Speed and Quality of Collective Decision Making: Imperfect Information Processing*

Elisabeth SCHULTE†
University of Mannheim

Hans Peter GRÜNER
University of Mannheim and CEPR, London

November 2003, revised February 2005,
2nd revision September 2005, this version: February 2006

Abstract
This paper studies efficient programmed hierarchies as introduced by Radner (1993) in which agents cannot process information perfectly. A group of \( P \) identical managers has to make a choice between \( n \) alternatives. In order to learn which is the best option, the alternatives have to be compared. The evaluation of an alternative takes time and managers are only able to identify the better one of two alternatives with a positive probability. The skip-level reporting tree proposed by Radner is found to be efficient in terms of the dimensions decision cost, decision delay, and decision quality.

Keywords: Information processing, hierarchies, bounded rationality.

JEL: D23, D70, D83, L22, P51

*We thank summer school participants at Hebrew University, especially Timothy Van Zandt, seminar participants at the 2002 Silvaplanar Workshop on Political Economy, the Theory Workshop and the Workshop on International Negotiations, University of Mannheim, the ZEI conference on Federalism, University of Bonn, ESEM 2004, and seminars at Erasmus University Rotterdam and UCLA, Enno Mammen, Thomas Gall, and in particular two anonymous referees and an associate editor for helpful comments and suggestions.

†Correspondence address: Elisabeth Schulte, University of Mannheim, Department of Economics, L7, 3-5, D-68131 Mannheim, Germany; email: elisabeth.schulte@vwl.uni-mannheim.de.
1 Introduction

In a seminal paper, Radner (1993) studies the efficient design of hierarchical structures when information processing takes time. Radner departs from the conventional assumption that individuals process information at infinite speed. He studies the problem of aggregating $n$ data items at maximum speed when $P$ information processors are available. This leads him to propose a hierarchical structure within which information is processed at maximum speed, the "reduced tree".1 The virtue of the reduced tree is that processors on all levels simultaneously process information. This minimizes the delay of the entire information processing procedure. Radner’s model can be applied to any information processing problem which requires repetitions of associative and commutative operations. One is the "max"-operation used in the collective decision problem which we consider in this paper. Information processing in this case implies the pairwise comparison of possible alternatives and the identification of the best one.

Radner’s analysis is focused on the efficient organization of information processing with respect to three dimensions: (i) the size of the information processing task, (ii) the size of the organization, and (iii) the delay within which the task is completed. An organizational form is considered to be efficient, if given the number of processors involved, the delay cannot be reduced, and at the same time this delay cannot be achieved with a smaller number of processors.

In this paper, we add a new dimension to this evaluation of hierarchies: the quality of a decision. In Radner’s original paper, which draws on a model brought forward by computer scientists (e.g. Gibbons and Rytter (1988)), it is assumed that processors work perfectly when they perform their task. But, in many real life situations individuals may make mistakes. In our analysis, we study a hierarchy which is composed of agents with imperfect calculation ability. Consequently, the evaluation of a hierarchy is carried out in terms of three dimensions: (i) the speed as well as (ii) the cost of information processing (i.e. the number of agents involved), and (iii) the quality of the decision.

We consider the project selection example proposed by Radner (1993). The organization’s task is to select one item out of a class of $n$ items. This corresponds, for example, to the choice of an investment project out of a set of competing investment opportunities.

Our mathematical analysis focuses on two measures: the probability that the best and the probability that the worst object is chosen by the hierarchy. We take these to be our measures of quality. Our main result is that reduced trees maximize quality for any

---

1The reduced tree will be described in greater detail in Section 2.3.
number of data items, any number of processors and any potential for making mistakes. Thus, the reduced tree is efficient in terms of all three dimensions, speed, cost, and decision quality. Consequently, no trade-off exists between decision speed and decision quality in hierarchy design.

Our paper is related to a recent literature that extends Radner’s framework into various directions. The reduced tree is designed for one-shot problems (to which we restrict attention). These are problems in which there is only one set of data to be processed, or in which the processing of the data is finished before another calculation task occurs. There are several contributions which assess the design of a hierarchy, or more generally a network of agents, when this is not the case (e.g. Van Zandt (1997, 1998); Meagher, Orbay and Van Zandt (2001)). Meagher and Van Zandt (1998) modify Radner’s work with respect to the payment of managers. Orbay (2002) adds the frequency with which new data arrives as a new dimension to the analysis of efficient hierarchies. Prat (1997) studies hierarchies in which some managers are able to work faster than others and the wage a manager is paid is a function of his ability. It turns out that with these modifications – except for the one made by Prat (1997) – the reduced tree is still (close to) efficient.

To our knowledge, all previous models of time consuming information processing in hierarchies treat the information processing agents more or less like machines, which perfectly do what they are programmed to do. The value added of this paper is to take into account the fact that human beings may make mistakes.\footnote{One can argue that machines may make mistakes as well. In a companion paper, we study organization design in a moral hazard setup, where agents are free to decide whether to obey the program. Interestingly, the efficiency result extends to this setup.}

In this regard, the problem studied in this paper has certain similarities to that in Sah and Stiglitz (1986). Sah and Stiglitz assess the relative performance of two economic systems, namely a hierarchy and a polyarchy, when agents make mistakes in the assessment of projects. They find that a hierarchy is less likely to accept both bad projects and good projects than the polyarchy because the polyarchy gives a second chance to rejected projects and the hierarchy performs a second test on accepted projects. Sah and Stiglitz assume that agents use some benchmark for the assessment of projects and that they may implement as many projects as they like. Whereas in our model, the objective is to choose the best object out of a given set.

