# INFORMATION AND COMPUTATION

J. F. Traub
H. Wozniakowski

CUCS-64-83

# Information and Computation

J. F. TRAUB

*Department of Computer Science*
*Columbia University*
*New York, New York*

H. WOŹNIAKOWSKI

*Institute of Informatics*
*University of Warsaw*
*Warsaw, Poland*
*and*
*Department of Computer Science*
*Columbia University*
*New York, New York*

ISBN 0-12-012123-9

## 1. Overview and Summary

Science has been called the study of invariants, seeking laws which are valid in varied domains.

An archetypal example is provided by Newtonian mechanics. Before Newton, any "reasonable" person believed that apples and planets were very different objects obeying different laws. For some characteristics this is true, as can be verified by biting into an apple and a planet. But if the correct quantities, which are force, mass, and acceleration, are considered, then there are laws which apply equally to apples and planets as well as to carriages, waterwheels, and buildings.

For most computational problems we have only partial or approximate information and consequently such problems can be solved only with uncertainty in the answer. Examples of such problems include computations arising in science and engineering, decision theory, prediction, estimation, computer science, mathematics, design of experiments, remote sensing, and signal processing. Such problems are as different from each other as apples and planets. For such seemingly unrelated problems we believe that we have identified the basic quantities and a fundamental invariant, which we call the radius of information.

The radius of information measures the intrinsic uncertainty in the solution of a problem due to the available information. In this article we shall see some examples of the many and varied domains which can be unified by using the concept of radius of information. As we shall see, the radius of information is fundamental; it provides limits on how well a problem can be solved and leads to very widely applicable notions of optimal information and optimal algorithms.

To emphasize this, we state the **Information principle:** There exists a quantity called the radius of information which measures the intrinsic uncertainty of solving a problem if certain information is known.

Our work is based on two theses: (1) most problems are approximately solved; that is, we live with uncertainty; (2) for problems with partial or approximate information, the usual algorithm-centered approach can be supplemented, and sometimes replaced, by the information-centered approach.

We briefly discuss these two theses here. Much of this article will be devoted to their expansion, illustrated by numerous examples.

We begin with the first thesis. It is very common for the information concerning a problem to be partial or approximate. Then the problem can only be solved with uncertainty. If, on the other hand, the information is complete and exact, then a solution without uncertainty may be possible. Even in this case the cost of computing an exact solution may cause us to settle for an approximate solution because we are willing to give up certainty to reduce complexity. Finally, many problems will be solved in a distributed environment and this may cause uncertainty in the answer.

We briefly discuss this last point. There are two reasons for using a distributed system. The first is that although a centralized system could be used, we select a distributed system for the sake of, say, efficiency. The second reason is that the problem is inherently distributed; examples include resource allocation in a decentralized economy, traffic networks, and reservation systems. Because the information flow in a distributed environment may have to be limited, this causes uncertainty in the solution. Even problems capable of exact solution on a uniprocessor will be solved only under uncertainty in the distributed environments of the future because complete, exact information on the current state of the distributed system will not be available.

We proceed to the second thesis. Currently, the algorithm-centered approach is in widespread use. In this approach, an algorithm is obtained, often on the basis of ad hoc criteria. This algorithm is then analyzed. Another algorithm is then proposed and analyzed, and so on.

We contrast this with the information-centered approach. In this approach, one merely states how well a problem should be solved and indicates the *type* of information available. The theory then discloses an optimal information and an optimal algorithm, and yields the bounds on the problem complexity, that is, how much it must cost to solve.

A simple example may illustrate the drawbacks of the algorithm-centered approach. Consider the Gauss algorithm for approximate integration. (We use this numerical example because it is widely known. However, numerical calculations are just one particular application of the general theory.) The algorithm is based on the ad hoc criterion that it exactly integrates polynomials of maximal degree. This is the characteristic criterion. There are additional criteria (see Section 5). There is no a priori reason why the Gauss algorithm should be good for nonpolynomial integrands. Indeed, as we shall show in Section 5.2, even for analytic integrands we can pay an exponential penalty for using Gauss information rather than the optimal information.

Another drawback of the algorithm-centered approach is that it does not give lower bounds on problem complexity. The information-centered approach yields both lower and upper bounds; these are often very tight.

What is the place of the algorithm-centered approach? The information-centered theory provides general notions of optimal information, optimal algorithm, and problem complexity. For a particular problem it may be technically very difficult to obtain these and, to do the job, an algorithm must be created and analyzed. This is particularly the case for complicated "real-world" models. With time, we expect the technical problems to be overcome for harder and harder problems.

