A One-shot Dynamic Coordination Algorithm for Distributed Sensor Networks *

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

Abstract

This paper presents a simple, fast coordination algorithm for the dynamic reorganization of agents in a distributed sensor network. Dynamic reorganization is a technique for adapting to the current local problem-solving situation that can both increase expected system performance and decrease the variance in performance. We compare our dynamic organization algorithm to a static algorithm with lower overhead. 'One-shot' refers to the fact that the algorithm only uses one meta-level communication action.

The other theme of this paper is our methodology for analyzing complex control and coordination issues without resorting to a handful of single-instance examples. Using a general model that we have developed of distributed sensor network environments [Decker and Lesser, 1993a], we present probabilistic performance bounds for our algorithm given any number of agents in any environment that fits our assumptions. This model also allows us to predict exactly in what situations and environments the performance benefits of dynamic reorganization outweigh the overhead.

Introduction

The distributed sensor network (DSN) domain has been a fertile source of examples for the study of cooperative distributed problem solving [Carver and Lesser, 1991; Durfee *et al.*, 1987; Lesser, 1991]. A key result of the early work in DSNs has been the demonstration of the advantages available to groups of agents that communicate about their current problem solving situation. Algorithms for coordinating DSN agents can be divided into two classes on the basis of their communication patterns: *static* algorithms communicate only the results of tasks and no other information about the local state of problem solving; *dynamic* algorithms use meta-level communication about their local problem-solving states to adapt to a situation (examples of this include partial global planning [Durfee and Lesser, 1991] and many negotiation algorithms).

This paper presents a simple one-shot dynamic algorithm for reorganizing agents' areas of responsibility in response to a particular DSN problem-solving episode, and analyzes the agents' resulting behaviors. The class of one-shot dynamic algorithms is interesting because it is the class of coordination algorithms with the lowest communication overhead (only one meta-level communication action) that still allows agents to adapt to a particular situation during problem solving. This low overhead allows dynamic algorithms to be used in environments where the higher costs of multiple meta-level communications and negotiation are not warranted. The particular algorithm presented here, called *one-shot dynamic reorganization*, allows agents to very quickly resolve to a new organization by limiting each agents' area of responsibility to a rectangular shape.

We will analyze the performance of the dynamic algorithm relative to a static one, the effect of some environmental assumptions such as the cost of communication, and the reduction of variance in performance caused by dynamic adaptation (which can be exploited by real-time scheduling algorithms[Decker *et al.*, 1990; Garvey and Lesser, 1993]). The model we will use for our analysis[Decker and Lesser, 1993a] grew out of the set of single instance examples of distributed sensor network (DSN) problems presented in [Durfee *et al.*, 1987]. 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 experimental result[Cohen, 1991] that our modeling and analysis method was designed to address—we wish to *explain* this result, and better yet, to *predict* which algorithm will do the best in each situation. We wish to identify the characteristics of the DSN environment, or the organization of the agents, that cause one algorithm to outperform another. Our approach relies on a statistical characterization of an environment rather than single instance examples.

The first section will summarize our model of DSN environments and the results of our previous analysis of static coordination algorithms. The next section will discuss dynamic coordination in general, and then we will present the

*This work was supported by ARPA under ONR contract N00014-92-J-1698, ONR 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.

one-shot dynamic reorganization algorithm and confidence intervals on its performance. Finally, we will present our relative performance, communication cost, and variance reduction results.

A Simple Model of DSN Environments

Our task environment model assumes that several independent *task groups* arrive at multiple physical locations over a period of time called an *episode*. In a distributed sensor network (DSN) episode a single vehicle track corresponds to a task group. The movements of several independent vehicles will be detected over a period of time (the episode) by one or more distinct sensors, where each sensor is associated with an agent. For example, on the left side of Figure 2 we see a single episode with 5 vehicle tracks and the outlines of 9 non-overlapping sensor areas.

