Rev. Econ. Design 3, 329–345 (1998)

Review of Economic Design

Managerial costs for one-shot decentralized information processing*

Kieron Meagher[1], **Timothy Van Zandt**[2]

[1] Economics Program, RSSS Australian National University, Canberra, ACT 0200, Australia
(Tel.: +61 (6) 249 4135; Fax: +61 (6) 249 0182; e-mail: kmeagher@coombs.anu.edu.au)
[2] Department of Economics, Princeton University, Princeton, NJ 08544-1021, USA
(Tel.: +1 (609) 258-4050; Fax: +1 (609) 258-6419; e-mail: tvz@princeton.edu,
Web: www.princeton.edu/-tvz)

Received: 28 February 1997 / Accepted: 30 September 1997

Abstract. Radner (1993) proposed a model of decentralized associative computation as a means to understand information processing in organizations. In the model, in which an organization processes a single cohort of data, resources are measured by the number of managers. This paper (i) explains why resources should instead be measured by the time the managers are busy, (ii) shows that, nevertheless, the characterization of sufficient conditions for efficient networks in Radner (1993) and Keren and Levhari (1979) is valid for either measure, (iii) shows that measuring resources by the number of operations leads to sharper results on necessary conditions for efficiency, (iv) strengthens Radner's results on the irregularity of efficient hierarchies, and (v) compares the relative costs of parallelization under the two measures.

JEL classification: D83, D23

Key words: Organizations, decentralization, parallel processing

1 Introduction

Working with information is a major area of employment in the economies of the developed world. This is demonstrated empirically by Baumol et al. (1989), Gittleman and Wolff (1995) and Radner (1992), who put the fraction of information or data workers in the U.S. economy at over 40% of those employed. However, information processing is not simply a production activity; instead, it is the process by which organizations make decisions.

* Parts of this paper appeared previously in the working papers "Efficient Hierarchies: Equivalence under Differing Employment Regimes" (by Meagher) and "The Scheduling and Organization of Periodic Associative Computation" (by Van Zandt).

Organizations make decisions over time from a flow of information about the environment. To study information processing in an organization, it is possible to explicitly model the organization's temporal decision problem and the endogenous determination of the decision rule and information processing procedure, such as in the real-time decentralized computation models of Radner and Van Zandt (1992), Van Zandt and Radner (1997) and Van Zandt (1997a,b). However, a simpler approach to studying information processing is to take as given the decision rule that must be computed and to ask simply what is the best way to compute it.

A further simplification is to view the ongoing computation of a temporal decision rule as a flow of separate calculations or jobs, which overlap temporally and are handled by the same administrative staff, but which do not share data and partial results. The resulting model of the administrative process is analogous to a parallel processing machine or distributed computing network that must handle a flow of separate computation jobs. The problem is to design algorithms for each job and to schedule the jobs to the machine or network, taking into account criteria such as computational (managerial) resources and the delay in processing each job. This is called decentralized batch processing, and was first proposed (in the early 1980's) by Mount and Reiter (1990) as a means to understand information processing in organizations.

Subsequently, Radner (1993) studied a more specialized model of decentralized associative batch computation, which focused on the interrelationship between four key factors in the decision making process: The quantity of information, the wages of the administrative staff, the delay, and the pattern of communication between information processing agents. The purpose of the current paper is to clarify the measurement of wages in this model.

Note that, generally, the delays and managerial wages depend on the entire flow of problems. The delay, which is the time between when the data for a job arrives and when the answer is computed, is the sum of (i) the time between when the job is started and when it is finished, and (ii) the time the job spends in a queue before it is started. The formation of queues has to do with the irregular and random nature of the flow of jobs. Malone and Smith (1988) and Beggs (1995) explicitly model the random arrival of jobs and focus on the queuing delays.

Similarly, the managerial wages are the sum of (i) the wages paid while the managers (used generically to refer to information processing agents) are actually processing jobs and (ii) the wages that are paid while the managers are idle. The idle time also depends on the nature of the flow of jobs. One of the models in Radner (1993), which is studied further in Van Zandt (1997c, 1998), focuses on the managerial wages due to idle time. In that model, an administrative staff of fixed size must process jobs of associative computation, which arrive periodically and are all of the same size.

However, there is one set of (unrealistic) assumptions under which queuing delays and wages for idle time are not present: When there is a potentially unlimited number of information processing agents who can be immediately organized to process each job that arrives and when the agents must only be paid

for the time they are busy, i.e., on an hourly basis. The optimal algorithm for each job and its delay and costs are independent of the other jobs the organization must process. This leads one to a model of "one-shot" computation, also studied by Radner (1993) for the case of associative computation, in which one studies the computation of a single job in isolation from others the organization may face.

Even when the assumptions that justify the one-shot model are not satisfied—for example, when an organization of fixed size must process the flow of jobs and all managers are paid salaries, even when idle—the one-shot model is a useful approximation for measuring the delay and managerial costs for each job. This is analogous to the approach usually taken in the design of parallel algorithms. If someone designs an algorithm for a particular problem, without knowledge of the other jobs that will be processed by the parallel computer, it is reasonable to ignore how the interaction with other jobs will lead to idle time and instead simply try to minimize the CPU time taken up by the job. This is an opportunity cost and is the time the managers could not possibly be attending to other jobs.

Whether the assumptions of the one-shot model actually hold or whether the model is simply meant to be an approximation, managerial costs should be measured by the time the managers are busy. (This would also be true in the hypothetical scenario in which an organization is assembled to process a single job and is then disassembled.) However, Radner (1993) measures resources in the one-shot model by the number of managers. This is also true of Keren and Levhari (1979, 1983), which can be interpreted as models of associative computation as described in Van Zandt (1996).[1] It is true that one could realistically add a fixed cost per manager in the one-shot model. However, if one has to choose between measuring managerial wages only by the number of managers or only by the amount of time managers are busy, the latter is more accurate.

The main goals of this paper are (i) to point out, as we have already done, that managerial wages in the one-shot model should be measured by the time the managers are busy rather than by the number of managers, (ii) to show that, nevertheless, the characterizations of sufficient conditions for efficient networks in Radner (1993) and Keren and Levhari (1979) are valid for either measure, (iii) to show also that measuring wages by the time managers are busy leads to sharper necessary conditions for efficiency, (iv) to strengthen Radner's results on the irregularity of efficient hierarchies, and (v) to compare the relative costs of parallelization under the two measures of managerial wages.