We emphasize that although our model *formally* includes the case where the information is complete and exact, it generally does not yield interesting new results for that case. Complete and exact information is typical of many discrete problems such as the traveling-salesman problem. The radius of information is then zero and the problem can be exactly solved. (See, however, Section 3, where we discuss why even problems with complete and exact information are sometimes not solved

exactly.) Lower bounds on problem complexity must then be obtained by entirely different techniques than the one we will describe. The creation of good algorithms can depend critically on the particular problem being considered and is often very difficult. *We emphasize that for the most part our discussion and conclusions apply only to problems which are not exactly solved.*

An analogy from physics may be helpful in understanding what we believe should be the role of the information-centered approach. An architect designing a building must know the general laws of mechanics. In addition, he must take into account many particulars of this building, such as its site, the relation of the projected building to existing neighboring buildings, and the special needs and desires of his clients. Furthermore, the same laws of mechanics are used whether the design is a plan for a skyscraper, a bridge, or an auditorium.

We believe that the information-centered approach provides the algorithm designer with general concepts and laws analogous to those provided the architect or the civil engineer by the laws of mechanics.

We stress that actually obtaining the radius of information, an optimal information, and an optimal algorithm for a particular problem may or may not be hard. This is common in science. Although the laws of mechanics are quite simple, applying them may be difficult. (See Section 6.4.1 for further discussion.)

We give a somewhat idealized description of what we believe may be the new role of the algorithm designer. He decides which of the information-centered settings (worst case, average case, asymptotic, average asymptotic) are relevant to his problem. Depending on anticipated algorithm implementation (VLSI, program) and/or the computer architecture on which the algorithm will run (uniprocessor, vector computer, a non-von Neumann architecture, systolic array), he chooses a model of computation. He applies general results to guide him on his selection of the information his algorithm will use and on the selection of his algorithm. He may have to perform difficult analysis if his problem has not been previously investigated. He estimates the problem complexity. For numerical algorithms he also concerns himself with algorithm stability, which, at least today, is dependent on detailed analysis.

This article is an exposition of the information-centered approach to problems that are solved with uncertainty. We are calling the theory and application of the information-centered approach *$\varepsilon$-complexity*. (See Section 10 for a discussion of the history and nature of $\varepsilon$-complexity.)

To make this account widely accessible, we defer the abstract theory to Section 6. Even when we finally turn to the abstract formulations and some of the general results, we give them as simply as possible.

Readers seeking to know more are referred to research papers and especially to two research monographs: Traub and Woźniakowski (1980a) and Traub *et al.* (1983). A third research monograph, reporting on average-case models and probabilistic settings, is projected.

We briefly summarize this article. In Section 2 we introduce the fundamental ideas through the elementary example of integration. We are not particularly interested in integration, but have chosen it because readers of diverse backgrounds will find it familiar and because most of the basic issues arise even in this example.

In Section 3 we discuss our first thesis and examine the causes of uncertainty. Problems *cannot* be solved exactly because the information is partial or approximate or because the class of algorithms is restricted. On the other hand, we sometimes *choose* to live with uncertainty to lower the complexity. We provide four examples where this choice is made: heuristics in artificial intelligence, probabilistic algorithms, approximate solutions of hard problems, and iterative solutions of large linear systems.

We examine when nonadaptive information is just as powerful as adaptive information in Section 4. Nonadaptive information provides a natural decomposition of a problem for solution on a parallel or distributed computer. Zero finding, integration, and binary search are used as examples.

In Section 5 we return to our second thesis and discuss the limitations of the ad hoc methods of the algorithm-centered approach.

An abstract formulation of the worst-case model with uncertainty measured by a norm is given in Section 6. The treatment parallels the example of Section 2.

In Section 7 we discuss some mathematical applications. We confine ourselves to two kinds of applications. In Sections 7.1–7.3 we show three ways in which we can cut the search space for optimal information and optimal algorithms. In Section 7.4 we discuss whether smoother problems have lower complexity. We discuss a result regarding information requirements of mathematical economics in Section 7.5.

So far we have confined ourselves to the worst-case normed model. Additional models are indicated in Section 8. We discuss a model where uncertainty is measured without a norm as well as average-case and asymptotic models.

We have received many comments and questions concerning the information-centered approach. In Section 9 we present some of these, together with our responses.

In Section 10 we conclude by discussing whether ε-complexity is a new discipline, give a brief history, and indicate some of the directions for future research.