Another analysis of the quality of hierarchical decision processes has previously been carried out in Jehiel (1999). Jehiel considers the case where some information is lost during the aggregation procedure. With a certain probability, which depends on the size of a
hierarchical unit, information aggregation within this unit is imperfect. Jehiel’s measure of quality is the probability of perfect information aggregation. Optimal organizations consist of units with the same number of members.

Our formal analysis uses some results on the optimal (quality-maximizing) allocation of calculation tasks in van Zandt (2003). Van Zandt also studies the way in which the allocation of calculation tasks affects decision quality. In his setup, information aggregation is perfect, while the underlying information is changing over time. Delay generates costs because the decision becomes less and less appropriate over time. As in our paper, this creates a cost when information is processed unequally. Van Zandt shows that the highest decision quality can be achieved with a balanced calculation tree, i.e. a sequence of calculation steps that guarantees a symmetric treatment of all objects.

The corresponding lemma and some further results on the optimal organization of calculation tasks will be briefly presented in Section 3 of this paper. In Section 4, we use these results to study the quality-maximizing structure of the hierarchical network, taking resource constraints into account. Based on the results on optimal calculation trees, we derive a set of necessary and sufficient conditions for maximum quality networks with a given number of processors and objects. We then show that these conditions are satisfied by all reduced trees. Since they also minimize delay, we conclude that reduced trees are efficient. Moreover, any outcome of an efficient organization can be achieved by a reduced tree.

2 The Model

The aim of this paper is to identify the quality-maximizing organization of information processing tasks for a given number of managers with imperfect information processing ability and to relate this design to the organizational form that completes the task fastest (Radner, 1993).

We first describe the decision problem and the limitations of agents’ ability to deal with information. Next, we introduce our notion of hierarchies as well as their representation in trees and briefly recall Radner’s (1992, 1993) result regarding the optimal organizational form with respect to the dimension of time.

2.1 The decision problem

We consider a decision problem in which one out of a set of \( n \) alternatives has to be chosen by a group of \( P \) identical agents. The alternatives (objects) are indexed by \( i = 1 \ldots n \) and
the agents (managers) are indexed by \( p = 1 \ldots P \). The alternatives differ only with respect to quality. Ex ante (prior to information processing) the \( n \) objects are not distinguishable. However, there exists an objective ranking of the objects according to quality. We assume that quality is distributed in such a way that ties never occur, i.e., for two objects \( i \) and \( j \), either \( i \) is of higher quality than \( j \) or vice versa. All managers have identical monotonous preferences regarding quality. In order to find the best alternative, the objects have to be compared. This is the information processing task that we study in this paper. Agents are endowed with an inbox, a processing unit and a memory. In order to learn which alternative is the best one, an agent compares the objects pairwise. He reads an object from his inbox into his processing unit and compares it to the object in the memory. If the memory is empty, he stores the first object he reads without processing.

An information processing task, i.e., the comparison of two objects \( i \) and \( j \), will be denoted \( i \otimes j \). The result of the calculation, denoted \((i \otimes j)\), is meant to be the better of the two objects. However, information processing is imperfect: with probability \((1-q) < 1/2\), the agent makes a mistake\(^3\) such that \((i \otimes j) = i\), although \( j \) is better than \( i \). With probability \( q \), no mistake is made and the agent correctly calculates \((i \otimes j) = j\).

Memory capacity is limited. In particular, an agent can store only one object. After having performed an operation, the agent stores the object he assesses to be the better one in his memory and takes the other one out of the set of possible alternatives.

Time enters the analysis in the following way: It takes a manager one unit of time to read an object he is supposed to process. We assume that neither the operation itself, nor sending a report (i.e., submitting a partial result to another agent) takes time. Thus, in one unit of time, a manager can perform the following tasks: (i) taking an object out of the inbox into the processing unit, (ii) comparing an object in the processing unit to the one in the memory (given that there are objects in both), and (iii) sending a message to the superior.

We do not introduce any assumptions about agents’ preferences except for monotonicity in the chosen object’s quality. Instead, we focus on the two extreme outcomes, which are of relevance for all quality distributions and for all von Neumann-Morgenstern utility functions: the event that the best object is chosen and the event that the worst object is chosen. Since it is the hierarchy’s task to find the best alternative, it is natural to measure quality in terms of the probability that the best (worst) object will be the final outcome, i.e., in terms of the probability of success and the probability of complete failure.

