Section 4 describes static and dynamic organizational structures, and develops expressions for the expected performance of each organizational style. This section also describes a particular algorithm for generating dynamic structures, and discusses the actual performance of the algorithm. In Section 5 we then show how the variance in performance without communication can lead to the efficient use of meta-level communication to customize a dynamic organizational structure. Finally, we discuss how these results can be used by designers of distributed problem solvers, and how our methodology can be used by other researchers. Throughout each section, we will illustrate and confirm the analytical results experimentally, using as an example a simulated distributed sensor network similar to the Distributed Vehicle Monitoring Testbed (DVMT) [10].

2 Task Environment Model

Our task environment model of naturally distributed problems assumes that several independent groups of tasks arrive at multiple locations over a period of time called an *episode*. For example, in a distributed sensor network (DSN) episode 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 will be based on how long it takes them to process all the task groups, which will include the cost of communicating data, task results, and meta-level communication, if any. The organizational structure of the agents will imply which subsets of

which task groups are available to which agents and at what cost. For example, if DSN agents have overlapping sensors, either agent can potentially work on data in the overlapping area (from its own sensor) without any extra communication costs. We make several simplifying assumptions: that the agents are homogeneous (have the same capabilities with respect to receiving data, communicating, and processing tasks), that the agents are cooperative (interested in maximizing the system performance over maximizing their individual performance), that the data for each episode is available simultaneously to all agents as specified by their initial organization, and that there are only structural (precedence) constraints within the subtasks of each task group.[1]

Any single episode can be specified by listing the task groups, and what part of each task group was available to which agents, given the organizational structure. Our analysis will be based on the statistical properties of episodes in an environment, not any single instance of an episode. The properties of the episodes in a DSN environment are summarized by the tuple $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$ where A specifies the number of agents, η the expected number of task groups, r and o specify the structural portion of the organization by the *range* of each agent and the *overlap* between agents, and $\mathcal{T}$ specifies the homogeneous task group structure (Section 2.5 and Figure 4 describes how task group structures are specified).

Our analysis initially focuses on exactly what *a priori* knowledge agents have about the distribution of task groups in an episode. First we will look at the distribution of the lowest-level sensor subtasks of a single task group among multiple agents (deriving the maximum expected number of subtasks), and then we will look at the distribution of task groups themselves. These results will then be used in subsequent sections to derive the total amount of work, and therefore expected termination performance, under various organizational structures and control schemes. Section 2.2 derives the expected number of low-level sensor subtasks at the most heavily-loaded agent given the number of task groups that same agent sees, how many agents see a single task group, and the environmental parameters. Section 2.3 then derives the number of task groups at the most heavily-loaded agent given the number of agents that see a single task group and the environmental parameters. Section 2.4 derives the number of agents that will see a single task group given the environmental parameters. Finally, Section 2.5 will show the structure of a task group, which allows us to derive the amount of work an agent must do given the number of sensor subtasks it executes, the number of task groups at the agent, and the number of agents involved in a single task group. All of these parts together allow us to derive the expected termination time of a particular system of agents.

2.1 Task environment simulation

In the next section and for the rest of the paper, we will test the model we are developing against simulated DSN problems. Each simulated DSN episode will take place on a grid where the concepts of length and size correspond directly to physical distances. For example, Figure 1 illustrates several simple organizations imposed on such a grid in our simulation.

In the simulation we assume that each vehicle is sensed at discrete integer locations (as in the DVMT), randomly entering on one edge and leaving on any other edge. Inbetween the vehicle travels along a *track* moving either horizontally, vertically, or diagonally each time unit using a simple DDA line-drawing algorithm (see Figure 5). In an 18×18 grid, the (empirical) average length of a track is 14 units—the actual length of any one track will range from 2 to 19 units and is not distributed normally. Given the organization (r, o, and A, and the geometry), we can calculate what locations are seen by the sensors of each agent. This information can then be used along with the locations traveled by each vehicle to determine what part of each task group is initially available to each agent. Section 2.5 will detail what the structure of each task group is for the DSN simulation.

[1] In general there are usually more complex interrelationships between subtasks that affect scheduling decisions, such as *facilitation* [4].

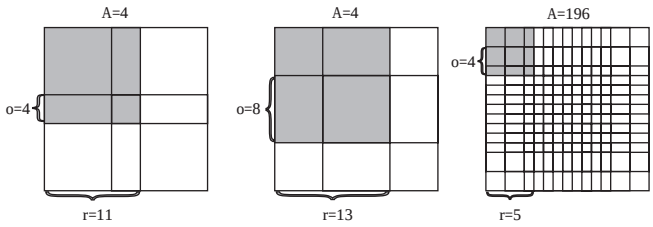

Figure 1: Examples of 18 × 18 DSN organizations

This specification is applicable to other interesting environments. For example: each task group may comprise several different types of subtasks; each agent may only respond to a certain type of subtask (its 'range'); multiple agents may respond to the same types ('overlap'). In general, the parameters of 'range' and 'overlap' can be multidimensional.

2.2 Expected number of sensor subtasks

In order to analyze the performance of a particular organization, we will want to know what proportion of each task group each agent is likely to process. There will be some upper limit on this proportion (related to the agent's range r), and sometimes the agent will process less than this upper limit. Especially in static organizational structures where tasks are not exchanged, the termination of the system as a whole can be tied to the completion of all tasks at the most heavily loaded agent. Normally, we would use the *average* part of a task group to be seen, but since the focus of our analysis is the termination of problem solving, we need to examine the *expected maximum* portion of a task group to be seen. This section will develop an equation for the expected maximum workload at an agent by counting the expected number of low-level sensor subtasks (each individually associated with a sensed vehicle location) that the maximally loaded agent will have.

The amount of a single task group seen by an agent (which is the same as the number of sensor subtasks in the DSN example) can be viewed as a random variable S with a probability density function and corresponding cumulative distribution function. In the DSN environment, S is discrete, and its probability function (determined empirically) is heavily weighted toward r (the maximum). To simplify the analysis, instead of letting S correspond to the number of subtasks in a single task group seen by an agent, let's create a new random variable $\mathbf{S}$ that we have equal 1 if the agent sees the maximum amount, and 0 otherwise. Now $\mathbf{S}$ has a Bernoulli (coin-tossing) distribution with parameter p corresponding to the chance of an agent seeing the maximum amount r of a task group. Let's assume we know that $N \leq n$ is the number of task groups at the maximally loaded agent, and that on average $a \leq A$ agents see a single task group (we'll remove these assumptions later). The number of times an agent sees the maximum out of N task groups (N coin flips) then has a binomial distribution ($b_{N,p}(s)$). We need to know, given that a agents each flip N coins, what the distribution is of the *maximum* number of 'heads' any agent sees—this is called the binomial max order statistic, $g_{a,N,p}(s)$:[2]

$$\begin{array}{rcll} b_{N,p}(s) & = & \binom{N}{s} p^s (1-p)^{N-s} & [\Pr[\mathbf{S} = s]] \\ B_{N,p}(s) & = & \sum_{x=0}^{s} b_{N,p}(x) & [\Pr[\mathbf{S} \leq s]] \\ g_{a,N,p}(s) & = & B_{N,p}(s)^a - B_{N,p}(s-1)^a & [\Pr[\hat{\mathbf{S}} = s]] \end{array}$$

The random variable $\mathbf{S}$ referred to *any* agent, the new random variable $\hat{\mathbf{S}}$ refers to the *maximally loaded agent*. It's probability function $g(s)$ has a much steeper shape and larger expected value than the binomial $b(s)$ (try it and see). In the DSN example we have $p = 0.5$[3] and the amount of a task group

[2] A more detailed derivation of this result is in our tech report.

