Econometrica, Vol. 61, No. 5 (September, 1993), 1109–1146

THE ORGANIZATION OF DECENTRALIZED INFORMATION PROCESSING

By Roy Radner[1]

In a decision-theoretic model of a firm, I represent managers as information processors of limited capacity; efficiency is measured in terms of the number of processors and the delay between the receipt of information by the organization and the implementation of the decision. I characterize efficient networks for both one-shot and repeated regimes, as well as the corresponding "production function" relating the number of items processed to the number of processors and the delay. I sketch some applications to common decision paradigms, and implications for decentralization and organizational returns to scale.

Keywords: Decentralization, organization of the firm, information-processing.

1. INTRODUCTION

The typical U.S. corporation is so large that a substantial part of its workforce is devoted to information-processing, rather than to "making" or "selling" things in the narrow sense. Although precise definitions and data are not available, a reasonable estimate is that more than one-half of U.S. workers (including managers) do information-processing as their primary activity.[2]

A related dichotomy is the one between "managers" and "workers." To paraphrase Frank Knight,[3] "workers do, and managers figure out what to do." If we add to the managers those who support managerial functions, we probably come out with roughly one-third or more of the workforce.[4]

We might think of the "information-processing" or "managerial" part of the firm as one huge decision-making machine, which takes signals from the environment and transforms them into actions taken by the "real workers." Of course, every worker on an assembly line or lathe, and every salesperson in the field, makes many decisions every day that are not precisely dictated by "management," and that has always been so, but the point I am making here is that in the modern corporation a large part of the workforce is *specialized* in management activities, or activities in support of management activities. It is in this sense that the management—or information-processing activities—are "decentralized," i.e., spread out among a large number of persons in the corpora-

[1] The views expressed here are those of the author, and do not necessarily represent those of AT&T Bell Laboratories. I am grateful for helpful discussions with P. B. Linhart, A. Rustichini, F. W. Sinden, P. W. Shor, B. D. Lubachevsky, C. B. McGuire, P. McGuire, C. Wilson, and T. Van Zandt. S. E. Page made valuable comments on an earlier draft. These persons are not, however, responsible for any errors of logic or judgement in the paper.

[2] See Radner (1992).

[3] Knight (1921, p. 268).

[4] Again, see Radner (1992).

tion. This is so, even though corporations are often thought to be highly "centralized" from the point of view of authority and supervision.

It is the main theme of this paper that "hierarchical" structures, which are usually thought of as the epitome of the centralization of authority, are also remarkably effective in decentralizing the activities of information processing. The decentralization of information-processing is dictated by the large scale of modern enterprises, which makes it impossible for any single person to manage everything. Thus the limited capacity of individuals for information-processing implies that this activity uses scarce resources (e.g., people), and it would seem that these uses constitute a substantial part of total economic activity. Hence information-processing and management would appear to be natural objects of economic study. On the other hand, it is the computer scientists, more than any others, who have specialized in studying how to organize effectively the resources used in information processing.

The present paper is thus close to the boundary between economics and computer science. I believe, however, that its subject is relevant to several concerns that are, or are becoming, central to economic theory. First, there is the question of whether market forces are sufficient to push organizational forms of large firms close to efficiency, i.e., can we "explain" current organizational forms on the grounds of efficiency (much as we sometimes explain the emergence of markets as taking place because of their efficiency). To do this, it would be helpful to know just what are the organizational forms that are efficient under various circumstances. Second, one would like to know how the costliness of information processing contributes to organizational economies or diseconomies of scale. Third, the present paper provides one possible model of "bounded rationality," a topic that is beginning to receive much attention in decision theory and game theory. I hasten to add that in this paper I shall not address these issues directly, but in Sections 7 and 8 I do briefly sketch some ideas on what directions such an effort might take, and also provide references to related research.

Other modes of decentralization follow from the limited capacities of individuals for information-processing and decision-making. It is typically economical to have most decisions, if not all, based on only partial information, and in many cases to have different decisions based upon different information; this latter situation is called *decentralization of information*.[5] The heterogeneity of information among different decision-makers[6] leads inevitably to the *decentralization of incentives*, and thence to problems of moral hazard and misrepresentation.[7] In other words, private information confers power.

In the present paper I shall consider explicitly only the decentralization of information-processing for decision-making, and I shall consider three applications: (i) linear decision rules, (ii) project selection, and (iii) pattern-matching.

[5] See, e.g., Radner (1961, 1962) and Marschak and Radner (1972).
[6] Sometimes called "asymmetric information."
[7] See, e.g., Radner (1987).

In the first, the decision is a linear function of the environmental signals. This corresponds, in particular, to the typical processing of accounting information. Numerical data are rescaled to common units, e.g., dollars, or minute-miles, and then added up. Thus we may think of the calculation of a linear function as occurring in two stages:

1. Each variable is multiplied by its respective coefficient (conversion to a common unit).
2. The resulting products are added up (aggregation).

The items to be aggregated might well be vectors, not just numbers, so that the coefficients are matrices. The decentralization of information processing is dictated by the fact that the number of items to be added (numbers or vectors) is very large.

In the third application, the decision-maker (decision-making organization) compares the "pattern" of data about the environment with the members of a finite set of reference patterns, picking the one that is "closest" in some sense. To each reference pattern corresponds a decision, so that the problem of choosing a decision is reduced to one of finding which reference pattern is closest to (has minimum distance from) the environmental pattern. For example, the data and reference patterns might be represented as vectors in a space of very large (but finite) dimension, and the measure of "closeness" might be ordinary Euclidean distance.

These two applications, as well as project selection, are discussed in Section 6. It is interesting that both "addition" and "finding a minimum" (or a maximum) are associative operations, and (as we shall see) lend themselves naturally to the decentralization of information processing, or what the computer scientist would call *parallel computation*.[8] (In fact, from a formal point of view these concepts are the same.[9]) The present paper is entirely concerned with associative operations, except for a few remarks in Section 7.

Various aspects of the processing of information are costly, and therefore should be "economized:"

1. The observation of the data about the environment.
2. The capabilities and numbers of the individual processors (persons?).
3. The communication network that transmits and switches the data (both original and partly processed) among the processors.
4. The delay between the observation of the data and the implementation of the decision(s).

The last aspect, delay, is costly to the extent that delayed decisions are obsolete (not "timely").

In fact, computer scientists have been largely concerned with delay, that is, the time it takes to compute a particular function. In the present paper, I give

[8] See Schwartz (1980).

[9] Another computer-science term for decentralized processing is "distributed computation." In the computer-science literature, the terms "parallel" and "distributed" connote different sets of research problems (and researchers), but both are concerned with what I have called decentralized information-processing.

equal attention to economizing on the number of processors. I do not consider the cost of communication; this has, I think, some empirical justification. The problem of "information overload" is apparently particularly acute in modern times, and is a reflection of the relative cheapness of communication vs. processing (digestion) of information.

Also, in the present paper I take the amount of environmental data as given, but in fact it should be an endogenous variable, determined by the balance between its cost and its value. The consideration of this complication is, however, deferred to a forthcoming paper.[10]

The basic construct that I shall use to represent decentralized information processing is that of *a (programmed) network of individual processors*. Although I shall consider very general networks, it may be helpful to introduce the idea with the special case of a hierarchical network. Roughly speaking, a hierarchy of processors will be defined as an inverted tree in which the processors are partitioned into levels. There is a single processor at the top of the hierarchy, and, except for the top one, every processor "reports to" exactly one processor directly above it. The levels are ranked, and if one processor is above another, it is also at a higher level; on the other hand, if two processors are at the same level, neither one is above the other in the tree. (For a more formal definition, see Section 2.)

Data about the environment enter the hierarchy at the bottom, different first-level processors receiving data about the different variables. The data items are processed at successive levels, the final "answer" (decision) coming out at the top. Each processor has limited capacity, in that there is a maximum number of items it can process per unit of time. For example, suppose that the hierarchy's task is to calculate the sum of the original data items. (More generally, addition could be replaced by any associative operation.) Suppose further, that each processor has an "in-box" and a register; in each unit of time it can take one item from its in-box and add it to its register. (For simplicity, assume that all of the processors are identical.) When a first-level processor has finished adding all the items in its in-box, it sends the sum in its register to the processor immediately above it in the tree, and resets its register to zero. This procedure is repeated at successively higher levels (with partial sums playing the role of the original data items) until the top processor puts out the final result. The number of units of elapsed time between the entry of the data at the first level and the output of the sum at the top is the *delay*.

Note that each processor is adding items from its in-box in a *serial* fashion, whereas processors at the same level are doing their work *in parallel*. Generally speaking, for a given number of data items, increasing the amount of parallel

[10] For analyses of the cost and value of information in an organizational setting, see Radner (1961) and Marschak and Radner (1972). Geanakoplos and Milgrom (1991) study team-theoretic models of a hierarchical form in which the *acquisition* of information by managers is time-consuming. Their study is thus, in a sense, complementary to the present paper.

processing (and decreasing the amount of serial processing) leads to a decrease in delay and an increase in the number of processors.

I shall in fact consider more general networks in which the processors are connected by some arbitrary pattern—not necessarily hierarchical—of one-way links. A program prescribes the original assignment of data items to the in-boxes of the several processors, the times at which each individual processor sends the contents of its register to the processor(s) to which it is directly linked, and the time at which one designated processor sends out the final answer.

In a decision-making organization, new data about the environment arrive from time to time, and decisions are correspondingly revised. For example, suppose that a "cohort" of N new data items arrives for processing every T units of time; I shall call one unit of time a "cycle." The *delay* for a cohort is the number of elapsed cycles from the time that the cohort arrives until its processing is completed. Suppose, for the purpose of this introduction, that every cohort experiences the same delay, say C. (In general, this need not be the case, even if N and T are the same for all cohorts; see Section 5.) Finally, let P denote the number of individual processors in the network. I shall call a network *efficient*, given N and T, if it is not possible to decrease the delay, C, without increasing the number of processors, P, and vice versa.

This paper focuses on two sets of issues concerning efficient networks. The first is structural: What is the "architecture" of efficient networks? If this architecture is complicated, are there simpler architectures that are close to efficient? In fact, we shall see that efficient networks are characterized by a collection of hierarchical structures, but that individual processors are rotated among these structures in a somewhat complicated way as successive cohorts are processed. On the other hand, we shall also see that some fairly simple hierarchical networks are relatively close to being efficient, when the number of items in each cohort is large. *The reader should not infer from this a justification or explanation of the usual hierarchy of authority in a firm*; in fact, a rather contrary inference is suggested in Section 7.

The second set of issues concerns the properties of the "production function" that corresponds to efficient, or almost efficient, networks: What is the shape of the (P, C) efficiency frontier, given N and T? How does this frontier depend on N and T? It is clear that the answers to these questions will have implications for the standard economic questions that one asks about a "production technology," such as (1) how does the choice of a network depend on the relative "prices" of processors and delay? (2) what are the returns to scale in information processing? We shall see that, when N, P, and $N/(P-N/T)$ are large, the points on the (P, C) efficiency frontier lie approximately on the rectangular hyperbola,

$$C\left(P-\frac{N}{T}\right)=\frac{N}{T}. \tag{1.1}$$

It follows that if the price of processors relative to that of delay is p (finite and strictly positive), then for the optimal choice of C and P,

$$P = \frac{N}{T} + \left(\frac{N}{pT}\right)^{1/2}, \tag{1.2}$$

$$C = \left(\frac{pN}{T}\right)^{1/2}. \tag{1.3}$$

Thus, if the relative price p is held constant while one increases N, the number of items per cohort, then there will be "increasing returns to scale" (although the returns to scale will in the limit be constant as n increases without bound).