\(^3\) Another interpretation of the assumption that agents make mistakes is that they receive an imperfect signal about which object is better suited to fit their needs.
Accordingly, these probability measures quantify the quality of a decision in our paper.\footnote{In particular, these measures deliver a complete description of situations in which the hierarchy’s task is to identify a certain object and there exists only one of its kind, e.g. a murderer or a thief (whom one would like to choose as a police department – in this case he represents the best object – and avoid choosing as a recruitment team).}

2.2 Trees

We restrict attention to hierarchical organizations. We follow Radner (1993) in defining a hierarchy (an organizational tree) as follows:

**Definition 1** A hierarchy $\mathcal{H}$ is a collection of objects ($n$ data items and $P$ managers), together with a relation among them, called ”superior to”. The relation has the following properties:

1. Transitivity: If $p$ is superior to $p'$, and $p'$ is superior to $p''$, then $p$ is superior to $p''$.
2. Antisymmetry: If $p$ is superior to $p'$, then $p'$ is not superior to $p$. $p'$ is called $p$’s subordinate.
3. There is exactly one object, called the root, that is superior to all the other objects.
4. Except for the root, every object has exactly one immediate superior.

Figure 1 illustrates an example for an organizational hierarchy $\mathcal{H}$ that processes 6 objects and the calculation tree $T(\mathcal{H})$ induced by the program.\footnote{What we call a ”calculation tree” is called a ”binary tree” in Van Zandt (1993).} In this figure (as in those following), $\Box$ represents manager $p$, and the objects are represented by their indices, 1...6. The ”superior to” relation is represented by a link between the objects. A link from an object $i$ to a manager $p$ means that $p$ reads object $i$, a link from a manager $p$ to $p'$ means that $p$ reports the result of his calculation activities to $p'$.

The organizational hierarchy determines who performs which task and who reports to whom. We refer to the assignment of processing tasks as the program and call the organization a programmed hierarchy. The program gives rise to a calculation tree, illustrating the operations to be performed on the objects. There are $n$ objects (represented as the leaves in the tree) to be processed, i.e. $(n-1)$ operations to be performed until the final result is obtained.

The program which underlies the trees depicted in Figure 1 gives the following instructions. In the first unit of time: agent 1, read object 1; agent 2, read object 3; agent 3, read object 5. In the second unit of time: agent 1, read 2, perform $1 \otimes 2$, report $(1 \otimes 2)$ to agent 4, and similar instructions for agents 2 and 3. In the third unit of time: agent 4,
read agent 1’s report, in the fourth unit of time, read 2’s report, perform $(1 \otimes 2) \otimes (3 \otimes 4)$, and in the fifth unit of time, read agent 3’s report, perform $((1 \otimes 2) \otimes (3 \otimes 4)) \otimes (5 \otimes 6)$, and report the result.

Note that a calculation tree can be induced by different organizational structures. For example, the calculation tree in Figure 1 is also obtained if agent 4’s tasks are performed by one of the managers on the lower hierarchy level. However, the organizational design matters in terms of speed.

In our setting with imperfect information processing ability, the associative law of binary operations does not hold anymore. This is why quality depends on the order of calculations. What matters in terms of quality is the calculation tree. The particular organizational tree by which it is induced does not matter in our setup as managers are assumed to be identical with respect to their calculation ability. Hence, who performs a particular operation is irrelevant for quality. However, feasibility restrictions with respect to the hierarchical organization (as the number of managers) limit the set of calculation trees that can be implemented.

2.3 Radner’s efficiency result

In this section, we briefly recall Radner’s (1993) results concerning the optimal organizational design. Decision delay is affected by the extent of parallel computation, i.e. how much information is handled simultaneously. Radner (1992, 1993) derives the following delay-minimizing organization of information processing: (i) Number the $P$ managers subsequently from 1 to $P$ and assign $n/P$ objects to each manager. (If $n/P$ is not an
integer, assign \( \lfloor n/P \rfloor \) to each manager and another one to the last \( n \mod P \) ones.)

(ii) Assign manager \( i \)'s partial result to \((i - 1)\), for each even \( i \). Manager \((i - 1)\) therewith becomes \( i \)'s immediate superior and \( i \) is \((i - 1)\)'s subordinate (if \( P \) is odd, one manager remains unconnected). (iii) Rename the managers who are not yet somebody’s subordinate, assigning the number 1 to the manager with the largest number. Repeat (ii) and (iii) until a single manager remains (the top manager).

Figure 2 depicts a reduced tree in which 4 managers process 12 objects. The virtue of this design is that information is processed in parallel as far as possible. The \( n \) objects are aggregated to the final result in \( \lfloor n/P \rfloor + \lceil \log_2 (P + n \mod P) \rceil \) units of time. Note that in order to maximize the speed of information processing, the efficient number of managers never exceeds \( \lfloor n/2 \rfloor \).

3 Calculation trees with maximum quality

In this section, we study the optimal organization of calculation tasks without paying attention to feasibility constraints with respect to the hierarchical form by which it could be induced. We are interested in the decision quality associated with the organization of information processing.

We take a calculation tree \( T \) as given. The calculation tree determines a calculation.

---

6\(|x|\) denotes the largest integer \( \leq x \), and \([x]\) is the smallest integer \( \geq x \).

7This construction slightly deviates from the one proposed by Radner (1993), which does not have an effect on speed, but – as will become clear later on – may affect quality.

8See Radner (1993) for the proof that this is indeed the delay minimizing organization.

9A different problem with a similar mathematical representation is studied in van Zandt (2003). We thank a referee for pointing out to us that van Zandt’s results might help us to generalize our earlier (Schulte and Grüner, 2004) results on the comparison of different hierarchies.
path for each object, and in particular how many comparisons an object has to pass before being selected. Let $\delta(i, T)$ be the length of the path from leaf $i$ to the root in tree $T$,\(^ {10} \) and let $\delta^T = (\delta(1, T), \ldots, \delta(n, T))$. Given that ex ante the objects are of the same expected quality, the decision quality associated with $T$ is fully described by $\delta^T$.