The performance of agents in such an environment will be based on how long it takes them to process all the task groups (vehicle tracks), 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 (which portions of which vehicle tracks) are available to which agents and at what cost (an agent can get information from its own sensor more cheaply than by requesting information from another agent's sensor). Usually DSN agents have overlapping sensors, and either agent can potentially work on data that occurs in the overlapping area 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 (vehicle tracks), 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 simple 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 physical *range* of each agent's sensor and the physical *overlap* between agent sensors[2], and $\mathcal{T}$ specifies the homogeneous task group structure (an example of the task group structure is shown in Figure 1). 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 a Poisson distribution with an expected value of η.

In a DSN episode, each vehicle track is modeled as a task group. The structure of each task group is based loosely on the processing done by a particular DSN, the Distributed Vehicle Monitoring Testbed (DVMT)[Lesser and Corkill, 1983]. Our simple model is that each task group $\mathcal{T}_i$ is associated with a track of length l_i and has the following structure: (l_i) vehicle location methods (VLM's) that represent processing raw signal data at a single location resulting in 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 relationships exist between each method at one level and the appropriate method at the next level as shown in Figure 1—two VLMs precede each VTM, and all VTM'S precede the lone VCM. Besides executing methods, agents may also execute communication actions and information gathering actions (such as getting data from the sensors or communications from other agents). We assume that communication and information gathering are no more time consuming that problem solving computation (in practice they are often much quicker—see the analysis in the final section of this paper). A more complete description of our modeling framework, which can handle much more complexity than this simple model illustrates, can be found in [Decker and Lesser, 1993b], in this volume.

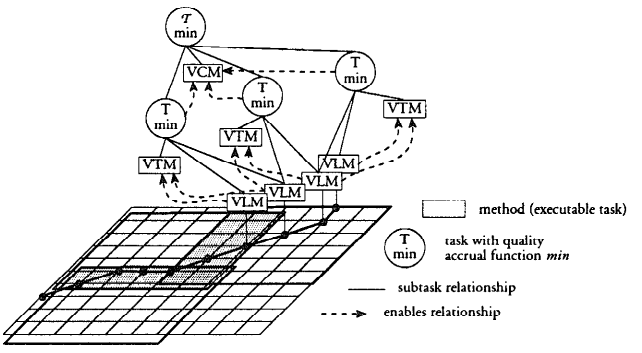

Figure 1: Task structure associated with a single vehicle track.

Later analysis in this paper will be verified by comparing model-based predictions against a DSN simulation, which generates and simulates the execution of arbitrary environments $\mathcal{D}$. 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 (for example, see Figure 2). 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

[1] In general there are usually more complex interrelationships between subtasks that affect scheduling decisions, such as *facilitation* [Decker and Lesser, 1991].

[2] We will also assume the sensors start in a square geometry, i.e, 4 agents in a 2 × 2 square, 25 agents arranged 5 × 5.

along with the locations traveled by each vehicle to determine what part of each task group is initially available to each agent. The analysis summaries in the next section were also verified by simulation; please see [Decker and Lesser, 1993a] for the derivation and verification of these early results; later in this paper we will discuss our verification methodology.

Environmental Analysis Summary

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* number of methods to be executed, but since the focus of our analysis is the termination of problem solving, we need to examine the *expected maximum* size ($\hat{S}$) of an initial data set seen by some agent as a random variable. This basic environmental analysis result is taken from the derivation in [Decker and Lesser, 1993a]; it is equivalent to the expected number of VLM methods seen by the maximally loaded agent in an episode. This value also depends on the expected number of task groups ($\hat{N}$) seen by that same agent, another random variable. For example, the observed value of the random variable ($\hat{S}$) in the particular episode shown on the left side of Figure 2 is 22 sensed data points at agent $A4$, and the number of task groups (tracks) seen by that same agent is 4 ($\hat{N} = 4$). The average number of agents that see a single track (which we represent by the variable a) is 3.8 in the the same episode.

If the system of agents as a whole sees n total task groups, then the discrete probability distributions of $\hat{N}$ and $\hat{S}$ are:

$$\Pr[\hat{N} = N|n] = g_{A,n,\frac{a}{A}}(N) \quad (1)$$

$$\Pr[\hat{\mathbf{S}} = s|\hat{N} = N] = g_{a,N,0.5}(s) \quad (2)$$

$$\hat{S} = (r\hat{\mathbf{S}} + (r/2)(N - \hat{\mathbf{S}})) \quad (3)$$

The function $g_{a,n,p}(s)$ is called the max binomial order statistic, and is defined in terms of the simple binomial probability function $b_{n,p}(s)$ as follows:

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

The variable a represents the average number of agents that see a single task group, and is estimated as follows:

$$a = A^{(\frac{r^2+o^2}{2r^2})} \quad (4)$$

These results, derived and verified in [Decker and Lesser, 1993a], will be used in the following sections within formulae for the predicted performance of coordination algorithms.