2 Model

The model is the one-shot model in Radner (1993), but we use the notation and formal definitions of Van Zandt (1997c).[2] We provide only a brief summary; see the Appendix and Van Zandt (1997c) for details.

[1] As explained in Van Zandt (1996), their model is not one of periodic computation because it does not take into account throughput.

[2] Since all the data here is from the same period, we omit the time index used in Van Zandt (1998).

The problem is to calculate, for some finite $N \geq 2$, the aggregate $X_1 \oplus \cdots \oplus X_N$ of N elements of a set $\mathscr{X}$, which is endowed with an associative operation $\oplus$. X_n is said to come from data source n. The unit of time is called a cycle and the data arrive at the beginning of cycle 0. The calculations are performed by a group of identical managers who can aggregate information and exchange partial results. The interpretation is that this calculation is part of the calculation of a decision by an organization. The managers are members of the organization's administrative staff.

The managers' capabilities are as follows: Each manager has a buffer to which raw data and messages (which are all elements of $\mathscr{X}$) arrive and are stored, and to which the manager has random access (meaning that she can retrieve the value of any message or raw datum without regard to the order in which data has arrived at the buffer). There is no bound on the capacity of this buffer. Each manager also has a register that can store a single element of $\mathscr{X}$. During one cycle, a manager can perfom one operation, which means that she reads one value from her buffer and either (i) stores it in her register, or (ii) aggregates it to the value in her register and then stores the result in her register. Finally, at the beginning of each cycle, a manager can instantly and costlessly send the value of her register to the buffer of any other manager.

Hence, a single manager can calculate $X_1 \oplus X_2 \oplus X_3 \oplus X_4$ in four cycles by reading the four data one at a time. In the first cycle, she reads X_1 and stores it in her register. In the second, she reads X_2, calculates $X_1 \oplus X_2$, and stores the value in her register, and so on.

In contrast, two managers can calculate $X_1 \oplus X_2 \oplus X_3 \oplus X_4$ in three cycles, as follows: Denote the managers by q_1 and q_2. In cycles 0 and 1, q_1 calculates $X_1 \oplus X_2$ and q_2 calculates $X_3 \oplus X_4$. At the beginning of cycle 2, q_2 sends the value $X_3 \oplus X_4$ to manager q_1, who then reads it in cycle 2 and calculates $(X_1 \oplus X_2) \oplus (X_3 \oplus X_4)$.

In the formal definition of the model, the activities of the managers are specified by strings called instructions. The instruction types are INPUT, SEND, OUTPUT, LOAD and ADD, and they have parameters, given in the Appendix, which specify when and by whom they are executed, and what information is read from the buffers. INPUT, SEND and OUTPUT instructions are called messages and are executed during an instant at the beginning of each cycle: An INPUT instruction transmits a datum from an input device to the managers, a SEND instruction sends the value of one manager's register to the other managers' buffers, and an OUTPUT instruction sends the value of one manager's register to an output device. LOAD and ADD instructions are called operations and are executed during the rest of each cycle. When executing a LOAD instruction, a manager reads the value of a designated message from her buffer and stores it in her register. When executing an ADD instruction, a manager reads the value of a designated message from her buffer, calculates the aggregate of this value and the value in her register, and stores the result in her register. In either case, an operations is said to process the message whose value is read.

A network is a pair $\langle \mathscr{Q}, \mathscr{F} \rangle$, where $\mathscr{Q}$ is a finite set of managers and $\mathscr{F}$ is a finite set of instructions, which satisfy certain consistency conditions so that

the instruction can be feasibly performed by the managers. For example, if a manager has an instruction to read the value of a message, another instruction that sent that message must have been executed previously. Also, a manager cannot cannot send a message or perform an ADD operation without first storing some value in her register, and hence her first instruction must be a LOAD. See Van Zandt (1997c) for details. In a network, it is possible to keep track of the value of each register and message as a function of the realizations of the raw data $\{X_1, \ldots, X_N\}$. A network is *functional* if it has a single OUTPUT instruction, whose value is the aggregate of the raw data.

The performance of a functional network is inversely measured by two factors: Managerial resources and the delay. The delay of a network $\langle \mathcal{Q}, \mathcal{F} \rangle$ is the time between when the data arrive and when the aggregate of the data is sent to the OUTPUT device. It is equal to the cycle in which the OUTPUT instruction is executed, and is denoted by D. Delay captures the importance of making a decision quickly, because information becomes obsolete and opportunities can be missed.

The managerial resources could include a variety of goods that are used in the computation, but in this computation model there are no memory or information transmission costs and the only resource that is used by a network $\langle \mathcal{Q}, \mathcal{F} \rangle$ is the managers' time. This is equal to the number of operations,[3] which we denote by W.

Within the class of functional networks for processing the N data, we define dominance, weak dominance and efficiency in the standard way with respect to D and W, given that less of either is better.

The number of managers in $\mathcal{Q}$ is called the size of the network and is denoted by Q. In order to compare with Radner (1993), we also consider Q as a measure of resources, in which case we use the terms size-dominance, weak size-dominance and size-efficiency.

The organizational structure that we infer from a network $\langle \mathcal{Q}, \mathcal{F} \rangle$ is a directed graph, called the *communication graph*, which is a static summary of the flow of information between data sources and managers. Let $\mathcal{N} \equiv \{1, \ldots, N\}$ be the set of data sources. The communication graph is the directed graph whose set of nodes is $\mathcal{Q} \cup \mathcal{N}$ and in which (i) there is an edge from manager q_1 to manager q_2 if and only if q_1 sends a message that is processed by q_2, and (ii) there is an edge from a data source n to a manager q if and only if q processes an INPUT instruction for data source n. Note that the communication graph does not show the topology of the physical network through which messages are transmitted (which is not modeled), but rather it shows links between senders

[3] For this computation model, a manager cannot tend to other information processing tasks while holding a value in her register that she will use for subsequent calculations. If the wages represent the opportunity cost of a manager's attention rather than compensation for disutility of effort, such idle time should also be included in the managerial costs. It is noted at the end of Sect. 4 that including this cost would further refine the set of efficient networks but would not substantially change the results of this paper.