[3] Empirically determined through simulation.

seen when an agent does not see the maximum amount averages $r/2$. Now we can convert back from the easy to analyze random variable $\hat{\mathbf{S}}$ to the variable we are really interested in, $\hat{S}$, by the equation $\hat{S} = (r\hat{\mathbf{S}} + (r/2)(N - \hat{\mathbf{S}}))$. The expected heaviest load seen by any agent when a agents see N task groups with a probability p of seeing r and probability $1 - p$ of seeing $r/2$ is:

$$E[\hat{S}|N, a] = \sum_{s=0}^{N} g_{a,N,p}(s)(rs + \frac{r}{2}(N - s)) \tag{1}$$

To reiterate: out of N task groups the maximally loaded agent sees the maximum r some $\hat{\mathbf{S}}$ (a random variable) times, and the other $(N - \hat{\mathbf{S}})$ times it sees only $r/2$. Eq. 1 shows the expected value of a new random variable $\hat{S}$ that indicates the number of sensor subtasks at the *maximally* loaded agent when there are N task groups in an episode. N is itself a random variable, we'll look at its distribution in the next section.

Figure 2 shows the heaviest load actually observed and averaged over 1000 runs, plotted against the expected value, for n tracks and all square DSN organizations $[2 \leq r \leq 10, 0 \leq o \leq r, 1 \leq \sqrt{A} \leq 10, 1 \leq N \leq 10]$ $(R^2 > .98)$[4]. The colors of the points refer to the value of r; lighter grey corresponds to larger values of r.

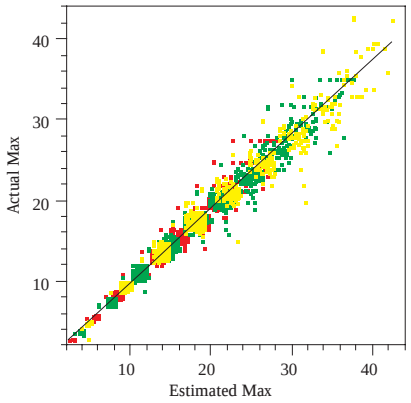

Figure 2: Actual versus predicted heaviest load $\hat{S}_N$ for various values of A, r, o, and N

2.3 Expected Number of Task Groups

Given the maximum number of task groups seen by an agent (N), we can calculate the expected heaviest agent load using Equation 1. But this begs the question of what is the maximum number of task groups an individual agent will see, given the actual (n) or expected number (η) that the entire system will see. The solution is similar—each agent either sees or does not see each of the n task groups, another binomial process. Let N_i be the number of task groups sensed by agent i, with a binomial distribution of parameters n and q. If a is again the number of agents that see a single task group and A the total number of agents, then $q = a/A$, the probability that each agent will see a particular track (we'll give an equation for a next). By the same derivation as in the last section, the max order statistic $\hat{N}$ has the probability function $g_{A,n,a/A}(s)$, and expected value:

$$E[\hat{N}|n, a] = \sum_{s=0}^{n} s g_{A,n,a/A}(s) \tag{2}$$

[4] R^2 is the squared correlation coefficient, a measure of goodness of fit. It may be interpreted as the proportion of total variabity in the observed data that is explained by the model.

Figure 3 shows the actual mean value of the maximum number of tracks seen by an agent over 1000 runs of the DSN simulation, versus the predicted value, for n tracks and square DSN organizations $[2 \leq r \leq 10, 1 \leq \sqrt{A} \leq 10, 1 \leq n \leq 10]$ without any overlap ($R^2 > .96$)

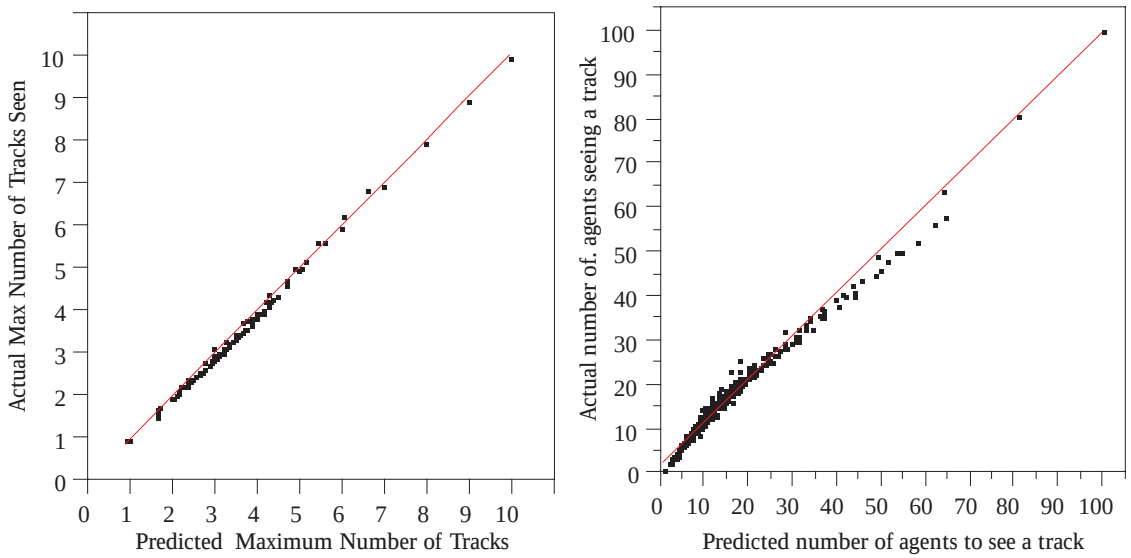

Figure 3: On the left, actual versus predicted maximum number of task groups (tracks) seen by any one agent for various r, A, and n. On the right, actual versus predicted average number of agents seeing a single task group (track) for various r, o, and A.

2.4 Expected Number of Agents

The only remaining term we need to analyze before deriving an expression for system performance is a, the expected number of agents that will see a single task group. In general, a will depend on the total number of agents A and the organization (r and o). When there is only one agent, it will see every task group ($a = 1$). When the agents overlap completely, every agent sees every task group ($[o = r] \rightarrow [a = A]$). When the agents in a square environment do not overlap, $a = \sqrt{A}$. The relationship follows the ratio of the area solely covered by an agent plus the area of the overlapping section, to the total area covered alone:

$$a = A^{(\frac{r^2+o^2}{2r^2})} \tag{3}$$

Note that a is not a random variable, it is just derived directly from environmental parameters. Figure 3 shows a regression of the actual average value of a over 1000 runs verses the predicted value for all 630 DSN organizations $[2 \leq r \leq 10, 0 \leq o \leq r, 1 \leq \sqrt{A} \leq 10]$ ($R^2 > .98$).

2.5 Work Involved in a Task Structure

Finally we turn to modeling the performance of the system as a whole, which is based on the structure of the tasks involved. We have developed a characterization of task environments that formally captures the range of features, processes, and especially interrelationships that occur during computationally intensive coordination and scheduling [5]. The model of environmental and task characteristics we propose has three levels: *objective*, *subjective*, and *generative*; the subjective level is not discussed here.

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. In this paper we will concentrate only on duration as a performance metric. The basic

model is that task groups $\mathcal{T}$ occur 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. An individual task that has no subtasks is called a method M and is the smallest schedulable chunk of work. The quality and duration 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'). When local or non-local effects exist between tasks that are known by more than one agent, they become *coordination relationships* [4, 3]. The basic purpose of the objective model is to formally specify how the execution and timing of tasks affect quality and duration.