To describe more completely how the entire efficiency frontier depends on N and T requires a more precise description than the approximation (1.1), which is valid only under the stated conditions. The reader will have to wait for Section 5 to see such a description, but I do want to state here two lower bounds on the number of processors and the delay:

$$P \geqslant \frac{N}{T}, \tag{1.4}$$

$$C \geqslant 1 + \log_2 N. \tag{1.5}$$

These bounds have interesting implications for returns to scale (in N). The first one states that the number of processors must be at least proportional to the number of items per cohort. The second states that the delay can grow even more slowly than in (1.3), but nevertheless must go to infinity if N does. In fact, one might question the concept of returns to scale that is implicit in (1.2) and (1.3). One could argue that, in this technology, the inputs are the processors, the output is the service of processing the N items, and the delay is a *quality* of the output. From this point of view, (1.5) states that *it is not possible to maintain a constant quality* (C) *while increasing the scale* (N). Thus, in contrast to (1.2)–(1.3), we get a picture of strongly decreasing returns to scale. (For further remarks on this point, see Section 7.)

The results summarized above are presented in Section 5, which contains the core of the paper. Section 2 introduces the idea of a network. Sections 3 and 4 deal with the case in which there is only one cohort of items to be processed; I call this the "one-shot mode," in contrast to the "systolic mode" of Section 5. Although the one-shot mode may be of limited interest in itself, its analysis proves to be central to the analysis of the systolic mode.

In Section 6, I sketch how these ideas can be applied to models of (i) accounting and control (linear operations), (ii) project selection, and (iii) pattern recognition. Section 7 presents some brief concluding remarks about a variety of loose ends. Bibliographic notes are gathered in Section 8, rather than being scattered through the paper. However, no attempt has been made to survey the relevant literature thoroughly.

2. NETWORKS AND HIERARCHIES IN THE ONE-SHOT MODE

The considerations of Section 1 now lead me to consider the following problem: Given N items to be added, and P processors, arrange and program the processors to add the N items in minimum time. (Here, for "add" we can read any associative operation, and the "items" are anything amenable to the associative operation.) As I have stated it, this problem is not well-defined, because I have not been precise about what a processor is. In what follows, a *processor* is an object with an *in-box*, a *register*, and a *clock*. Time is measured in cycles; in one cycle a processor can take one item from its in-box and add it to its register. From each processor there may be one or more one-way communication links to other processors. At prescribed times, a processor can also send the contents of its register to the in-boxes of other processors to which it is directly linked, and then re-initialize its own register to "zero;" this can be done in any cycle without additional elapsed time. Finally, there is a particular processor that, at a designated time, sends out the contents of its register as the result of the computation, i.e., the grand total. The set of processors and links will be called a *network*. The *program* prescribes the original assignment of the items to the several processors' in-boxes, and the times of communication and final output. The combination of a network and a program will be called a *programmed network*. The number of cycles used to perform the computation will be called the *delay*. (For a more formal definition, and its relation to computer science models of parallel processing, see Appendix A1.)

I can now restate the problem: *given P processors, construct a programmed network to add the given N items with a minimum delay.*

A special subclass of networks, called hierarchies, will turn out to be important. In fact, I shall start by defining a more general concept, that of a *tree*. (I fear that the mathematical name may be misleading, since it corresponds more or less to the botanical object with the same name, but upside-down!) Formally, a *tree* is a collection of objects, together with a relation among them, to be called here "superior to." This relation has the following properties:

1. Transitivity—if A is superior to B, and B is superior to C, then A is superior to C.

2. Antisymmetry—if A is superior to B, then B is not superior to A; in this case I shall say that B is *subordinate to* A.

3. There is exactly one object, called the *root*, that is superior to all the other objects.

I shall say that A is the *immediate superior* of B if there is no object that is "between" A and B in the relation. A fourth property required of a tree is:

4. Except for the root, every object has exactly one immediate superior.

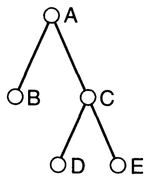

FIGURE 1.—A tree.

Figure 1 shows a tree with 5 objects. The root is at the top! Notice that not all the objects in the tree need be "comparable;" for example, B is not superior to D, nor is D superior to B.

In everyday language, the word "hierarchy" not only connotes an upside-down-tree-like structure, but also an assignment of rank or level. By a ranking of a tree I shall mean an assignment of a number (rank) to each object such that:

1. if A is superior to B, then it has a higher rank (larger number);
2. if A and B have the same rank, then they are not comparable, i.e., A is not superior to B, nor is B superior to a.

I shall adopt the convention that the lowest rank is 1.

I can now define a *hierarchy*; it is a ranked tree. I note that there may be more than one way to rank a tree (in a way that satisfies properties 1 and 2 above); Figure 2 illustrates this. The hierarchies in Figure 2 look like organization charts (for a small organization!). With this interpretation, the relation "superior to" is that of formal authority. However, as we shall see later, efficient networks for information processing need not coincide with hierarchies of authority.

Figure 3 illustrates a hierarchical network, with 15 processors indicated by circles, and 40 items. The links joining the processors are to be understood as pointing upward. One processor is the immediate superior of another if there is a direct link pointing upward from the second to the first. The successive levels in the diagram indicate the ranks, of which there are 4. Here is the way the program works. The 40 items are originally assigned equally to the processors of the lowest level (rank 1). This is indicated in the figures by the 5 lines coming up into each of the 8 lowest-level processors. The computation starts with each first-level processor adding its 5 items into its register. At the end of the 5th cycle each first-level processor sends its partial sum to its immediate superior at

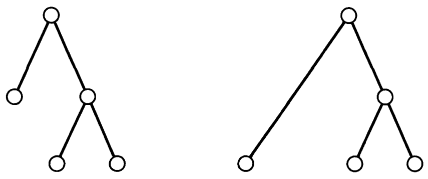
FIGURE 2.—The same tree organized in levels in two different ways.

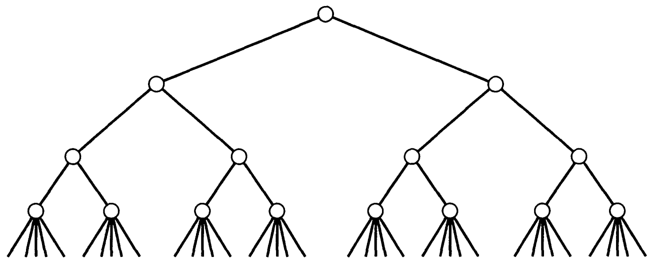
FIGURE 3.—A hierarchical network with 15 processors, 40 items, and delay 11.

the second level. Each second-level processor then takes 2 cycles to add its items and send its partial sum to its third-level immediate superior, etc. At the end of 11 cycles the single fourth-level processor (the root of the tree) puts out the grand total of the 40 items; thus the delay is 11.

As we shall see below, the minimum delay is actually 8, not 11. Indeed, we shall also see that one can achieve the same delay, 8, with fewer than 15 processors, namely 8.

In the subsequent sections, a significant role will be played by hierarchies that display a certain symmetry (as in Figure 3). I shall call a hierarchy *strictly balanced* if (i) all the immediate subordinates of any processor are at the next lower level and (ii) at each level above the first, all members of the same level have the same number of immediate subordinates. All of the processors at one level that are the immediate subordinates of the same processor at the next higher level will be called a *cadre*. The hierarchy in Figure 3 is not strictly balanced, but each cadre has two members. In general, however, in a strictly balanced hierarchy cadres at different levels need not be of the same size.

3. EFFICIENCY OF HIERARCHIES IN THE ONE-SHOT MODE

I shall say that a *programmed network is efficient* for a given number of items if the number of processors cannot be decreased without increasing the delay, or vice-versa. By extension, a network is efficient if it can be programmed to be efficient. Thus, as noted above, we shall see that the network in Figure 3 is not efficient. Finally, I shall say that *the pair* (P, C) *is efficient for* N *items* if there is a programmed network with P processors that is efficient for N items and adds them with a delay equal to C.

Although the hierarchical network of Figure 3 is not efficient, one can show that hierarchical networks are sufficient to attain efficiency, in the following sense: *For any number of items,* N*, and any pair* (P, C) *that is efficient for* N*, there is a tree with* P *processors that can add the* N *items in* C *cycles.* This proposition is based on the following solution to the delay-minimization problem.

For any real number x, let $\lceil x \rceil$ denote the smallest integer $\geqslant x$ (ceiling), and $\lfloor x \rfloor$ the largest integer $\leqslant x$ (floor).

TABLE I

MINIMUM DELAY (C) FOR ADDING 40 ITEMS[a]

P	1	2	3	4	5	6	7	8	9	10	11	12	13	⋯	20	⋯	40
Min C	40	21	15	12	11	10	9	8	8	8	8	7	7	⋯	7	⋯	7
	*	*	*	*	*	*	*	*				*					

[a]P = number of processors, * denotes an efficient network.

THEOREM 1: *Given $N \geqslant P \geqslant 1$, the minimum number of cycles needed to add N items with P processors, using any network, is*

$$\hat{C} = \left\lfloor \frac{N}{P} \right\rfloor + \lceil \log_2(P + N \bmod P) \rceil, \tag{3.1}$$

and is attainable by a tree.

Although this proposition seems to be more or less familiar to computer scientists interested in parallel computation (see, e.g., Gibbons and Rytter (1988)) a complete statement and proof appears not to be easily accessible. I have therefore provided a proof in Appendix A2.

Table I illustrates formula (3.1) for $N = 40$ and P varying from 1 to 40. A one-processor hierarchy is the slowest, with a delay of 40 cycles. The minimum delay is 7 cycles, and can be attained with 12 processors; the use of more processors will not further reduce the delay. Thus a network with more than 12 processors is not efficient for 40 items. Similarly, we see from the table that networks with 9, 10, and 11 processors are not efficient, either, since a delay of 8 cycles can be attained with 8 processors. Thus, although the formula gives the minimum number of cycles for any number of processors, not every pair (P, C) generated by the formula is efficient. Figure 4 shows a graph of Table I, and illustrates the same phenomenon. For very large numbers of items, however, these inefficiencies will not be very significant unless the number of processors is relatively large. For example, Figure 5 shows a plot of the minimum delay vs.

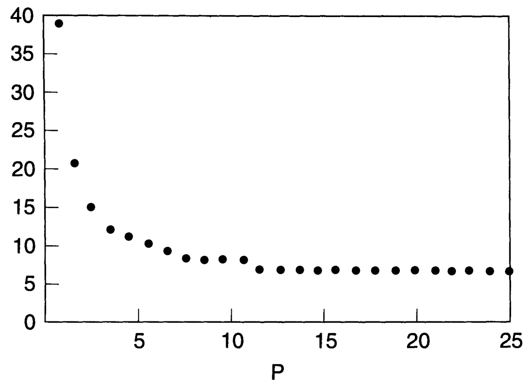

FIGURE 4.—$N = 40$. Min C vs. P.

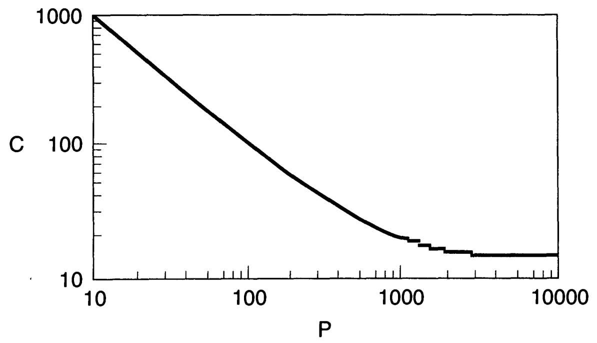

FIGURE 5.—$N = 10{,}000$. Min C vs. P.

the number of processors for $N = 10{,}000$; inefficiency does not become a significant problem until the number of processors exceeds 2000 (approximately).

Figures 4 and 5 illustrate the trade-off between delay and number of processors. This tradeoff is a reflection of the trade-off between serial and parallel processing. With few processors, there is a lot of serial processing of the items at the first level, which causes a large delay. Many processors permit a lot of parallel processing, which reduces the delay.