Static Coordination Algorithm Analysis summary

In our static algorithm, agents always divide up the overlapping sensor areas evenly between themselves so that they do not do redundant work, and never have to communicate about their areas of responsibility. The total time until termination for an agent receiving the maximum 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. Because this agent has the *maximum* amount of initial data, it will not finish any more quickly than any other agent and therefore we can assume it will not have to wait for the results of other agents. The termination time of this agent (and therefore the termination time of the entire statically organized system) can be computed from the task structure shown earlier, and a duration function $d_0(M)$ that returns the duration of method M:

$$\begin{aligned} T_{\text{static}} = \\ & \hat{S}\mathbf{d}_0(\text{VLM}) + (\hat{S} - \hat{N})\mathbf{d}_0(\text{VTM}) + \\ & (a-1)\hat{N}\mathbf{d}_0(\text{VTM}) + \hat{N}\mathbf{d}_0(\text{VCM}) + \\ & 2\mathbf{d}_0(I) + 2\mathbf{d}_0(C) \end{aligned} \quad (5)$$

We can use Eq. 5 as a predictor by combining it with the probabilities for the values of $\hat{S}$ and $\hat{N}$ given in Eqns. 3 and 1. Again, we refer the interested reader to [Decker and Lesser, 1993a] for derivations and verification.

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 with each other about the current local state of problem solving directly. A clearer distinction is that in a one-shot dynamic organization each agent makes its initial decision (about changing boundaries or shipping raw data) without access to non-local information. By contrast, in a full meta-level communication algorithm (like Partial Global Planning) the agent has access to both its local information and a summary of the local state of other agents. In this paper the decision to dynamically change the organization is made only once, at the start of an episode after the initial information-gathering action has occurred.

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 more than the default amount of 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 5, and adding a communication action to indicate the shift and an extra information gathering action to receive the results. The following section 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 load balancing case, an agent communicates some proportion ρ of its initial sensed 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 5 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 (see the next section).

Whether or not a dynamic reorganization is useful is a function of both the agent's local workload and also the load at the other agent. The random variable S again represents the number of initially sensed data points at an agent. Looking first at the local utility, to do local work under the initial static organization with n task groups, any agent will take time:

$$S\mathbf{d}_0(\text{VLM}) + (S-n)\mathbf{d}_0(\text{VTM}) \quad (6)$$

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

$$\mathbf{d}_0(C_{\text{short}}) + S''\mathbf{d}_0(\text{VLM}) + (S''-n)\mathbf{d}_0(\text{VTM}) + \mathbf{d}_0(I) \quad (7)$$

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 the new range r''. When balancing the load directly, local actions will take time

$$\mathbf{d}_0(C_{\text{long}}) + \rho S\mathbf{d}_0(\text{VLM}) + \rho(S-n)\mathbf{d}_0(\text{VTM}) + \mathbf{d}_0(I) \quad (8)$$

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 an extra 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 local work at another agent from Equation 6 by replacing S with $\bar{S}$, the probability distribution of the *average* initial sensed data at an agent. This probability distribution is the same as Eq. 3 except that we replace the probability function of the max order statistic $g_{a,N,p}(s)$ in Eq. 2 with the simple binomial probability function $b_{N,p}(s)$ (we'll restate the equations for our implementation in the next section). Therefore without any meta-level communication an agent can estimate how busy its neighbors are, and a system of agents could choose intelligently between static, dynamic overlap reorganization, and dynamic load balancing given these constraints.