## 2. Fundamentals

We introduce the fundamental ideas through the elementary example of integration. We stress that we are not particularly interested in integration per se but that it provides a common ground with which all our readers are familiar. A general formulation is given in Section 6. When possible, the same notation is used in Sections 2 and 6.

### 2.1 Problem Formulation

We wish to compute $\int_0^1 f(t)\,dt$. Because "few" integrands can be integrated exactly, we have to settle for an approximate answer. We want to compute a number $x$ such that

$$\left|x - \int_0^1 f(t)\,dt\right| < \varepsilon$$

for some preassigned positive $\varepsilon$. If $\varepsilon = 0$, we want to compute a number $x$ such that $x = \int_0^1 f(t)\,dt$. (This distinction between $\varepsilon > 0$ and $\varepsilon = 0$ is made for technical reasons.) We say $x$ is an *ε-approximation.*

We must know something about the integrand to compute an ε-approximation. We will assume that we sample $f$ at $n$ given points $t_1, t_2, \ldots, t_n$. Thus we know the vector $[f(t_1), f(t_2), \ldots, f(t_n)]$. We denote this vector by $N(f)$ and call $N(f)$ the *information.*

It is easy to show that if the number of sample points is fixed, then we cannot guarantee that an ε-approximation is determined. To see this define

$$g(t) = f(t) + c\prod_{i=1}^{n}(t - t_i)^2.$$

Then $N(g) = N(f)$ and the two integrands are indistinguishable under the information $N(f)$. By choosing $c$ sufficiently large, $\int g$ and $\int f$ can be made to differ by an arbitrary amount. Hence we cannot guarantee that an ε-approximation is determined.

To guarantee that an ε-approximation is determined we must restrict the class of integrands. To fix ideas, assume $f$ is any function whose first derivative is bounded. Without loss of generality we can assume $|f'(t)| \leq 1$ on $[0, 1]$, because if the bound were $L$ we could scale our error results by $L$. Call the set of all such functions $F$. [This definition of $F$ serves our present purpose; for a precise definition see Traub and Woźniakowski (1980b, pp. 90, 109).]

Thus we can formulate our problem as follows. Given information $N$, compute an ε-approximation to $\int_0^1 f(t)\,dt$ for all $f \in F$.

In Section 1 we stated that in the information-centered approach "one merely states how well a problem should be solved and indicates the *type* of information available." We can now be more specific about what we mean, in terms of this example. The problem is specified as approximating $\int_0^1 f(t)\,dt$ for all $f \in F$. How well the problem should be solved is specified by the condition

$$\left|x - \int_0^1 f(t)\,dt\right| < \varepsilon \qquad \text{for all} \quad f \in F.$$

The type of information is

$$N(f) = [f(t_1), f(t_2), \ldots, f(t_n)].$$

### 2.1.1 *Why We Must Indicate the Type of Information*

Recall that it was stated in Section 1 that in the information-centered approach the *type* of information available must be indicated. We now amplify this remark.

The type of information appropriate for the ε-approximation of an integral might be the values of an integrand or its derivatives at a number of points. In the example of Section 2.1 we assumed for simplicity that the type of information was integrand values. The problem would become trivial if we permitted $\int_0^1 f(t)\,dt$ to be information.

There are, however, problems for which it is reasonable to permit the value of an integral as information. An example is provided by the problem of approximating a zero of a nonlinear function $f$. For that problem, Kacewicz (1976a,b, 1979) has shown that information consisting of $f, f'$, and an integral of $f$ is useful.

## 2.2 Radius of Information

As we observed in Section 1, there exists a quantity called the radius of information which measures the intrinsic uncertainty of solving a problem when certain information is available. We use the integration example to provide the reader with an intuitive feel for this quantity.

Let $V$ be the set of integrands, $\tilde{f}$, which has two properties: (1) $N(\tilde{f}) = N(f)$; thus $\tilde{f}$ is indistinguishable from $f$ under the information $N$; (2) $\tilde{f} \in F$; thus $|\tilde{f}'| \le 1$ on $[0, 1]$.

There is, in general, an infinite number of integrands in $V$. The integrals of these functions are, of course, numbers, and it is easy to show that they form a finite interval. Because all the $\tilde{f}$ are indistinguishable from $f$, we do not know which point in this interval is the exact answer. The uncertainty is minimized by choosing the answer as the interval midpoint. For this example, the radius of information is the distance from the interval midpoint to an end point. The formula for the radius of information for a particular choice of $N$ is provided in Section 2.7.