It is interesting to see how a strictly balanced hierarchy can be "reduced" so as to decrease both the total number of processors (P) and the delay (C). I suppose we start with the N items allocated equally (or as equally as possible) among the lowest-level processors. The reduction will be done in stages. At stage 1, we eliminate 1 member of each cadre at level 1, and assign its items to its corresponding immediate superior. Figure 6a shows the hierarchy of Figure 3, but with each group of 5 items at the bottom replaced by a triangle, or "fan." Figure 6b shows the result of applying the first stage of reduction. Since there were originally 4 cadres at level 1, 4 processors have been eliminated at level 1, reducing the total number of processors from 15 to 11. Each second-level processor now has 5 items plus 1 first-level processor assigned to it.

I shall call the items and/or immediately subordinate processors assigned to a processor its *predecessors*. Let R denote the number of levels in the hierarchy. The reduction procedure is completed in stages as follows: at stage $r < R$, one processor is eliminated from each level-r cadre, and its predecessors are assigned to its immediate superior at level $r + 1$.

Figures 6c and 6d show the second and third stages of reduction for the hierarchy of Figure 6a. The number of processors has been reduced from 15 to 8, and the number of cycles from 11 to 8. (One can show that it is no coincidence that the new number of processors equals the old number of first-level processors.)

In general, the minimum delay in (3.1) is attained by a hierarchy of the type that one gets from reducing a balanced hierarchy, as described above. Indeed, it

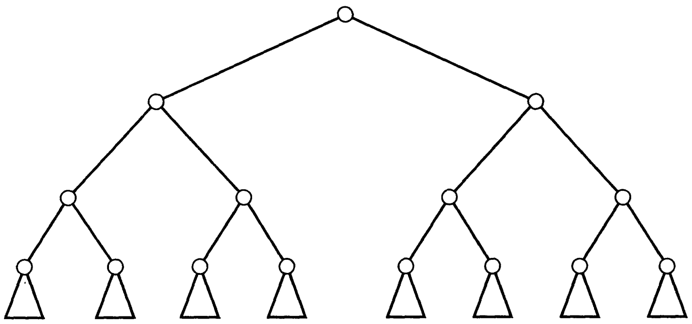
FIGURE 6a.—Balanced hierarchy before reduction.

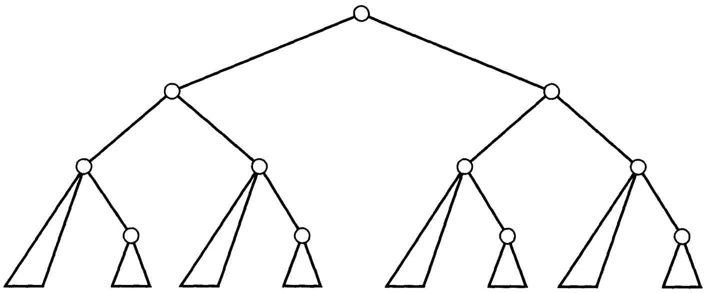
FIGURE 6b.—Balanced hierarchy after 1 stage of reduction.

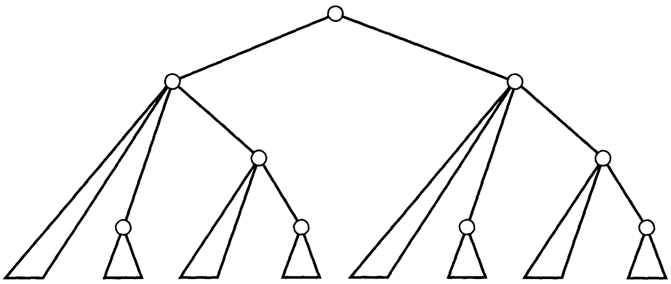
FIGURE 6c.—Balanced hierarchy after 2 stages of reduction.

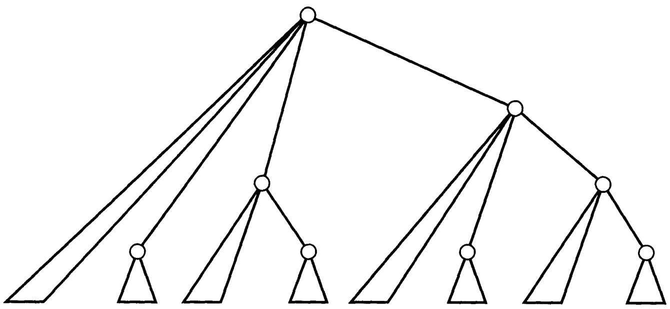
FIGURE 6d.—Balanced hierarchy after 3 stages of reduction (final).

is attained by reducing a balanced hierarchy whose cadres originally all have size two! (Note that the hierarchy of Figure 6a does have this property.)

There is, however, something odd about Figure 6d, at least as a picture of an organizational hierarchy. We see that the top-ranking processor has immediate subordinates at all levels. In fact, a similar phenomenon is repeated at each lower level. Reporting through skipped levels is not unheard of in corporate hierarchies (in fact, at AT&T this is called "skip-level reporting"), but the practice does not seem to be as widespread as the above reduction process would suggest.

I now explore how the set of efficient pairs (P, C) depends on the number of items, N. First even if the processors are unlimited, the minimum delay increases like $\log N$. In fact, from (3.1), we have:

COROLLARY 1: *For* $N \geqslant 1$,

$$\min_P \hat{C} = 1 + \lceil \log_2 N \rceil, \tag{3.2}$$

and this minimum is attained for all $P > N/2$.

PROOF: See Appendix A2.

If N is a multiple of P and P is a power of 2, then (3.1) becomes

$$\hat{C} = \frac{N}{P} + \log_2 P \equiv f(P). \tag{3.3}$$

In fact, if N, P, and N/P are all large, then (3.3) will be a good approximation to (3.1) in any case. If we think of P as a continuous variable (rather than integer-valued), then $f(P)$, the right-hand side of (3.3), is strictly decreasing and convex in P for $1 \leqslant P < N \ln 2$. In fact, by the Corollary, we want to restrict P in (3.3) so that

$$1 \leqslant P \leqslant \frac{N}{2}. \tag{3.3a}$$

(Note that $N/2 < N \ln 2 \approx .693N$.)

Since f is strictly decreasing and convex on the interval (3.3a), every point on the graph of f minimizes some positive linear function of C and P, say

$$L = \gamma C + \phi P. \tag{3.4}$$

We can interpret ϕ as the unit cost of processors, and γ as the unit "cost" of cycles of delay. However, since the "cost" of delay derives from the decision problem in which the information is used, the assumption of linearity may or may not be appropriate; see Section 7. Nevertheless, the rates at which the minimum value of L and the corresponding optimal pairs (P, C) increase with N tell us something about how the efficiency frontier moves out with N.

We can rewrite (3.3) as

$$\hat{C} = \frac{N}{P} + \frac{\ln P}{\ln 2}.$$

Substituting the right-hand side for C in the total cost, we see that the first-order condition for minimum cost, given N, is

$$-\frac{\gamma N}{P^2} + \frac{\gamma}{P \ln 2} + \phi = 0,$$

or equivalently,

$$\alpha P^2 + \frac{P}{\ln 2} - N = 0, \quad \alpha \equiv \frac{\phi}{\gamma}.$$

Solving this quadratic equation for the positive root, we get

$$P = \frac{-\frac{1}{\ln 2} + \left(\frac{1}{(\ln 2)^2} + 4\alpha N\right)^{1/2}}{2\alpha}. \tag{3.5}$$

One easily verifies that, as N increases without bound,

$$P \sim \left(\frac{N}{\alpha}\right)^{1/2}, \quad C \sim (\alpha N)^{1/2}, \quad \min L \sim (\gamma \phi N)^{1/2}. \tag{3.6}$$

$$\lim_{N \to \infty} \frac{C}{P} = \alpha.$$

Hence, if we look at points on the efficiency frontier where the slope, dC/dP, equals $-\alpha$, then these points move out asymptotically at the rate $\sqrt{N}$, approximately along a ray of slope α. Also, from (3.3) and (3.6), the efficiency frontier is approximately

$$PC \approx N$$

for large N, P, and N/P.

4. BALANCED HIERARCHIES

Balanced hierarchies were defined in Section 2. For reasons that will be given below (Sec. 5), in the discussion of the "systolic mode," balanced hierarchies may be interesting to study, even though the efficient one-shot trees of the previous section were not balanced.

Recall that a balanced hierarchy has levels (or ranks), with the bottom level numbered 1 and the top level numbered R. Let p_r denote the number of processors at level r. I shall use here a slightly more general definition of balancedness than that of Section 2. The N items are divided as equally as possible among the p_1 processors at the first level. For $r \geqslant 2$, let k_r denote the maximum number of immediate subordinates of any processor at level r; the number of processors at level r is determined by

$$(4.1) \qquad p_r = \left\lceil \frac{p_{r-1}}{k_r} \right\rceil, \quad 2 \leqslant r \leqslant R,$$

and the number of levels, R, by

$$(4.2) \qquad p_R = 1.$$

Thus each processor (except possibly one) at level $r \geqslant 2$ has k_r immediate subordinates (also called the *fan-in*); if p_{r-1} is not an integer multiple of k_r then the last processor at level r has $p_{r-1} \bmod k_r$ immediate subordinates. Each processor at level 1 has either k_1 or $k_1 - 1$ original data items assigned to it, where

$$(4.3) \qquad k_1 = \left\lceil \frac{N}{p_1} \right\rceil.$$

Thus the parameters of the balanced hierarchy are $p_1, k_2, \ldots, k_R$. (Note that if $p_{R-1} < k_R$, then the single processor at level R—the root—will have only p_{R-1} immediate subordinates.)

I shall not give a detailed analysis of balanced hierarchies here.[11] I note only that, if one wants to minimize a positive linear function of the delay and the number of processors, as in (3.4), then (roughly speaking) the optimal fan-in, k_r, will decrease rapidly with r, for sufficiently large N. This suggests studying a special case of a balanced hierarchy, which I shall call a *preprocessing/tree* (PPT) network. In a PPT, $k_2 = \cdots = k_R \equiv K \geqslant 2$. Thus—given N—a PPT has only 2 parameters: the number of preprocessors at the first level, and K, the fan-in at every level above that. Corresponding to (4.1), the number of processors at each level r above the first is

$$(4.4) \qquad p_r = \left\lceil \frac{p_{r-1}}{K} \right\rceil, \qquad 2 \leqslant r \leqslant R.$$

Roughly speaking, the number k_1, in (4.3), measures the amount of serial "preprocessing" done by first-level processors, and K measures the amount of serial processing done at higher levels. Figure 3 showed a PPT with $N = 40$, $p_1 = 8$ and $K = 2$. The delay is $C = 11$, and the total number of processors is $P = 15$.

I shall say that a PPT is (*relatively*) *optimal* if it minimizes the cost

$$(4.5) \qquad L \equiv \gamma C + \phi P$$

within the class of PPTs, where γ and ϕ are strictly positive parameters. We shall see that in a relatively optimal PPT, when N is large, k_1 is much larger than K. I shall in fact give formulas for the optimal k_1 and K. In particular, these formulas imply that k_1, C, and P are all $O(\sqrt{N})$ for optimal PPTs, and

[11] See Radner (1990).

TABLE II
RELATIVE INEFFICIENCY OF PREPROCESSING / TREE NETWORKS

N	10^4	10^6	10^8	10^{10}
P	10^2	10^3	10^4	10^5
$\hat{C}$	107	1010	10014	100017
$\hat{C}_{PPT}$	131	1095	10294	100922
K	10	27	79	242
$\dfrac{\hat{C}_{PPT}}{\hat{C}}$	1.224	1.084	1.028	1.009

that K/k_1 goes to zero. To be precise, we shall see that, for optimal PPTs,

$$(4.6) \qquad C \sim (\alpha N)^{1/2}, \qquad P \sim \left(\frac{N}{\alpha}\right)^{1/2},$$

$$k_1 \sim (\alpha N)^{1/2}, \qquad K \sim \frac{(4\alpha N)^{1/4}}{\ln N},$$

where $\alpha = (\phi/\gamma)$.

