Computational and Mathematical Organization Theory 2:3 (1996): 219-246
©1996 Kluwer Academic Publishers, Manufactured in the Netherlands

Designing Tree-Structured Organizations for Computational Agents

YOUNG-PA SO AND EDMUND H. DURFEE
Department of EE and CS, University of Michigan, Ann Arbor, MI 48109, Email: frege@eecs.umich.edu, durfee@umich.edu, http://ai.eecs.umich.edu

Abstract

We describe a framework for defining the space of organization designs for computational agents, use our framework for analyzing the expected performance of a class of organizations, and describe how our analyses can be applied to predict performance for a distributed information gathering task. Our analysis specifically addresses the impact of the span of control (branching factor) in tree-structured hierarchical organizations on the response time of such organizations. We show quantitatively how the overall task size and granularity influence the design of the span of control for the organization, and that within the class of organizations considered the appropriate span of control is confined to a relatively narrow range. The performance predicted by our overall model correlates with the actual performance of a distributed organization for computer network monitoring. Consequently, we argue that our framework can support aspects of organizational self-design for computational agents, and might supply insights into the design of human organizations as well.

Introduction

A human or computer agent acting in a world populated by other agents should typically coordinate its actions with the actions of others, and this often requires models of what the other agents have done, are doing, and/or will do. In many cases, these models are mutually held. In fact, in multiagent systems that act in environments that reward cooperation, the agents might actively seek to establish a shared set of models. A shared organization structure, for example, provides agents with descriptions about their roles and responsibilities in the multiagent context, and thus represents guidelines for intelligent cooperation and communication among the agents.

An organization is thought of as a long-term commitment made by the agents to a particular way of jointly handling the cooperative tasks. In the situation for which the organization was devised, the agents should cooperate effectively and reliably. However, many interesting real-world organizations are situated in complex, often uncertain, and changing worlds. If the collection of agents is to be effective in such circumstances, it must adapt to such changes in at least one of a number of ways. One way is for the agents to actively change their environment such that it once again matches their organizational structure. A second way is for the agents to adopt, at the outset, an organizational structure that provides sufficient flexibility to the agents such that they can dynamically adapt to new circumstances within the same organizational structure. Finally, a third way is for the agents to reorganize themselves into an organizational structure that they have designed specifically for the expected situation.

Clearly, any system of agents that will survive for an extended time should be capable of adapting in all of these ways. It should be able to perform *task-environment (re)design* in cases when no organizational structure could possibly succeed or where the advantages of the current organization can continue to be realized. The task could be simplified, for example by relaxing some goals. Or the agents could maintain their environment in some way; a fascinating example is the process of sociogenesis, whereby individuals in a population differentiate to form an emergent organized colony with its own internal environment which buffers the individual members from changes in the outside world (Novak, 1982). However, organizations can be even more robust if the differentiated individuals have retained some degree of flexibility, such that they can adapt as a whole to changing circumstances while still remaining within the same organizational framework. In such organizations, individual *sophistication* becomes a more prized asset, since individuals must be able to undertake, within organizational guidelines, different tasks or roles as circumstances change, such as an underling taking on the duties of a missing manager. Pushed to the extreme, of course, even large degrees of latitude might not salvage an obsolete organizational structure, and the agents within it must be capable of more full-scale *organizational self-design*.

This paper is thus about organizations of communicating, autonomous, computational agents: how the organizations should be designed, and ultimately how agents can, among themselves, design organizations in order to jointly perform a set of tasks in an efficient, flexible, and reliable manner. While issues of task-environment (re)design are important, in this paper we concentrate on the other two approaches to using organizations to provide agents with models of others. Specifically, we present an analytical method, with empirical validation, about how to predict the impact that alternatives in the *span-of-control* (the branchiness of an organizational tree) will have on the speed of organizational problem solving. We begin in Section 1 with a framework for describing the organizational design problem, and explore some initial strategies for forming organizational designs depending on possible performance measures, task-environment parameters, and agent capabilities. In Section 2, we focus on a specific part of this framework to analyze the effects of different spans-of-control in tree-structured organizations. To validate our analysis, we map our model to a running application in Section 3, and confirm that our model's predictions correlate with experimental data. Next, we then briefly touch on techniques for making organizations more robust, including how agents could use the analytical techniques we have outlined to evaluate new candidate organizational designs as they engage in organizational self-design (Section 4). Finally, we summarize our results and outline future research directions in Section 5.

1. Models of Organizations

1.1 Organization-Theoretic Perspective

Human organizations are the subject of study for organization theory, and many models of various types of organizations have been generated (Scott, 1992). The variety of emphases within organization theory attests to the complexity of the phenomena in human organizations. Since the subject of study is about *human* organizations, various human factors and relations are a primary concern for organization theory. However, we agree with other researchers within both computer science and organization theory that many concepts and ideas can be shared between the two disciplines to better understand human organizations and to design more efficient and flexible distributed systems (Carley and Prietula, 1994; Cohen, 1986; Fox, 1981; Malone, 1987).

Of the various models of organizations in organization theory, we find two models particularly relevant and useful to our research on organizational self-design. The first is *contingency theory*. Lawrence and Lorsch (1967), who coined the label *contingency theory*, stressed the importance of organization-environment match in determining organizational performance.

The main theses of contingency theory are as follows (Scott, 1992):

1. There is no one best way to organize. There are no general principles applicable to organizations in all times and places.
2. Any way of organizing is not equally effective. Organizational structure is not irrelevant to organizational performance.
3. The best way to organize depends on the nature of the environment to which the organization relates. Organization design decisions depend—are contingent—on environmental conditions.

We are indebted to contingency theory in developing our model of organizational performance where we classify the various factors that affect the performance of the organization into two broad classes: *task-environmental factors* and *organizational factors*.

The second school of thought in organization theory relevant to our research is the *socio-technical systems* perspective (Trist, 1981). Influenced by human relations research, it proposed that the distinguishing feature of organizations is that they are both social and technical systems, and insisted that the two systems follow different "laws" and their relationship represents a "coupling of dissimilars" (Emery, 1959). Also, they proposed designing systems that emphasized discretionary behavior, internalized regulations, and work-group autonomy rather than top-down, manager controlled, technical bureaucracies (Scott, 1992).

Our model of intelligent cooperation via organizational self-design has a very similar spirit with the socio-technical systems approach since both are interested in endowing the agents themselves (whether human or machine) with the capability to determine the way they could work together more effectively in various task environments, and to understand exactly under which task environments organizations consisting of members with such capabilities are "good". Also, since our research stems from the distributed artificial intelligence perspective, we are aware of the many rich theories of machine intelligence (e.g. learning, reasoning, planning) which are quite different from the theories of computation and communication in distributed systems. So, in a sense, we are also attempting a "coupling of dissimilars" by investigating ways to design more intelligent distributed systems.

1.2 Distributed Artificial Intelligence Perspective

As the availability of processing systems, and specifically networked processing systems, increased, it became clear that there was power to harness in distributed computing systems if only the various systems could be coordinated to work together. Naturally, notions of organizing the systems came to the fore. Initially, the ideas were simply to decompose large tasks into component pieces that could be carried out in parallel, with little or no interaction among them. More tightly-coupled systems were also considered, with the idea that the interactions *between* the systems could be programmed much like the instructions within the individual systems. These approaches, along with others, have powered distributed computing into increasingly significant applications.

However, another feature of the intuitive notion of an organization, as understood (and populated) by people, was not well captured in these approaches. Human organizations are populated with amazingly versatile and sophisticated agents, at least compared to the computer programs that (still) generally populate distributed computer systems. Organizations, in human terms, do not dictate action at the "instruction" level, by and large; rather, they define roles and responsibilities for organizational participants, who are then expected to elaborate those into action depending on the current task and environmental demands.

This alternative perspective was introduced to computational systems within the field of distributed artificial intelligence, which has typically considered distributed systems where each individual in such systems possesses, in some degree, what might be labeled as intelligence.[1] With such capabilities, the individual agents in an organization have the capacity to make reasonable local decisions about what they should do given what they know about their tasks and the environment, as well as what they know about what others are likely to be doing. It is precisely this last bit of knowledge—about what others are likely to be doing—that is available thanks to knowing the organizational structure. That is, if a participant knows the roles and responsibilities of others, it can make more informed decisions about what to do locally and how to interact with them. Of course, this in turn means that the agents should abide by their designated roles, so each agent must be able to focus its local decisions toward fulfilling its responsibilities.