In order to choose the best object, all calculations performed on this object have to be correct. To choose the worst object, all operations performed on it have to entail mistakes. Since the combinatorics for choosing the best or the worst item out of the given set are the same, the calculations are analogous. To simplify the exposition, we focus on the probability of choosing the best object. All of our results apply to quality measured in terms of the probability of picking the worst object as well. This can easily be seen by replacing $q$ with $(1 - q)$ in the calculations and is therefore omitted.\(^ {11} \)

If leaf $i$ is the best object and if $i$ is supposed to be processed $\delta(i, T)$ times in tree $T$, it will be chosen with probability $q^{\delta(i, T)}$. Object $i$ is the best object with probability $1/n$. Hence, our quality measures can be defined as follows.

**Definition 2** $\pi(x, n, T) = \frac{1}{n} \sum_{i=1}^{n} x^{\delta(i, T)}$.

**Lemma 1** With calculation tree $T$, the probability of choosing the best object is $\pi(q, n, T)$, and the probability of choosing the worst object is $\pi(1 - q, n, T)$.

Let $\mathcal{T}$ denote the set of calculation trees with $n$ leaves. With respect to decision quality, we use the following optimality criterion.

**Definition 3** A calculation tree is optimal for the processing of $n$ objects, if it solves $\max_{T \in \mathcal{T}} \pi(q, n, T)$ and $\min_{T \in \mathcal{T}} \pi(1 - q, n, T)$.

Van Zandt (2003) has shown in a different context that calculation trees in which the distance from a leaf to the root differs too much among the leaves are dominated by trees in which these distances are more equal. The same is true in our setup, as we will show in the following. Consider an arbitrary calculation tree $T$, characterized by $\delta^T$. Arrange the tree such that it starts with the deepest nodes (as shown in Figure 3), and label the leaves in descending order as they appear in the tree. We make use of a quality enhancing reorganization of the leaves in order to derive the optimal calculation tree.

\(^{10}\)When there is no ambiguity, we will suppress the argument $T$ and use the notation $\delta_i$.

\(^{11}\)Note that it is not universally valid that an organization which chooses the best object with a higher probability than another organization also has a lower chance to pick the worst object (see Schulte and Grünner (2004)). However, in our model, this is a feature of the optimal organization.
Manipulation (*), ”switching” (van Zandt, 2003):

1. Delete operation \( n \otimes (n - 1) \) and perform the successive operations in the affected branch on \( (n - 1) \) instead of \( (n \otimes (n - 1)) \).
2. Add operation \( 1 \otimes n \).

Call the new tree \( T' \). Manipulation (*) is illustrated in Figure 3.

This manipulation has the following effects:

1. object 1 was formerly processed \( \delta(1, T) \) times, now it is processed \( \delta(1, T) + 1 \) times,
2. object \( n \) was formerly processed \( \delta(n, T) \) times, now it is processed \( \delta(1, T) + 1 \) times,
3. object \( (n - 1) \) was formerly processed \( \delta(n, T) \) times, now it is processed \( \delta(n, T) - 1 \) times.

Thus, the effect of the manipulation on the probability of choosing the best object is:

\[
\pi(q, n, T) - \pi(q, n, T') = \frac{1}{n} \left( -2q^{\delta(n, T)} - q^{\delta(1, T)} + 2q^{\delta(1, T) + 1} + q^{\delta(n, T) - 1} \right) \\
= \frac{1}{n} (2q - 1) \left( q^{\delta(1, T)} - q^{\delta(n, T) - 1} \right). \tag{1}
\]

This expression is positive, if \( \delta(1, T) < \delta(n, T) - 1 \). Hence, the proposed manipulation increases the probability of choosing the best object as long as this inequality holds. If we replace \( q \) with \( 1 - q \) in (1), the expression is negative, verifying that Manipulation (*) leads to a smaller probability of choosing the worst outcome as well.

The intuition for this effect is the following. In the original tree, object 1 had the best position concerning the probability of being chosen, because it had to survive the
smallest number of comparisons. Object 1 is the best (worst) object with probability $1/n$. Manipulation (*) replaces object 1 with an object which has a higher chance to be the best one and a lower chance to be the worst one. This is so because processing a subset of items always increases quality as compared to randomly choosing an object. In the other affected branch, the partial result $(n \otimes n - 1)$ is replaced with object $(n - 1)$, to which a lower probability is attached that it is the best object and a higher probability that it is the worst one. But this object’s position is the worst one in the calculation tree concerning the probability of being chosen. Hence, the first effect dominates the latter.

We now show that by repeatedly applying Manipulation (*) (with appropriate renaming of the objects) the best (highest quality) calculation tree is obtained. Each manipulation increases (decreases) the probability of choosing the best (worst) object, as long as $\delta_1 < \delta_{n - 1}$. One can stop this procedure as soon as $\delta_1 \geq \delta_{n - 1}$, i.e. when the objects are processed as equally as possible. Equation (1) implies that the calculation tree cannot be improved further with this manipulation. In particular, applying Manipulation (*) again only changes the position of the objects in the tree, but does not affect the probability of choosing the best object. Following van Zandt (2003), we call the resulting tree a balanced tree (see Figure 4).