An interesting consequence of these formulas is that *asymptotically, relatively optimal PPTs are relatively as efficient as optimal (nonbalanced) trees*, in the sense that, as N increases without bound, the percentage loss due to using optimal PPTs tends to zero. On the other hand, the difference in the costs tends to infinity. (Compare the preceding formulas with (3.6).) The relative inefficiency of PPTs is illustrated in Table II.

If p_1 is a divisor of N and a power of K I shall say that the PPT is *strictly balanced* (this corresponds to the definition in Section 2). In this case one can give closed-form expressions for P and C. These expressions are also useful approximations when N is large. For a strictly balanced PPT, (4.3) and (4.4) imply

$$(4.7) \qquad p_1 k_1 = N,$$

$$(4.8) \qquad p_r = K^{R-r}, \qquad 1 \leqslant r \leqslant R.$$

Hence the number processors in levels 1 to R is

$$K^{R-1} + K^{R-2} + \cdots + 1 = \frac{K^R - 1}{K - 1}.$$

On the other hand, from (4.7) and (4.8),

$$(4.9) \qquad \frac{N}{k_1} = K^{R-1},$$

so the total number of processors is

$$(4.10)\quad P = \frac{\dfrac{KN}{k_1} - 1}{K-1}.$$

The total number of cycles is

$$C = k_1 + (R-1)K.$$

Hence, by (4.9)

$$(4.11)\quad C = k_1 + K \log_K\left(\frac{N}{k_1}\right) = k_1 + \left(\frac{K}{\ln K}\right)(\ln N - \ln k_1).$$

For the remainder of this section I shall focus on strictly balanced optimal PPTs using (4.10) and (4.11), which can also be interpreted as approximations to the precise values if the assumption of strict balancedness is not satisfied. This procedure gives the correct asymptotics when N is large. One can show (see Appendix A3), that, for relatively optimal PPTs,

$$(4.12)\quad k_1 \sim (\alpha N)^{1/2}, \qquad p_1 \sim \left(\frac{N}{\alpha}\right)^{1/2}, \qquad K \sim \frac{(4\alpha N)^{1/4}}{\ln N}.$$

It then follows from (4.10) and (4.11) that C, P, and L are all $O(\sqrt{N})$.

5. THE SYSTOLIC MODE

5.1. *Periodic Arrival of Data*

I must now turn our attention to a fact about real organizations that we have thus far ignored, namely, that typically new data about the environment will be coming in periodically, with the consequence that new decisions must be calculated periodically. Adopting a term from computer science, I shall call this the *systolic mode*, to distinguish it from the *one-shot* mode discussed in the preceding sections. I shall assume that, in the systolic mode, every T cycles a new *cohort* of data arrives to be processed; each cohort is made up of N data items. The delay for a cohort is the number of cycles that elapse from the arrival of the cohort to the completion of its processing; this cohort delay includes (1) the *waiting time*, if any, between the arrival of the cohort and the beginning of the processing, and (2) the actual *processing time*. The *delay for a network* is the long-run average of the successive cohort delays.[12]

For example, consider the one-shot-efficient networks described in Section 3 (e.g., as illustrated in Figure 6d). If the "one-shot" delay, say $\hat{C}$, does not exceed T, the time between cohorts, then the same tree can be used for every successive cohort, with a long-run average delay equal to $\hat{C}$. However, if $\hat{C} > T$, and one tries to use the same tree, the waiting times of successive cohorts will

[12] Note that I am implicitly considering only networks for which this long-run average exists.

increase without bound. This is because, in the one-shot-efficient tree, the top processor is busy all the time, and hence the processing of one cohort cannot begin until the processing of the previous cohort has been completed.

One remedy for the situation just described, with $\hat{C} > T$, is to assign a new tree to each succeeding cohort until the first tree is free, after which the same trees can be re-used cyclically; I shall call this *replication* of the one-shot-efficient tree (ROSE). If $\lceil \hat{C}/T \rceil = r$, then r replicates will be needed; if each tree uses $\hat{P}$ processors, then the total number of processors in the ROSE network will be $r\hat{P}$, and the network delay will be $\hat{C}$.

Another remedy is to use a balanced hierarchy (cf. Sec. 4) in which the processing time at each level does not exceed T. For example, if the processing time at each level is exactly T, then as soon as a level completes its work on one cohort it can go on to work on the next cohort, and there will be no idle processors in any cycle. This contrasts with the ROSE network, in which there are bound to be idle processors if $\hat{P} > 1$.

The concept of efficiency can be extended to the systolic mode in the natural way; in the remainder of this section, "efficient" will mean "systolic-mode-efficient" unless otherwise indicated. The problem of characterizing efficient networks in the systolic mode has been solved by T. Van Zandt (1990), but his results cannot be described precisely in a brief and simple way. Here, I shall give a simple lower bound for the set of feasible pairs, (P, C). Van Zandt's result implies that this bound is fairly sharp under certain conditions. In addition, I shall describe some fairly simple, but not fully efficient, networks that get close to the lower bound.[13]

5.2. *A Lower Bound for Steady Networks*

Let $\boldsymbol{P}$ be a network with P processors, let Q^m be the number of processors actually used by cohort m, and C^m be the delay for cohort m $(m = 1, 2, \dots)$. I shall call $\boldsymbol{P}$ *steady* if the two long-run-averages,

$$\bar{C} \equiv \lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} C^m, \tag{5.1}$$

$$\bar{Q} \equiv \lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} Q^m, \tag{5.2}$$

exist. For every positive integer N, define the function f_N by

$$f_N(x) = \frac{N}{x} + \log_2 x, \qquad x \geqslant 1. \tag{5.3}$$

THEOREM 2: *Let $\boldsymbol{P}$ be a network with P processors, in which a cohort of N items arrives every T cycles. If $\boldsymbol{P}$ is steady, with $\bar{C}$ and $\bar{Q}$ defined as in* (5.1) *and*

[13] These results are apparently not in the computer science literature.

(5.2), *resp., then*

$$(5.4)\qquad \bar{C} \geqslant f_N(\bar{Q}) \equiv C_L,$$

$$(5.5)\qquad P \geqslant \frac{N-1+\bar{Q}}{T} \equiv P_L.$$

PROOF:[14] By the theorem of Section 3, for every cohort m,

$$(5.6)\qquad C^m \geqslant \left\lfloor \frac{N}{Q^m} \right\rfloor + \lceil \log_2 (Q^m + N \bmod Q^m) \rceil ;$$

cf. (3.1). We need the following:

LEMMA 1: *For every strictly positive integer $q \leqslant N$,*

$$(5.7)\qquad \left\lfloor \frac{N}{q} \right\rfloor + \lceil \log_2 (q + N \bmod q) \rceil \geqslant f_N(q).$$

PROOF: Write

$$N = nq + r, \qquad 0 \leqslant r < q.$$

We wish to show that

$$n + \lceil \log_2 (q+r) \rceil \geqslant \frac{nq+r}{q} + \log_2 q,$$

or that

$$\lceil \log_2 (q+r) \rceil - \log_2(q) - \frac{r}{q} \geqslant 0.$$

For this, it is sufficient to show that

$$\log_2(q+r) - \log_2(q) - \frac{r}{q} \geqslant 0,$$

or that

$$(5.8)\qquad g(x) \equiv \log_2(1+x) - x \geqslant 0, \qquad 0 \leqslant x < 1.$$

However, g is concave, and $g(0) = g'(1) = 0$, which proves (5.8), and hence the lemma.

From the lemma and (5.6), we see that, for every cohort m,

$$(5.9)\qquad C^m \geqslant f_N(Q^m).$$

[14] Unlike the corresponding theorem for the one-shot mode, this result appears to be new to the computer science literature.

It is straightforward to verify that the function f_N is continuous and convex on the interval $[1, N]$; hence, for every M, by (5.9),

$$\frac{1}{M}\sum_{m=1}^{M} C^m \geqslant \frac{1}{M}\sum_{m=1}^{M} f_N(Q^m) \geqslant f_N\left(\frac{1}{M}\sum_{m=1} Q^m\right),$$

and, letting M increase without bound we get (5.4).

To prove (5.5), first note that the addition of N items requires exactly $(N-1)$ additions; hence at least $(N-1)$ processor-cycles are needed. Furthermore, if Q^m processors are actually used for cohort m, then an additional Q^m processor-cycles are needed to add the first item in every processor's in-box into its register. Hence the total number of processor-cycles used in cohort m is at least

$$W_m \equiv N - 1 + Q^m,$$

and the long-run-average number of processor-cycles used *per cycle* is at least as large as $(N - 1 + \bar{Q})/T$, which proves (5.5), and completes the proof of the theorem.

The preceding theorem provides a curve L of points $(P_L(\bar{Q}), C_L(\bar{Q}))$, parameterized by $\bar{Q}$ as in (5.4) and (5.5), which forms a lower bound on all points $(P, \bar{C})$ that can be attained by steady networks (given N and T). It is interesting to compare this bound with the corresponding one for the one-shot mode (Sec. 3). As in that case, we have

$$\text{(5.10)} \qquad \bar{C} \geqslant 1 + \log_2 N,$$

and this bound can actually be attained with equality if N is a power of 2, for example by taking $\bar{Q} = N/2$ and

$$P = \left\lceil \frac{1 + \log_2 N}{T} \right\rceil \frac{N}{2}.$$

In other words, the systolic mode can be as fast as the one-shot mode if one is willing to use enough processors (this was already evident from the preceding discussion of ROSE networks). On the other hand, from (5.5), $P \geqslant N/T$, so P must grow at least proportionality to N, for fixed T. This contrasts strongly with the one-shot case, in which P can be reduced to 1 (at the expense, of course, of making $C = N$).

5.3. *Efficient Networks*

The previously cited work by Van Zandt (1990) has implications for the sharpness of the bounds (5.4) and (5.5). He considers a slightly smaller class of networks, called *stationary*,[15] in which Q^m (the number of processors used by cohort m) and C^m (the delay for cohort m) are each the same for all cohorts m;

[15] I shall not give a precise definition here.

call the respective values Q and C. Let $\hat{P}(C,N,T)$ be the minimum number of processors in a stationary systolic network needed to achieve a delay C when a cohort of N items arrives every T cycles, and let $P_L(C,N,T)$ be the corresponding lower bound in (5.5). Fix T, and for each N let C_N be a feasible delay, i.e.,

$$1+\lceil \log_2 N \rceil \leqslant C_N \leqslant N.$$

Finally, let q_N denote the minimum number of processors needed to process N items in C_N cycles in a *one-shot* network.