One-shot 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 2, 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 static local scheduling algorithm described in [Decker and Lesser, 1993a]. This is consistent with our view of coordination as a *modulating* behavior [Decker and Lesser, 1991]. This simple local scheduler basically runs a loop that finds all local methods that can currently be executed that are also tied to data within the agent's static, non-overlapping sensor area, and then executes one. If no methods can be executed, the current set of results (if new) are broadcast to the other agents, and an information gathering action is executed to receive any new communication from other agents. The only modification to the local scheduler for the dynamic system 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 precede 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 under the default static organization (Eq. 6), 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.

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 W using Eq. 6 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 $W(r-\rho, S, N)$ and the *average* work $\bar{W}(r+\rho, \bar{S}, \bar{N})$ at the other agent:

$$\begin{aligned} S(r,s,N) &= (rs + \frac{r}{2}(N-s)) \\ W(r,s,N) &= S(r,s,N)\mathbf{d}_0(\text{VLM}) \\ &\quad +(S(r,s,N)-N)\mathbf{d}_0(\text{VTM}) \\ E[\bar{W}|r] &= \sum_{N=0}^{n}\sum_{s=0}^{N} b_{n,\frac{s}{A}}(N)b_{N,0.5}(s)W(r,s,N) \quad (9) \end{aligned}$$

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.

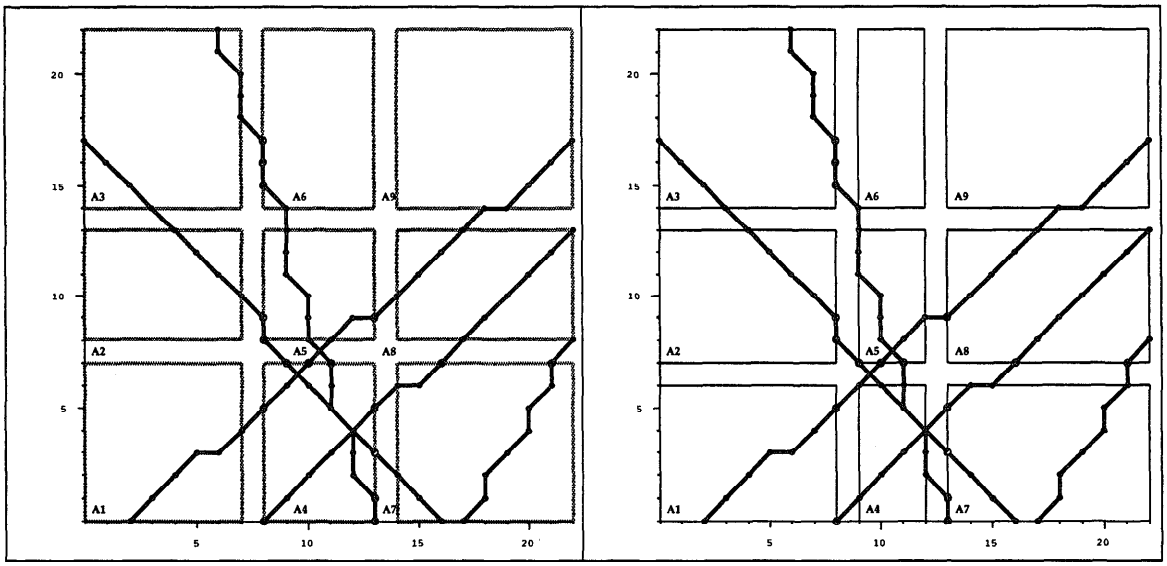

Figure 2: On the left is a 3x3 static organization, on the right is the dynamic reorganization result after agents 3, 4, 5 and 7 attempt to reduce their areas of responsibility by one unit. These are actual screen visualizations (originally in color) from our simulation package.

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. Competing votes of the same magnitude sum to zero, and cancel each other. The summed value becomes the final direction and amount of movement of that corridor. Figure 2 shows a particular example, where four agents each vote to reduce their areas of responsibility by one unit.

At this point the agent has a new static area that does not overlap with any other agent (since all agents will see the same information and follow the same decision procedure), and it enters the final *normal processing* phase, and the local scheduler schedules all further actions as described earlier (scheduling only tasks in the new, non-overlapping range).