Corkill and Lesser (1983; Corkill, 1982), for example, developed computational representations for organizations in terms of *interest areas* for agents. An agent's interest areas would indicate what kinds of data-processing tasks it was willing and able to tackle, and to what degree. Faced with a variety of possible actions to take, therefore, an agent would be influenced (to a degree that could be modified by an experimenter) by how well those actions fit within its most preferred areas of interest. Moreover, because each agent knew the interest areas of the others, each could identify processing tasks, or information, that would be potentially of interest to them, focusing communication among them to eventually converge to a state where all the most important tasks were accomplished in a coherent, distributed manner.

[1]Practically speaking, this means that the programs running on the different computing processes capture, to some extent, the state-of-the-art techniques available within Artificial Intelligence.

Of course, the careful design of the interest areas was of primary importance to the success of this approach. Interest areas that were too narrow could mean that processing tasks became unevenly distributed among the agents, leading to longer delays until overall task completion. Moreover, if a subset of the agents failed to participate (they crashed or the network that connected them failed), then some tasks would be left unaccomplished and the overall task would fail to be completed. On the other hand, if interest areas were broadly defined so as to increase reliability and the chances that every agent would have something useful to do, then the situation could quickly deteriorate such that agents were duplicating effort and working at cross purposes. In addition, by making every agent more of a "generalist," communication among agents would explode because everyone would potentially be interested in everything!

In the work of Corkill and Lesser, and in subsequent work (Durfee, Lesser, and Corkill, 1987), a variety of organizational structures were experimented with and evaluated. Techniques were developed whereby agents working within some general organizational structure could also communicate about their more immediate plans, so as to avoid redundant activity and improve differentiation of effort. Performance of these systems was evaluated not only in terms of how quickly the organized agents would complete the overall task, but also in terms of the utilization of the agents, the overhead of communication and coordination among the agents, the tolerance of the organization to faulty communication channels, and the reliability of the organization to agent failures.

Qualitatively, the conclusions from all of this work closely resembled many of the theses of contingency theory, specifically, that different organizations are more or less effective under different task and environmental conditions. While the agreement with contingency theory is heartening, it is disappointing that the computational realization of organizations provides only domain-specific quantitative prescriptions about organizations to adopt under various conditions. Among the reasons for this difficulty is that the studies were embedded in complex task-environments, with agents whose abilities and behaviors were difficult to clearly characterize, and with performance measures that were not clearly articulated.

1.3 Our Model

We adopt the main assumptions of contingency theory in our model, but go further in the direction of making the concept of "best match" between organization and environment more precise. At the same time, we avoid tackling too many features of organizations, tasks, environments, and performance measures at once, in an effort to characterize the space of effective organizations within well-defined bounds. In our model, the organizational performance is *jointly determined* by the features of the organization and the features of the task environment. That is, we pursue a closed-form formula for each of the performance metrics for the organization. The formula, however, need not be expressible in algebraic form. It can be any algorithm that can compute the values of performance metric given the organizational and task-environmental factors as input.

Our model of *organization design* (OD) is directly related to our model of organization and the relevant ontologies surrounding it. In order to design an organization, we must first

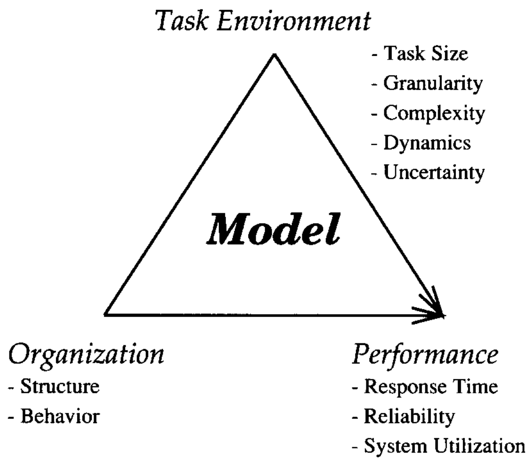

Figure 1.

know what is to be designed, that is, know what are the components and features of the organization we can select and combine. These we call *organizational factors.* Secondly, we need to know what is a good design or a bad design, that is, we need some criteria for evaluation. These we call the *performance metrics.* Thirdly, we need to know how an organization works in order to link the structure of elements and their features to the evaluation criteria. This we call the *organizational performance model.* If we see an organization as a closed system, these three things may suffice. However, many interesting organizations are open systems (Katz and Kahn, 1966), and interact with non-organizational external elements and processes. These we call *task-environmental* factors. Moreover, in many cases, these task-environmental factors affect the performance of the organization. Thus we require the knowledge of task-environmental factors in addition to the organizational factors for organization design. Figure 1 is a diagram showing our model of organizational performance. The arrows indicate the dependency relationships.

The organizational and task-environmental factors define an *organizational design space* (ODS). A designed organization is a point in ODS. The performance measure is a function over ODS onto the performance metric space. Then, we can define the OD process as a search through ODS for an organization with acceptable performance. In other words, our design process model for OD is a generate-and-test or search process model, typically seeking a satisficing (rather than optimal) solution. We note that our model requires the designer to have predictive knowledge concerning exactly how the various factors determine the performance of the organization. This knowledge is then embodied in the performance evaluation function to be used in the design process. This allows automation of computational organization design.

In the following sections, we elaborate on the components of our model. It consists of three components: the organization model, the task-environment model, and the performance model. In its general form, it is still informal and verbal at this stage of research, and thus not amenable to precise computational implementations. However, for specific cases, as we show later in this paper, it can be realized computationally. Our goal is to formalize our model more generally so that we can embody it as *organizational knowledge* into autonomous agents capable of intelligent cooperation, enabling them to use it in the process of organizational self-design (OSD).

1.3.1 Organization Model We initially focus our attention on work organizations, that is, organizations that are designed for some definite work to be done. In particular, we mainly deal with computational organizations where the type of work done by the organization is the computation (or execution) of a complex function which is functionally decomposable into subfunctions, and the elements of the organization are agents capable of decomposing and distributing a set of tasks, transferring and routing the (sub)tasks and (sub)results, and capable of executing a set of primitive functions.

We think that the model of organization is tightly related to the model of the task the organization is used for. More concretely, we think that a specification of a work organization should at least include the following elements:

1. The set of tasks and subtasks to be done.
2. The set of agents participating in the organization.
3. An assignment of the tasks and subtasks to the participating agents.
4. A work flow structure which dictates how the tasks and subtasks are to be distributed among agents and how the results and partial results are to be synthesized.
5. Optionally, a set of resources aside from the agents and a set of constraints on the usage of those resources may apply to agents.

1.3.2 Task Environment Model By a task environment, we mean task and environmental characteristics that affect the performance of the organization. For example, type, size, rate of change, and structure of the task and the world are common important characteristics affecting the performance of many organizations.

In our research, we seek to be able to come up with a more comprehensive model of tasks and environments so that we can explicitly represent and reason about different kinds of task characteristics, and also incorporate task and environmental uncertainties, complexities, and dynamics into the task environment model. We believe that many terms such as task complexity, task dynamics, task uncertainty, environmental complexity, environmental uncertainty, and environmental dynamics must be precisely defined.

1.3.3 Performance Model The following are potential performance measures.

1. **Response Time** is the total time taken to accomplish a task. It is also called the turnaround time.
2. **Throughput** is the number of tasks accomplished per unit time. Without a definition of a unit task this measure is ill-defined.
3. **System Utilization** is the fraction of the total system capacity being used at any given time. For a given resource, it is the fraction of time the resource is busy.
4. **Communication Cost** is the cost of transmitting a number of bits across the channel. If time is used as the cost, it may include the connection time plus the time to transmit

a number of messages across the channel. Alternatively, the number of bits or message packets transmitted across the communication channel may be used as a measure for communication cost.

5. **Reliability** refers to the probability that the system or a component under consideration does not experience any failures in a given time interval. It is typically used to describe systems that cannot be repaired (as in space-based computers), or where the operation of the system is so critical that no downtime for repair can be tolerated. When a system is composed of multiple subsystems and/or components, the reliability of each component can be used to evaluate the reliability of the total system. By using redundant components, the system reliability can be improved.[2]
6. **Availability** refers to the probability that the system is operational according to its specification at a given point in time. Availability can be used as some measure of "goodness" for those systems that can be repaired and which can be out of service for short periods of time during repair.
7. **Solution Quality** refers to some objective measure of the quality of task results defined for the particular task domain.