Proposition 1 (Van Zandt, 1990):[16]
1. *If* $N \to \infty$ *and* $q_N/N \to 0$, *then*

$$\frac{\hat{P}(C_N,N,T)}{P_L(C_N,N,T)} - 1 = o\left(\left[\frac{q_N}{N}\right]^2\right).$$

2. *If* $q_N < N^{1/2}$ *and* $T \leqslant (N^{1/2}/2) - 2$, *then*

$$\hat{P}(C_N,N,T) - P_L(C_N,N,T) \leqslant 2.$$

To interpret the proposition, observe that any *upper* bound on the rule at which q_N increases with N corresponds to a *lower* bound on the rate at which C_N increases with N. To make this more precise, let $\hat{c}(q,N)$ denote the minimum number of cycles needed to process N items in the one-shot mode with q processors. From (3.1),

$$\begin{aligned}\hat{c}(q,N) &= \left\lfloor \frac{N}{q} \right\rfloor + \lceil \log_2 (q + N \bmod q) \rceil \leqslant \frac{N}{q} + \log_2(2q) + 1 \\ &= \frac{N}{q} + \log_2 q + 2 = f_N(q) + 2;\end{aligned}$$

cf. (5.3). Hence, for any $\bar{q}$ between 1 and N,

$$C_N > f_N(\bar{q}) + 2 \Rightarrow C_N > \hat{c}(\bar{q},N) \Rightarrow q_N < \bar{q}.$$

In particular, taking $\bar{q} = N^{1-\alpha}/\beta$, with $\beta > 0$, $0 < \alpha \leqslant 1$, we get

$$C_N > \beta N^{\alpha} + (1-\alpha)\log_2 N - \log \beta + 2 \Rightarrow q_N < N^{1-\alpha}/\beta \Rightarrow q_N/N \to 0,$$

so that Part 1 of the Proposition is applicable. To apply Part 2, take $\alpha = 1/2$ and $\beta = 1$ in the above, so that

$$C_N > \sqrt{N} + \log_2 \sqrt{N} + 2 \Rightarrow q_N < \sqrt{N}.$$

Figure 7 illustrates efficient (stationary) networks and the lower bound, for $N = 10^6$.

[16] See also Radner and Van Zandt (1992) for a summary of this and related results.

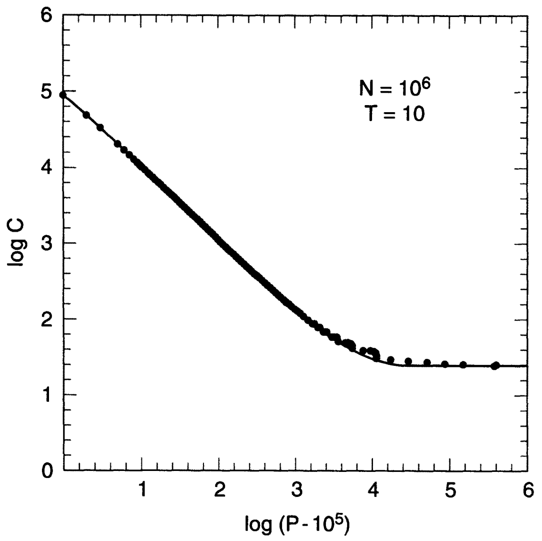

FIGURE 7.—Efficient systolic-mode networks and the lower bound.

5.4. *Some "Good" Balanced Trees*

In this subsection I consider two families of rather simple networks that come "close" to efficiency, and also close to the lower bound of Section 5.2. The first family is adapted from the preprocessing/tree networks of Section 4. For reasons that will become clear in a moment, I shall call them *preprocessing/overhead* (PPO) networks.

A PPO network is characterized by two parameters, p_0 and K, with

$$(5.11)\quad \begin{aligned} &1 \leqslant p_0 \leqslant N, \\ &2 \leqslant K \leqslant T. \end{aligned}$$

Each cohort of N items is assigned to a group of p_0 *preprocessors* as equally as possible. After the group of preprocessors has finished, their partial sums are passed up to the first level of a balanced hierarchy with fan-in K at every level; this hierarchy will be called the *overhead tree*. The group of preprocessors will then turn to the next available cohort. If the preprocessing time of the first cohort exceeds T, then a second group of preprocessors will be assigned to the second cohort, etc., until the first group is free. On the other hand, since $K \leqslant T$, only one overhead tree is needed; after K cycles spent on one cohort, the first level is free to process the preprocessed partial sums from the next cohort. Typically, different levels of the overhead tree will be working simultaneously on different cohorts (this is sometimes called "pipelining"). Note that the overhead tree is acting like a PPT of Section 4 with respect to each group of preprocessors.

If we number the levels in the overhead tree $r = 1, \ldots, R$, and denote the number of processors at level r by p_r, then

$$(5.12) \quad p_r = \left\lceil \frac{p_{r-1}}{K} \right\rceil, \qquad 1 \leqslant r \leqslant R,$$

and R is determined by

$$(5.13) \quad p_R = 1;$$

cf. (4.4). The number of preprocessing cycles in each cohort is

$$(5.14) \quad k \equiv \left\lceil \frac{N}{p_0} \right\rceil.$$

Hence the number of groups of preprocessors is

$$(5.15) \quad \left\lceil \frac{k}{T} \right\rceil = \left\lceil \left(\left\lceil \frac{N}{p_0} \right\rceil / T \right) \right\rceil,$$

the total number of preprocessors is

$$p_0 \left\lceil \frac{k}{T} \right\rceil,$$

and the total number of processors is

$$(5.16) \quad P_0 \equiv p_0 \left\lceil \frac{k}{T} \right\rceil + \sum_{r=1}^{R} p_r.$$

For $1 \leqslant r < R$, each cohort spends K cycles at level r of the overhead tree; at level 1 it spends $(p_{R-1} \bmod K)$ cycles. Hence the delay for each cohort is

$$(5.17) \quad C_0 \equiv \left\lceil \frac{N}{p_0} \right\rceil + (R-1)K + p_{R-1} \bmod K.$$

In the special case in which p_0 is a divisor of N and a power of K, and T is a divisor of k, I shall say that the PPO is *strictly balanced*, and (5.16) and (5.17) become:

$$(5.18) \quad P_0 = \frac{N}{T} + \frac{p_0 - 1}{K - 1},$$

$$(5.19) \quad C_0 = \frac{N}{p_0} + K \log_K p_0.$$

Given N and T, as we vary the parameters p_0 and K, subject to (5.11), we get a family of points (P_0, C_0). Since we have a two-parameter family, it is not implausible that some pairs are "more efficient" than others. I shall say that a PPO is *relatively efficient* if it is efficient within the family of PPOs. We can study the relatively efficient PPOs in a way that corresponds to (3.3)–(3.6) for the one-shot mode. Given N and T, we wish to choose p_0 and K to minimize

$$L_0 \equiv C_0 + \alpha P_0, \tag{5.21}$$

where α is a fixed, positive parameter. One can show (see Appendix A4) that, as N gets large with T fixed, for the optimal choice of p_0 and K,

$$K = T, \tag{5.22}$$

$$p_0 \sim \left[\frac{(T-1)N}{\alpha}\right]^{1/2}, \tag{5.23}$$

$$C_0 \sim \left[\frac{\alpha N}{T-1}\right]^{1/2}, \tag{5.24}$$

$$P_0 - \frac{N}{T} \sim \left[\frac{N}{\alpha(T-1)}\right]^{1/2}, \tag{5.25}$$

$$L_0 - \frac{\alpha N}{T} \sim 2\left[\frac{\alpha N}{T-1}\right]^{1/2}. \tag{5.26}$$

Thus, if the cost of delay is linear in delay, there are increasing returns to scale, but asymptotically returns to scale are constant. In fact, we know from (5.5) that the latter is unavoidable, and this is an important respect in which the systolic mode differs from the one-shot mode.

It is interesting to observe, in addition, that

$$\lim_{N\to\infty} \frac{C_0}{P_0 - \frac{N}{T}} = \alpha, \tag{5.27}$$

$$C_0\left(P_0 - \frac{N}{T}\right) \sim \frac{N}{T-1}. \tag{5.28}$$

Note that, by (5.5), for any steady network $P \geqslant N/T$, so that $(P-(N/T))$ represents the part of the total number of processors that can actually be controlled.

The asymptotic results (5.22)–(5.25) enable us to compare the performance of optimal PPOs with the lower bound (5.4)–(5.5), for large N. One can show (see Appendix A4) that, if we choose $\overline{Q}$ in (5.5) to make $P_L = P_0$, then

$$\lim_{N\to\infty} \frac{C_0}{C_L} = \frac{T}{T-1}. \tag{5.29}$$

Thus, for large N, the delay for an optimal PPO (in the strictly balanced case) exceeds the lower bound by at most a constant factor, when the number of processors is the same as the lower bound. Asymptotically, this factor is at most $2(T=2)$, and approaches 1 as T gets large.

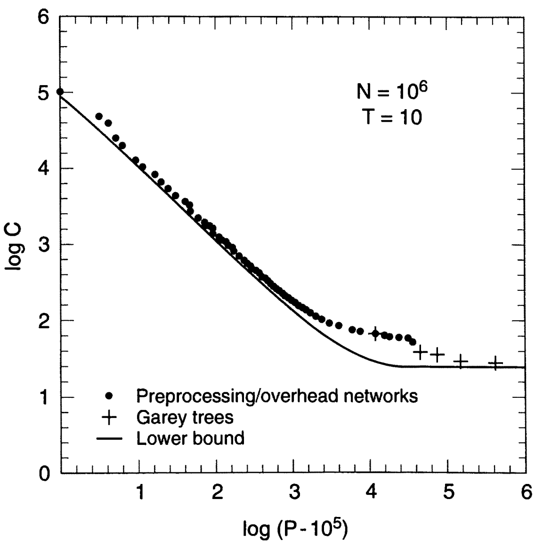

FIGURE 8.—Some good balanced trees.

From (5.24) we see that, for "optimal" PPO networks, the delay grows asymptotically like $\sqrt{N}$, *when we fix the "relative prices" of processors and delay.* On the other hand, we know from (5.10) and the discussion following it that delay can be made to grow more slowly, i.e., like $\log_2 N$, provided the number of processors is increased fast enough. The latter corresponds to letting the relative price of processors approach zero as N increases. In fact, one can show that the fastest PPO networks are attained for $K = 3$, and that for the fastest (strictly balanced) PPO networks

(5.30) $\quad \min C_0 \sim 1.89 \log_2 N.$

Another family of simple networks, suggested by M. R. Garey,[17] performs well in the (P, C) region where C is close to its minimum. The basic idea is to construct a balanced hierarchy of "modules," in which each module is itself a one-shot-efficient tree whose delay equals T (or is as close to T as possible without exceeding it). I shall call these *Garey trees*; a precise definition and analysis is given below.

Figure 8 illustrates the foregoing ideas for the case in which $N = 10^6$ and $T = 10$. It shows (1) the lower bound discussed in Section 5.2, represented by the continuous curve, (2) selected points corresponding to PPO networks, represented by solid dots, and (3) selected points corresponding to Garey trees, represented by "plus" marks. The horizontal axis measures $P - (N/T) = P - 10^5$, and the vertical axis measures C, both on logarithmic scales. We see that (i) the lower bound looks similar to the efficiency frontier in the one-shot-mode (Fig. 5); (ii) the PPO networks are quite close to the lower bound, except when

[17] Private communication.

the number of processors is very large; (iii) the Garey trees are also close to the lower bound, but their range is limited to the region of faster networks with many processors.

A more precise description of Garey trees and their performance characteristics is given in Appendix A5.

6. APPLICATIONS

6.1. *Linear Operations in Accounting and Control*

I have already mentioned (in Sec. 1) the use of linear operations in typical accounting processes, and I shall not repeat those remarks here. I only note here that in computing a linear function, say

$$c_1x_1 + \cdots + c_Nx_N,$$

the individual products, c_ix_i, can be computed in parallel, and then added as described in Sections 3–5.

Linear operations are also common in the methods of statistical prediction and control, especially under the popular assumptions of Gaussian uncertainty and quadratic payoff functions. (For material on this subject in a specifically organizational setting, see Marschak and Radner (1972), Aoki (1986), and Radner and Van Zandt (1992).) Here the situation is usually explicitly dynamic, involving both the periodic aggregation of contemporaneous data and the updating of previous predictions and/or actions using the new aggregates. The reader is referred to Radner and Van Zandt (1992) for an analysis of examples of this kind in the context of parallel computation.