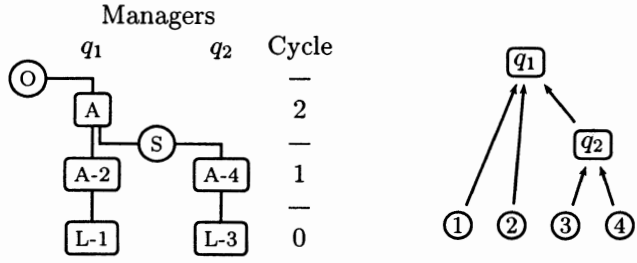

Fig. 1. The left-hand side is a graphical representation of a network with 2 managers, for $N = 4$. The graph is a reduced form of the execution graph, which is defined in Van Zandt (1997c). The nodes of the graph are instructions; L, A, S and O denote LOAD, ADD, SEND and OUTPUT instructions, respectively. The INPUT instructions are not shown, but A-2, for example, denotes an ADD operation that processes the INPUT instruction for data source 2. This network has 5 operations and a delay of 3. The graph on the right is the communication graph

and final recipients of messages. If the communication graph is a tree, then the network is said to be *hierarchical*.

Recall the example given earlier in this section of two managers who aggregate 4 data. The left-hand side of Fig. 1 gives a graphical representation of this network. (The interpretation of this kind of graph, which is used to represent other networks in this paper, is explained in the caption.) This network has a delay of 3 cycles and has 5 operations. The right-hand side of Fig. 1 shows the communication graph, which is a tree.

3 Essential and simple networks

A network $\langle \mathcal{Q}, \mathcal{J} \rangle$ is defined in Van Zandt (1997c) to be *essential* if it is functional and if it is not possible to delete a manager $q \in \mathcal{Q}$ or an instruction $i \in \mathcal{J}$ and still obtain a well-defined network $\langle \mathcal{Q} \setminus \{q\}, \mathcal{J} \rangle$ or $\langle \mathcal{Q}, \mathcal{J} \setminus \{i\} \rangle$ that is functional. In searching for efficient networks, we can restrict attention to essential networks, because any network is weakly dominated and weakly size-dominated by an essential network (which can be constructed by iteratively deleting extraneous managers and instructions). An efficient or size-efficient network might not be essential, because an efficient network can contain messages that are never processed or managers who execute no operations, and a size-efficient network can contain operations that never affect the value of the output. However, if we were to treat lexicographically the number of instructions or the number of managers as a cost, then efficient networks would have to be essential. For this reason, we restrict attention to essential networks when deriving necessary conditions for efficiency.

Proposition 3.1 below provides a useful characterization of essential networks. The main message of this proposition is that, in an essential network, each manager's activities can be divided into one or more *tasks*. Intuitively, a task consists of the steps by which a manager aggregates a number of pieces of information into a single message and sends that message off. A task begins

with a LOAD, and then each subsequent piece of information is aggregated with an ADD. The resulting value is then sent with a SEND or OUTPUT instruction. Specifically:

Definition 3.1 *Let $\langle \mathcal{Q}, \mathcal{J} \rangle$ be a network. A set $\mathcal{H} \subset \mathcal{J}$ of instructions is called a* task *if and only if (i) all instructions in $\mathcal{H}$ are executed by the same manager (who is said to perform the task), (ii) the first instruction in $\mathcal{H}$ is a LOAD and the last is a SEND or OUTPUT, (ii) all other instructions in $\mathcal{H}$ are ADDs, and (iv) there is no instruction in $\mathcal{J} \setminus \mathcal{H}$ executed by the manager who performs $\mathcal{H}$ after the LOAD and before the message in $\mathcal{H}$.*

Proposition 3.2 (Van Zandt 1997c) *A network $\langle \mathcal{Q}, \mathcal{J} \rangle$ is essential if and only if:*

1. *Each manager's instructions can be partitioned into one or more tasks.*
2. *There is a single OUTPUT instruction.*
3. *For each of the N data, there is a single INPUT instruction.*
4. *Each INPUT and SEND is processed by a single operation.*

For example, the network in Fig. 1 is essential, and each manager has a single task. Also, the networks in Radner (1993) and Keren and Levhari (1979, 1983) satisfy the properties of this characterization and hence are essential.

This characterization is a useful tool for proving the results of this paper. To show that a network is efficient, we only have to show that it is not dominated by any networks with the properties listed in the proposition. Furthermore, we can show that a network is functional by checking that it has these properties.

An essential network is not necessarily hierarchical. For example, in the network on the left side of Fig. 2, manager q_1 executes instructions in one task and sends a message to manager q_2, and then, in a subsequent task, processes a message from q_2. However, this is not possible if each manager in $\mathcal{Q}$ has a single task, as in the network on the right side of Fig. 2. Such a network is called *simple*.

Definition 3.3 *A network $\langle \mathcal{Q}, \mathcal{J} \rangle$ is* simple *if it is essential and if each manager performs a single task.*

Proposition 3.4 *A simple network is hierarchical.*

Proof: In a simple network, each manager sends a single message, and only one manager's message is an OUTPUT. Each of the remaining $Q - 1$ messages is a SEND, which is processed by a manager other than the sender. (A manager cannot process her own SEND because she has no further instructions in a simple network.) In an essential network, each datum is processed by a single manager. Hence, (i) there are $N + Q - 1$ edges in the communication graph. Furthermore, a manager's SEND must be executed before the SEND of her immediate successor in the graph, and so (ii) the graph is acyclic. A directed graph with properties (i) and (ii) is a tree. □

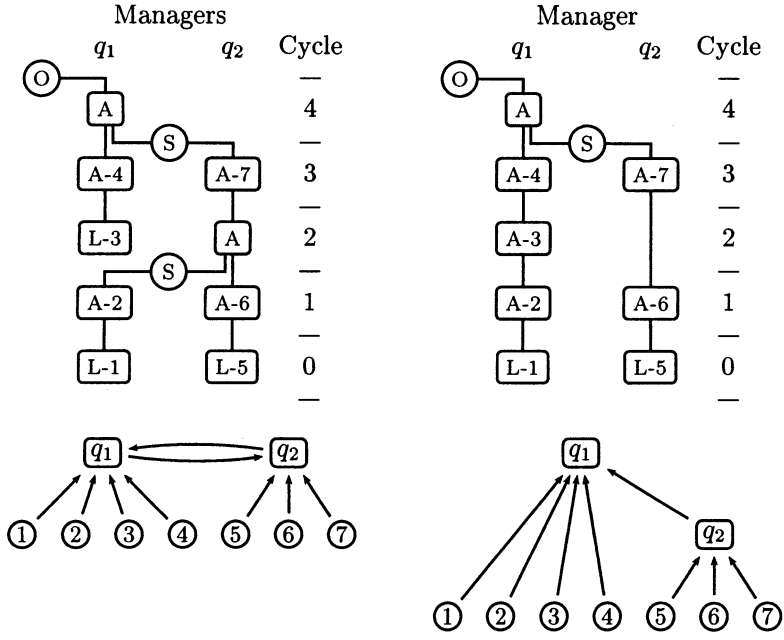

Fig. 2. Two networks for $N = 7$. The bottom row shows the communication graphs. Both networks have 2 managers, but the l.h.s. has 8 operations whereas the r.h.s. has 7. The l.h.s. is not simple, because manager q_1 has two tasks. The l.h.s. is also not hierarchical. The r.h.s. is simple and hierarchical

4 Efficient versus size-efficient networks

Henceforth, a network, with no further qualification, is assumed to be essential.