2. Analyses of Alternative Tree Organizations

As we have mentioned previously, an impediment in studying computational organizations has been that the study of these organizations often involves making sense of a multiplicity of task-environment factors, of organizational parameters, and of performance metrics. Using our model of the organizational design space, we have been studying organizations by restricting these various features of task-environments, organizations, and performance models, and then incrementally extending our investigations.

For example, our initial studies (So and Durfee, 1993; 1994) considered a simple computational task-environment (which is exemplified by a distributed addition task) characterized by its *size* and by its *granularity,* defined as the ratio between unit task execution time and the unit message transmission time. The organizations are *tree-structured,* and comprised of homogeneous agents that simply perform their tasks. The performance is measured in terms of the *response time* of the organizations to accomplish their tasks. Our initial studies focused on how different tree organizations (branching factors and number of levels) performed on different task-environments (sizes and granularities).

We denote the problem size of the task by N. We assume that each agent or node is capable of performing any of the tasks, and each task requires τ time units. For example, in the addition task, loading a number in an accumulator requires τ time units, as does adding a number to the accumulated total, so adding N numbers takes $N\tau$ time units. For simplicity, we assume that any message to be sent can fit into a packet of fixed size, and that the communication delay between any two agents takes constant δ time units. Thus, the task granularity is $\gamma = \tau/\delta$.

[2]The concept of reliability for distributed systems, however, is a little tricky since they may involve redundant components for improving reliability, and therefore failure of some part of the overall system may degrade the performance of the system along other dimensions (e.g. response time) rather than making the entire system fail.

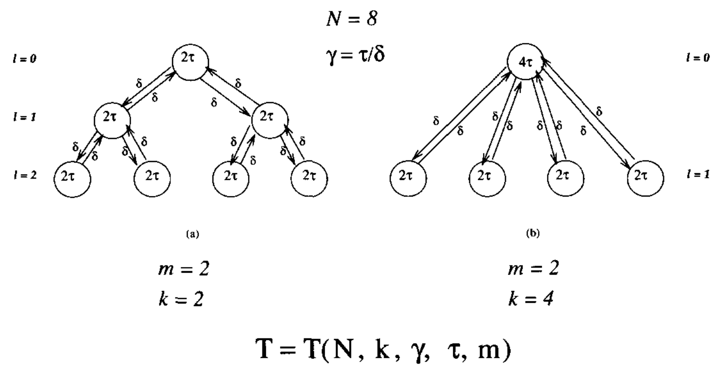

Figure 2. Examples of tree organizations.

We can begin with two extreme examples of tree-structured organizations: *l-level binary tree* and the *one-level k-ary tree.* The *height* of a rooted tree is the length of the longest path from the root to a leaf node. The *level* or *depth* of a node, in a rooted tree, is the length of the unique path from the root to that node. The *degree* (arity, branching factor) of a node is the number of "children" nodes incident to it. Thus by a *l-level binary tree,* we mean a binary tree of height l, and by a *one-level k-ary tree* we mean a tree of height one with the degree of the root equal to k. With a task size N of 8, and assuming the number of tasks assigned to each leaf node, m, is 2, we get the tree organizations in Figure 2.

Note that in the above, and in general, we adopt mathematical terminology to describe these organizations. The organizational literature has corresponding terminology as well, some of which we alluded to in the introduction to this paper. Specifically, the organization-theoretic notion of "span of control," indicating the number of individuals a manager is responsible for supervising, is equivalent to the "degree" or "branching factor" of nodes in the tree organization. A one-level k-ary tree is sometimes referred to as a "team" in the organizational literature, while a multilevel tree with smaller arity is referred to as a hierarchy. While we in general use the mathematical terms, the correspondence between the mathematical and organizational concepts should be kept in mind. Indeed, a central topic of this paper is in designing organizations in terms of selecting an appropriate branching factor, which is equivalent to choosing an appropriate span of control.

Unless otherwise noted, in this paper we assume that the performance measure T of a given organization for a given task is the time taken to complete the task, which is a function of the size of the overall task N, the branching factor k, the task granularity γ, the unit task execution time τ, and the number of unit tasks assigned to each leaf m. Note, moreover, that from these parameters we can derive other useful parameters, such as unit message delay δ $(= \tau/\gamma)$ and number of levels l $(= \log_k \frac{N}{m})$. We assume that the overall task initiates at the root of the tree, and is progressively decomposed and subtasks are distributed as one works down the tree. The results of subtasks must similarly be synthesized together working back up the tree. For simplicity, we also assume that the overall problem is of a size that can be neatly decomposed; for binary trees, $N = 2^{l+1}$, while for one-level trees, $N = 2k$. We now make these notions more precise:

Definition 2.1 *A* complete k-ary tree *is a tree in which every internal node has degree k where an internal node is a non-leaf node.*

Definition 2.2 *A* balanced complete k-ary tree *is a tree in which all internal nodes at the same level have degree k.*

Definition 2.3 *A* general balanced complete tree *is a tree in which all nodes at the same level have the same degree.*

Note that in a general balanced complete tree, the degree of internal nodes for different levels may differ. If we call a tree rooted on one of the nodes at level l a level-l subtree, we can see that for each level l in a general balanced complete tree, all level-l subtrees are of the same structure. The following lemma, which allows us to compute the performance (response time) of a general balanced complete tree organization for an overall task, allows us to derive performance equations for the two special kinds of general balanced complete tree organizations that are of interest to us.

Lemma 2.1 *Let D_l denote the time taken for a level-l subtree of a general balanced complete tree to complete its task. Let the degree of the root of that tree be k. Then, for* $l \geq 0$,

$$\text{If } \delta \leq \tau \text{ then } D_l = D_{l+1} + 2\delta + k\tau \text{ (see (a) below).}$$
$$\text{If } \delta > \tau \text{ then } D_l = D_{l+1} + (k+1)\delta + \tau \text{ (see (b) below).}$$

Proof. Let Δ_{l+1}^{i} denote the ith subtree of level $l+1$. Since the level-l subtree has degree of k, there will be k subtrees on level-$(l+1)$, i.e., from Δ_{l+1}^{1} to Δ_{l+1}^{k}. Denote the subtask completion time of Δ_{l+1}^{i} by T_i, assuming that the time at which the level-l node assigns its first subtask to the first subtree is 0. Note that $T_1 = \delta + D_{l+1}$ since it takes δ to assign the task to Δ_{l+1}^{1}, and it takes D_{l+1} for Δ_{l+1}^{1} to complete the assigned task.

Then, since subtasks are assigned sequentially, with each task assignment time δ, and since each subtask takes the same amount of time, $T_{i+1} = T_i + \delta$ for $1 \leq i \leq k-1$.

Figure 3 shows a timing diagram where each δ after T_i represents the time taken for sending the result of Δ_{l+1}^{i} to the node in level l, and each τ represents one unit task (addition) operation at a level l node after receiving result from Δ_{l+1}^{i}.

If we define $\mathcal{T}_k$ as the time at which the level l node receives the result of Δ_{l+1}^{k} and combines it with other received results (adds it to the partial sum):

(a) when $\delta \leq \tau$,

$$\mathcal{T}_k = T_1 + \delta + k\tau;$$

(b) when $\delta > \tau$,

$$\mathcal{T}_k = T_1 + k\delta + \tau.$$

Since $\mathcal{T}_k$ represents the time taken to complete the task at level l by assigning k subtasks to k level-$(l+1)$ nodes (or subtrees), and receiving results from those nodes and

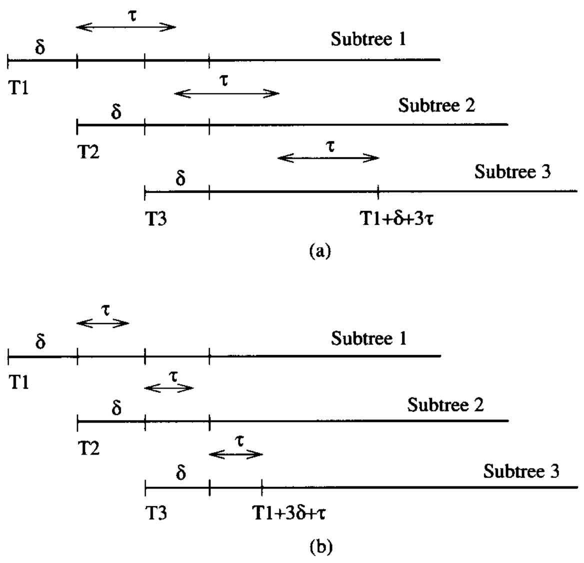

Figure 3. Timing diagrams for (a) $\delta \le \tau$ (b) $\delta > \tau$

combining k results, we can see that $D_l = \mathcal{T}_k$. But since $T_1 = \delta + D_{l+1}$, we get the following by substituting it into the above equations for $\mathcal{T}_k$:

1. if $\delta \le \tau$ then,

$$\begin{aligned} D_l &= \mathcal{T}_k \\ &= T_1 + \delta + k\tau \\ &= \delta + D_{l+1} + \delta + k\tau \\ &= D_{l+1} + 2\delta + k\tau; \end{aligned}$$

2. if $\delta > \tau$ then,

$$\begin{aligned} D_l &= \mathcal{T}_k \\ &= T_1 + k\delta + \tau \\ &= \delta + D_{l+1} + k\delta + \tau \\ &= D_{l+1} + (k+1)\delta + \tau. \end{aligned}$$

Definition 2.4 Let $\gamma = \frac{\tau}{\delta}$ where τ is the unit task execution time, and δ is the unit message transmission time. We call γ the *task environment granularity.*

Definition 2.5 A task environment is of *Coarse Granularity* when $\gamma > 1$ (i.e., $\delta < \tau$). A task environment is of *Medium Granularity* when $\gamma = 1$ (i.e. $\delta = \tau$). A task environment is of *Fine Granularity* when $\gamma < 1$ (i.e., $\delta > \tau$).

2.1 Binary Tree Organization

For simplicity, we assume that $N = 2^{l+1}$ where l is the level or height of the binary tree. That is, we assume in the addition task that each node in the binary tree adds two numbers. We assume that the root node divides the task of adding N numbers into two subtasks of adding $N/2$ numbers. The two subtasks are assigned to two other nodes sequentially. In this way, each subtask is divided in half and assigned to the next level down until the size of the subtask is 2. Since we assume the problem size is 2^{l+1}, a binary tree of l-levels will be sufficient and necessary. After the leaf nodes accomplish their tasks (add their two numbers), the result is propagated up to the node that assigned the subtasks. The node that receives the two results from one level down combines them and again propagates the partial result up one level, and so on. When the root node finishes combining the last two results the task is completed.

Another assumption about each node is that it is basically a serial computer, and therefore can only perform one task at a time which takes τ time units. Thus, even if two tasks (numbers to add) arrive at the same time, it will take 2τ time units to do them both (add them together). Also, when subtasks are assigned down the tree, we ignore the time taken in each node to divide the received subtask into two equal subtasks since it can be considered as taking constant time and thus counted as a part of the communication delay.[3]

The binary tree is a special case when the degree of all internal nodes is 2. In such a case, $D_l = D_{l+1} + 2\delta + 2\tau$ when $\delta \leq \tau$. Since such a finite difference relation holds for each pair of levels, we get $D_0 = D_l + l \times (2\delta + 2\tau)$. Thus for a l-level binary tree performing 2^{l+1} tasks according to our scheme, $D_l = 2\tau$ since all lth level nodes (i.e. leaf nodes) are assumed to perform two tasks. Thus the time taken for $N = 2^{l+1}$ tasks using an l-level binary tree is as follows.

If $1 \leq \gamma$, then

$$\begin{aligned} T(N, 2, \gamma, \tau, 2) &= D_0 \\ &= D_l + l \times (2\delta + 2\tau) \\ &= 2\tau + 2l(\delta + \tau) \\ &= 2l\delta + 2(l+1)\tau \\ &= 2\delta(\log_2 N - 1) + 2\tau \log_2 N \end{aligned}$$

If $1 = \gamma$, then

$$\begin{aligned} T(N, 2, \gamma, \tau, 2) &= 2\delta(\log_2 N - 1) + 2\tau \log_2 N \\ &= 2\tau(\log_2 N - 1) + 2\tau \log_2 N \\ &= 2\tau(2\log_2 N - 1) \end{aligned}$$

[3]This assumption cannot hold in DAI domains where task decomposition itself is a non-trivial time-consuming task.

If $1 > \gamma$, then

$$\begin{aligned}T(N, 2, \gamma, \tau, 2) &= D_0 \\ &= D_l + l \times ((2+1)\delta + \tau) \\ &= 2\tau + l(3\delta + \tau) \\ &= 3l\delta + (l+2)\tau \\ &= 3\delta(\log_2 N - 1) + \tau(\log_2 N + 1)\end{aligned}$$

We can thus determine the conditions under which cooperation via binary tree organization is more effective than centralized computation through comparison. More specifically, we want to know the condition under which $T(N, 2, \gamma, \tau, 2) < T(N, 0, \gamma, \tau, N)$. Knowing that $T(N, 0, \gamma, \tau, N) = N\tau$, we can derive such conditions.

We find that, for Coarse Granularity Binary Tree Organization,

$$\begin{aligned}&T(N, 2, \gamma, \tau, 2) < T(N, 0, \gamma, \tau, N) \Leftrightarrow \\ &\quad (N > 12) \bigvee [(4 < N \leq 12) \bigwedge (\gamma > \frac{2(\log_2 N - 1)}{N - 2\log_2 N})]\end{aligned}$$

For Medium Granularity Binary Tree Organization,

$$T(N, 2, \gamma, \tau, 2) < T(N, 0, \gamma, \tau, N) \Leftrightarrow (N > 12)$$

For Fine Granularity Binary Tree Organization,

$$T(N, 2, \gamma, \tau, 2) < T(N, 0, \gamma, \tau, N) \Leftrightarrow (N > 12) \bigwedge (\frac{3(\log_2 N - 1)}{(N-1) - \log_2 N} < \gamma < 1)$$

2.2 *One-Level k-ary Tree Organization*

In this case, there is only one-level of k nodes to which the task of size $N = 2k$ is distributed. Each of the k nodes will perform 2 unit tasks (add 2 numbers) and return the result to the root node.

We model this organization by using the previous Lemma with

$$T(N, N/2, \gamma, \tau, 2) = D_0, \quad D_1 = 2\tau, \quad \text{and} \quad N = 2k.$$

If $1 \leq \gamma$, then

$$\begin{aligned}T(N, N/2, \gamma, \tau, 2) &= D_0 \\ &= D_1 + 2\delta + k\tau \\ &= 2\tau + 2\delta + (N/2)\tau \\ &= 2\delta + \frac{N+4}{2}\tau\end{aligned}$$

If $1 = \gamma$, then

$$\begin{aligned} T(N, N/2, \gamma, \tau, 2) &= 2\delta + \frac{N+4}{2}\tau \\ &= 2\tau + \frac{N+4}{2}\tau \\ &= \tau(4 + \frac{N}{2}) \end{aligned}$$

If $1 > \gamma$, then

$$\begin{aligned} T(N, N/2, \gamma, \tau, 2) &= D_0 \\ &= D_1 + (k+1)\delta + \tau \\ &= 2\tau + ((N/2) + 1)\delta + \tau \\ &= \frac{N+2}{2}\delta + 3\tau \end{aligned}$$

Comparing the performance of this organization with single node performance, we find that, for Coarse Granularity One-level *k*-ary Tree Organization,

$$\begin{aligned} &T(N, N/2, \gamma, \tau, 2) < T(N, 0, \gamma, \tau, N) \Leftrightarrow \\ &\quad (N \geq 8) \bigvee [(4 < N < 8) \bigwedge (\gamma > \frac{2(\log_2 N - 1)}{N - 2\log_2 N})] \end{aligned}$$

For Medium Granularity One-level*k*-ary Tree Organization,

$$T(N, N/2, \gamma, \tau, 2) < T(N, 0, \gamma, \tau, N) \Leftrightarrow N > 8$$

For Fine Granularity One-level *k*-ary Tree Organization,

$$T(N, N/2, \gamma, \tau, 2) < T(N, 0, \gamma, \tau, N) \Leftrightarrow (N > 8) \bigwedge (\frac{N+2}{2(N-3)} < \gamma < 1)$$

In summary, we can see that the performance of a given type of organization depends both on the size of the task and the granularity of the task-organization configuration. For instance, our model shows that adding four numbers using a tree organization gives worse performance than adding them sequentially at one node. Tree organizations outperform single node performance for increased task size since in our model the organization size grows with problem size, but even this tendency is conditional on the speed of the communication links relative to the unit task execution rate of processors. Thus, for fine granularity, binary tree organizations usually outperform single nodes but only if the granularity is above some bound. That is, for fine granularity, if the communication delay is too large relative to the unit task execution time, it may still be better to execute the task in a single node.