**Definition 4** Consider a calculation tree $T$ with $n$ leaves. Name the leaves $i = 1, \ldots, n$ such that $\delta_1 \leq \delta_2 \leq \ldots \leq \delta_n$, where $\delta_i$ is the length of the path from $i$ to the root. A tree with the property $\delta_1 \geq \delta_{n - 1}$ is called a balanced tree.

Is there now an alternative manipulation that enhances decision quality even further? Assume that this is the case. Then either (i) the manipulation again leads to a balanced tree, or (ii) it yields a tree that is not balanced. In case (i), the manipulation does not increase decision quality because all balanced trees (given $n$ and $q$) produce the same quality. In case (ii), the algorithm described above can be applied to the resulting non-balanced tree, resulting in a balanced tree which is superior in terms of quality. This
implies that the resulting balanced tree is better than the original balanced tree, and hence a contradiction.

This yields Lemma 2.

**Lemma 2 (van Zandt, 2003)** Consider a calculation task on $n$ objects. All optimal calculation trees are balanced trees, i.e. $\delta_i - \delta_j \leq 1$, for all objects $i$ and $j$.

We can also reverse the argument applied in this section. It is straightforward that if one applies a manipulation converse to Manipulation ($\ast$), quality decreases. By repeatedly doing so, one gets a calculation tree with the property $\delta_1 = 1$, $\delta_i = \delta_{i-1} + 1$ for all $i \neq 1, n$, and $\delta_n = \delta_{n-1}$. Call this tree a *serial tree*.

**Corollary 1** Consider a calculation task on $n$ objects. Serial processing yields the lowest quality.

The serial tree and serial subtrees will play an important role in the our analysis of efficient organizations. Two further useful results are stated in Lemmata 3 and 4.

**Lemma 3 (van Zandt, 2003)** In a serial tree $S$ with $n$ leaves, the probability of choosing the best object is

$$\pi(q, n, S) = \frac{1}{n} \frac{q}{1-q} \left(1 + q^{n-2} (1 - 2q)\right).$$

To save on notation we introduce Definition 5.

**Definition 5** $s(x) = \frac{q}{1-q} \left(1 + q^{x-2} (1 - 2q)\right)$. 

Figure 5: Serial tree.
Consider a partial result, produced by serially processing a subset of \( n_p \) (out of \( n \)) items, e.g. the result of the first three operations in Figure 5, \(((7 \otimes 6) \otimes 5) \otimes 4\).

**Lemma 4** (i) The probability that the partial result produced in a serial subtree containing \( n_p \) raw data items is the best object is \((1/n)s(n_p)\).

Moreover, (ii) for \( x > 0 \)

\[
s(n_p + x) = s(n_p) + q^{n_p-1}(s(x + 1) - 1).
\]

**PROOF.** Part (i) follows from Lemma 3. To see (ii), note that in two serial subtrees, one containing \( n_p \) items, and the other one containing \( n_p + x \) items, where \( x > 0 \), the \( n_p - 1 \) items processed last have the same probability of reaching the root in both trees. The item processed first in the smaller tree is the best object with probability \( 1/n \). This object is replaced in the larger tree with a partial result (produced within another serial subtree) which is the best object with probability \((1/n)s(x + 1)\).

Q.E.D.

### 4 Efficient Organizations

We use the following efficiency criterion in our analysis.

**Definition 6** A hierarchy is efficient if no alternative organization exists for processing a given set of \( n \) objects which performs better on one of the dimensions (i) quality, (ii) speed, and (iii) cost, and at least equally well on the other dimensions.

In this section, we show that for a given information processing problem and a given number of managers \( P \), reduced trees maximize quality. Since the reduced tree is also the fastest way to deal with \( n \) objects, this organization is efficient.

The efficiency of the reduced tree can directly be established for an organization with \( \lfloor n/2 \rfloor \) managers. Recall the properties of a reduced tree from Section 2.3. In the first phase of information aggregation, each manager aggregates the raw data items assigned directly to his inbox. A manager’s processing activity in this phase can be depicted by a serial tree. With \( \lfloor n/2 \rfloor \) managers, each of the serial trees has two leaves.\(^\text{12}\) Moreover, the reporting structure induces a balanced calculation tree, where the ”leaves” are the results of the first aggregation phase. Hence, a reduced tree with \( \lfloor n/2 \rfloor \) managers induces a balanced calculation tree.

**Proposition 1** A reduced tree with \( \lfloor n/2 \rfloor \) managers is efficient.

\(^\text{12}\)If \( n \) is odd, one of the trees has three leaves.
Consequently, there is a non-trivial upper bound for the size of efficient hierarchies.

**Corollary 2** No efficient hierarchy employs more than \( \lfloor n/2 \rfloor \) managers.

We now turn to a complete characterization of efficient hierarchies. For this purpose, it is useful to distinguish between an object that has been compared to another one and an object that has not yet been processed. We call the former a *partial result* and the latter a *raw data item*. Let \( n_p \) denote the number of raw data items \( p \) is supposed to handle, with \( \sum_{p=1}^{P} n_p = n \). If fewer than \( \lfloor n/2 \rfloor \) managers are available, there must be more than one manager \( p \) such that \( n_p > 2 \). In a hierarchy with fewer than \( \lfloor n/2 \rfloor \) managers, the balanced calculation tree is in general not implementable. The reason is that limited memory capacity forces each manager to process the information he is supposed to handle serially.