The following proposition tells us that if we restrict attention to a class of simple networks, then the number of managers is monotonically increasing in the number of operations, and vice-versa. Hence, a network is efficient within the class if and only if it is size-efficient.

Proposition 4.1 *If* $\langle \mathcal{Q}, \mathcal{F} \rangle$ *is simple, then* $W = N + Q - 1$. *Otherwise,* $W > N + Q - 1$.

Proof: Because $\langle \mathcal{Q}, \mathcal{F} \rangle$ is essential, each datum and each SEND are processed by a single operation. Also, the number of SENDs is equal to $H - 1$, where H is the number of tasks. Therefore, the number of operations is equal to $N + H - 1$. If $\langle \mathcal{Q}, \mathcal{F} \rangle$ is simple, then $H = Q$ tasks. Otherwise, $H > Q$. □

The important point here is that decentralization (parallelization), which is measured by the number of managers, incurs managerial costs. As explained in Van Zandt (1996), this is due to an implicit communication cost: Each manager in an essential network sends a message that is processed by another manager, and this reading of messages takes time. That decentralization has such overhead is robust, although there are special, unrealistic models in which this is

not true. In the PRAM model of parallel computation, there are no communication costs and there is no increase in the number of operations as parallelization increases (for associative computation); hence, all efficient networks have maximal parallelization. For the Radner computation model, in contrast, there is a class of efficient networks and decentralization entails a trade-off between higher managerial costs and lower delay.

The rest of this involves showing that we can and should restrict attention to simple networks, and noting that Radner (1993) and Keren and Levhari (1979,1983) do restrict attention to simple networks. Thus, their results on size-efficient networks also provide sufficient conditions for efficient networks.

For example, Keren and Levhari (1979,1983) restrict attention to a class of simple networks whose communication graphs are balanced hierarchies. An example is shown in Fig. 3. In such a hierarchy, the managers are arranged in tiers. Managers in the same tier process approximately the same number of messages. All data are processed by managers in the lowest managerial tier, who send their partial results to managers in the next tier. Managers in this tier process only these messages, and send their results to the next tier, and so on. Because these hierarchies are simple, the characterization in Keren and Levhari (1979) of size-efficient hierarchies within this class also yields a characterization of efficient hierarchies within this class.

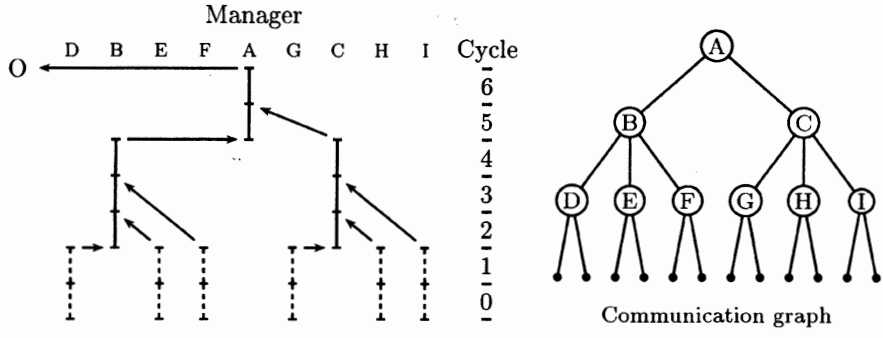

Fig. 3. A balanced hierarchy. (The l.h.s. shows an even more reduced version of the execution graph, with *dashed* segments representing operations that process raw data and *solid* segments representing operations that process messages from other managers)

Radner (1993) does not restrict attention a priori to simple hierarchies. However, he characterizes a set of size-efficient simple networks that span the efficiency frontier, and thereby proves:[4]

Proposition 4.2 (Radner 1993) *Every network is weakly size-dominated by a simple network.*

For efficient networks, we obtain a sharper characterization:

Proposition 4.3 *Every efficient network is simple (and hence hierarchical).*

[4] See Lemma A.3 in the Appendix for a direct proof of this result.

Proof: It follows from Proposition 4.1 that if a network $\langle \mathcal{Q}, \mathcal{F} \rangle$ is not simple, then the simple network $\langle \mathcal{Q}', \mathcal{F}' \rangle$ that weakly size-dominates $\langle \mathcal{Q}, \mathcal{F} \rangle$ (Proposition 4.2) has fewer operations than $\langle \mathcal{Q}, \mathcal{F} \rangle$. □

Figure 2 illustrates how we can eliminate operations in a network that is not simple by concatenating a manager's multiple tasks. The network on the left side of Fig. 2 is not simple because manager q_1 has two tasks. The first task aggregates X_1 and X_2 and then sends the result to manager q_2. The second tasks aggregates X_3, X_4 and the message received from q_2, and then OUTPUTs the result. An alternative design, shown on the right side of the figure, is obtained by concatenating q_1's task so that q_1 aggregates X_1, X_2, X_3 and X_4 within a single task. The SEND from q_1 to q_2 and the ADD by q_2 which processes the SEND can then be deleted because they are no longer needed. This reduces the number of operations in the network by one.

Corollary 4.1 *A network is efficient if and only if it is size-efficient and simple.*

Proof: If $\langle \mathcal{Q}, \mathcal{F} \rangle$ is efficient, it is simple (Proposition 4.3). If $\langle \mathcal{Q}, \mathcal{F} \rangle$ is not size-efficient, it is size-dominated by a simple network $\langle \mathcal{Q}', \mathcal{F}' \rangle$ (Proposition 4.2). Since $W' = N + Q' - 1$ and $W \geq N + Q - 1$ (Proposition 4.1), and since $Q' \leq Q$, we have $W' \leq W$. Furthermore, either $D' < D$ or $Q' < Q$, and in the latter case $W' < W$. Hence, $\langle \mathcal{Q}', \mathcal{F}' \rangle$ also dominates $\langle \mathcal{Q}, \mathcal{F} \rangle$, which is therefore not efficient.