2.3 Comparisons Between Organizational Structures

In the previous sections, we saw conditions in which an organization of nodes could outperform a single, centralized strategy for solving the same problem, where the problem task-environment, the organizational structure, and the performance measure were kept simple and well-defined. In this section, we stay mostly within these same parameters, but ask a slightly different question: which organizational structure will perform best for a particular problem? To study this, we extend the range of structures to include a third category:

1. Binary Trees.
2. k-ary trees of one level.
3. General Balanced Complete Trees.

We continue to make the following assumptions:

1. Uniform processing rate for all nodes.
2. Uniform transmission rate for all links.
3. Uniform task size for all nodes at the same level.
4. Uniform packet size for all messages.
5. Unlimited supply of nodes.
6. Allow only complete trees.
7. Number of nodes grow as problem size increases.
8. Task decomposition takes negligible constant time.

Let us begin with the two simple organizational structures from the previous section. We can think of the l-level binary tree organization as representing a tall-thin hierarchical organization, and the one-level k-ary tree organization as representing a short-fat hierarchical organization. Thus, by determining the condition under which $T(N, 2, \gamma, \tau, 2) < T(N, N/2, \gamma, \tau, 2)$, we can gain intuitions about the conditions under which tall-thin hierarchical organizations outperform short-fat hierarchical organizations.

We find that, for Coarse Granularity task environment:

$$T(N, 2, \gamma, \tau, 2) < T(N, N/2, \gamma, \tau, 2) \Leftrightarrow$$
$$(N < 4) \vee (N \geq 26) \vee [(8 < N < 26) \wedge (\gamma > \frac{\log_2 N - 2}{N/4 - \log_2 N + 1})]$$

For Medium Granularity task environment,

$$T(N, 2, \gamma, \tau, 2) < T(N, N/2, \gamma, \tau, 2) \Leftrightarrow (N < 4) \vee (N \geq 26)$$

For Fine Granularity task environment,

$$T(N, 2, \gamma, \tau, 2) < T(N, N/2, \gamma, \tau, 2) \Leftrightarrow$$
$$(N < 4) \vee (N \geq 26) \vee [(16 < N < 26) \wedge (\gamma < \frac{N/2 - 3\log_2 N + 4}{\log_2 N - 2})]$$

Thus, in general, tall-thin hierarchies outperform short-fat hierarchies when problem size is sufficiently large (above some bound). However, for some problem sizes (e.g. $N = 24$), short-fat hierarchies may outperform tall-thin ones if the granularity of the task environment is neither too large nor too small.

Of course, even remaining within the realm of tree organizations, there are other options between short-fat and tall-thin hierarchies. If we consider more arbitrary k-ary tree organizations, for example, their performance function could be expressed in algebraic terms as follows.

$$\begin{aligned} T(N, k, \gamma, \tau, m) = {} & \chi_{\gamma \le 1}(\gamma) \cdot \{(k+1)l\delta + (l+m)\tau\} \\ & + \chi_{\gamma > 1}(\gamma) \cdot \{2l\delta + (kl+m)\tau\} \end{aligned}$$

where

$$l = \log_k(\frac{N}{m})$$

$$\delta = \frac{\tau}{\gamma}$$

$$\chi_{\gamma \le 1}(\gamma) = \begin{cases} 1 & \text{if } \gamma \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

$$\chi_{\gamma > 1}(\gamma) = \begin{cases} 1 & \text{if } \gamma > 1, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

For k-ary trees, we have found that there usually exists a branching factor k which is larger than 2 but smaller than the branching factor for a single level tree such that the k-ary tree outperforms both the binary tree and the single level tree. For example, for a task of size 32 ($N = 32$) and $m = 2$, the 2-level 4-ary tree organization outperforms both the 4-level binary tree and the one-level 16-ary tree organization *for any task environment granularity* γ. This is interesting, since it means that the 4-ary organization is better than the binary and the one-level organization no matter how the processing speeds of the nodes (τ) and/or the communication delays between the nodes (δ) change (albeit the assumption that τs and δs are uniform). That is, when the above equation is applied to the case where $N = 32$, $m = 2$, and $k = 2, 4, 16$, it can be verified that $\forall \gamma, \tau$:

$$T(32, 4, \gamma, \tau, 2) \le T(32, 2, \gamma, \tau, 2)$$
$$T(32, 4, \gamma, \tau, 2) \le T(32, 16, \gamma, \tau, 2)$$

A tree with possibly varying branching factors at each node might, at times, perform even better. Unfortunately, however, expressing a performance function for them in algebraic terms is extremely difficult, and instead we have devised a recursive performance evaluation function. However, a full explanation of our recursive algorithm is beyond the scope of this paper.

2.4 Generalizing the Choice of Span-of-Control

While it is interesting to observe, above, an instance where an intermediate span-of-control (branching factor k) leads to a better organization than either the extremely tall-thin hierarchy (binary tree) or the short-fat team (one-level k-ary tree), how general is this result? And was the choice of $k = 4$ the best choice, or are there other values that would be even better? In other words, what can we say more generally about a good choice of k given the performance metric of minimizing task execution time?

To answer this question, we have studied the proper choice of k within these simple tree-structured organizations to determine the degree to which an optimal k depends on N, m, and γ. To do this we have further analyzed Equation 1, taking the partial derivative of it with respect to the span of control k, and looking for the optimal value of k as being where the partial derivative equals zero. We have done this for various values of N, m, and γ.

Our preliminary results indicate that the optimal k (k_{opt}) does vary, but varies within only a narrow band depending on γ. We find the following:

1. for fine granularity: as γ approaches 0, k_{opt} approaches approximately 3.6;
2. for medium granularity: as γ approaches 1, k_{opt} approaches approximately 4.4;
3. for coarse granularity: as γ approaches ∞, k_{opt} approaches approximately 2.8.

The above results appear to hold for all sufficiently large N, independently of the value of m. To illustrate our claim, we provide three contour plots (Figure 4) for $T = T(N, k)$ with m fixed at 2 and γ of 0.1, 1.0, and 10.0 for (a), (b), and (c), respectively.

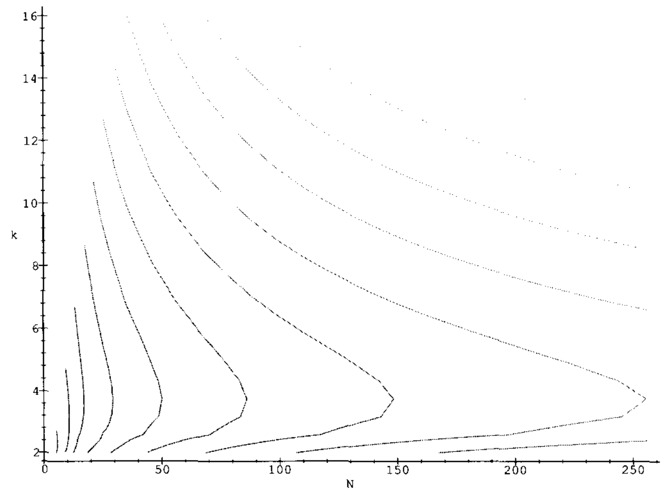

Figure 4(a). Contour plot of performance (T) given N and k (a)$\tau = 1.0, \delta = 10.0$.

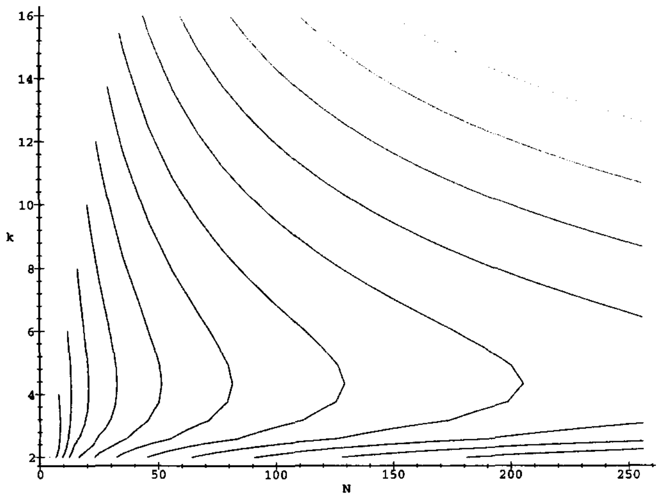

Figure 4(b). $\tau = 1.0, \delta = 1.0$.