We now derive the quality-maximizing hierarchy design with fewer than \( \lfloor n/2 \rfloor \) managers. As it turns out, reduced trees are optimal. To gain an intuition why this is the case, consider Figure 6, which represents the calculation tree induced by the reduced tree depicted in Figure 2.

![Figure 6: Calculation tree induced by the reduced tree in Figure 2.](image)

The calculation tree contains four serial subtrees of equal length, hence any partial result is the best object with equal probability. We know from the previous section that there is no better way to organize the processing of the partial results than a balanced tree. Moreover, given that the partial results are processed in a balanced tree, i.e. all partial results have the same distance \( m_p \) to the root, a reassignment of raw data items cannot increase quality: Consider a reassignment of one object from one agent to another. This reorganization affects two serial subtrees: Both have the same number \( n_p \) of leaves initially. After the reassignment, one of them has \( n_p - 1 \) leaves, and the other one has...
\( n_p + 1 \) leaves. The effect on \( \pi(\cdot) \) is
\[
\Delta \pi(\cdot) = \frac{1}{n} q^{m_p} (s(n_p - 1) + s(n_p + 1) - 2s(n_p)) = \frac{1}{n} q^{m_p} (1 - 2q) q^{s_p - 1} (1 - q) < 0 \iff q > \frac{1}{2}.
\]

However, the reasoning above does not tell us anything about the reduced tree’s performance relative to hierarchies in which neither every agent processes the same number of raw data items, nor the partial results are aggregated in a balanced tree. Our strategy to exclude such structures is to consider an arbitrary hierarchy \( \mathcal{H} \) and to look for a feasible reorganization of calculation tasks such that quality increases. If we find such a reorganization, hierarchy \( \mathcal{H} \) is not optimal with respect to quality. This yields necessary conditions for optimal hierarchy design. Step by step, we restrict attention to hierarchies in which the necessary conditions are met. We finally show that the set of necessary conditions is sufficient.

Consider a programmed hierarchy \( \mathcal{H} \) with \( P < n/2 \) managers and its calculation tree \( T(\mathcal{H}) \). Let \( n_p \geq 0 \) be the number of raw data items processed by agent \( p \) and let \( r_p \geq 0 \) be the number of reports processed by agent \( p \). If \( p' \) reports to \( p \), we denote by \( r_{p'p} \) the report \( p' \) sends to \( p \). We may restrict attention to the case that \( p' \) processes strictly more than one object, since otherwise \( \mathcal{H} \) could be reprogrammed assigning the object directly to \( p \) and firing \( p' \). We denote by \( \delta(i, T(\mathcal{H})) \) the length of the path from leaf \( i \) to the root in the calculation tree induced by \( \mathcal{H} \), and with \( \delta(r_{p'p}, T(\mathcal{H})) \) the length of the path from report \( r_{p'p} \) to the root. The first two necessary conditions on quality-maximizing hierarchies are provided in Lemmata 5 and 6.

**Lemma 5** In a quality-maximizing hierarchy with \( P < n/2 \) managers, every agent \( p \), for whom \( n_p > 0 \) and \( r_p > 0 \), processes the raw data items before processing a report.

**Proof.** Suppose \( p \) was processing report \( r_{p'p} \) before being finished with the raw data. Then there exists an object \( i \) such that \( \delta(i, T(\mathcal{H})) < \delta(r_{p'p}, T(\mathcal{H})) \). The probability that \( i \) is the best object is \( 1/n \), whereas \( r_{p'p} \) – containing more than one raw data item – is the best object with probability higher than \( 1/n \) and the worst one with probability lower than \( 1/n \). Exchanging the positions of object \( i \) and report \( r_{p'p} \) increases the probability that report \( r_{p'p} \) will be chosen from \( q^{\delta(r_{p'p}, T(\mathcal{H}))} \) to \( q^{\delta(i, T(\mathcal{H}))} \) (if it is the best object), and vice versa for object \( i \). Hence, quality increases. Q.E.D.

A consequence of the lemma above is that for any agent \( p \) with \( n_p > 0 \) the calculation subtree describing this agent’s task contains a serial subtree connecting the \( n_p \) raw data items. The reorganization of the hierarchy proposed in the proof of Lemma 5 and the associated effect on the calculation tree are illustrated in Figure 7.
Lemma 6  In a quality-maximizing hierarchy with $P < n/2$ managers, every agent processes raw data.

**Proof.** Suppose $n_p = 0$ for some agent $p$. Assign $p$’s tasks to one of his direct subordinates and preserve the order of calculations in this subtree. Note that quality is unaffected. Look for the agent $p'$ who processes the largest amount of raw data. If $n_{p'} > 3$, assign two items to $p$ and let $p$ report the result to $p'$. Let $p'$ read the report after finishing his raw data. Note that this yields a manipulation of the calculation tree in the spirit of Manipulation (*) and hence increases quality. If $n_{p'} = 3$, search for another agent $p''$ processing three raw data items. Assign an item from each of these managers to $p$. Denote by $N_{p'p''}$ the set of remaining raw data items processed by $p'$ and $p''$. Identify the agent who processes the object $i$, for which $\delta(i, H) = \min \{\delta(j, H)\}_{j \in N_{p'p''}}$, and let $p$ report to this agent. Let $p$’s report be read immediately after the raw data. Again, the reorganization yields a manipulation of the calculation tree in the spirit of Manipulation (*) and hence increases quality. Q.E.D.