From this discussion it should be clear that the radius of information has the following important property. The information determines an ε-approximation if and only if the radius of information is smaller than ε. We denote the radius of information by $r(N)$. We have

**Theorem 2.1** The information $N$ is strong enough to determine an ε-approximation for all $f \in F$ iff $r(N) < \varepsilon$. ∎

This result holds very generally and not just for our integration example (see Traub and Woźniakowski, 1980a, Chapter 1, Sect. 2; Traub *et al.*, 1983, Chapter 1, Sect. 3).

Note that the existence of an ε-approximation depends only on the information and is independent of any notion of algorithm. This illustrates our information point of view. (See Section 5 for further discussion of the contrast between the information and algorithm points of view.)

## 2.3 Algorithms

An *idealized algorithm* (or simply, algorithm) is any rule for computing an approximation knowing the information

$$N(f) = [f(t_1), f(t_2), \ldots, f(t_n)],$$

and knowing that $f \in F$. To indicate the dependence on information, we write an idealized algorithm as $\varphi(N(f))$. Examples of algorithms are

$$\varphi(N(f)) = \sum_{i=1}^{n} a_i f(t_i), \qquad \varphi(N(f)) = \frac{\sum_{i=1}^{n} b_i f(t_i)}{\sum_{i=1}^{n} c_i f(t_i)}. \tag{2.1}$$

Our definition of an idealized algorithm is far more general than the notions of algorithms prevalent in computer science. Motivation for this generality and its relation to notions common in computer science are discussed in Section 3.1.3.

## 2.4 Optimal Algorithms

Which algorithms are best? Indeed, what do we mean by best algorithm? We will introduce two notions of best algorithm in this article; one of these will be introduced in this section.

The error of approximating $\int_0^1 f(t)\,dt$ using the algorithm $\varphi$ is defined as

$$e(\varphi, f) = \left|\varphi(N(f)) - \int_0^1 f(t)\,dt\right|.$$

The *algorithm error*, $e(\varphi)$, is the worst $e(\varphi, f)$ for all $f \in F$.

The radius of information is a lower bound on the algorithm error. We have

**Theorem 2.2** For any algorithm $\varphi$ which uses the information $N(f)$

$$e(\varphi) \geq r(N).$$

Furthermore, this lower bound is the best possible. That is, there cannot be a lower bound which is larger. ■

Because we want to make the algorithm error as small as possible, we are interested in algorithms whose error equals that of the radius of information. We say $\varphi$ is an *optimal error algorithm* (or simply, an optimal algorithm) if

$$e(\varphi) = r(N).$$

We denote an optimal algorithm by $\varphi^*$.

We shall discuss in Section 2.4.1 how to obtain optimal or nearly optimal algorithms. A second notion of optimality (optimal complexity algorithm) is defined in Section 2.11.

### *2.4.1 How to Generate Good Algorithms*

We define two paradigms for generating optimal or near-optimal algorithms.

Recall that we defined in Section 2.2 the set, $V$, of integrands which belong to $F$ and which are indistinguishable under the information $N$. The integrals of functions in $V$ form a certain interval. An *interpolatory algorithm*, $\varphi^I$, chooses any point in this interval as an approximation to $\int_0^1 f(t)\,dt$. A *central algorithm*, $\varphi^c$, chooses the midpoint. We have

**Theorem 2.3**

$$e(\varphi^I) \leq 2r(N),$$

$$e(\varphi^c) = r(N). \quad ■$$

Because $r(N)$ is a sharp lower bound on the error of any algorithm, interpolatory and central algorithms provide tight upper bounds.

To create an interpolatory algorithm, choose a "simple" function $\tilde{f}$ from $F$ which is indistinguishable from $f$ under $N$. Then the interpolatory algorithm is the integral of $\tilde{f}$. For example, $\tilde{f}$ can be chosen as a polynomial or a piecewise polynomial which interpolates $f$ at the $n$ points $(t_1, f(t_1)), \ldots, (t_n, f(t_n))$. Then $\int_0^1 \tilde{f}(t)\,dt$ is an interpolatory algorithm.

Central algorithms are in general more difficult to generate. The algorithm discussed in the example of Section 2.7 is a central algorithm.

## 2.5 Linear Algorithms

We defined an algorithm as any rule for computing an approximation knowing the information $N$. The simplest rule for combining the integrand values is

$$\varphi(N(f)) = \sum_{i=1}^{n} a_i f(t_i)$$

where the $a_i$ are constants. We call $\varphi$ a *linear algorithm*.