To summarize: the agents first perform information gathering to discover the amount of local sensor data in the episode. They then use this local information to decide how to divide up the overlapping regions they share with other agents, using the assumption that the other agents have the average amount of local work to do. This parameter is then communicated to all the affected agents, and the agent works on data in its unique area (if any)—the part of the agent's sensor range that is never the subject of negotiation because only that agent can sense it. After completing and communicating this unique local work, the agent performs another information gathering action to receive the parameter values from the other agents, and a simple algorithm produces a compromise for the way the overlap will be divided up. The agents now proceed as in the static case until the end of the episode.

Analyzing the Dynamic Restructuring Algorithm

As we did in [Decker and Lesser, 1993a], 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:

$$T_{\text{dynamic}} = \max[T_{\text{static}}[r = r-\rho], T_{\text{static}}[r = r+\rho, \hat{S} = \bar{s}, \hat{N} = \bar{N})] \quad (10)$$

where ρ is computed as described in the last section using the 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. 3 and 1). For example, we can produce an expression for the expected

termination of the algorithm:

$$\sum_{\hat{N}=0}^{n}\sum_{\hat{S}=0}^{\hat{N}}\sum_{\bar{N}=0}^{n}\sum_{\bar{s}=0}^{\bar{N}} g_{A,n,\frac{a}{A}}(\hat{N})\cdot g_{a,\hat{N},0.5}(s)\cdot$$

$$b_{n,\frac{a}{A}}(\bar{N})\cdot b_{\bar{N},0.5}(\bar{s})\cdot T_{\text{dynamic}}[r,\hat{S},\hat{N},\bar{s},\bar{N}] \quad (11)$$

We tested the predictions of Equation 11 versus the mean termination time of our DSN simulation over 10 repetitions in each of 10 randomly chosen environments from the design space $[2 \le r \le 10, 0 \le o \le r, 1 \le \sqrt{A} \le 5, 1 \le N \le 10]$. 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 shown in the next section. We used the simulation validation statistic suggested by Kleijnen [Kleijnen, 1987] (where $\hat{y}$ = the predicted output by the analytical model and y = the output of the simulation):

$$z = \frac{y - \hat{y}}{(\text{Var}(y) + \text{Var}(\hat{y}))^{1/2}} \quad (12)$$

where $\text{Var}(\hat{y})$ is the predicted variance.[3] 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, thus validating our formal model.[4]

Increasing task durations

Figure 3 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: $\mathbf{d}_0(\text{VLM}) = 6$, $\mathbf{d}_0(\text{VTM}) = 2$, and $\mathbf{d}_0(\text{VCM}) = 2$.[5] Note that the dynamic organization often does significantly better than the static organization, and rarely does much worse—remember that in 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).

Comparative Analyses

The next figure demonstrates the effect of the ratio of computation duration to communication duration. This and

[3]The predicted variance of Equation 5 can be easily derived from the statistical identity $\text{Var}(x) = E[x^2] - (E[x])^2$.

[4]For non-statisticians: 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.

[5]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.

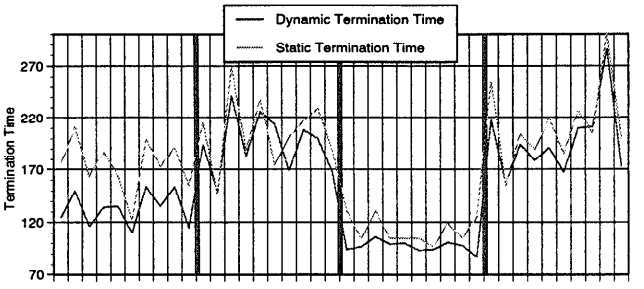

Figure 3: Paired-response comparison of the termination of static and dynamic systems in four different environments (ten episodes in each). Task durations are set to simulate the DVMT (see text).

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 the previous section. Figure 4 shows how the expected value and 50% confidence 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 that of the DSN example described in Section . In Figure 4 we see a clear separation emerge between static and dynamic termination. The important point to take from this example is not this particular answer, but that we can do this analysis for *any* environment $\mathcal{D}$.