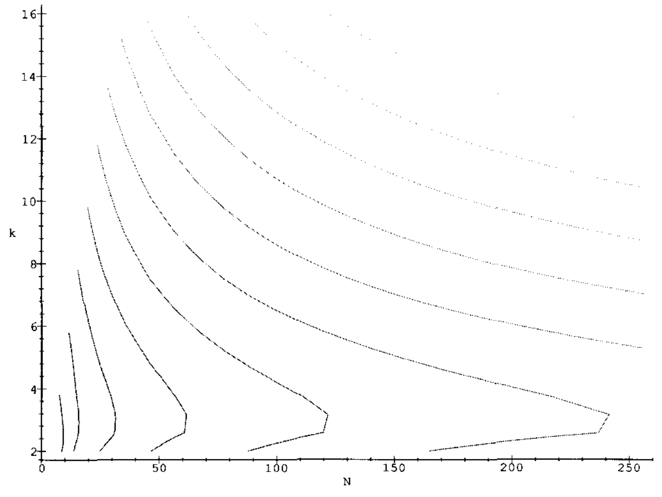

Figure 4(c). $\tau = 1.0, \delta = 0.1$.

We currently conjecture that the typically non-integer values for k_{opt} may imply that, for certain task-environments, hierarchical organizations with different spans of control at different levels may outperform such organizations with constant spans of control across the levels. Our future plans include testing this conjecture.

2.5 Summary

It has been shown many times that different organizations are more effective in different circumstances. For example, Carley summarizes work that has shown that, when it comes to organizational performance and learning, multi-level hierarchies are slower but more robust, while flatter teams tend to learn and respond faster (Carley, 1995). To some extent, our results confirm previous observations, such as that tall hierarchies can be at a disadvantage in fine granularity task-environments because of the delays in passing information up and down multiple hops in the hierarchy. By focusing more narrowly on a particular performance metric and a parametric definition of an organization, however, we have been able to reach more quantitative, categorical conclusions:

1) Cooperative distributed problem solving using tree-organizations is better than centralized problem solving as long as the task is big enough (thus exploiting the benefits of parallelism) and communication is fast enough relative to computation.
2) Tall-thin hierarchies outperform (have faster response time than) short-fat hierarchies when problem size is sufficiently large (i.e. above some bound).
3) For certain intermediate problem sizes, short-fat hierarchies may outperform tall-thin ones if the granularity of the task environment is neither too large nor too small.
4) For k-ary trees, there exists a branching factor k_{opt} which is larger than 2 but smaller than the branching factor for a single level tree such that the k_{opt}-ary tree generally outperforms both the binary tree and the single level tree.
5) Determining an appropriate span of control can be done independently of m and N (assuming large N), and varies within a narrow band ($2.8 \leq k \leq 4.4$) depending on γ.

3. Application to a Distributed Network Monitoring System

Consider a Distributed Network Monitoring (DNM) system where nodes in a large network are endowed with the capability to communicate and cooperate in monitoring the network. The wide area can be divided into several regions or subnetworks and one or more agents can be employed to be responsible for each subnetwork. Within each subnetwork, a set of agents may be jointly responsible for maintaining up-to-date models of host performance and availability. Monitoring, or more generally network management, agents may have disjoint functions or potentially overlapping responsibilities for increased reliability. Since network monitoring involves polling information about each component in the network, each agent may be responsible for monitoring a subset of those components.

Our previous work led to a distributed network monitoring system called Distributed Big Brother (DBB) (So and Durfee, 1992). DBB uses a static organizational structure to decompose the management task, and within that structure uses contracting to assign specific tasks to nodes. The basic organizational structure, shown in Figure 5, identifies several management roles in a hierarchical structure. The TOP manager provides information to the Network Administrator, and oversees multiple LAN managers, which are responsible

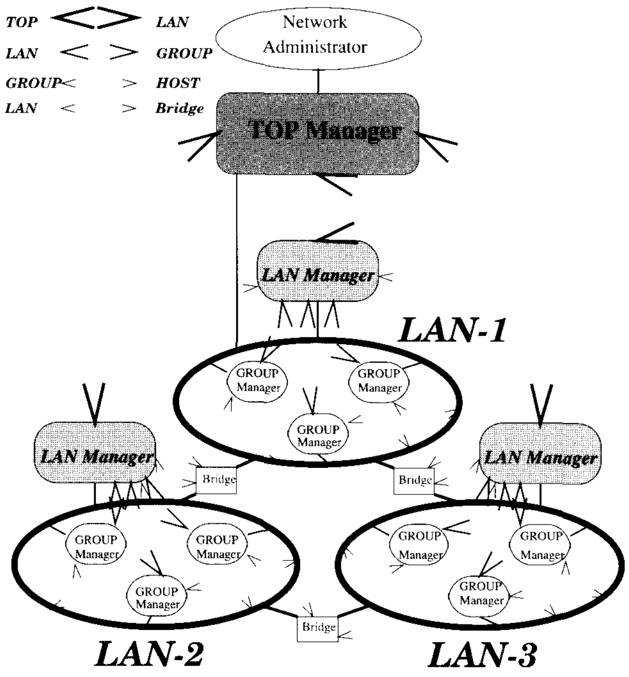

Figure 5. Distributed big brother organization.

for providing the TOP manager with summaries of performance of their respective LANs. A LAN manager, similarly, oversees one or more GROUP managers, which are responsible for directly monitoring a subset of hosts on the LAN and periodically reporting back to the LAN manager.

3.1 Analytical Model

The type of task going on in distributed network monitoring is actually quite similar to the abstract task characterization introduced in Section 2, exemplified in the addition task. Like the addition task, the goal of network monitoring is to accomplish a large combination of relatively simple tasks (assessments of host statuses). Like the addition task, the distribution involves decomposing the overall task into subsets (LANs and host groups to monitor) and combining the results collected (summarizing received data). We can even depict the network monitoring task in a similar way (Figure 6), although there are some new features to the model.

The model represents the following:

- k : The number of GROUP managers per LAN.
- δ : Time to transmit one message packet from one agent to another. A packet was approximately 2000 bytes.
- τ : Time to gather information from one host. This is the unit task execution time.

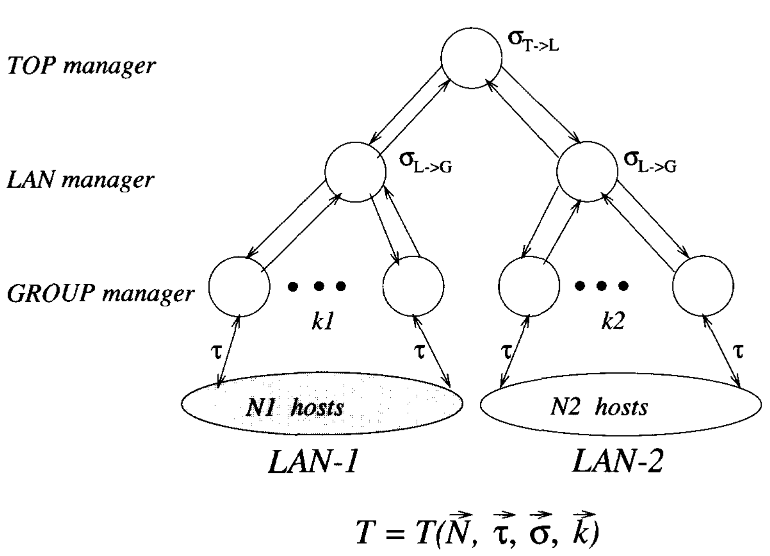

Figure 6. Model of the DNM task.

- $\sigma_{T\to L}$: Time between task assignment from TOP manager to one of LAN managers. This is the inverse of TOP to LAN task assignment rate.
- $\sigma_{L\to G}$: Time between task assignment from LAN manager to one of GROUP managers. This is the inverse of LAN to GROUP task assignment rate.
- N_i : The number of hosts in LAN i.