Researchers often restrict themselves to considerations of linear algorithms. Because they *assume* a linear algorithm, they cannot rule out the existence of a much better nonlinear algorithm. Without assuming anything about the structure of the algorithm we will often be able to *conclude* the existence of a linear optimal algorithm. In particular, this is true for our integration example.

## 2.6 Optimal Information

We have assumed that the sample points $t_1, t_2, \ldots, t_n$ are fixed. We now want to consider the best choice of the sample points.

We hold the number of sample points $n$ fixed. (The number of sample points is varied in Section 2.8.) Because the radius of information is a sharp lower bound on the error of any algorithm, we say $N$ is *optimal information* if the sample points are chosen to minimize $r(N)$. We denote optimal information by $N^*$.

## 2.7 Example

We illustrate the concepts of Section 2.6. Recall that the information $N(f)$ consists of $n$ function samples and that the class of integrands consists of functions whose derivative is bounded by one on the unit interval.

Optimal information consists of sampling $f$ at the points $t_i = (2i - 1)/2n$, $i = 1, \ldots, n$. Thus

$$N^*(f) = [f(1/2n), f(3/2n), \ldots, f(1 - 1/2n)].$$

Note that if the unit interval is imagined to be transformed into a circle with $t = 0$ coincident with $t = 1$, then the sample points are equally spaced.

For the optimal information $N^*$ the radius of information is given by

$$r(N^*) = 1/4n. \tag{2.2}$$

There exists an optimal algorithm which uses optimal information, and this optimal algorithm is linear. Its formula is given by

$$\varphi^*(N^*(f)) = \frac{1}{n}\sum_{i=1}^{n} f\left(\frac{2i-1}{2n}\right). \tag{2.3}$$

Thus the optimal algorithm is just a Riemann sum! Indeed, it is simply the average of integrand values at equidistant points. In other words, the continuous average of $f$, which is of course $\int_0^1 f(t)\,dt$, is best approximated by the discrete average of $f$ at properly chosen points. We call this the *averaging algorithm*.

## 2.8 Is the Information Strong Enough?

So far we have fixed $n$, the number of sample points. Recall that we want to compute an $\varepsilon$-approximation for all $f \in F$. It may turn out, even using optimal information, that $n$ is not large enough.

We therefore vary $n$ and ask for the smallest $n$ such that the information is strong enough to compute an $\varepsilon$-approximation. Recall that, in general, we can compute an $\varepsilon$-approximation for all $f \in F$ iff $r(N) < \varepsilon$. Furthermore, for optimal information $N^*$, $r(N^*) = 1/4n$. It follows that if $m$ denotes the smallest number of sample points which can determine an $\varepsilon$-approximation, then $m = \lfloor 1/4\varepsilon \rfloor + 1$.

To fix ideas, if $\varepsilon = 10^{-8}$, we must sample the integrand at 25,000,001 points. No smaller number of samples will do. If the class of integrands is smoother (we assume the functions in $F$ have only one derivative), then fewer samples are required (see Section 7.4).

We require that the error be less than $\varepsilon$ for any $f \in F$. This is a worst-case criterion. Intuition might suggest that, on the average, substantially fewer samples are required. This intuition is often incorrect, as we shall see in Section 8.2.

## 2.9 Computational Complexity

Until now our concern has been with whether an $\varepsilon$-approximation can be computed. If the answer is affirmative, we want to know how much it must cost; that is, what is the computational complexity?

We shall restrict ourselves to time costs. Assume, for simplicity, that each arithmetic operation costs unity and that each function evaluation costs $c$. We define the $\varepsilon$-complexity to be the smallest cost of computing an $\varepsilon$-approximation by *any* algorithm. We denote the $\varepsilon$-complexity as comp($\varepsilon$). We sometimes refer to $\varepsilon$-complexity as problem complexity or computational complexity.

Observe that the $\varepsilon$-complexity is the smallest cost for solving the problem. This concept should not be confused with the *algorithm complexity*, which is the cost of a particular algorithm. The phrases *algorithm cost* and *algorithm complexity* are used interchangeably. Obtaining the $\varepsilon$-complexity is very difficult and we almost always have to be content with upper and lower bounds. The upper bound is obtained by exhibiting an algorithm whose cost then gives the upper bound. The lower bound is established by a theorem which states that there cannot exist an algorithm with lower cost. We shall see that under certain conditions the gap between the lower and upper bounds is small (see Sections 2.10 and 6.11).