At the lowest level, each method (leaf task) M at time t can produce, if executed, some maximum quality $\mathbf{q}(M,t)$ in some amount of time $\mathbf{d}(M,t)$ (each method has an initial maximum quality and duration $\mathbf{q}_0(M)$ and $\mathbf{d}_0(M)$). $Q(M,t)$ will denote the quality of method M at time t; $Q(M,t) = 0$ before a method is executed, and in the simplest case $Q(M,t) = \mathbf{q}(M,t) = \mathbf{q}_0(M)$ after the method is executed. In this paper, we will be concerned primarily with method durations, and the choice of method quality accrual function $Q(M,t)$ is not significant. Any task execution that starts before the execution of T completes may potentially affect T's execution through *non-local effects*. There are precisely two possible non-local effects on T under our model: duration effects, where $\mathbf{d}(T,t)$ (duration) is changed for some or all of the methods for a task; and quality effects, where $\mathbf{q}(T,t)$ (quality) is changed similarly. A non-local effect on a task is initiated by the execution of some other task in the task group. The effect is dependent on the relative timing of the two task executions, the quality of the task causing the effect, and whether information was transmitted between the two tasks.

Other work has considered the case of *facilitation*, a non-local effect where the availability of a result from one task alters the quality and duration of another task, but the non-communication of a result has no effect [4]. This work considers a different relationship, *precedence*. If task A precedes task B, then the maximum quality $\mathbf{q}(B,t) = 0$ until A is completed and the result is available, when the maximum quality will change to the initial maximum quality $\mathbf{q}(B,t) = \mathbf{q}_0(B)$.

2.5.1 Execution Model

For this paper we use an extremely simple model of execution. Agents can perform three actions: method execution, communication, and information gathering. The control component of an agent determines the next action an agent will perform based on the agent's current set of beliefs [1, 13]. A method execution action, of method M, that is begun at time t will conclude at time $t+\mathbf{d}(M,t)$. An information gathering action has duration $\mathbf{d}_0(I)$ and updates the agent's set of beliefs with any new information in the environment, for example, the arrival of data at the start of an episode, or communications from other agents. A communication action has duration $\mathbf{d}_0(C)$ and, after a communication delay, makes information (such method execution results) available to other agents. The agent on the receiving side must perform an information gathering action before the communication can affect its local beliefs.

2.5.2 Simple Objective DSN Model

Recall that the summary of a DSN environment was the tuple $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$; this will become our generative model, especially the parameter η (expected number of task groups). A particular episode in this environment can be described by the tuple $D =< A, r, o, \mathcal{T}_1, \ldots, \mathcal{T}_n >$ where n is a random variable drawn from an unknown distribution with location parameter (central tendency) of η. Note that we make almost no assumptions about this distribution; its characteristics will differ for different environments. For example, in the description of our DSN simulation early in Section 2 we noted the physical process by which vehicle tracks were generated and that the length of the tracks was not normally distributed.

Each task group $\mathcal{T}i$ is associated with a track of length l_i and has the same basic objective structure, based on the DVMT:

- l_i Vehicle Location Methods (VLM's) that represent processing raw signal data at a single location to a single vehicle location hypothesis.
- $l_i - 1$ Vehicle Tracking Methods (VTM's) 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 VTM's together into a complete vehicle track hypothesis.

Non-local precedence effects exist between each method at one level and the appropriate method at the next level as shown in Figure 4—two VLMs precede each VTM, and all VTM'S precede the lone VCM.

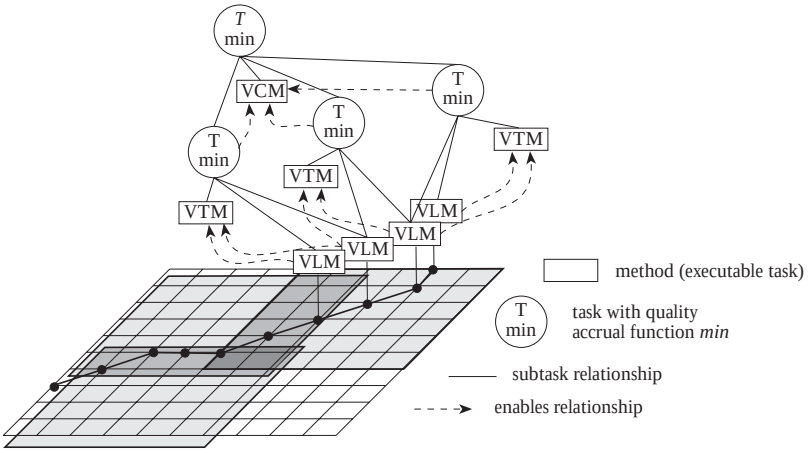

Figure 4: Objective task structure associated with a single vehicle track.

If we assume that each VLM has initial duration $\mathbf{d_0}(\text{VLM})$ and each VTM has the initial duration $\mathbf{d_0}(\text{VTM})$, then we can see from the task structure that for each task group the total execution time taken by a single processor agent will be:

$$l_i\mathbf{d_0}(\text{VLM}) + (l_i - 1)\mathbf{d_0}(\text{VTM}) + \mathbf{d_0}(\text{VCM}) \tag{4}$$

This task structure is a simplification of the real DVMT task structure. For example, there is no sensor noise (which will cause *facilitation* relationships between tasks, and there is no confusion caused by 'ghost tracks'. Adding these features to the task structure will cause some interesting phenomena that we will discuss briefly in the conclusions.

3 Model Summary

This section has described our basic model of the DSN environment, beginning with the basic parameters $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$. A particular episode in this environment can be indicated as $D =< A, r, o, \mathcal{T}_1, \ldots, \mathcal{T}_n >$, where each of the n task groups has the structure mentioned above. This section also developed expressions for $\hat{S}$, the number of low-level sensor subtasks (i.e., VLM's) at the most heavily loaded agent, the number of task groups $\hat{N}$ at the most heavily loaded agent, and the average number of agents a that see a single task group. Another way to look at these results is that we have derived the probability distributions of these variables, i.e., if the system of agents as a whole sees n total task groups, then the distributions of $\hat{N}$ and $\hat{S}$ are:

$$\Pr[\hat{N} = N|n] \quad = \quad g_{A,n,\frac{a}{A}}(N) \tag{5}$$

$$\Pr[\hat{\mathbf{S}} = s | \hat{N} = N] = g_{a,N,0.5}(s) \quad (6)$$

$$\hat{S} = (r\hat{\mathbf{S}} + (r/2)(N - \hat{\mathbf{S}})) \quad (7)$$

Finally Eq. 4 derived the duration, or amount of processing work, involved in a single task group from its structure.

In the next section we will combine these results with simple local scheduling and coordination algorithms appropriate for static and dynamic organizations. This will allow us to describe the performance of a system of agents following one of these organizational algorithms in a particular episode or environment.

4 Static vs. Dynamic Organizational Structures

Now we have the necessary background to analyze static and dynamic organizational structures. We have equations for the maximum expected number of subtasks at an agent given the number of task groups seen $\hat{S}$, the maximum expected number of task groups seen given the total number $\hat{N}$, and the predicted number of agents sensing part of a task group a. The key to static structures is to divide up the overlap area *a priori* (rather than to penalize agents for doing redundant work in the overlap area [7]). The key to dynamic organizational structures is to transfer tasks so that all the agents' resources are used efficiently.