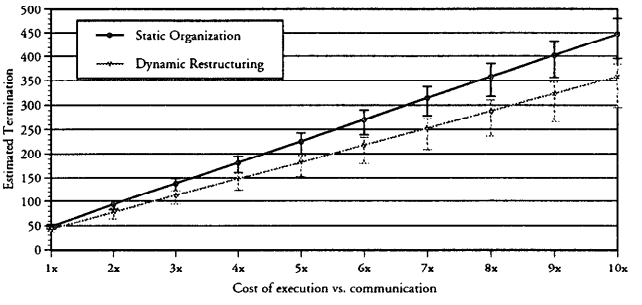

Figure 4: Predicted effect of decreasing communication costs on expected termination under a static organization and dynamic restructuring (expected value and 50% confidence interval, $A = 25, r = 9, o = 9, n = 7$).

Decreasing Performance Variance

The earlier figure assumes 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 5 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:

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

We can use this probability in conjunction with Eqns. 3, 6, and 9 to calculate the expected value, 50%, and 95% confidence intervals on termination in the static or dynamic organizations. An example of this calculation for one environment is shown in Figure 5. Note in Figure 5 both the large increase in variance when n is random, and the small decrease in variance in the dynamic restructuring organization. Note also that the mean termination time for the dynamic organization is less than that for the static organization.

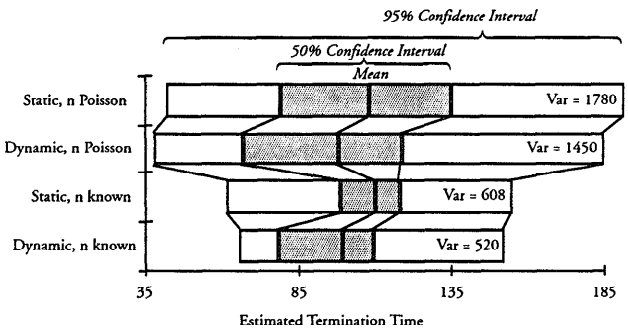

Figure 5: Demonstration of both the large increase in performance variance when the number of task groups n is a random variable, and the small decrease in variance with dynamic restructuring coordination $[A = 9, r = 22, o = 9]$. Where n is known, $n = 5$. Where n is a random variable, the expected value $\eta = 5$.

Conclusions

This paper described a one-shot dynamic coordination algorithm for reorganizing the areas of responsibility for a set of distributed sensor network agents. When performance is measured in terms of the time for a system of agents to terminate, the class of dynamic algorithms can often outperform static algorithms, and reduce the variance in performance (which is a useful characteristic for real-time scheduling [Decker *et al.*, 1990; Garvey and Lesser, 1993]). This paper presented a formula for the expected value (or variance, or confidence interval) of the termination time for a particular one-shot dynamic reorganization algorithm. It showed how this result can be used to predict whether the extra overhead of the dynamic algorithm was worthwhile compared to a static algorithm in a particular environment. Other questions were examined, such as the effect of decreasing communication costs, or increased uncertainty about the task environment.

We hope these results can be used directly by designers of DSNs to choose the number, organization, and control algorithms of agents for their particular environments, and that they inspire the DAI community to move beyond the development of ideas using single-instance examples.

We are currently analyzing a simple extension of this algorithm that uses two meta-level communication actions to provide agents with non-local information with which to make decisions about how to reorganize. We have observed only a small reduction in mean performance but a greater reduction in variance. Future work we have planned includes the analysis of a multi-stage communication, PGP-style dynamic coordination algorithm, and the use of our expanded model that includes faulty sensors and ghost tracks [Decker and Lesser, 1993b].

References

Carver, N. and Lesser, V.R. 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, 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, K.S. and Lesser, V.R. 1991. Analyzing a quantitative coordination relationship. Technical Report 91–83, University of Massachusetts. To appear, *Group Decision and Negotiation*, 1993.

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

Decker, K.S. and Lesser, V.R. 1993b. Quantitative modeling of complex computational task environments. In *Proc. of the Eleventh National Conference on Artificial Intelligence*, Washington.

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

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

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

Garvey, A.J. and Lesser, V.R. 1993. Design-to-time real-time scheduling. *IEEE Trans. on Systems, Man, and Cybernetics* 23(6). Special Issue on Scheduling, Planning, and Control.

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

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

Lesser, V.R. 1991. A retrospective view of FA/C distributed problem solving. *IEEE Trans. on Systems, Man, and Cybernetics* 21(6):1347–1363.