Let $\langle \mathcal{Q}, \mathcal{F} \rangle$ be simple. If $\langle \mathcal{Q}, \mathcal{F} \rangle$ is not efficient, there is a simple network $\langle \mathcal{Q}', \mathcal{F}' \rangle$ that dominates $\langle \mathcal{Q}, \mathcal{F} \rangle$. Recall that $W = N + Q - 1$ and $W' = N + Q' - 1$. Since $W' \leq W$, $Q' \leq Q$. Furthermore, either $D' < D$ or $W' < W$, and in the latter case $Q' < Q$. Hence, $\langle \mathcal{Q}', \mathcal{F}' \rangle$ also size-dominates $\langle \mathcal{Q}, \mathcal{F} \rangle$, which is therefore not size-efficient. □

Corollary 4.2 *Let $Q \geq 1$ and let $W = N + Q - 1$. Then $\langle W, D \rangle$ is an efficient performance if and only if $\langle Q, D \rangle$ is a size-efficient performance.*

In conclusion, the class of size-efficient simple networks in Radner (1993) that span the size-efficiency frontier is also a class of efficient networks that span the efficiency frontier, and the size-efficiency frontier in Radner (1993) is easily mapped to an efficiency frontier via $\langle Q, D \rangle \mapsto \langle N + Q - 1, D \rangle$.

Note, however, that efficiency is a sharper criterion than size efficiency. According to Proposition 4.3, all efficient networks are hierarchical, whereas there are size-efficient networks that are not hierarchical, such as the one on the left side of Fig. 2. In such networks, there is slack that allows managers to engage in wasteful message exchanges; these increase the number of operations and hence are ruled out by efficiency.

There are several other costs that, when introduced to the model, might refine but would not significantly change the set of efficient networks. For example, introducing a cost to communication links or messages would not change the set of efficient networks at all, because efficient networks have the minimum number $N + Q - 1$ of messages and of edges in the communication graph for

the communication graph to be connected, and hence they already economize on such costs. As shown in Van Zandt (1998), each network is weakly dominated by an efficient network in which each manager only needs to hold one item in her buffer at any time and in which managers are not idle after beginning and before finishing a task. Hence, including memory costs or wages for managers when idle within tasks (during which time they could not attend to other information processing activities) would refine but not otherwise change the set of efficient networks.

5 Irregularity of efficient and size-efficient networks

Radner (1993) showed that balanced hierarchies, such as the ones in Keren and Levhari (1979) or the network in Fig. 3, although appealing because of their regular structure, are typically not size-efficient. They can be "reduced" by combining the operations of a manager in one tier with the operations of a manager in a lower tier, because these operations are not concurrent. The reduction not only eliminates managers, but it also eliminates the messages these managers send along with the operations that process these messages, and so Radner's reduction also shows that these hierarchies are not efficient.

Proposition 5.1 generalizes this reduction process, and shows that each manager must process at least two raw data. This property, which is shared by the efficient networks in Radner (1993), is thus shown to be a necessary property of efficient and size-efficient networks.

Proposition 5.1 *In efficient and size-efficient networks, each manager performs at least two operations before any other manager executes her last instruction, and each manager processes at least two raw data.*

Proof: See Appendix. □

Figure 4 illustrates the generalized reduction process that is used in the proof of Proposition 5.1. In the network at the top of Fig. 4, manager q_2 only performs one operation before manager q_1 sends a message. Therefore manager q_2 is free during all but one of the cycles in which q_1 processes data, and hence q_2 can perform all but one of q_1's operations. The reassignment of these operations from q_1 to q_2 is the first step in the reduction, and results in a network in which there is a manager, q_1, who performs only one operation. This network is shown at the bottom-left of Fig. 4. The second step involves eliminating q_1 entirely. q_1 now processes just one piece of data, X_3, and then SENDs a message containing just X_3 to manager q_3. X_3 is thus handled twice. q_3 can instead receive X_3 directly from the input device rather than from q_1. This means that we eliminate q_1 and her instructions, and q_3 processes X_3 in the same cycle in which she processed q_1's message in the original network. We thereby obtain the network at the bottom-right of Fig. 4, which has the same delay but one less manager and one less operation than the original network.