We will repeat the assumptions we discussed at the start of Section 2 on page 1: the agents are homogeneous (have the same capabilities with respect to receiving data, communicating, and processing tasks), the agents are cooperative (interested in maximizing the system performance over maximizing their individual performance), the data for each episode is available simultaneously to all agents as specified by their initial organization, and there are only structural (precedence) constraints within the subtasks of each task group.

4.1 Analyzing Static Organizations

In a static organization, agents divide the overlapping areas of their ranges as evenly as possible. The result is a new area of responsibility $r' = r - \frac{o}{2}$ for each agent with no overlap (see Figure 5).[5] Given the task structure as described in Section 2.5 and shown in Figure 4, and any raw data or communicated task results provided by information gathering actions, the agent can at any time build a list of currently executable methods (under the set of precedence constraints). Also, at any time an agent can build a list of methods that need to be executed, but cannot be because their precedence constraints have not yet been met. The communication action in this algorithm is a broadcast of the highest level results of all the task groups an agent has worked on. Each agent follows the same control algorithm (remember, all the raw data is available at the start) and terminates when all task groups are completed (either locally or by reception of the result from another agent):

[5] The reason for overlap will be apparent in dynamic structures—multiple agents can work in an overlapping area without paying any cost for communicating raw data between them. Overlap can also provide redundancy in case of agent failure.

```
(Repeat
        Do Information-Gathering-Action
        (Repeat
                Let E = [get set of currently executable methods]
                (For method In E
                        Do Method-Execution-Action(method))
        Until (null E))
        Do Communication-Action(broadcast highest-level results)
        Let W = [get set of methods still waiting on precedence constraints]
Until (null W))
```

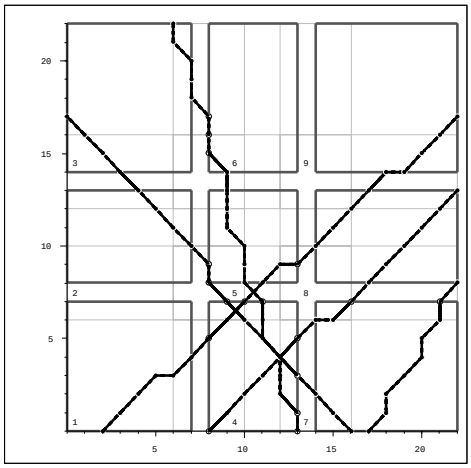

Figure 5: Example of a 3x3 organization, $r = 11$, $o = 5$, with 5 tracks. The thick dark grey boxes outline the default static organization, where there is no overlap.

First, let us analyze this algorithm assuming that only *one* task group (vehicle track) is present. In the environment $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$, if we let S' represent the largest amount of low-level data in *one* task group seen by any agent, and a the total number of agents that see the task group (from Equation 3), then the amount of time it will take that agent to construct a complete solution is equal to the amount of time it will take for the initial information gathering action $(\mathbf{d}_0(I))$ plus the amount of time to do all the local work $(S'\mathbf{d}_0(\text{VLM}) + (S' - 1)\mathbf{d}_0(\text{VTM}))$, communicate that work $(\mathbf{d}_0(C))$, get the other agents' results $(\mathbf{d}_0(I))$, plus the amount of time to combine results from the other $a - 1$ agents $((a - 1)\mathbf{d}_0(\text{VTM}))$, plus time to produce the final complete task group result $(\mathbf{d}_0(\text{VCM}))$, plus communicate that result to everyone $(\mathbf{d}_0(C))$. For simplicity we will assume that $\mathbf{d}_0(I)$ and $\mathbf{d}_0(C)$ are constant and do not depend on the amount of data. Note also that since this agent is the most heavily loaded, by definition all other agents will finish their work and have communicated it by the time this agent finishes its local work.

In the general case, if the system sees $n = \eta$ total task groups, then the expected amount of low-level sensor data (size of the initial data set) at the maximally loaded agent can be derived from the marginal expected value for $\hat{S}$ given the joint distribution of $\hat{S}$ (Eqns. 6, 7) and $\hat{N}$ (Eq. 5):

$$E[\hat{S}] = \sum_{N=0}^{n} \sum_{s=0}^{N} g_{A,n,\frac{a}{A}}(N) g_{a,N,p}(s)(rs + \frac{r}{2}(N - s)) \tag{8}$$

Similar to the single task group case, 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:

$$\hat{S}\mathbf{d}_0(\text{VLM}) + (\hat{S} - \hat{N})\mathbf{d}_0(\text{VTM}) + (a - 1)\hat{N}\mathbf{d}_0(\text{VTM}) + \hat{N}\mathbf{d}_0(\text{VCM}) + 2\mathbf{d}_0(I) + 2\mathbf{d}_0(C) \tag{9}$$

We can use Eq. 9 as a predictor by combining it with the probabilities for the values of $\hat{S}$ and $\hat{N}$ given in Eqns. 6, 7, and 5 (this is very similar to the treatment in Eq. 8).

We tested these predictions of Equation 9 versus the mean termination time of our DSN simulation over 10 repetitions in each of 43 randomly chosen environments from the design space $[2 \leq r \leq 10, 0 \leq o \leq r, 1 \leq \sqrt{A} \leq 5, 1 \leq N \leq 10]$. The durations of all tasks were set at 1 time unit, as were the duration of information gathering and communication actions; we will demonstrate and discuss the effect of this assumption later in the paper. We used the simulation validation statistic suggested by Kleijnen [9] (where $\hat{y} =$ the predicted output by the analytical model and $y =$ the output of the simulation):

$$z = \frac{y - \hat{y}}{(\text{Var}(y) + \text{Var}(\hat{y}))^{1/2}} \tag{10}$$

where $\text{Var}(\hat{y})$ is the predicted variance.[6] The result z can then be tested for significance against the standard normal tables. In each case, we were unable to reject the null hypothesis that the actual mean termination equals the predicted mean termination at the $\alpha = 0.05$ level. For non-statisticians: this is a good thing. The null hypothesis is that our prediction is the same as the actual value, we did not wish to reject it, and we did not. However, such a test has problems since there are other possible reasons which might prevent us from rejecting the null hypothesis. The best approach to avoiding this problem is to report not only the alpha level of the test but also the test's power. With any statistical hypothesis test there are four possible outcomes: we are unable to reject the null hypothesis when it is in reality true , we are unable to reject the null hypothesis when it is in reality false (called a *Type II error*), we reject the null hypothesis when it is in reality false, and we reject the null hypothesis when it is in reality true (called a *Type I error*). The α level is essentially the probability of a Type I error. The *power* of a test is $(1 - \beta)$, where β is essentially the probability of a Type II error (accepting a false null hypothesis). Unfortunately, estimating the power of a test is often difficult, and requires detailed knowledge of the distribution of the test statistic under the condition that the alternative hypothesis is true (this is turn usually requires the computation of the power for several likely alternative hypotheses). Future work will involve the computation of the power for this statistic, or the development of a new test based on other possible statistics that have better-understood power characteristics. Figure 6 shows the mean of 10 repetitions in each environment versus the expected value and its likelihood intervals. Thus the analytical model describes the implementation fairly well, and we could use the analytical model to design a good static organization for a given environment, using standard heuristic optimization techniques such as simulated annealing.