## 2.10 Example

We obtain upper and lower bounds on the $\varepsilon$-complexity of our integration example. We begin with the upper bound. Recall that our optimal algorithm is the averaging algorithm

$$\varphi^*(N^*(f)) = \frac{1}{n}\sum_{i=1}^{n} f\left(\frac{2i-1}{2n}\right).$$

We saw in Section 2.8 that the minimal number of function samples $m$ to compute an $\varepsilon$-approximation is

$$m = \lfloor 1/4\varepsilon \rfloor + 1. \tag{2.4}$$

The cost of evaluating $f$ at $m$ points is $mc$. In addition, $m$ arithmetic operations are sufficient to combine the samples. Thus the cost of $\varphi^*$ is $mc + m$, and using Eq. (2.4),

$$\text{cost}(\varphi^*) = (\lfloor 1/4\varepsilon \rfloor + 1)(c + 1). \tag{2.5}$$

That gives us an upper bound on comp($\varepsilon$).

We turn to the lower bound. Because $m$ samples are required to compute an $\varepsilon$-approximation, the cost of evaluating $f$ must be at least $mc$. In addition, at least $m - 1$ arithmetic operations are needed to combine the samples. Hence a lower bound is given by $mc + m - 1$, and hence

$$\text{comp}(\varepsilon) \geq (\lfloor 1/4\varepsilon \rfloor + 1)(c + 1) - 1. \tag{2.6}$$

From Eqs. (2.5) and (2.6)

$$(\lfloor 1/4\varepsilon\rfloor + 1)(c + 1) \geq \mathrm{comp}(\varepsilon) \geq (\lfloor 1/4\varepsilon\rfloor + 1)(c + 1) - 1. \quad (2.7)$$

Note that the gap between the upper and lower bounds is very small. Hence we essentially know comp($\varepsilon$). The tight bounds are typical of "linear optimal algorithms." (See Section 6.11 for further discussion.)

### 2.11 Optimal Complexity Algorithms

We introduce our second notion of optimal algorithm. In Section 2.4 we introduced the notion of optimal error algorithm. We define an *optimal complexity algorithm* as an algorithm whose cost is the least among all algorithms for computing an $\varepsilon$-approximation.

For example, it follows from Eqs. (2.5) and (2.6) that the averaging algorithm,

$$\varphi^*(N^*(f)) = \frac{1}{n}\sum_{i=1}^{n} f\left(\frac{2i-1}{2n}\right),$$

is within, at most, one unit of being an optimal complexity algorithm.

Recall that this algorithm is also an optimal error algorithm. This connection between optimal error algorithms and optimal complexity algorithms is often the case. (See Section 6.11 for further discussion.)

## 3. Why Are Most Problems Solved with Uncertainty?

In the previous section we introduced the fundamental ideas of $\varepsilon$-complexity through the elementary example of integration. In contrast, this section is devoted to a rather general examination of the causes of uncertainty.

We solve problems with uncertainty because we *cannot* solve exactly or we *choose* not to solve exactly. We will discuss three reasons why we *cannot* solve exactly: the information is *partial*, the information is *approximate*, or the class of algorithms is *restricted*.

As we observed in Section 1, even problems capable of exact solution on a uniprocessor will be solved only under uncertainty in the distributed environments of the future because complete, exact information on the current state of the distributed system will not be available.

We will give four examples of *choosing* not to solve exactly: heuristics, approximate solution of hard problems, probabilistic algorithms, and iterative solution of large linear systems. To date we have used $\varepsilon$-complexity to contribute only to the last of these. The first three are included here to

indicate the variety of areas where we choose not to solve exactly. We are hopeful that $\varepsilon$-complexity will prove useful in some of these areas.

### 3.1 Why We Cannot Solve Exactly

We have given three general reasons why problems cannot be solved exactly. There are other causes of uncertainty. For example, we might not be able to solve a problem arising in nature because we cannot give a mathematical formulation. In the general abstract formulation to be presented in Section 6, however, only the causes of uncertainty arising from partial and approximate information or from a restricted class of algorithms are considered. We now discuss each of these causes.

#### *3.1.1 Partial Information*

Recall that in the integration example we consider integrands $f \in F$, where $F$ is the set of integrands for which $|f'(t)| \leq 1$ on the unit interval. We are given the information $N(f) = [f(t_1), f(t_2), \ldots, f(t_n)]$. In general, there are many other functions belonging to $F$ which have the same values as $f$ at the sample points. All of these functions are indistinguishable using the preceding information. Let $V$ be the set of all functions which belong to $F$ and which are indistinguishable knowing $N(f)$. Of course, $f \in V$.