We denote the response time of DBB with k GROUP managers per LAN as T_k. Since the response time of DBB is dominated by the LAN with the largest number of hosts we need only consider the time to gather full information about that LAN by the TOP manager.

Some values of the above variables for our experiment were as follows.

- $\delta \cong 10\ ms$.
- $\tau \cong 13.33\ s$. This value is obtained from the Centralized Big Brother data where information on 117 hosts were gathered in 26 minutes, resulting in 13.3 seconds per host.
- $\sigma_{T\to L} = 60\ s$.
- $\sigma_{L\to G} = 60\ s$.
- $N_1 = 117$. This is the number of hosts on the LAN with the larger number of hosts.

We measured the response times of each DBB configuration as follows. First, the start time was defined as the closest time *before* the first task assignment to a GROUP manager at which the TOP manager received a report from the LAN manager. Since the TOP manager requests a LAN manager to report on the status of the LAN every 60 seconds (i.e. $\sigma_{T\to L}$), the time from start time as we define it to the first GROUP task assignment, which we denote by σ_{Start}, cannot be greater than 60 seconds. That is, $0\ s < \sigma_{Start} < 60\ s$.

Secondly, the end time was defined as the closest time *after* the full information of the dominant LAN was gathered at which the TOP manager received a report from the LAN

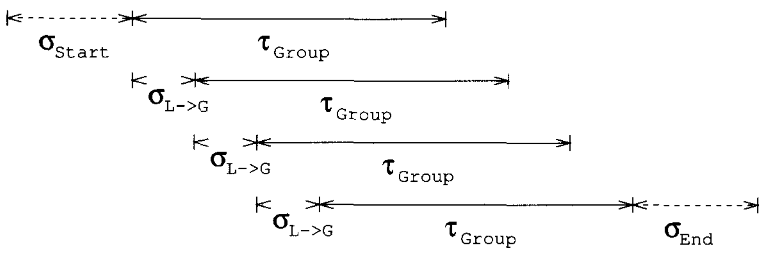

Figure 7. DBB response time analysis.

manager. The time from full information gather time to end time, which we denote by σ_{End}, cannot be greater than 60 seconds for similar reasons. That is, $0\ s < \sigma_{End} < 60\ s$.

Figure 7 shows a timing diagram analysis of the DBB response time for four GROUP manager configuration (i.e. $k = 4$). Here, τ_{Group} refers to the total execution time for each GROUP manager. Since in our experimental environment $\delta(0.1\text{s}) \ll \tau(13\text{s}) < \sigma\ (60\text{s})$, we ignore δ in our analysis. We also assume that every message fits into one packet, and that assigning a task using the Contract Net Protocol (Smith, 1980) requires three message transfers. We assume that the time required to assign a task to an agent is less than 1 second, which is reasonable since transmitting three packets requires 0.3 ($= 3 \times 0.1$) seconds, and since we use the current number of assigned tasks as the criterion for task assignment, which requires simple one value fetch operation per bidder, and a function to return the bidder with the least number of tasks assigned, which together we assume to take less than 0.7 seconds.

The formula for total response time for DBB with k GROUP managers per LAN is as follows.

$$T_k = \sigma_{Start} + (k - 1)\sigma_{L \to G} + \tau_{Group} + \sigma_{End} \tag{2}$$

Since the total LAN task is divided into eight subtasks, we assume that each GROUP manager is assigned $\lceil \frac{8}{k} \rceil$ subtasks. Since the LAN manager has a fixed task assignment rate, a GROUP manager which finished its previously assigned subtask might have to wait for next LAN to GROUP task assignment time. We denote such wait time by σ_{Wait}, and note that $0s < \sigma_{Wait} < \sigma_{L \to G}$. Since we only consider such wait times after the first subtask execution and before the last subtask execution, there are $(\lceil \frac{8}{k} \rceil - 1)$ many such occasions per GROUP manager. Thus, the formula for τ_{Group} is as follows.

$$\tau_{Group} = \frac{N_l \times \tau}{k} + \sigma_{Wait} \times (\lceil \frac{8}{k} \rceil - 1)$$

The first term refers to the total time each GROUP manager spends gathering host information. In our case $N_l \times \tau = 26\ min$, which is the time taken to gather 117 hosts' information by a centralized BB.

3.2 Experimental Comparison

Table 1 shows the response times computed with our formula (Equation 2) with $\sigma_{Start} = (0, 1)$, $\sigma_{End} = (0, 1)$, $\sigma_{L \to G} = 1$, $\sigma_{Wait} = (0, 1)$ where $X = (\alpha, \beta)$ is equivalent to $\alpha < X < \beta$. It shows that worst case results of the model closely approximate the experimental results.

As we can see, the performance of DBB changes as the number of agents in a GROUP changes. If we have a utility function that takes into account the benefits and the costs incurred by the addition of more agents, we can use an analytical model like the one above to predict the overall best number of GROUP managers per LAN.

We thus see that the performance (response time) of DBB is dependent on the number of hosts in a LAN (N_i), the unit task execution time (τ), and the task assignment rates (σ). In the analysis, we ignored the transmission delay (δ) because it was so much smaller than τ or σ. However, depending on the amount of network traffic, the communication link technology, and the distance between the agents, δ may well vary and become a significant factor affecting the performance of the system. Likewise, depending on the computational power of the hardware of each agent, and the current load of the processor in multitasking situations, the value of τ may vary. Finally, there may be long-term changes in the number of hosts (i.e. size of tasks) in one or more subnets which will also change the performance graph as well. Thus, a configuration or an organizational structure that maximizes the overall utility in one task and resource environment may not be optimal in a different environment.

Therefore, if the DBB agents are to adaptively reconfigure themselves as the task and resource environment changes, they must be able to continuously monitor the changes in the environment (e.g. values of N, τ, δ) and determine whether a change in the organizational structure (e.g. span of control k) can lead to higher payoff. In order to do that, there must be a way to generate the possible changes in organization, and a way to evaluate each possible organization given the current set of environmental parameter values or the predicted future values of those parameters.

Table 1. Response time comparison for model and experiment.

Configuration	(min,max) Time Expected by Model (mins)	Experimental Time (mins)
DBB-2L-1G	(26.0, 35.0)	34
DBB-2L-2G	(14.0, 19.0)	18
DBB-2L-3G	(10.7, 14.7)	14
DBB-2L-4G	(9.5, 12.5)	12

4. Toward Organizational Self-Design

The type of organization developed and evaluated for DBB is only effective if all of the roles are filled—if a LAN manager disappears, the TOP manager loses touch with an entire LAN; if a GROUP manager fails then a LAN manager loses track of a subset of hosts. The *reconfiguration problem* (Pattison, Corkill, and Lesser 1987), within DBB, thus involves having managers at the various levels *monitor* each other to detect breakage of the organizational hierarchy, and having the managers *reimplement* the hierarchy by reassigning roles among themselves to restore the functionality. DBB performs reconfiguration in response to agent failures, requiring complex algorithms for monitoring the state of the organization and choosing new agents to populate various organizational roles (So and Durfee, 1996).

Solving the reconfiguration problem allows only limited adaptation to changing task-environments, however, because it always assumes that the organizational structure is appropriate and is only concerned with maintaining it. But, in a task like distributed network monitoring, what happens as the number of LANs increases to tens or hundreds? Is only one TOP manager going to be able to keep up, or should there be new layers of management in between? Or, conversely, if few managers are available such that only one GROUP manager is available in a LAN, does it still make sense to have one LAN manager supervising only a single GROUP manager?

For such cases, agents actually need to redesign the structure of their organization, rather than just reassigning roles in the current organization. The process of organizational self-design (OSD) by agents involves not only monitoring and (re)implementation of an organization, but also a search through the space of possible organization designs and an evaluation of how well they can be expected to perform. There are a variety of ways that OSD can be done; see So and Durfee (1996) for an overview of OSD and how it has been addressed by the distributed AI community. Our ongoing approach is to build off of the model described in this paper, so that agents can design an organization in terms of determining the proper k (and m, along with a parameter specifying the amount of redundant task assignment) based on an analysis of the expected performance of the organization given what is known of N, τ, and δ (So and Durfee, 1995).