4.2 Control Costs

The control algorithm presented above is simple and not necessarily optimal. By communicating only when there is no local work to be done, the heaviest-loaded agent gives up the chance for other agents to do the high-level composition in a task group by incrementally transmitting each result (or set of results on a single task group) as it is completed—a maximum potential savings of $(\hat{N} - 1)(a - 1)\mathbf{d}_0(\text{VTM}) + (\hat{N} - 1)\mathbf{d}_0(\text{VCM})$. However, this needs to be balanced with the cost of multiple communication actions, which is $(\hat{N} - 1)\mathbf{d}_0(C)$. Thus the question of 'when to communicate' (when to incrementally transmit partial results) rests directly on the cost of communication relative to $(a - 1)\mathbf{d}_0(\text{VTM}) + \mathbf{d}_0(\text{VCM})$ (which depends on both the basic method durations and the agents' organizational structure).

This simple control algorithm can be analyzed easily, unlike many other systems where control costs are ignored. If we view the cost of control as the time spent by an agent when *not* performing an action

[6] The predicted variance of Equation 9 can be easily derived from the statistical identity $\text{Var}(x) = E[x^2] - (E[x])^2$.

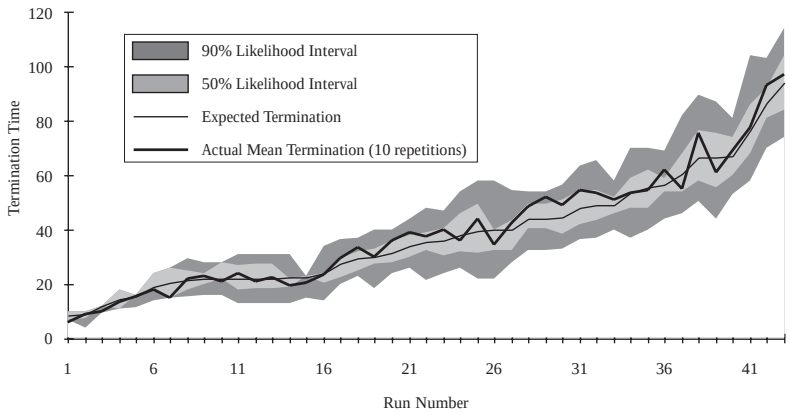

Figure 6: Actual system termination versus analytic expected value and analytically determined 50% and 90% likelihood intervals. Runs arbitrarily ordered by expected termination time.

(executing a method, information gathering, communication), then our algorithm runs in constant time between actions except for the two tests [*get set of currently executable methods*] and [*get set of methods still waiting*]. Each of these in the worst case requires a constant-time test of each element of the full task structure, which is of size $O(\eta r)$. Thus we see how the control costs, too, are related to organizational structure.

4.3 Analyzing Dynamic Organizations

In the dynamic organizational case, agents are not limited to the original organization and initial distribution of data. Agents can re-organize by changing the initial static boundaries (changing responsibilities in the overlapping areas), or by shipping raw data to other agents for processing (load balancing). We will assume in this section that the agents do not communicate about the current local state of problem solving directly (see Section 5 on using meta-level communication). A clearer distinction is that in the dynamic organization each agent makes its initial decision (about changing boundaries or shipping raw data) without access to non-local information. In the meta-level communication situation discussed later the agent has access to both its local information and a summary of the local state of other agents. In either case the decision to dynamically change the organization is made only once, at the start of an episode.

In the case of reorganized overlapping areas, agents may shift the initial static boundaries by sending a (very short) message to overlapping agents, telling the other agents to do all the work in the overlapping areas. The effect at the local agent is to change its effective range parameter from its static value of $r' = r - o/2$ to some value r'' where $r - o/2 \geq r'' \geq r - o$, changing the first two terms of Equation 9, and adding a communication action to indicate the shift and an extra information gathering action to receive the results. Section 4.4 discusses a particular implementation of this idea that chooses the partition of the overlapping area that best reduces expected differences between agent's loads and averages competing desired partitions from multiple agents.

In the second case, an agent communicates some proportion ρ of its initial data to a second agent, who does the associated work and communicates the results back. Instead of altering the effective range and overlap, this method directly reduces the first two terms of Equation 9 by the proportion ρ. The proportion ρ can be chosen dynamically in a way similar to that of choosing where to partition the overlap between agents (Section 4.4).

Whether or not a dynamic reorganization is useful is a function of both the agent's local utility and also the load at the other agent. We will again be concentrating on the agent with the heaviest load.

Looking first at the local utility, to do local work under the initial static organization with n task groups, the heaviest loaded agent will take time:

$$S'\mathbf{d}_0(\text{VLM}) + (S' - n)\mathbf{d}_0(\text{VTM}) \tag{11}$$

When the static boundary is shifted before any processing is done, the agent will take time

$$\mathbf{d}_0(C_{\text{short}}) + S''\mathbf{d}_0(\text{VLM}) + (S'' - n)\mathbf{d}_0(\text{VTM}) + \mathbf{d}_0(I) \tag{12}$$

to do the same work, where C_{short} is a very short communication action which is potentially much cheaper than the result communications mentioned previously, and S'' is calculated using r''. When balancing the load directly, local actions will take time

$$\mathbf{d}_0(C_{\text{long}}) + \rho S'\mathbf{d}_0(\text{VLM}) + \rho(S' - n)\mathbf{d}_0(\text{VTM}) + \mathbf{d}_0(I) \tag{13}$$

where $\mathbf{d}_0(C_{\text{long}})$ is potentially much more expensive than the communication actions mentioned earlier (since it involves sending a large amount of raw data). If the other agent had no work to do, a simple comparison between these three equations would be a sufficient design rule for deciding between static and either dynamic organization.

Of course, we cannot assume that the other agent is not busy; the best we can do *a priori* (without meta-level communication during a particular episode) is to assume the other agent has the *average* amount of work to do. We can derive *a priori* estimates for the average initial work at another agent from Equation 9 by replacing the probability function of the max order statistic $g_{a,N,p}(s)$ with the simple binomial probability function $b_{N,p}(s)$. Therefore without any meta-level communication, a system of agents could choose intelligently between static, dynamic overlap reorganization, and dynamic load balancing given these constraints.

4.4 Dynamic Coordination Algorithm for Reorganization

This section describes a particular implementation of the general idea described earlier of dynamically reorganizing the partitions between agents for the DSN simulation. This implementation will keep each agent's area of responsibility rectangular, and relaxes competing constraints from other agents quickly and associatively (the order of message arrival does not affect the eventual outcome). To do this, the message sent by an agent requests the movement of the four *corridors* surrounding an agent. The northern corridor of Agent 1, for example, is the northern agent organizational responsibility boundary shared by every agent in the same *row* as Agent 1. As can be seen in Figure 7, a 3x3 organization has four corridors (between rows 1 and 2, 2 and 3, and between columns 1 and 2, 2 and 3).

The coordination algorithm described here works with the local scheduling algorithm described earlier in Section 4.1. This is consistent with our view of coordination as a *modulating* behavior [4]. The only modification to the local scheduler is that we prevent it from scheduling local method execution actions until our initial communications are completed (the *initial* and *reception* phases, described below).

The coordination algorithm is then as follows. During the *initial* phase the local scheduler schedules the initial information gathering action, and we proceed to the second phase, *reception*. In the second phase we use the local information to decide what organizational design to use, and the parameter values for the design we choose. To do this we calculate the duration of our (known) local work (Eq. 11), and then estimate that duration under the alternative organizations (dynamic reorganization or load-balancing). When a parameter needs to be estimated, we do so to minimize the absolute expected difference between the amount of work to be done locally and the amount of work done at the remote agent that is impacted the most by the proposed change.