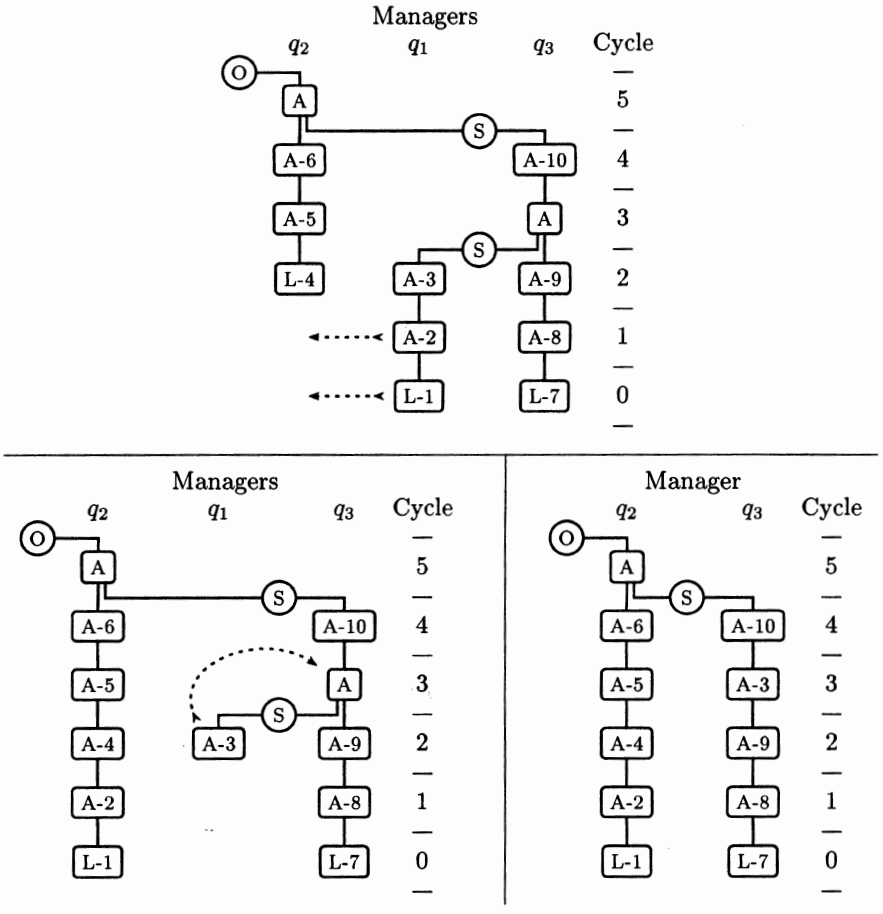

Fig. 4. The *top network* does not have strongly overlapping managers. The *bottom-left network* is obtained by transferring all but one of q_1's operations to manager q_2, as indicated by the *dashed lines*. The *bottom-right network* is then obtained by transferring q_1's remaining operation to manager q_3 in the cycle in which q_3 processed q_1's message. The resulting network also processes 10 items with a delay of 6, but it has 1 less manager and 1 less operation

6 Costs and returns to parallelization

Section 4 tells us that the structure of efficient networks is not drastically different from that of size-efficient networks. Of course, the magnitudes of the measures of managerial resources are quite different. This matters when optimal hierarchies are chosen, given a cost of managers and an explicit or implicit cost of delay, such as in the analysis of returns to scale in Radner (1993) and Keren and Levhari (1983).

As an example, suppose that $N = 1000$. A manager can process the data in 1000 cycles, whereas a simple network with 2 managers can process the data in 501 cycles. This reduction in delay is the benefit of parallelization. If we measure resources by the number of managers, it seems that the benefit comes at the cost

of a doubling of the managerial resources. However, when measuring resources by the number of operations, the increase in managerial costs is relatively minor, from 1000 to 1001.

Nevertheless, it turns out that the conclusions of Radner (1993) and Keren and Levhari (1983) on returns to scale are also not affected significantly by the measure of resources. In Radner (1993), the study of returns to scale is brief and exploratory. He studies a case in which the cost is linear in both managers and delay, which is sensitive to the measure of resources. However, the more realistic case that he studies is where the payoffs are $Nf(D)$, for a function f that decreases to zero as $D \to \infty$. The decreasing returns to scale obtained for this case depend only on the fact that delay increases inexorably in N, even with maximal parallelization. This does not depend on the measure of managerial resources. Keren and Levhari (1983) study a similar but more general payoff function, and for the same reasons their results depend mainly on the rate at which delay increases with problem size.

Appendix

The formal definition of instructions and networks takes advantage of the fact that, when there are no information transmission costs, it is not necessary to specify who receives a message, but rather only who processes a message. It is as if all messages are broadcast to all managers.

In a network $\langle \mathcal{Q}, \mathcal{J} \rangle$, instructions are strings of the following forms:

$$\text{INPUT}(t,n,a) \quad \text{SEND}(t,q,a) \quad \text{OUTPUT}(t,q) \quad \text{LOAD}(t,q,a) \quad \text{ADD}(t,q,a)$$

The argument $t \in \mathbb{N}$ is called the execution time, $q \in \mathcal{Q}$ is the manager who executes the instruction, and $a \in \mathbb{N}$ is the message ID. As long as no confusion should arise, we use a message ID a to denote both the ID itself and the message with ID a, which is required to be unique in a network. $\text{INPUT}(t,n,a)$ means that the input device sends X_n with message ID a. $\text{SEND}(t,q,a)$ means that manager q sends the value of her register to the other managers with ID a. $\text{OUTPUT}(t,q)$ means that manager q sends the value of her register to the output device. $\text{LOAD}(t,q,a)$ (respectively, $\text{ADD}(t,q,a)$) means that manager q retrieves from her buffer the value of message a and stores it in her register (respectively, aggregates it to the current value in her register). See Van Zandt (1997c) for details.

In this appendix, all networks are essential. For a network $\langle \mathcal{Q}, \mathcal{J} \rangle$ and for manager $q \in \mathcal{Q}$, we use the following notation:

$$\begin{aligned}
\mathcal{H}(q) &\equiv \text{instructions in } \mathcal{J} \text{ executed by } q. \\
b(q) &\equiv \text{cycle in which } q \text{ begins (execution time of } q\text{'s first LOAD).} \\
c(q) &\equiv \text{cycle in which } q \text{ finishes (execution time of } q\text{'s last message).} \\
w(q) &\equiv \text{number of operations performed by } q.
\end{aligned}$$

We will also denote some networks by $\langle \mathcal{Q}', \mathcal{J}' \rangle$ or $\langle \mathcal{Q}, \mathcal{J}' \rangle$, in which case primes are added to the above notation: $\mathcal{H}'$, c', w', D', W', and Q'.

The proof of Proposition 5.1 is divided into Lemmas A.1 and A.2.[5] The first lemma shows how to improve networks in which some manager has a single operation. The construction is illustrated by two networks at the bottom of Fig. 4. (See also the discussion of this figure in Sect. 5.)

Lemma A.1 *Let $\langle \mathcal{Q}, \mathcal{J} \rangle$ be a simple network such that a manager q^* has a single operation. Then there is a simple network $\langle \mathcal{Q} \setminus \{q^*\}, \mathcal{J}' \rangle$ such that $\forall q \in \mathcal{Q} \setminus \{q^*\}$, $b'(q) = b(q)$, $c'(q) = c(q)$ and $w'(q) = w(q)$. Hence, $Q' = Q - 1$, $W' = W - 1$ and $D' = \max\{c(q) | q \in \mathcal{Q} \setminus \{q^*\}\} \leq D$.*

Proof: We delete q^*'s instructions $\mathcal{H}(q^*)$ and then reroute the message processed by q^*'s single operation either to the output device or to the operation that processes q^*'s message. That is, we obtain $\mathcal{J}'$ from $\mathcal{J} \setminus \mathcal{H}(q^*)$ by modifying one instruction, as follows. Let i_1 and i_2 be the operation (a LOAD) and message (a SEND or OUTPUT), respectively, in $\mathcal{H}(q^*)$. Let i_0 be the message processed by i_1. (The values of i_0, i_1 and i_2 are the same.) (i) If i_2's type is OUTPUT, then (since $N \geq 2$), i_0 is a SEND. Change i_0's type to OUTPUT. (ii) If i_2's type is SEND, then it is processed by an operation i_3. Change the message ID in i_3 to the message ID of i_0, so that it processes i_0 instead of i_2. Then $\langle \mathcal{Q} \setminus \{q^*\}, \mathcal{J}' \rangle$ has the desired properties. □

Remark A.1 Let i_1 and i_2 be consecutive instructions in the same task, with execution times t_1 and t_2, respectively. The manager who performs the task is idle in cycles $t_1 + 1, \ldots, t_2 - 1$, and so i_1's execution time can be increased as long as it does not exceed $t_2 - 1$. i_1 is then still executed after the message it processes and is part of its original task. Hence, the resulting network is also essential and has the same delay as the original network. This fact is used in the proof of Lemma A.2.