Let $f_1, f_2 \in V, f_1 \neq f_2$. Then $\int_0^1 f_1(t)\,dt$ can differ substantially from $\int_0^1 f_2(t)\,dt$. Thus we cannot guarantee that an $\varepsilon$-approximation is determined for both $f_1$ and $f_2$, unless $r(N) < \varepsilon$. Because we cannot distinguish among the functions in $V$, we cannot guarantee an $\varepsilon$-approximation for $f$ because $f$ could be either $f_1$ or $f_2$.

*It is crucial to understand that the algorithm does not use $f$. It uses only $N(f)$ and the fact that $f$ belongs to $F$.*

We comment on this point. In engineering and natural science we typically do not know $f$. What we might know are some measurements of $f$, and these measurements have experimental error. In mathematical science we sometimes know $f$. Thus we may want to compute $\int_0^1 f(t)\,dt$ where $f(t)$ is a known function. However, the algorithm used to compute an $\varepsilon$-approximation does not use $f(t)$; it uses a finite number of evaluations of $f$.

We say the information is *partial* if knowing $N(f)$ and $f \in F$ does not determine $f$ uniquely. Partial information causes uncertainty. (An abstract definition of partial information is given in Section 6.)

If information is not partial, it is *complete*. An example of complete information may be found in Section 3.1.3.

### 3.1.2 *Approximate Information*

Information is approximate for many reasons; some of these are listed below. We begin with how approximate information might occur in our integration example.

So far, we have assumed that the sample values were exact. In practice, the sample values are often approximate for a number of reasons: they are computed with uncertainty; even if they are exact, rounding errors occur when they are entered into the computer; or they are obtained by experiments with error. More generally, information is approximate for many reasons. These include the stochastic nature of information, computer errors, transmission errors, limitations of number representation and arithmetic, adversary's lies, limitations on measurement accuracy due to instrument limits, and intrinsic measurement limitations due to Heisenberg uncertainty.

We comment on just two of these. The case of stochastic information is of great importance in many applications. It is not included in the models we have studied so far, but will be incorporated in future models.

As an example of adversary's lies we consider binary search (which is popularly called "20 Questions"). Is there a good strategy for playing 20 Questions with a liar? More precisely, what questions should you ask if you know $k$ of your adversary's answers will be lies. (Of course you do not know which answers are lies.) Rivest *et al.* (1980) show that if $k$ is not too large relative to the total number of questions, then there is a sequence of questions such that the complexity is not much greater than for the case of no lies. An $\varepsilon$-complexity formulation is given by Traub *et al.* (1983) for both the discrete and continuous versions of this problem.

Traub *et al.* (1983, Chapter 2) introduce the concept of *approximate information*, $N_\rho(f)$, where $\rho$ is a measure of error, and the notion of approximate radius of information $r(N_\rho)$. As the generalization of Theorem 2.1, we have the following theorem.

**Theorem 3.1** The information $N_\rho$ is strong enough to determine an $\varepsilon$-approximation iff $r(N_\rho) < \varepsilon$. ■

### 3.1.3 *Realizable Algorithms*

Even if the information is complete and exact we may not be able to compute an exact answer because we restrict what we mean by an algorithm.

Recall that an idealized algorithm is any rule using the information $N(f)$. One of the reasons for this extremely general notion of algorithm is the following: if we want to show that an $\varepsilon$-approximation cannot be

computed (because the information is not strong enough), then it is desirable to establish this with the most general notion of algorithm. If, on the other hand, the information is strong enough to determine an $\varepsilon$-approximation and we want to compute an $\varepsilon$-approximation, we are faced with the fact that we may not be able to implement an idealized algorithm.

We may therefore restrict the algorithms under consideration to a class of realizable algorithms. What we define to be a realizable algorithm is up to us. We emphasize that restricting the notion of algorithm can only increase uncertainty.

We illustrate the idea of a realizable algorithm by a simple and practical example. Let $F$ be the set of nonnegative real numbers and let $N(f) = f$. We wish to compute an $\varepsilon$-approximation to $\sqrt{f}$.

The information is certainly complete because we know $f$. It is also exact. If we permit any algorithm, then we take $\varphi(N(f)) = \varphi(f) = \sqrt{f}$. There is no uncertainty ($\varepsilon = 0$). This is as we would expect. If the information is complete and exact and if idealized algorithms are allowed, the answer can be computed exactly.