The intuition for the lemma above is straightforward: By involving more processors in the processing of raw data items, serial subtrees can be transformed into "more balanced" ones. We know from the previous section that this enhances quality. The reorganization of the hierarchy is illustrated in Figure 8.

From Lemmata 5 and 6 we know that every quality-maximizing hierarchy induces a calculation tree containing $P$ serial subtrees. It remains to be studied (i) how many leaves ($n_p$) these subtrees optimally have, (i.e. how many raw data items shall be assigned to
each manager), and (ii) how the serial subtrees are optimally arranged, (i.e. the optimal reporting structure). Note that with regard to the arrangement of the $P$ serial subtrees, any tree can be implemented, because we are endowed with $P$ processors, and there are only $P-1$ operations left to perform.

Let $m_p$ denote the distance from $p$’s result of the raw data processing phase to the root.

**Lemma 7** In a quality-maximizing hierarchy, $(m_p + n_p) - (m_{p'} + n_{p'}) \leq 1$ for all $p$ and $p'$.

**Proof.** The probability of choosing the best object is $\pi(.) = \sum_{p=1}^{P} (1/n)q^{m_p}s(n_p)$. Consider the effect of a reassignment of a raw data item from $p$ to $p'$. We have:

\[
\Delta \pi(.) = \frac{1}{n}(q^{m_p}(s(n_p) - 1) - s(n_p)) + q^{m_{p'}}(s(n_{p'}) + 1) - s(n_{p'})
\]

\[
= \frac{1}{n}(q^{m_p}(1-2q)q^{n_p-2} + q^{m_{p'}}(2q-1)q^{n_{p'}}-1)
\]

\[
= \frac{1}{n}(2q-1)(q^{m_{p'}+n_{p'}-1} - q^{m_p+n_p-2}) > 0 \iff m_p + n_p > m_{p'} + n_{p'} + 1.
\]

Hence, we can increase quality using the proposed reorganization as long as there exist $p$ and $p'$ such that $(m_p + n_p) - (m_{p'} + n_{p'}) > 1$. Q.E.D.

According to Lemma 7, the distance from a serial subtree’s deepest leaf to the root of the calculation tree differs by at most one unit among all serial subtrees. The last step is to determine the optimal reporting structure. The optimal assignment of raw data items then follows from Lemma 7.

**Lemma 8** In a quality-maximizing hierarchy, for all $p$ and $p'$, (i) $m_p - m_{p'} \leq 1$, and (ii) $n_p - n_{p'} \leq 2$, where $n_p - n_{p'} = 2$ only if $m_{p'} - m_p = 1$.

**Proof.** (i) Name the agents such that $m_1 = m_2 \geq \ldots \geq m_P$. If $m_p = m_{p'}$, and $n_p < n_{p'}$, name $p$ and $p'$ such that $p$ gets the lower number. Hence, agent 1 processes the lowest
number of raw data items, and his partial result has the longest distance to the root. Agent \(P\)'s partial result has the shortest distance to the root. Note that the serial subtree representing agent 1’s raw data processing activities must be connected directly to another serial subtree. That is, there exists an agent \(p'\) such that either agent 1 or agent \(p'\) is engaged only in raw data processing, and the other agent reads the report immediately after his raw data items.

Now, assume that \(m_p - m_{p'} \leq 1\) does not hold for all \(p\), i.e. assume that there exists an integer \(k > 1\) such that \(m_1 = m_P + k\). We show that in this case the following reorganization enhances quality: Reassign a raw data item from agent \(P\) to agent 1. Let agent 1 report the result of his raw data processing activities to agent \(P\), and let agent \(P\) read this report after finishing his remaining raw data processing activities. Assign agent 1’s remaining tasks (if there are any) to agent \(p'\). Note that the raw data items processed by \(p'\) move up one position in the calculation tree when removing agent 1’s partial result.

The effect of this reorganization on quality is the following:

\[
\Delta \pi(.) = (1/n)\{q^{n_1-1}((1 - q)s(n_{p'}) - qs(n_1)) + q^{n_{p'}+1}(s(n_1 + 1) - 1)\}. \quad (2)
\]

The effect of removing the serial subtree representing agent 1’s raw data processing activities is captured by (2), and the effect of reattaching it is summarized in (3). Note that the serial subtree representing agent 1’s tasks replaces the raw data item processed last by agent \(P\) in the original program. Lemma 7 requires that \(n_{p'} \in \{n_1, n_1 + 1\}\). Let \(D\) assume the value 1 if \(n_{p'} = n_1 + 1\), and 0 otherwise. We have:

\[
\Delta \pi(.) = (1/n)\{q^{n_{p'}+k-1}((1 - q)s(n_1 + D) - qs(n_1)) + q^{n_{p'}+1}(s(n_1 + 1) - 1)\}. \quad (3)
\]

\[
\Delta \pi(.) > 0 \iff q^{k-2}((1 - 2q)s(n_1) + Dq^{n_1-1}(2q - 1)(1 - q)) + qs(n_1) + q - 1 > 0
\]
\[
\iff (1 - q)(s + (1 - 2q)q^{k-3})s(n_1) - 1 + Dq^{n_1+k-3}(2q - 1) > 0
\]
\[
\iff (1 - q)(s(k - 1)s(n_1) - 1) + Dq^{n_1+k-3}(2q - 1)(1 - q) > 0 \quad (4)
\]