5. Summary and Future Work

We have presented a model of organizations for computational agents, and have described an approach for designing organizations based on a concrete understanding of their task-environments and the performance metrics. Our specific results reported here focus on tree-structured organizations, and how to design the span of control (branching factor) for such organizations, which impacts the height and width of the hierarchy. We have shown how the design is dependent on the nature of the task-environment (characterized by task size and granularity) and the performance measure (we restricted our considerations here to response time).

A distinguishing feature of our model compared to other models of distributed hierarchical problem solving such as in (Montgomery, 1992) is that this model takes into account the effect of task assignment overhead. Although parallel asynchronous communication is not uncommon, in many applications synchronous communication such as TCP/IP is common, and when we think of human organizations, due to the biological limitation of a single agent, task assignment to other agents is often a sequential process. However, the assumption here that the task assigner has to wait for an acknowledgment before he can start assigning the next task to another individual may be unrealistic since people often do continue doing other things after sending off a task but before it arrives at the destination. Similarly, in computer networks, asynchronous communication of this type is also possible.

Our work only begins to explore the space of task-environments, organizational structures, and performance measures embraced by our model. Clearly, many open problems exist in extending our results along each of these dimensions. For example, another performance measure of particular interest is reliability. Whether comprised of humans or computational agents (or both), an organization should be able to accomplish its goals satisfactorily despite failures of individual organization participants. Strategies for increasing this performance measure can involve assigning more forgiving tasks to the organization, or stabilizing the environment, but when the task environment is fixed, it is once again the organization that must be designed to maximize performance. In our ongoing work, we are exploring the use of redundant task assignments as part of an organizational specification, and their impacts on reliability and also on response time.

An essential feature that is lacking in our model of dynamic organizations is that it does not take into account the effect of organizational design on the task environment. That is, not only is an organization influenced by the task environment, but also its behavior and actions influence the task environment. Thus, a more complete model of organizational performance should have a second class of equations which has the task environmental factors as dependent variables and the organizational factors as independent variables, in addition to the current set of equations which have various organizational performance metrics as dependent variables and both organizational and task environmental factors as independent variables jointly determining the organizational performance. More complex models such as these can result in many possible classes of organization-environment system behaviors as identified in theories of complex systems such as Chaos, Punctuated Equilibria, etc. (Kephart, Hogg, and Huberman, 1989).

Despite limitations of this sort, our belief is that pushing our model along its various dimensions will illuminate in a more precise manner, as exemplified in this paper, the relationship between task-environmental, organization, and performance characteristics. Our methodology emphasizes beginning with simple combinations of these characteristics, which is suited to fielding increasingly complicated organizations of computational agents, while (at least in the short term) having perhaps limited applicability to human organizations where simplifications might be impossible. Nonetheless, our hope is that our work might provide insights to human organizations, while reaping more immediate rewards in computational applications.

Acknowledgments

We would like to thank our colleagues at the University of Michigan, especially Michael Cohen, and other members of the Distributed Intelligent Agents Group (DIAG), for their suggestions and insights regarding this work. We would also like to thank the anonymous reviewers for their comments. This work was supported, in part, by the National Science Foundation, under PYI award 91-58473.

References

Carley, K. M. and M. J. Prietula (Eds.)(1994), *Computational Organization Theory.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Carley, K. M. (1995), "Computation and Mathematical Organization Theory: Perspective and Directions," *Computational and Mathematical Organization Theory* 1(1), 39–56.

Cohen, M. D. (1986), "Artificial Intelligence and the Dynamic Performance of Organization Designs," in James G. March and Roger Weissinger-Baylon, (Eds.) *Ambiguity and Command: Organizational Perspectives on Military Decision Making,* Marshfield, Mass.: Pitman Publishing.

Corkill, D. D. (1983), *A Framework for Organizational Self-Design in Distributed Problem Solving Networks.* Ph.D. thesis, University of Massachusetts. (Also published as Technical Report 82–33, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1982.).

Corkill, D. D. and V. R. Lesser (1983), "The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* pp. 748–56, ed. Alan H. Bond and Les Gasser, Karlsruhe, Federal Republic of Germany.

Durfee, E. H., V. Lesser and D. D. Corkill (1987), "Coherent Cooperation Among Communicating Problem Solvers," in *IEEE Transactions on Computers,* C-36(11), 1275–1291.

Emery, F. E. (1959), *Characteristics of Socio-Technical Systems.* Tavistock Document 527. London: Tavistock.

Fox, M. S. (1981), "An Organizational View of Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics,* 11(1), 70–80. (Also published in *Readings in Distributed Artificial Intelligence,* pp. 140–150, ed. A. H. Bond and Les Gasser, Morgan Kaufmann, 1988.).

Katz, D. and R. L. Kahn (1966), *The Social Psychology of Organizations.* New York: John Wiley & Sons.

Kephart, J. O., T. Hogg and B. A. Huberman (1989), "Dynamics of Computational Ecosystems: Implications for DAI," in Les Gasser and M. Huhns (Eds.) *Distributed Artificial Intelligence,* Vol. 2, London: Pitman; San Mateo, CA: Morgan and Kaufmann.

Lawrence, P. R. and J. W. Lorsch (1967), *Organization and Environment: Managing Differentiation and Integration.* Boston: Graduate School of Business Administration, Harvard University.

Malone, T. W. (1987), "Modeling Coordination in Organizations and Markets," *Management Science,* 33(10), 1317–1332.

Montgomery, T. A. and E. H. Durfee (1992), "Search Reduction in Hierarchical Distributed Problem Solving," in *Proceedings of the 1992 Distributed AI Workshop.*

Novak, V. J. A. (1982), *The Principle of Sociogenesis.* Praha: Academia Pub. House of the Czechoslovak Academy of Sciences.

Pattison, H. E., D. D. Corkill and V. R. Lesser (1987), "Instantiating Descriptions of Organizational Structures," in Michael N. Huhns, (Ed), *Distributed Artificial Intelligence,* London: Pitman.

Scott, W. R. (1992), *Organizations: Rational, Natural and Open Systems.* Englewood Cliffs, NJ Prentice-Hall.

Smith, R. G. (1980), "The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver," IEEE Transactions on Computers, C-29(12): 1104–1113.

So, Y. and E. H. Durfee (1992), "A Distributed Problem Solving Infrastructure for Computer Network Management," *International Journal of Intelligent and Cooperative Information Systems,* 1(2), 363–392.

So, Y. and E. H. Durfee (1993), "An Organizational Self-Design Model for Organizational Change," *Paper presented at the 1993 AAAI AI and Theories of Groups and Organizations Workshop,* Washington, DC.

So, Y. and E. H. Durfee (1994), "Modeling and Designing Computational Organizations," *Paper presented at the 1994 AAAI Computational Organization Design Symposium,* Palo Alto, CA: Stanford.

So, Y. and E. H. Durfee (1995), "Local Sophistication and Organizational Performance," paper presented at the *1995 Workshop on Mathematical and Computational Organization Theory,* Los Angeles, CA.

So, Y. and E. H. Durfee (1996), "Designing Organizations for Computational Agents," to appear in *Computational Organization Theory 2,* K. Carley, M. Prietula, and L. Gasser (Eds.) Menlo Park, CA: AAAI/MIT Press.

Trist, E. L. (1981), "The Evolution of Sociotechnical Systems as a Conceptual Framework and as an Action Research Program," in Andrew H. Van de Ven and William F. Joyce, (Eds.), *Perspectives on Organization Design and behavior,* New York: John Wiley & Sons.

Young-Pa So

Young-pa So was born in Seoul, Korea, attended the Seoul National University (1984–1985) as an undergraduate majoring in chemistry, transferred to the University of Michigan in 1985 and earned his bachelor's degree in Computer Science in 1989. He received his Master's degree at the Electrical Engineering and Computer Science Department of the University of Michigan in 1991, and is currently pursuing his Ph.D. degree in the same department. His current field of research is in Multiagent Systems and Computational Organization Theory, and his research interests include Artificial Intelligence, Organizational Self-Design, and Intelligent Network Management.

Edmund H. Durfee

Edmund H. Durfee is an associate professor in the Department of Electrical Engineering and Computer Science at the University of Michigan, where he conducts research in multiagent systems, real-time intelligent control, and cooperative problem solving for applications ranging from interacting unmanned vehicles to digital libraries. He received his undergraduate degree in 1980 from Harvard University, and his Ph.D. in 1987 from the University of Massachusetts. In 1991, he was named a Presidential Young Investigator.