The main new point in the next lemma (in Step 2 of the proof) is to show that, if a manager q_1 has only one operation after another manager q_2 starts processing, then we can obtain a network in which q_1 has a single operation by transferring this managers other operations to q_2. This is illustrated by the two networks at the top and bottom-left of Fig. 4.

Lemma A.2 *Let $\langle \mathcal{Q}, \mathcal{J} \rangle$ be a simple network with managers $q_1, q_2 \in \mathcal{Q}$ such that q_2 does not perform two operations before q_1 is finished and such that $c(q_2) \geq c(q_1)$. Then there is a simple network $\langle \mathcal{Q} \setminus \{q_1\}, \mathcal{J}' \rangle$ that has the same delay as and one less operation than $\langle \mathcal{Q}, \mathcal{J} \rangle$ such that:*

1. $\forall q \in \mathcal{Q} \setminus \{q_1, q_2\}$: $b'(q) = b(q)$, $c'(q) = c(q)$ *and* $w'(q) = w(q)$.
2. $b'(q_2) \geq b(q_1)$, $c'(q_2) = c(q_2)$ *and* $w'(q_2) = w(q_1) + w(q_2) - 1$.

[5] These results are also used in Van Zandt (1998) to characterize a class of efficient networks in the periodic model.

Proof: If $\mathscr{H}(q_1)$ has a single operation, then the network $\langle \mathscr{Q} \setminus \{q_1\}, \mathscr{J}' \rangle$ given in Lemma A.1 has the desired properties.

Suppose instead that $\mathscr{H}(q_1)$ has at least two operations. We reroute the last message processed by q_1 to the operation that processes q_1's message and we transfer q_1's remaining operations to q_2.

Step 1. It is assumed that q_2 executes at most one operation before cycle $c(q_1)$. If there is such an operation and it is executed before cycle $c(q_1)-1$, change its time of execution to cycle $c(q_1) - 1$ and let $\mathscr{J}''$ be the resulting set of instructions. (Otherwise, let $\mathscr{J}'' = \mathscr{J}$.) Since q_2's subsequent instruction is either an operation executed in or after cycle $c(q_1)$ or is q_2's message, which is executed in cycle $c(q_2) \geq c(q_1)$, it follows from Remark A.1 that $\langle \mathscr{Q}, \mathscr{J}'' \rangle$ is essential and has the same delay as $\langle \mathscr{Q}, \mathscr{J} \rangle$.

Step 2. Let i^* and i^{**} be q_1's last operation and message, respectively. Let $\mathscr{H}^*(q_1) = \mathscr{H}(q_1) \setminus \{i^*, i^{**}\}$. Since we are assuming that $\mathscr{H}(q_1)$ has at least two operations, $\mathscr{H}^*(q_1)$ is non-empty. We construct $\mathscr{J}'''$ from $\mathscr{J}''$ by transferring the instructions in $\mathscr{H}^*(q_1)$ to manager q_2 as follows:

1. Change i^*'s type to LOAD, so that $\{i^*, i^{**}\}$ constitute a task for q_1.
2. Because i^* is performed no later than in cycle $c(q_1) - 1$, the operations in $\mathscr{H}^*(q_1)$ are executed before cycle $c(q_1) - 1$. As constructed in Step 1, all the instructions in $\mathscr{H}''(q_2)$ are executed during or after cycle $c(q_1)-1$. Therefore, we can change the manager who executes the instructions in $\mathscr{H}^*(q_1)$ from q_1 to q_2, and q_1 still performs only one operation in each cycle.
3. Change the type of q_1's first operation in $\mathscr{H}''(q_2)$ from LOAD to ADD so that $\mathscr{H}^*(q_1) \cup \mathscr{H}''(q_2)$ constitute a task for q_2.

The resulting set $\mathscr{J}'''$ of instruction is such that $\langle \mathscr{Q}, \mathscr{J}''' \rangle$ satisfies the properties of a simple essential network and has the same delay as $\langle \mathscr{Q}, \mathscr{J} \rangle$, and $b'''(q_2) = b(q_1)$, $c'''(q_2) = c(q_1)$ and $w'''(q_2) = w(q_1) + w(q_2) - 1$. This modification is illustrated by the top and bottom-left networks in Fig. 4.

Step 3. In the network $\langle \mathscr{Q}, \mathscr{J}''' \rangle$, manager q_1 has a single operation. Then by Lemma A.1, the desired network $\langle \mathscr{Q} \setminus \{q_1\}, \mathscr{J}' \rangle$ can be obtained by eliminating manager q_1 and her instructions from $\langle \mathscr{Q}, \mathscr{J}''' \rangle$. (Note that because $c(q_2) \geq c(q_1)$, $\langle \mathscr{Q} \setminus \{q_1\}, \mathscr{J}' \rangle$ has the same delay as $\langle \mathscr{Q}, \mathscr{J} \rangle$.) □

Proof of Proposition 5.1: We first prove the result for a simple network $\langle \mathscr{Q}, \mathscr{J} \rangle$. Suppose that either (i) there is a manager $q \in \mathscr{Q}$ who processes fewer than 2 data or (ii) there are managers $q_1, q_2 \in \mathscr{Q}$ such that q_2 has fewer than 2 operations before q_1 finishes. If (i) holds, then q's second operation, if it exists, cannot be performed until the first SEND, and since $\langle \mathscr{Q}', \mathscr{J}' \rangle$ is simple, this does not occur until the first manager is finished. Hence, (ii) also holds. If (ii) holds and if q_2 has a single operation, then $\langle \mathscr{Q}, \mathscr{J} \rangle$ network is neither efficient nor size-efficient according to Lemma A.1. Suppose instead q_2 has at least two operations; then $c(q_2) \geq c(q_1)$. Therefore, according to Lemma A.2, $\langle \mathscr{Q}, \mathscr{J} \rangle$ is neither efficient nor size-efficient.