6.2. *Project Selection*

In this section I shall sketch the application of the preceding ideas to some models of project selection. I start with the simplest case. There is given a set of N projects, $X_1, \dots, X_N$. I shall write $X_i \gtrsim X_j$ if X_i is "at least as good as" X_j. The task is to find a best project. In one cycle a processor can compare a project in its in-box to one in its register, and keep the better one in its memory, discarding the other. (If they are equally good, the one that was in the memory is kept there.) If the memory is empty, then it takes one cycle to put a project from the in-box into the memory. The interpretation is that it takes one cycle to "evaluate" a project; a project in the memory has already been evaluated, but one in the in-box has not. A processor can also send a project from its memory to another processor's in-box, in zero time. However, the next processor must then "evaluate" the project all over again. (One processor cannot simply accept another processor's evaluation of a project!) As just stated, the task of finding a best project out of N is isomorphic to the task of adding N items, and the results of Sections 3–5 are immediately applicable.

The situation is more complicated if the task is one of finding the best M projects out of the original N. To handle this task, the capability of the processors must be enhanced. For example, suppose that each processor can store M evaluated projects in its memory, in rank order. Furthermore, in one cycle a processor can evaluate a project from its in-box and insert it in the proper order in its memory, discarding the worst project if it has $M+1$ projects there. Such a processor can take n projects from its in-box and pick the best M of them in n cycles. The delay for a tree of such processors can be calculated as follows. At the preprocessing stage (lowest level), a processor receives $n \geqslant N$ projects, which it processes in n cycles; the best M of these are sent up to its immediate superior. Whenever a processor receives a batch of M projects from an immediate subordinate, it requires M cycles to process the batch. With this modification, the analyses corresponding to Sections 3–5 can be carried out in the obvious way.

If M is large, it may be of interest to adopt a procedure in which processors at level r send up a number of projects, say m_r, where m_r is significantly smaller than M at the lower levels. However, with such a procedure, there will be a positive probability that the M projects finally selected will not be the M best in the original set of N. The problem is then to balance this risk against the reduction of delay. A thorough discussion of this and other methods is beyond the scope of this paper. One would also want to consider more sophisticated models of project selection. For example, different projects may require investments of different magnitudes, with an upper limit on the total investment (capital budgeting). Also, the "technology" of project comparison may differ from that described above.[18]

6.3. *Nearest-Neighbor Classification*

According to some models of decision-making, actions are chosen by a process of pattern-recognition or classification. In such a model, there is a given finite partition of the set of possible alternative observations of the environment (a "classification scheme"), and to each element of the partition there corresponds a particular action to be taken. For example, in the *nearest-neighbor* classification scheme, the observation of the environment takes the form of a vector, say X, whose possible values are contained in some set, say $\mathscr{H}$. The partition of $\mathscr{H}$ is generated by a finite set of *reference* vectors, $Y_1, \ldots, Y_M$, which are also in $\mathscr{H}$. The set $\mathscr{H}$ is partitioned into sets $A_1, \ldots, A_M$ such that, for every m and every X in A_m,

$$(6.1) \qquad \|X - Y_m\| = \min_{m'} \|X - Y_{m'}\|,$$

where $\| \ \|$ denotes Euclidean length. (Note that the classification of vectors on the boundary between two sets in the partition is arbitrary.) Thus X is classified as being in set A_m ("of type m") if Y_m is its nearest neighbor among the reference vectors $Y_1, \ldots, Y_M$. To each set A_m corresponds an action, say a_m; if

[18] For material on selection see Plaxton (1980), Kruskal et al. (1988), Megiddo (1982), and the references cited therein.

the decision-maker observes that the environment is of type m (in A_m) then he takes action a_m.

Now suppose that the vector X has components $x_1, \ldots, x_N$, which are themselves vectors, and that different components are observed by different members of the same organization (see Section 1). Furthermore, think of N as large, and also that the dimension of each component is sufficiently large so that the process of finding the nearest neighbor needs to be "decentralized." I shall describe one way to do this using a tree of processors.[19]

For each m, let (y_{mn}) be the components of the reference vector Y_m, corresponding to the N components of X. Define

$$(6.2) \qquad z_{mn} \equiv \|x_n - y_{mn}\|^2,$$

$$(6.3) \qquad w_m \equiv \sum_{n=1}^{N} z_{mn} = \|X - Y_m\|^2.$$

Thus the nearest-neighbor classification can be performed in three stages: (i) for each m and n, calculate z_{mn}; (ii) for each m, calculate w_m; (iii) find the $\hat{m}$ that minimizes w_m.

Since both addition and choosing a minimum are associative operations, the nearest-neighbor computation easily lends itself to parallel processing. Here is one way to do it:

1. Assign y_{mn} to a processor (m, n); there will be $M \times N$ of these, and they will constitute the lowest level of a tree.
2. When the input vector $X = (x_n)$ arrives, assign a copy of x_n to each processor (m, n), which then computes z_{mn}.
3. For each m, calculate w_m using a tree; thus there will be a separate tree corresponding to each component.
4. Using a tree, find the $\hat{m}$ that minimizes w_m.

Note that three types of processor have been used, one to calculate the distances z_{mn}, one for pairwise addition of numbers, and one for choosing the minimum of two numbers. The use of more powerful processors would, of course, permit one to use smaller trees. For example, it is clear how one might do this if one had processors that could add vectors of dimension M, and/or could find the minimum of more than two numbers at a time. (On the other hand, the computation of the distances z_{mn} could itself be made parallel with processors that can only add and processors that can only multiply.)

7. CONCLUDING REMARKS

In this paper I proposed a model of parallel processing to describe the decentralized computation of organizational decisions. In fact, in certain cases efficiency—in terms of the number of processors and computational time—in

[19] However, no attempt is made here to characterize efficient networks for this task. For other material on parallel pattern-recognition algorithms using trees, see Gorin et al. (1990).

the class of *all* architectures could be attained by trees (or hierarchies). I offer here some tentative remarks on the interpretation of these results.

1. Organizations are called upon to make many different decisions, i.e., compute many different functions. In principle, different trees might be used to compute different decisions, or different classes of decisions. In particular, this might be a way to interpret the practice called "matrix management." Also, even where there is only one tree of authority (which is by far the most common case), it is well known that many—or most—of the information flows do not follow it.[20]

2. The decentralization of *information* typically implies that the amount of data in any cohort is subject to stochastic variation, a circumstance that gives rise to stochastic queuing in the network of processors. (This problem will be explored in a forthcoming paper.)

3. I have considered here only the case in which the computation of organizational decisions requires primarily associative operations. Indeed, there seems to be a view in the computer science literature that associative operations are the most amenable to parallel computation (see, e.g., Schwartz (1980)), although I am not aware of any precise result along these lines. If this view is correct, then the results presented in this paper provide lower bounds for all operations.

4. Finally, I want to make some brief observations about returns to scale in information processing. The material of Sections 3–5 (specifically (3.4)–(3.6), (4.5)–(4.6), and (5.21)–(5.26)) can be interpreted as showing that there are increasing returns to scale (decreasing average costs), *provided that the "cost" of delay is linear, with a coefficient that is independent of the number, N, of items processed.* In this section I illustrate with two examples how quite different returns to scale can be implied by different, but plausible, cost functions. To simplify the discussion, I consider only the "one-shot" mode of Section 3, although similar remarks can be made about the systolic mode. (A fuller treatment will be found in Radner and Van Zandt (1992).)

First, imagine that N reflects the "size" of the enterprise, and that the cost of delay is proportional to N. Thus replace the cost function (3.4) by

$$L = N\gamma C + \phi P. \tag{7.1}$$

It is immediate that the optimal number of processors is now proportional to N:

$$P = \Psi N, \quad \Psi = \frac{-\dfrac{1}{\ln 2} + \left[\dfrac{1}{(\ln 2)^2} + 4\alpha\right]^{1/2}}{2\alpha}.$$

(In equation (3.5), replace α by α/N; one can check that $0 < \Psi < 1$.) One

[20] See, for example, Mintzberg (1979).

verifies that

$$C = \frac{1}{4} + \log_2 \Psi + \log_2 N, \qquad \frac{L}{N} = \gamma C + \phi \Psi,$$

so that the cost per item is increasing in the number of items (decreasing returns to scale).

In the second example, suppose that the gross *benefit* from processing N items is proportional to N but decreases exponentially to zero as a function of the delay. (For a class of organizational decision problems with this property see Marschak and Radner (1972, Ch. 7).) To be precise, the gross benefit is

$$(7.2) \qquad B = N\beta e^{-\gamma C},$$

where β and γ are positive constants, and the net benefit, or value, is

$$(7.3) \qquad V = B - \phi P.$$

In this example, for sufficiently large N the optimal number of processors is 1, and indeed, for sufficiently large N the maximum net benefit, V, will actually be negative. To see this, first note that

$$C \geqslant 1 + \log_2 N = 1 + \ln N^{(1/\ln 2)}$$

(the lower bound is attained when $P = N$). Hence, from (7.2),

$$B \leqslant e^{-\gamma} N^{1-(1/\ln 2)}.$$

Thus the maximum gross benefit approaches zero as N increases without bound; for sufficiently large N this gross benefit will in fact be less than the cost of a single processor. (In a statistical decision problem, this would imply that the decision should be based on prior information, without using any observations.)[21]

8. BIBLIOGRAPHIC NOTES

Two strands of literature have influenced the ideas presented here, one from economics, and one from computer science. T. A. Marschak and C. B. McGuire (1971) were probably the first to propose the model of a finite automaton as a formalism of the notion of a boundedly rational decision-maker. The model of a decision-making organization as a network of information processors was explored by J. Marschak and R. Radner (1972, Ch. 9), but their analysis was concerned more with the decentralization of information than of information processing. In a similar spirit, T. A. Marschak and S. Reichelstein (1992) studied conditions under which a "hierarchical" structure of decision-making would be efficient in a broader set of structures. In their model, every processor is also responsible for the final decision about some action variable, and the

[21] Keren and Levhari (1983) studied a hierarchical model in which each manager's decision-making time is a linear function of his span of control, and output is a decreasing function of delay. Taking the spans of control at different levels as the organizational parameters to be optimized, they derive conditions for eventual decreasing returns to scale. It is difficult, however, to fit their model into the present framework.

only cost of processing is that of communication. Their analysis derived some conditions under which hierarchy would be preferred, but I shall not attempt to summarize their results here. In the "economics strand," the current research of K. R. Mount and S. Reiter is most closely related to the present paper, and I have benefited from exposure to it in unpublished papers (Mount and Reiter (1982, 1990)), and private communications. Recently, several other authors have explored the question of organizational design from a similar viewpoint, notably Moore et al. (1992), Bolton and Dewatripont (1992), and Jehiel (1992). (I have already noted the work of Keren and Levhari (1983); for further references see Radner (1992).)

T. A. Marschak (1972) and J. Marschak and R. Radner (1972, Ch. 7) studied the effect of delay on the value of decisions, particularly in the context of decentralization of information. As pointed out in Section 7, the cost of delayed decision may not be simply proportional to the delay; the functional form of the dependence of cost on delay will depend on the intertemporal statistical properties of the stochastic process of environmental data.

From the "computer-science strand," the reader familiar with the article, "Ultracomputers," by J. T. Schwartz (1980), will recognize how heavily I have relied on the ideas in that paper. I have also benefited from the lecture notes on the course by T. Leighton, et al. (1988), and the paper by Kruskal et al. (1988).

The significance of the boundedness of the rationality of economic decision-makers, especially in an organizational context, has been emphasized by H. Simon (1972), who also has an interesting discussion of the significance of hierarchy in the design and organization of tasks (1981, Ch. 5).

Recently, some game-theorists have used the automaton model to explore how the boundedness of rationality might alter the predictions of Nash equilibrium theory, especially in sequential games (see, e.g., Rubinstein (1986); Neyman (1985); Kalai and Stanford (1988); Abreu and Rubinstein (1988); and the references cited there). In most of these explorations, however, the boundedness of a player's rationality is expressed only by a bound on the number of states of the automaton, a direction that is quite orthogonal to the one taken here.