The second term in (4) is either zero, or positive if \(q > 1/2\), and negative if \(q < 1/2\). The first term is positive if and only if \(s(k - 1)s(n_1) - 1\) is positive. Note that \(s(x) > 1\) for \(x \geq 1\) if and only if \(q > 1/2\). Hence, \(\Delta \pi(.) > 0\) if and only if \(q > 1/2\).

Part (ii) follows from Lemma 7 and part (i). Q.E.D.

Note that, if \(n_p - n_{p'} = 2\) and \(m_{p'} - m_p = 1\), a reassignment of one raw data item from \(p\) to \(p'\) yields a permutation of \(\delta^T\), but has no effect on quality. That is, a quality-maximizing hierarchy which has the property \(n_p - n_{p'} = 2\), for some \(p\) and \(p'\), coexists
with another quality-maximizing organization with the property $n_p - n_{p'} \leq 1$, for all $p$ and $p'$.

Part (i) of Lemma 8 states that an optimal hierarchy has a reporting structure such that the $P$ partial results of the first processing phase are aggregated in a balanced calculation tree. Part (ii) requires an equal assignment of the raw data items (up to "integer leftovers"). We are in a position to state our main results.

**Proposition 2** A hierarchy with $P$ managers maximizes quality if and only if it has the properties stated in Lemmata 5-8.

**Proof.** Necessity has been established already. To verify sufficiency, note that any feasible reorganization has either no effect on quality or yields a violation of at least one of the necessary conditions stated in Lemmata 5-8. Q.E.D.

**Proposition 3** All reduced trees are efficient.

**Proof.** Reduced trees have the properties described in Lemmata 5-8, hence they maximize quality given $P$. We know from Radner (1993) that a reduced tree achieves the minimum delay for a given number of processors. Q.E.D.

As a reduced tree maximizes the speed of the decision procedure given the number of managers, as well as the quality of the decision, no slower working hierarchy with the same number of managers can be efficient.
Corollary 3 A hierarchy with $P$ managers processing $n$ objects is efficient only if it achieves minimum delay. Any outcome of an efficient organization can be achieved by a reduced tree.

5 Conclusion

We studied a problem of efficient decentralized information aggregation in a setup with information processing imperfections. Our results indicate that a hierarchy designer does not face a trade-off between the speed and the quality of information aggregation. The reduced tree proposed by Radner (1993) is a hierarchy in which information processing takes minimum time and delivers maximum quality for a given number of processors and objects. Moreover, we find that highest quality can be obtained by the reduced tree with $\lfloor n/2 \rfloor$ managers (which is again the fastest of its class).

There are several useful extensions of our framework. First, it would be desirable to consider more general measures of decision quality. This would require the specification of a von Neumann-Morgenstern utility function, as well as assumptions on the distribution of quality. Our results hold regardless of the form the utility function takes or how quality is distributed.

As a referee pointed out to us, processing imperfections may have an impact on the properties of the efficient set of hierarchies only if the associative law of binary operations does not hold anymore. For combinations of calculation tasks and processing imperfections other than the ones considered in our paper, it may still hold. It is worth studying which combinations yield a violation of the associative law and whether alternative specifications would affect our efficiency results.

We introduced an information processing imperfection into the analysis of decentralized information aggregation by restricting the calculation ability of agents in an intuitive, but rather simplifying manner. In our setup, agents’ mistakes do not depend on the quality of the two compared items. Intuitively, mistake making should depend on the task to be performed. To incorporate this consideration into our model, one could make use of probabilistic choice models, such as Luce (1959). Again, this modification would require the specification of the quality distribution and of the utility function.

One may also allow agents to influence the individual probability of making a mistake through effort. This issue is addressed in Grünert and Schulte (2004), where a game-theoretical approach is taken to study the incentives for effort provision in carrying out a decentralized information processing task.
Finally, we restricted attention to hierarchical organizations. In such an organization, calculations cannot be repeated and agents make one final report to their superior. If information processing is imperfect, one would like to repeat calculations to increase quality. However, the hierarchical structure precludes the possibility of sending an object to multiple processors. Whether or not a more general structure is feasible depends on the agents’ information-storing abilities and on the information structure. For instance, if the objects cannot be copied, and only one agent can work on an object at any time (e.g. job candidates who need to be interviewed), then it is impossible to send objects to multiple processors and hierarchies are the only feasible organization in a setting with limited memory capacity. In our notion of a hierarchy, we allowed for only one upward link for each manager. However, one might want to allow a manager to send more than one report to his superior. With respect to the speed of information processing, it is optimal to implement a hierarchical structure as defined in this paper. In terms of quality, it may make sense to allow for multiple reports to get closer to the balanced calculation tree. Concerning these modifications, one would have to specify a different information processing technology. Both modifications – multiple reports and repeated calculations – would involve a trade-off between speed and quality that does not play a role in our framework.

References