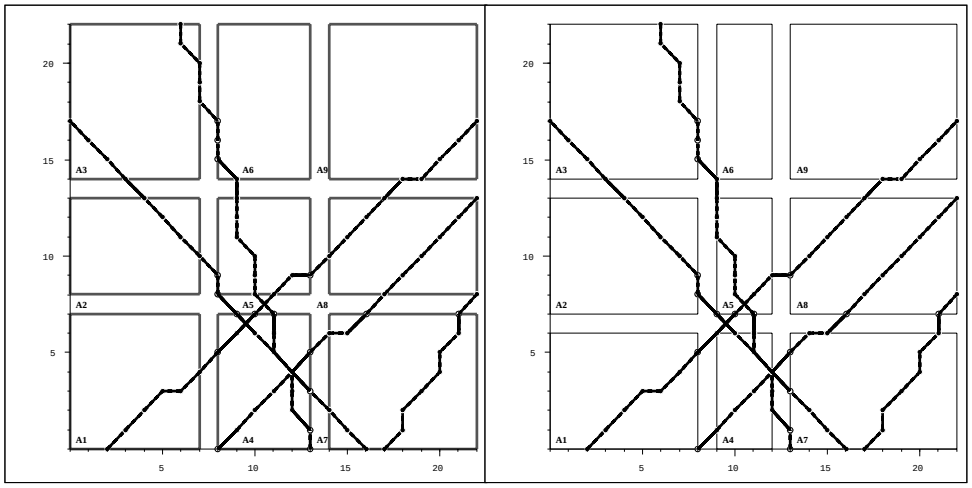

Figure 7: On the left is a 3x3 static organization, on the right is the dynamic reorganization result after agents 3, 4, 5 and 7 attempt to reduce their areas of responsibility by one unit. In this example the corridors running North to South have been moved closer by two units to reduce the load on agents 4, 5, and 6 in the second column.

For example, when dynamically restructuring, if the overlap between agents is more than 2 units, we have a choice of reducing the area an agent is responsible for by more than 1 unit (this is the organizational design parameter ρ in question). To decide on the proper reduction (if any), each agent computes its known local work $\hat{W}(\rho)$ using Eq. 11 with the actual (not estimated) S' and N computed assuming the agent's area is reduced by ρ. Then the agent finds the value of ρ that minimizes the difference in its known local work $\hat{W}(r-\rho)$ and the *average* work $\bar{W}(r+\rho)$ at the other agent:

$$
\begin{aligned}
S(r,s,N) &= (rs + \frac{r}{2}(N-s)) \\
W(r,s,N) &= S(r,s,N)\mathbf{d}_0(\text{VLM}) + (S(r,s,N)-N)\mathbf{d}_0(\text{VTM}) \\
E[\hat{W}(r)] &= \sum_{N=0}^{n}\sum_{s=0}^{N} g_{A,n,\frac{a}{A}}(N) g_{a,N,p}(s) W(r,s,N) \qquad (14) \\
E[\bar{W}(r)] &= \sum_{N=0}^{n}\sum_{s=0}^{N} b_{n,\frac{a}{A}}(N) b_{N,p}(s) W(r,s,N) \qquad (15)
\end{aligned}
$$

Equation 15 is just a restatement of Eqn. 14 (itself derived from Eqns. 5, 6, 7, and 11 for the case of the average, not maximally loaded, agent (thus the use of b, the binomial probability function rather than g, the max order statistic p.f.).

If $\rho = 0$, then the agent will not restructure. If $\rho \neq 0$, then the agent sends a message to all affected agents requesting a reduction of amount ρ in each corridor (north, east, south, and west). The agent sets its current area of interest to include only the unique (non-overlapping) portion of its area (if any), and enters the *unique-processing* phase. During this phase the regular local scheduler described earlier controls method execution actions.

When no more methods unique to this agent can be executed, the coordination algorithm checks the current time. If enough time has passed for the messages from other agents (if any) to arrive (this depends on the communication delays in the system), the coordination algorithm schedules an information-gathering action to retrieve the messages. Note that every agent may reach this point at a different time; agents with a large amount of unique local work may take some time, agents with no work at all will wait idle for the length of communication delay time in the system.

At this point each agent will relax its borders according to the wishes of the other agents. The relaxation algorithm we have chosen is fairly simple and straightforward, though several similar choices are possible. The algorithm is symmetric with respect to the four corridors surrounding the agent, so we will just discuss the relaxation of the northern corridor. There will be a set of messages about that corridor, some wanting it moved up by some amount and some wanting it moved down by some amount—we will consider these as positive and negative votes of some magnitude. The relaxation algorithm sums the votes, and returns the sum unless it is larger than the maximum vote or smaller than the minimum vote, in which case the max or min is returned, respectively. At this point the agent enters the final *normal processing* phase, and the local scheduler schedules all further actions as described earlier.

4.5 Analyzing the Dynamic Restructuring Algorithm

As we did in Section 4.1, we can develop an expression for the termination time of any episode where the agents follow this algorithm. To do so, we start with the basic termination time given all of the random variables. This equation is derived from Eqns. 14 and 15:

$$T(r, \hat{S}, \hat{N}, \bar{s}, \bar{N}) = \max[W(r - \rho, \hat{S}, \hat{N}), W(r + \rho, \bar{s}, \bar{N})] \tag{16}$$

where ρ is computed as described in the last section using the given values of $(r, \hat{S}, \hat{N}, \bar{s}, \bar{N})$. To turn this into a predictive formula, we then use the expressions for the probabilities of the terms $\hat{S}$, $\hat{N}$, $\bar{s}$, and $\bar{N}$ (from Eqns. 6, 7, and 5). For example, we can produce an expression for the expected termination of the algorithm:

$$\sum_{\hat{N}=0}^{n} \sum_{\hat{S}=0}^{\hat{N}} \sum_{\bar{N}=0}^{n} \sum_{\bar{s}=0}^{\bar{N}} g_{A,n,\frac{o}{A}}(\hat{N}) g_{a,\hat{N},0.5}(s) b_{n,\frac{o}{A}}(\bar{N}) b_{\bar{N},0.5}(\bar{s}) T(r, \hat{S}, \hat{N}, \bar{s}, \bar{N}) \tag{17}$$

We tested the predictions of Equation 17 versus the mean termination time of our DSN simulation over 10 repetitions in each of 10 randomly chosen environments. The durations of all tasks were set at 1 time unit, as were the duration of information gathering and communication actions, with the exception of the 4 environments described in the next section. Using the same validation statistic as before (Eq. 10) in each case we were unable to reject the null hypothesis that the actual mean termination equals the predicted mean termination at the $\alpha = 0.05$ level.[7]

4.5.1 Increasing Task Durations

Figure 8 compares the termination of static and dynamic restructuring organizations on identical episodes in four different environments. From left to right, the environments were $[A = 9, r = 9, o = 9, n = 7]$, $[A = 4, r = 9, o = 3, n = 5]$, $[A = 16, r = 8, o = 5, n = 4]$, $[A = 9, r = 10, o = 6, n = 7]$. Ten different episodes were generated for each environment. In order to see the benefits of dynamic restructuring more clearly, we chose task durations for each environment similar to those in the DVMT: $\mathbf{d}_0(\text{VLM}) = 6, \mathbf{d}_0(\text{VTM}) = 2\mathbf{d}_0(\text{VCM}) = 2$.[8] Note that the dynamic organization often does significantly better than the static organization, and rarely does much worse—remember that is many particular episodes that the dynamically organized agents will decide to keep the static organization, although they pay a constant overhead when they keep the static organization (one extra communication action and one extra information gathering action, given that the time for a message to reach all agents is no longer than the communication action time).