AT&T Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974 U.S.A.
and
Dept. of Economics, New York University, New York, NY 10003, U.S.A.

Manuscript received June, 1990; final revision received March, 1993.

APPENDIX

A1. A Formal Model of a Programmed Network: The model of a programmed network (in the one-shot mode) described in Section 2 is a specialization of what is sometimes called a parallel machine with distributed memory. The *network* consists of P processors connected by a given graph of one-way links. A pair of processors, say i and j, may be connected by no link, by one link, or by two links (one from i to j and one from j to i). A processor may (but need not) also have one "output" link; at least one processor has one.

Each individual processor consists of (1) a *processing unit* and (2) a finite number of *memory locations*, numbered 0 to M. A memory location may contain at most one item from some prescribed set, say X. In a single time period (cycle) a processor may do one or both of the following: (1) "read" the contents of two of its memory locations into its processing unit, perform a prescribed associative binary operation on them (say $*$, from X^2 to X), and "write" the result into one of its memory locations; (2) read the contents of one of its memory locations and write it (or copies of it) into a memory location of one or more other processors to which it is directly connected. (Some rule must specify what happens if two or more processors try to write simultaneously into the same memory location.) In any cycle, a processor may also be idle. One is given N data items from X, say $x_1, \ldots, x_N$, and it is required to calculate $y = x_1 * x_2 * \cdots * x_N$.

A *program* (1) initially assigns the N data items to a set of memory locations, one item per location, and (2) prescribes a sequence of activities for each processor. A program is *feasible* with finite *delay* d, if at the end of cycle d (but not before), the answer, y, is read out on the output link of some designated processor.

In a more general model, (i) different processors might have different numbers of memory locations, (ii) each processor might be able to compute several different functions of 2 or more variables stored in its own memory, (iii) different processors might have different sets of functions, and (iv) the final answer might be read out by more than one processor.

On the other hand, the model in the present paper is even more specialized, as follows. To each individual processor there corresponds a particular memory location, called its register; the remaining M locations constitute its *in-box*. Whenever a processor performs the operation $*$ on two items, one of the items must be in the register, and the result must be written into the register. Further, I make the convention that, if the register is "empty," then it contains the identity item 0 for the operation $*$, i.e., for all x in X, $0 * x = x * 0 = 0$. Because of this convention, it takes one cycle to read an item from a processor's in-box into its "empty" register. This assumption is intended to reflect situations in which processing time is required to "digest" (understand, interpret) each item before it can be operated upon.

Finally, it is assumed here that a processor can only send an item from its register to another processor's in-box.

A2. Proof of Theorem and Corollary of Section 3: The proof is in two parts; I first show that the right-hand side of (3.1) is a lower bound on $\hat{C}$, and then that this bound can be attained by a tree.

Let $n(c)$ denote the number of "items" remaining at the end of cycle c, where an "item" is either a partial sum in a processor's register or an original data item in a processor's in-box. Make the convention that before cycle 1 each processor has a partial sum equal to 0 in its register; with this convention,

$$n(0) = N + P. \tag{A2.1}$$

During any one cycle, the number of "items" can at most be reduced by half; also, it can be reduced by no more than P. Hence

$$n(c) - n(c+1) \leqslant \min\left\{\frac{n(c)}{2}, P\right\},$$

which is equivalent to

$$n(c+1) \geqslant \max\left\{\frac{n(c)}{2}, n(c) - P\right\}. \tag{A2.2}$$

Let $M(\cdot)$ denote the solution of the difference equation,

$$\begin{aligned} M(c+1) &= \max\left\{\left\lceil\frac{M(c)}{2}\right\rceil, M(c) - P\right\}, \qquad c \geqslant 0, \\ M(0) &= N + P, \end{aligned} \tag{A2.3}$$

and let $\tilde{C}$ be the first c such that $M(c) = 1$. Then $n(c) \geqslant M(C)$ and $\hat{C} \geqslant \tilde{C}$. I shall show that $\tilde{C}$ is given by the right-hand side of (3.1).

To calculate $\tilde{C}$, first note that after c cycles, the number of remaining cycles, $\tilde{C} - c$, is determined by the value of $M(c)$, through (A2.3). Accordingly, define $R(m)$ to be the remaining number of cycles if $M(c) = m$. In particular, $R(1) = 0$ and $R(N+P) = \tilde{C}$. For $2 \leqslant m < 2P$, if $M(c) = m$ then $M(c+1) = \lceil m(c)/2 \rceil$. Hence it is straightforward to show by induction that

$$\text{(A2.4)} \qquad R(m) = \lceil \log_2 m \rceil, \qquad 1 \leqslant m < 2P.$$

For $m \geqslant 2P$, if $M(c) = m$ then $M(c+1) = m - P$. Hence,

$$\text{(A2.5)} \qquad R(m) = \left\lfloor \frac{m-P}{P} \right\rfloor + R(P + m \bmod P), \qquad m \geqslant 2P.$$

Putting (A2.4) and (A2.5) together we have, for $N \geqslant P$,

$$\text{(A2.6)} \qquad \begin{aligned} \tilde{C} &= R(N+P) \\ &= \left\lfloor \frac{N}{P} \right\rfloor + \lceil \log_2 (P + N \bmod P) \rceil. \end{aligned}$$

It remains to show that (3.12) can be attained by a tree. Number the processors consecutively from 1 to P. Start by assigning the N data items as equally as possible; if $N \bmod P > 0$ then assign these remaining items to the first $N \bmod P$ processors.

Let $c_0 \equiv \lfloor N/P \rfloor$. In the first c_0 cycles, $c_0 P$ data items will be "preprocessed," leaving a partial sum in each processor, and $N \bmod P$ data items not yet preprocessed but assigned to the first $N \bmod P$ processors (one each). In describing cycle $(c_0 + 1)$ it is useful to distinguish two cases:

Case 1. $N \bmod P = 0$. For each even i make processor i the direct subordinate of $(i-1)$, and in cycle $(c_0 + 1)$ processor i's partial sum is added to $(i-1)$'s register and i's register is emptied.

Case 2. $N \bmod P > 0$. Each of the first $N \bmod P$ processors has one remaining data item, which is added to its register in cycle $(c_0 + 1)$. Treat the remaining $P - N \bmod P$ processors as in Case 1 (with appropriate renumbering).

In both cases, let $p \equiv n(c_0 + 1)$; this is the number of remaining processors with partial sums to be added after cycle $(c_0 + 1)$. Number these processors from 1 to p, and arrange them in a tree as follows:

1. For each even i, make $(i-1)$ the direct superior of i, retaining the links established previously.

2. Renumber the top-level processors consecutively. For each even i, make $(i-1)$ the direct superior of i, retaining the links established previously, etc.

Continue this procedure until the top level contains only 1 processor. The resulting tree attains the minimum number of cycles, (3.12).

Note that if N/P is an integer and P is a power of 2, then the above tree is exactly a "reduction" of a balanced hierarchy, as described in Section 3.

To prove the Corollary, first note that, for fixed N, $\hat{C}$ is nonincreasing in P for $1 \leqslant P \leqslant \lceil N/2 \rceil$. For $(N/2) < P \leqslant N$,

$$P + N \bmod P = n$$

$$\left\lfloor \frac{N}{P} \right\rfloor = 1,$$

and so (3.2) is satisfied. If $N/2$ is an integer, then for $P = N/2$, (3.2) is also satisfied. Finally, the argument leading up to (A2.4) shows that the delay cannot be reduced by using more than N processors.

A3. Balanced Hierarchies in the One-Shot Mode: Here I derive the asymptotics for relatively efficient PPTs (Section 4). One verifies (cf. (4.5)) that

$$\text{(A3.1)} \qquad \frac{\partial L}{\partial k_1} = \gamma - \frac{\gamma K}{k_1 \ln K} - \frac{\phi K N}{k_1^2 (K-1)},$$

$$\text{(A3.2)} \qquad \frac{\partial L}{\partial K} = \frac{\gamma \ln\left(\frac{N}{k_1}\right)(\ln K - 1)}{(\ln K)^2} + \frac{\phi\left(1 - \frac{N}{k_1}\right)}{(K-1)^2}.$$

From (A3.1) we see that, as a function of k_1, L is convex. On the other hand, rewrite (4.12) as

$$\text{(A3.3)}\quad \frac{\partial L}{\partial K}=\frac{\gamma(\ln p_1)(\ln K-1)}{(\ln K)^2}-\frac{\phi(p_1-1)}{(K-1)^2}.$$

Recall that $2 \leqslant K \leqslant p_1$. A detailed examination of (A3.3) shows that, as a function of K, L is initially decreasing, and then may increase; however, it need not be convex throughout. For $\alpha \equiv (\phi/\gamma)$ sufficiently small (given N), L will be minimized by taking $k_1 = K = 3$. For α sufficiently large, L will be minimized by taking $k_1 = N(p_1 = 1)$, in which case there is only one processor and one level, i.e., there is only preprocessing, and the "tree" is degenerate. Setting $K = p_1$ in (A3.3), we see that

$$\left.\frac{\partial L}{\partial K}\right|_{K=p_1}=\frac{\gamma(\ln p_1-1)}{\ln p_1}-\frac{\phi}{p_1-1},$$

which (for fixed γ and ϕ) becomes positive and small as p_1 becomes large. As we shall see, this implies that, for fixed α, for N large the optimal PPT will be "interior."

From (A3.1), the first-order condition for an optimum with respect to k_1 is

$$\text{(A3.4)}\quad \begin{aligned} &k_1^2-\left(\frac{K}{\ln K}\right)k_1-\frac{\alpha KN}{K-1}=0,\\ &\alpha\equiv\frac{\phi}{\gamma},\end{aligned}$$

which is a quadratic equation in k_1, and has the positive root

$$\text{(A3.5)}\quad k_1=\frac{b+\sqrt{b^2+4cN}}{2},$$

$$\text{(A3.6)}\quad b\equiv\frac{K}{\ln K},\quad c\equiv\frac{\alpha K}{K-1}.$$

On the other hand, the first-order condition corresponding to (A3.2) is

$$\text{(A3.7)}\quad \frac{(K-1)^2(\ln K-1)}{(\ln K)^2}=\frac{\alpha\left(\dfrac{N}{k_1}-1\right)}{\ln\left(\dfrac{N}{k_1}\right)}.$$

I shall now show that K must increase without bound as $N\to\infty$. Suppose to the contrary that K were bounded for some infinite subsequence of N; then b and c would also be bounded, and so, by (A3.5), k_1 would be $O(\sqrt{N})$. But the right-hand side of (A3.7) would diverge to ∞, whereas the left-hand side would be bounded (recall that $K \geqslant 2$), a contradiction. Hence K diverges to ∞:

$$\text{(A3.8)}\quad \lim_{N\to\infty} K=+\infty.$$

From (4.18) it follows that

$$\lim_{N\to\infty} c=\alpha,$$

that b diverges to ∞, and hence (by A3.5) that k_1 diverges to ∞.

I shall now show, in fact, that

$$\text{(A3.9)}\quad k_1\sim(\alpha N)^{1/2}.$$

Before I do so, however, note that this implies that

$$\text{(A3.10)}\quad p_1\sim\left(\frac{N}{\alpha}\right)^{1/2},$$

and that (as I shall show)

$$(A3.11)\qquad K \sim \frac{(4\alpha N)^{1/4}}{\ln N}.$$

It then follows from (4.10) and (4.11) that C, P, and L are all $O(\sqrt{N})$.