This proof is complete for efficient networks, as these must be simple. Lemma A.3 extends the proof to size-efficient networks $\langle \mathcal{Q}, \mathcal{J} \rangle$ that are not simple, because the weakly size-dominating simple network $\langle \mathcal{Q}', \mathcal{J}' \rangle$ given in that lemma satisfies (i) or (ii) if $\langle \mathcal{Q}, \mathcal{J} \rangle$ does. □

The following lemma is used to prove Proposition 5.1 and also provides a direct proof of Proposition 4.2. It construction is illustrated in Fig. 2.

Lemma A.3 *Let $\langle \mathcal{Q}, \mathcal{J} \rangle$ be a network. There is a simple network $\langle \mathcal{Q}', \mathcal{J}' \rangle$ with the same delay as $\langle \mathcal{Q}, \mathcal{J} \rangle$ such that $\mathcal{Q}' \subseteq \mathcal{Q}$ and, for each $q \in \mathcal{Q}'$, (i) $c'(q) = c(q)$, (ii) $w'(q) \leq w(q)$, (iii) $q \in \mathcal{Q}'$ processes the same number of raw data in $\langle \mathcal{Q}', \mathcal{J}' \rangle$ as in $\langle \mathcal{Q}, \mathcal{J} \rangle$, and (iv) if $q \in \mathcal{Q}'$ performs an operation in $\mathcal{J}'$ in cycle t, then q performs an operation in $\mathcal{J}$ in cycle t.*

Proof: Construct $\mathcal{J}'$ from $\mathcal{J}$ as follows:

1. Delete each SEND that is not a manager's last message in $\mathcal{J}$, and delete the operation that processes the SEND. Let $\mathcal{J}''$ be the remaining instructions.
2. If a manager has no operations in $\mathcal{J}''$, delete from $\mathcal{J}''$ the manager's remaining messages and the operations that process the messages and delete the manager from $\mathcal{Q}$. Repeat this step until all remaining managers have operations, and then let $\mathcal{Q}'$ be the set of remaining managers and let $\mathcal{J}'''$ be the set of remaining instructions.
3. For each $q \in \mathcal{Q}'$, set the type of $\mathcal{Q}$'s first operation in $\mathcal{J}'''$ to LOAD and set the type of q's other operations in $\mathcal{J}'''$ to ADD. Let $\mathcal{J}'$ be the resulting set of instructions.

We verify that $\langle \mathcal{Q}', \mathcal{J}' \rangle$ has the properties of an essential network listed in Proposition 3.2. (1) For $q \in \mathcal{Q}'$, we have constructed $\mathcal{H}'(q)$ so that it is a task. (2) The OUTPUT instruction in $\mathcal{J}$ is also in $\mathcal{J}'$, unless the manager $q \in \mathcal{Q}$ who executes the instruction is not in $\mathcal{Q}'$. This is impossible as follows. Consider the directed graph whose nodes are the managers in $\mathcal{Q}$ and in which, for $q_1, q_2 \in \mathcal{Q}$, there is an edge from q_1 to q_2 if and only if q_1's *last* message in $\mathcal{J}$ is processed by q_2. By the proof of Proposition 3.4, this graph must be a tree whose root is q. Furthermore, if $q_1 \in \mathcal{Q}'$, $q_2 \in \mathcal{Q}$ and q_2 is the parent of q_1, then q_2's operation that processes q_1's last message was not eliminated and hence $q_2 \in \mathcal{Q}'$. Hence, if $\mathcal{Q}' \neq \emptyset$, then $q \in \mathcal{Q}'$. $\mathcal{Q}' \neq \emptyset$ because any manager who processes raw data in $\langle \mathcal{Q}, \mathcal{J} \rangle$ is in $\mathcal{Q}'$. (3) $\mathcal{J}$ and $\mathcal{J}'$ have the same INPUT instructions. (4) Each INPUT and SEND in $\mathcal{J}'$ is processed by the same manager in the same cycle as in $\langle \mathcal{Q}, \mathcal{J} \rangle$.

Since the OUTPUT instructions in $\mathcal{J}$ and $\mathcal{J}'$ are the same, $\langle \mathcal{Q}, \mathcal{J} \rangle$ and $\langle \mathcal{Q}', \mathcal{J}' \rangle$ have the same delay. □

References

Baumol, W.J., Blackman, S.A.B., Wolff, E.N. (1989) *Productivity and American Leadership: The Long View*. MIT Press, Cambridge, MA

Beggs, A.W. (1995) *Queues and hierarchies.* Wadham College, Oxford University

Gittleman, M., Wolff, E.N. (1995) R&D activity and cross-country comparisions. *Cambridge Journal of Economics* 19:189–207

Keren, M., Levhari, D. (1979) The optimum span of control in a pure hierarchy. *Management Science* 11:1162–1172

Keren, M., Levhari, D. (1983) The internal organization of the firm and the shape of average costs. *The Bell Journal of Economics* 14:474–486

Malone, T.W., Smith, S.A. (1988) Modeling the performance of organizational structures. *Operations Research* 36:421–436

Mount, K., Reiter, S. (1990) A model of computing with human agents. The Center for Mathematical Studies in Economics and Management Science, Discussion Paper No. 890, Northwestern University, Evanston, Illinois

Radner, R. (1992) Hierarchy: The economics of managing. *Journal of Economic Literature* 30:1382–1415

Radner, R. (1993) The organization of decentralized information processing. *Econometrica* 62:1109–1146

Radner, R., Van Zandt, T. (1992) Information processing in firms and returns to scale. *Annales d'Economie et de Statistique* 25/26:265–298

Van Zandt, T. (1996) Organizations with an endogenous number of information processing agents. In: M. Majumdar (ed.) *Organizations with Incomplete Information.* Cambridge University Press, Cambridge

Van Zandt, T. (1997a) Real-time decentralized information processing as a model of organizations with boundedly rational agents. *Review of Economic Studies* (Forthcoming)

Van Zandt, T. (1997b) *Real-time hierarchical resource allocation.* Princeton University, Princeton

Van Zandt, T. (1997c) The scheduling and organization of periodic associative computation: Essential networks. *Review of Economic Design* 3:15–27

Van Zandt, T. (1998) The scheduling and organization of periodic associative computation: Efficient networks. *Review of Economic Design* 3:93–127

Van Zandt, T., Radner, R. (1997) Real-time decentralized information processing and returns to scale. Princeton University Princeton and New York University, New York