[7] The same caveats we discussed earlier still apply.

[8] The idea being that the VLM methods correspond to lowest three DVMT KSIs as a group, and the other methods correspond to single DVMT KSIs, and that a KSI has twice the duration of a communication action.

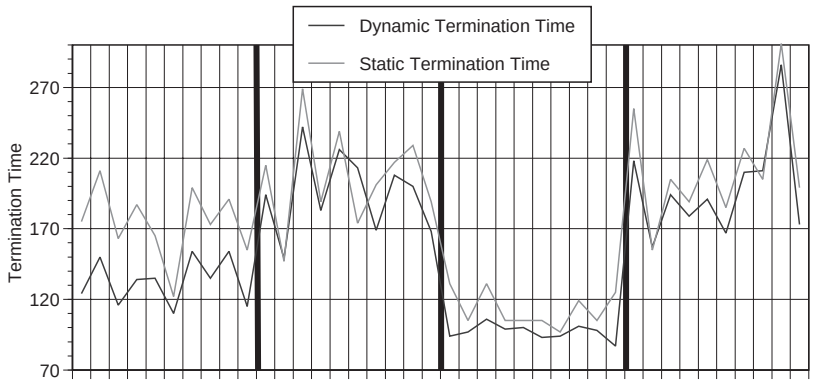

Figure 8: Paired-response comparison of the termination of static and dynamic systems in four different environments (ten episodes in each). Task durations are set to simulate the DVMT (see text).

5 Using Meta-Level Communication

For some environments $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$ one of the three organizational choices may be clearly better in the long run, but for most environments the choice is not so clear given the variance in system performance. The choice that optimizes performance over the long run is often not optimal in any particular episode. Taking the essential equations for local work in Section 4.3, we can compute likelihood intervals on the predicted performance of an organization under each of the three coordination regimes by combining the local likelihood interval on the expected load of the heaviest loaded agent, and the likelihood interval on the average agent load. These results, for the 50% likelihood interval, are shown in Figures 9 and 10. Again we have assumed that all execution, communication, and information gathering action durations have the same value (making communication relatively expensive). The first figure, Figure 9, highlights how the relationship between performance under a static organization and a dynamically load balanced organization changes as the number of agents increases. As expected, load balancing becomes more desirable as the number of agents increases (in relation to the average number of tracks): when there are many agents, the average agent load becomes very low, which offsets the cost of transferring tasks. In this figure the performance difference between static and overlap reorganization remains nearly constant relative to the number of agents.

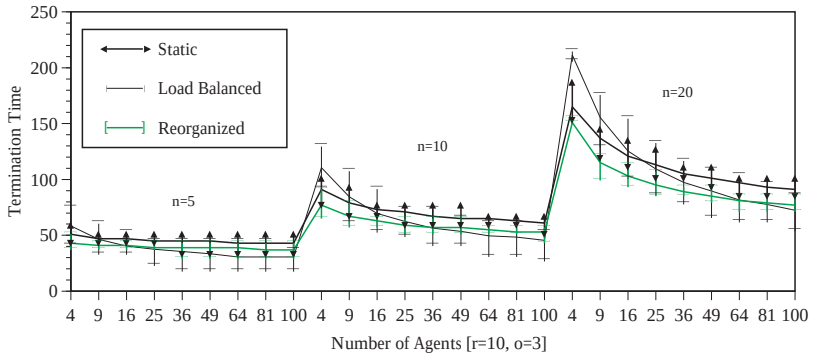

Figure 9: 50% likelihood intervals on the expected termination of a system under three coordination regimes, different numbers of agents, and three values of n

The second, Figure 10, points out how dynamically reorganizing the overlap area increases the performance over static organization as the amount of overlap increases. For this graph we assumed that the agents would shrink their entire area of responsibility (as opposed to minimizing the difference in

maximum versus average work as described in Section 4.4). This graph shows the need for dynamically calculating the shrinkage parameter (ρ) especially at high levels of overlap (note how the dynamically reorganized organization is predicted to do worse at high levels of overlap in the $n = 20$ portion). The expected performance difference between the static organization and load balancing remains relatively constant across changing values of o. In both figures we have let $\mathbf{d}_0(C) = \mathbf{d}_0(C_{\text{short}}) = \mathbf{d}_0(C_{\text{long}})$; increasing the differences in these values will move the corresponding curves directly up or down.

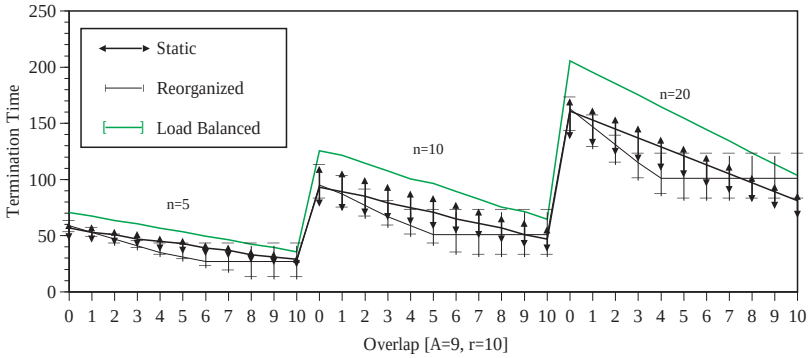

Figure 10: 50% likelihood intervals on the expected termination of a system under three control regimes, different overlaps, and three values of n. Likelihood intervals on the load balanced line are omitted for clarity.

The next series of figures demonstrate the effect of the ratio of computation duration to communication duration. This and subsequent figures assume that the dynamic restructuring shrinkage parameter ρ is set to minimize the difference between maximum and average local work as described in Section 4.4. Figure 11 shows how the expected value and 90% likelihood interval on system termination changes as the duration of a method execution action changes from equal to (1x) a communication action to 10 times (10x) that of a communication action. The task structure remains than of the DSN example described in Section 2.5. In Figure 11 we see a clear separation emerge between static and dynamic termination.

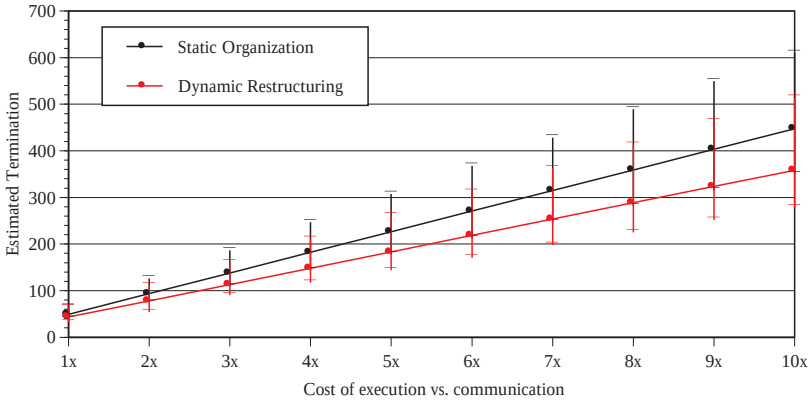

Figure 11: Effect of decreasing communication costs on expected termination under a static organization and dynamic restructuring (expected value and 90% likelihood interval, $A = 25, r = 9, o = 9, n = 7$).

These figures assume that the number of task groups n is known beforehand. The reason for this is to highlight the variance implicit in the organization, and minimize the influence of the external environment. Figure 12 shows how much extra variance is added when only the expected value of n, which is η, is known. We assume that the number of task groups n (in the DSN example, vehicle tracks) that occur during a particular episode has a Poisson distribution with an expected value of η.