I conclude this section by proving (A3.9) and (A3.11). First, since $K \to \infty$,

$$(A3.12)\qquad \text{LHS(A3.1)} \sim \frac{K^2}{\ln K},$$

and diverges to ∞; hence the right-hand side of (A3.7) also diverges to ∞, and hence

$$(A3.13)\qquad \lim_{N\to\infty}\left(\frac{N}{k_1}\right) = \infty$$

(recall that $p_1 = N/k_1$). From (A3.5) and the fact that $c \to \alpha$, we have

$$(A3.14)\qquad \frac{K}{k_1 \ln K} + \left[\left(\frac{K}{k_1 \ln K}\right)^2 + \frac{4\alpha N}{k_1^2}\right]^{1/2} \to 2.$$

Write

$$(A3.15)\qquad \frac{\alpha N}{k_1^2} = \left(\frac{K}{k_1 \ln K}\right)\left(\frac{\alpha N}{k_1}\right)\left(\frac{\ln K}{K}\right).$$

From (A3.7), (A3.12), and (A3.13),

$$(A3.16)\qquad \frac{K^2}{\ln K} \sim \frac{\alpha\left(\dfrac{N}{k_1}\right)}{\ln\left(\dfrac{N}{k_1}\right)},$$

and hence

$$(A3.17)\qquad \left(\frac{\alpha N}{k_1}\right)\left(\frac{\ln K}{K}\right) \sim K \ln\left(\frac{N}{k_1}\right) \to \infty.$$

If, for some subsequence of N,

$$\frac{K}{k_1 \ln K}$$

were bounded away from zero, then, by (A3.17), the left-hand side of (A3.14) would diverge to ∞, and not converge to 2. Hence

$$\lim_{N\to\infty}\left(\frac{K}{k_1 \ln K}\right) = 0,$$

which, by (A3.14), implies (A3.9). One can now derive (A3.10) from (A3.16).

A4. Preprocessing/Overhead Networks: This appendix section is devoted to deriving (5.22)–(5.29) of Section 5.

From (5.18), (5.19), and (5.21),

$$(A4.1)\qquad \begin{aligned} \frac{\partial L_0}{\partial p_0} &= \frac{N}{p_0^2} + \frac{K}{p_0 \ln K} + \frac{\alpha}{K-1}, \\ \frac{\partial L_0}{\partial K} &= \frac{(\ln p_0)(\ln K - 1)}{(\ln K)^2} - \frac{\alpha(p_0 - 1)}{(K-1)^2}. \end{aligned}$$

Setting the right side of (A4.1) to zero we get

$$p_0 = \left(\frac{K-1}{2\alpha}\right)\left(\frac{-K}{\ln K} + \left[\left(\frac{K}{\ln K}\right)^2 + \frac{4\alpha N}{K-1}\right]^{1/2}\right). \tag{A4.2}$$

Since K is bounded between 2 and T, (A4.2) shows that p_0 increases without bound with N; hence the right-hand side of (A4.1) eventually becomes negative, so the optimal $K = T$, which proves (5.2). It is now straightforward to verify (5.23)–(5.28).

To prove (5.29), first note that, from (5.5) and (5.18), $P_L = P_0$ implies

$$\frac{\bar{Q}-1}{T} = \frac{p_0 - 1}{T-1}, \tag{A4.3}$$

$$\bar{Q} = 1 + \frac{T(p_0 - 1)}{T-1}.$$

Hence, from (5.23)

$$\bar{Q} \sim T\left[\frac{N}{(T-1)\alpha}\right]^{1/2},$$

so from (5.3) and (5.4)

$$C_L \sim \left(\frac{1}{T}\right)[\alpha(T-1)N]^{1/2}. \tag{A.4}$$

Comparing (5.34) with (5.24) proves (5.29).

A5. Garey Trees: Call a triple (n, c, p) *one-shot-efficient* (OSE) if there is an efficient one-shot network that processes n items in c cycles with p processors. Recall that if (n, c, p) is OSE, then

$$c = \hat{c}(n, p) \equiv \left\lfloor \frac{n}{p} \right\rfloor + \lceil \log_2(p + n \bmod p) \rceil;$$

cf. (3.1). Fix N and T. A *T-module* with parameters (n, p) is a one-shot efficient network that achieves a OSE triple (n, c, p) such that

1. $c \leqslant T$;
2. if (n', c', p') is OSE and $c' > c$, then $c' > T$.

Let E be the set of parameter pairs corresponding to T-modules. Fix some (n, p) in E, and let

$$R \equiv \lceil \log_n N \rceil,$$

i.e., R is the smallest integer r such that $n^r \geqslant N$. A *Garey tree* is a R-level n-ary tree of T-modules with parameters (n, p). Assign the N items as equally as possible to the n^{R-1} bottom modules. No bottom module will have more than $n_1 \equiv \lceil N/n^{R-1} \rceil$ items, and at least one will have that many. Thus the number of cycles at the bottom level is $\hat{c}(n_1, p)$. At each higher level the number of cycles will be $\hat{c}(n, p)$. Hence the total number of cycles is

$$C_G = \hat{c}(n_1, p) + (R-1)\hat{c}(n, p). \tag{A5.1}$$

The total number of modules is

$$\sum_{r=0}^{R-1} n^r = \frac{n^R - 1}{n-1},$$

and so the total number of processors is

$$P_G = p\left(\frac{n^R - 1}{n-1}\right). \tag{A5.2}$$

Note that

$$\log_n N \leqslant R \leqslant \log_n N + 1, \quad n_1 \leqslant n, \quad C_G \leqslant T(\log_n N + 1).$$

In the case in which "everything comes out even," i.e., N is a power of n and $c = T$, then (A5.1-2) become

$$C_G = T \log_n N,$$

$$P_G = \frac{p(N-1)}{n-1}.$$

In this case the fastest Garey tree has

$$\min C_G = \left(\frac{T}{G-1}\right) \log_2 N$$

(I omit the details). Compare this with the corresponding results for PPO networks, (5.30), and fully efficient networks, (5.10).

Finally, I note that, in the definition of a Garey tree one could have allowed any $c \leqslant T$. However, it can be shown that relative efficiency (within the class of Garey trees thus defined) would imply that c must be as close to T as possible. Again, I omit the details.

REFERENCES

Abreu, D., and A. Rubinstein (1988): "The Structure of Nash Equilibrium in Repeated Games with Finite Automata," *Econometrica*, 56, 1259–1281.

Aoki, Masahiko (1986): "Horizontal vs. Vertical Information Structure of the Firm," *American Economic Review*, 76, 971–983.

Bolton, P., and M. Dewatripont (1992): "The Firm as a Communication Network," Discussion Paper TE/92/256/, Suntory-Toyota Centre, London School of Economics (unpublished).

Geanakoplos, J., and P. Milgrom (1991): "A Theory of Hierarchies Based on Limited Managerial Attention," *Journal of the Japanese and International Economies*, 5, 205–225.

Gibbons, A., and W. Rytter (1988): *Efficient Parallel Algorithms*. Cambridge: Cambridge University Press.

Gorin, A. L., D. B. Roe, and A. G. Greenberg (1990): "On the Complexity of Pattern Recognition Algorithms on a Tree-Structured Parallel Computer," in *Signal Processing, Part I: Signal Processing Theory*, ed. by L. Auslauder, T. Kailath, and S. Miller. New York: Springer-Verlag, 1990.

Jehiel, P. (1992): "Coordination and Organizations," CERAS-ENPC, Paris (unpublished).

Kalai, E., and W. Stanford (1988): "Finite Rationality and Interpersonal Complexity in Repeated Games," *Econometrica*, 56, 397–410.

Keren, M., and D. Levhari (1983): "The Internal Organization of the Firm and the Shape of Average Costs," *Bell Journal of Economics*, 41, 474–486.

Knight, Frank (1921): *Risk, Uncertainty, and Profits*. Cambridge, MA: Houghton Mifflin, 1921.

Kruskal, C. P., L. Rudolph, and M. Snir (1990): "A Complexity Theory of Efficient Parallel Algorithms," *Theoretical Computer Science*, 71, 95–132.

Leighton, T., C. E. Leiserson, B. Maggs, S. Plotkin, and J. Wein (1988): "Lecture Notes on Theory of Parallel and VLSI Computation," MIT/LCS/RSS 2, Laboratory for Computer Science, Massachusetts Inst. of Technology, Cambridge.

Marschak, T. A. (1972): "Computation in Organizations," in *Decision and Organization*, ed. by C. B. McGuire and R. Radner. Amsterdam: North Holland, 1972, pp. 237–282.

Marschak, T. A., and C. B. McGuire (1971): "Lecture Notes on Economic Models for Organization Design," Graduate School of Business Administration, Univ. of Calif., Berkeley.

Marschak, J., and R. Radner (1972): *Economic Theory of Teams*. New Haven: Yale University Press, 1972.

Marschak, T. A., and S. Reichelstein (1987): "Network Mechanisms, Informational Efficiency, and Hierarchies," Haas School of Business, University of California, Berkeley (unpublished).

McGuire, C. B., and R. Radner (1986): *Decision and Organization*. Amsterdam: North-Holland; 2nd ed., Minneapolis: University of Minnesota Press, 1986.

MEGIDDO, N. (1982): "Parallel Algorithms for Finding the Maximum and the Median Almost Surely in Constant Time," GSIA, Carnegie-Mellon University, October 1982 (unpublished).

MINTZBERG, H. (1979): *The Structuring of Organizations*. Englewood Cliffs, NJ: Prentice-Hall.

MOORE, J. C., H. G. RAO, AND A. B. WHINSTON (1992): "Information Processing for a Finite Resource Allocation Mechanism," Graduate School of Business, Univ. of Texas, Austin (unpublished).

MOUNT, K. R., AND S. REITER (1982): "Computation, Communication, and Performance in Resource Allocation," paper presented at the CEME-NBER Decentralization Seminar, University of Minnesota, Minneapolis (unpublished).

——— (1990): "A Model of Computing with Human Agents," Disc. Paper No. 890, Center for Mathematical Studies in Economics and Management Science, Northwestern University (unpublished).

NEYMAN, A. (1985): "Bounded Complexity Justifies Cooperation in the Finitely-Repeated Prisoners' Dilemma," *Economics Letters*, 19, 227–229.

PLAXTON, C. G. (1980): "Load Balancing, Selection, and Sorting on the Hypercube," *Journal of the Association for Computing Machinery*, 36, 64–73.

RADNER, R. (1961): "The Evaluation of Information in Organizations," in *Proceedings of the Fourth Berkeley Symposium on Probability and Statistics*. Berkeley: Univ. of Calif. Press, pp. 491–530.

——— (1962): "Team Decision Problems," *Annals of Mathematical Statistics*, 33, 857–881.

——— (1987): "Decentralization and Incentives," in *Information, Incentives, and Economic Mechanisms*, ed. by T. Groves, R. Radner, and S. Reiter. Minneapolis: Univ. of Minnesota Press, pp. 3–47.

——— (1992): "Hierarchy: The Economics of Managing," *Journal of Economic Literature*, 30, 1382–1415.

——— (1990): "The Organization of Decentralized Information Processing," AT&T Bell Laboratories (unpublished); presented at the NBER-CEME Decentralization Seminar, Northwestern University, April 27–29, 1990.

RADNER, R., AND T. VAN ZANDT (1992): "Information Processing in Firms and Returns to Scale," *Annales d'Economie et de Statistique*, No. 25-26, 265–298.

RUBINSTEIN, A. (1986): "Finite Automata Play the Repeated Prisoners' Dilemma," *Journal of Economic Theory*, 39, 83–96.

SCHWARTZ, J. T. (1980): "Ultracomputers," *ACM Transactions on Programming Languages and Systems*, 2, 484–521.

SIMON, H. A. (1972): "Theories of Bounded Rationality," in *Decision and Organization*, ed. by C. B. McGuire and R. Radner. Amsterdam: North-Holland, pp. 161–176.

——— (1981): *The Sciences of the Artificial* (2nd ed.). Cambridge: The MIT Press.

VAN ZANDT, T. (1990): "Efficient Parallel Addition," AT&T Bell Laboratories, Murray Hill, NJ (unpublished).