The discrete probability function for the Poisson distribution, given in any statistics book, is then:

$$p_\eta(y) \quad = \quad \frac{\eta^y}{y!}e^{-y} \quad [\Pr[n = y]]$$

We can use this probability in conjunction with Eqns. 8, 11, and 15 to calculate the expected value, 50%, and 95% likelihood intervals on termination in the static or dynamic organizations. Note in Figure 12 both the large increase in variance when n is random (between the top bar and the third bar in the figure), and more importantly the small decrease in variance in the dynamic restructuring organization (between the first and the second bars). Note also that the mean termination time for the dynamic organization is slightly less than that for the static organization.

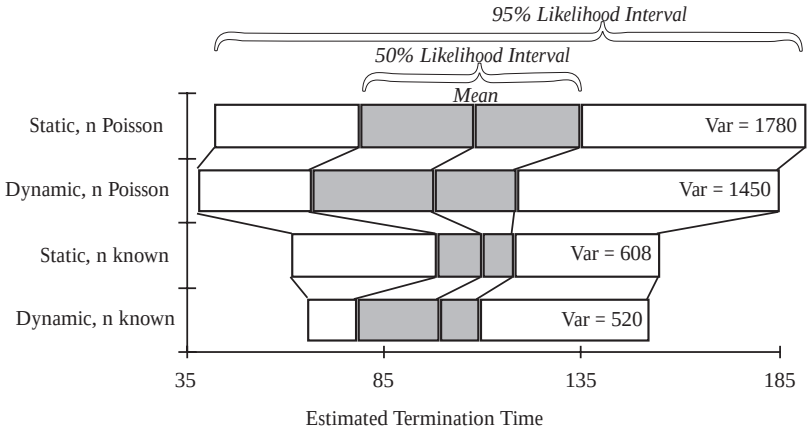

Figure 12: Demonstration of both the large increase in performance variance when the number of task groups n is a random variable, and the small decrease in variance with dynamic restructuring coordination $[A = 9, r = 9, o = 9]$. Where n is known, $n = 7$. Where n is a random variable, the expected value $\eta = 7$.

These figures bring us to the final point of this paper: often system performance can be improved significantly by dynamic reorganization, but it will rarely *always* be improved. Therefore, meta-level communication between agents about their local loads can, with a small communication cost, pinpoint the true costs and benefits of the various organizational structures, allowing an informed organizational decision to be made. Instead of an agent making a decision about restructuring or load balancing by assuming the *average* load, the agent will have the *actual* load for the neighboring agents. As we said in the introduction, the proper organization is often one that exploits *information that resolves uncertainties* about the current environment as it becomes available to the agents, allowing the agents to then create the most efficient organization for the situation.

6 Conclusions

The results of this paper can be looked at from three points of view. From the practitioner's viewpoint, the analysis presented here resulted in a set of design equations that can be used directly to optimize the performance of a simple DSN, or explore the design space given some model of how expensive agents are and what bounds (mean, median, 90% quantile) on their performance are required. Several of the simplifying assumptions we used, such as constant communication and information gathering costs, can be easily replaced with submodels chosen by the designer. From the viewpoint of the distributed AI community, we have returned to look at the some of the problems first studied by Durfee, Lesser, and Corkill[7]. They concluded that "Our intent is to show that overly specialized organizational structures allow effective network performance in particular problem-solving situations, but that no

such organization is appropriate in all situations." In this paper we reach the same abstract conclusion, but also show precisely what the effect is of a particular organizational structure (characterized by both its structural components and its coordination algorithm) in an environment (characterized by the structure and frequency of its tasks) in a clear way that not only allows us to predict performance but to explain it. The technique of using binomial approximations should also prove useful in different domains. From the viewpoint of the general research community this paper presents a methodology for answering questions about the design of a system by analysis and simulation. In such a methodology, the observation of particular phenomena in a complex system (the DVMT) leads to the building and verification of general models that predict and explain such phenomena.

In the short term, this work leads to the explanation of other interesting distributed problem solving phenomena displayed in [7]. The addition of noise at DSN sensors leads to the necessity of more complex coordination with the introduction of more complex task interrelationships (such as *facilitation*[4]). The addition of correlated noise in the environment can then cause these new, more complex coordination mechanisms to break down, producing the phenomenon recognized as *distraction*. In the long term, we are working towards a complete characterization of generalized partial global planning[3] as a first step towards a theory of coordination in distributed problem solving. We and other researchers are also using our task structure characterization for the analysis and simulation of problems in real-time and parallel scheduling.

References

[1] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3):213–261, 1990.

[2] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, January 1983.

[3] Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1(2):319–346, June 1992.

[4] Keith S. Decker and Victor R. Lesser. Analyzing a quantitative coordination relationship. *Group Decision and Negotiation*, 2(3):195–217, 1993.

[5] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 217–224, Washington, July 1993.

[6] E. H. Durfee and T. A. Montgomery. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1363–1378, November 1991.

[7] 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.

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

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

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

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

[12] Jasmina Pavlin. Predicting the performance of distributed knowledge-based systems: A modeling approach. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 314–319, Washington, D.C., August 1983.

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

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

A Distribution of the Binomial Max Order Statistic

To expand on the formulae in Section 2.2, the amount of a single task group seen by an agent can be viewed as a random variable S with probability density function $f(s)$ and corresponding cumulative distribution function $F(s)$. Let a be the number of agents that initially see part of a single task group. By elementary statistics, the density of the max order statistic $S_{max} = \max(S_1, S_2, \ldots, S_a)$ is $g_a(s) = aF(s)^{a-1}f(s)$. The expected heaviest load then is $\int_0^r s g_a(s) ds$. In the DSN environment, S is discrete, and its probability function (determined empirically) is heavily weighted toward r (the maximum). To simplify the analysis, instead of letting S correspond to the number of subtasks in a single task group seen by an agent, we have a new random variable $\mathbf{S}$ equal 1 if the agent sees the maximum amount, and 0 otherwise. Now $\mathbf{S}$ has a Bernoulli (coin-tossing) distribution with parameter p corresponding to the chance of an agent seeing the maximum amount r of a task group. $\mathbf{S}$ then corresponds to the number of times an agent sees the maximum r if the agent sees N task groups (tracks). $\mathbf{S}$ has a binomial distribution with parameters p and N. Now if a agents see N task groups each, what is the distribution of $\hat{\mathbf{S}} = \max(\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_a)$? This is the max order statistic for the binomial distribution, and because it is discrete, can be easily derived. The probability function and cumulative distribution function for the binomial distribution are:

$$\begin{array}{rcll} b_{N,p}(s) & = & \binom{N}{s} p^s (1-p)^{N-s} & [\Pr[\mathbf{S} = s]] \\ B_{N,p}(s) & = & \sum_{x=0}^{s} b_{N,p}(x) & [\Pr[\mathbf{S} \leq s]] \end{array}$$

If we assume each track is independent of the others, we can derive the cumulative distribution function:

$$\begin{array}{rcl} \Pr[\hat{\mathbf{S}} \leq s] & = & \Pr[\max(\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_a) \leq s] \\ & = & \Pr[\mathbf{S}_1 \leq s] \Pr[\mathbf{S}_2 \leq s] \cdots \Pr[\mathbf{S}_a \leq s] \\ & = & B_{N,p}(s)^a \end{array}$$

The probability function $g_{a,N,p}(s) = \Pr[\hat{\mathbf{S}} = s]$ is then:

$$g_{a,N,p}(s) = B_{N,p}(s)^a - B_{N,p}(s-1)^a$$