We now restrict our notion of algorithm. For this problem, a *realizable* algorithm is any rule which uses $N(f)$ and a finite number of arithmetic operations ($+, -, \times, \div$) and comparisons. Now there is uncertainty in the answer.

The computation of square roots is a special case of computing an $\varepsilon$-approximation to a polynomial zero. There has been much recent progress on computing $\varepsilon$-approximations with realizable algorithms (see Kuhn *et al.*, 1983; Murota, 1982; Schönhage, 1982; Shub and Smale, 1982a,b; Smale, 1981).

What we elect to call a realizable algorithm is up to us. Examples of realizable algorithms include Turing-machine algorithms, algorithms that are computable functions, on-line algorithms, algorithms that are linear functions of the input, and stable algorithms.

We discuss the relation between the concepts of algorithm used elsewhere in computer science and our notion of realizable algorithm. Recall that an idealized algorithm is an arbitrary rule for computing an approximation knowing certain information. Clearly, if an $\varepsilon$-approximation cannot be computed using an idealized algorithm, it cannot be computed by an algorithm in any formal system. Only if an $\varepsilon$-approximation can be computed by an idealized algorithm does it become of interest whether an $\varepsilon$-approximation can be computed in a formal system, say a Turing-machine model. Of course, for many computer science problems there is an exact solution and the issues of decidability and complexity in a formal system become paramount. [See Traub *et al.* (1983, Chapters 3 and 5) for further discussion of restricted classes of algorithms.]

## 3.2 Why We Choose Not to Solve Exactly

We *choose* not to solve problems exactly because it is significantly cheaper to solve approximately and we are content with an approximate solution. Hence we live with uncertainty to lower the complexity. We will illustrate this with four examples.

### 3.2.1 Heuristics

What is the difference between an algorithm and a heuristic? A thorough discussion of heuristics would carry us too far afield. Roughly speaking, the distinction is that an algorithm guarantees a correct answer whereas a heuristic is a rule of thumb; a correct answer is not guaranteed. [An informal discussion of algorithms and heuristics may be found in Traub (1978).]

Heuristics are used for a number of reasons: we could solve algorithmically, but it is too expensive; no algorithm is known; and/or the goal is not well defined.

We discuss only the first of these here, using chess as an illustration. The problem is to find a winning strategy for white (if it exists) against all possible strategies of black. The set $F$ consists of the rules of chess and the initial position. The information is complete and exact; there is therefore an idealized algorithm. Indeed, we have the following gedanken algorithm. Generate the tree of all possible moves. If there exist one or more winning strategies against all moves by black, choose one of these strategies. This is an algorithm which guarantees a win. If no such strategy exists, no algorithm for winning exists.

Such a "brute-force" approach would be far too expensive (McCorduck, 1979). We live with the uncertainty of the heuristics to decrease complexity.

### 3.2.2 Approximate Solution of Hard Problems

Consider problems which we could, in principle, solve exactly ($\varepsilon = 0$), but for which the algorithm complexity of all known algorithms is so high that we cannot solve the problem exactly on even the fastest computers. Hence we use an algorithm of low complexity and solve the problem approximately ($\varepsilon > 0$).

We illustrate the idea with a well-known instance of combinatorial optimization, bin packing. Let $f = [f_1, f_2, \ldots, f_n]$ be a given sequence of positive numbers on the unit interval. Let $\mathrm{BIN}_1, \mathrm{BIN}_2, \ldots$ be a sequence of bins, each of unit capacity. In bin packing, we assign each $f_i$ to a bin in

such a way that the sum of numbers in each bin does not exceed one and the total number of bins used is minimal.

Because bin packing has complete and exact information, the radius of information is zero. Hence the problem can be solved exactly. What is the $\varepsilon$-complexity (with $\varepsilon = 0$) as a function of $n$? Let comparison and the four arithmetic operations each cost unity. The cost of all known algorithms is an exponential function of $n$. It is known that bin packing is NP-complete (see Garey and Johnson, 1979) and it is therefore very likely that the $\varepsilon$-complexity (with $\varepsilon = 0$) is also an exponential function of $n$. In that case, we cannot solve bin packing on even the fastest computers for even moderate values of $n$.

A packing is an $\varepsilon$-approximation if it uses at most $(1 + \varepsilon)$ times more bins than the optimal one. Can we compute an $\varepsilon$-approximation at much lower cost? The answer is affirmative for arbitrarily small positive $\varepsilon$. [See Karmarkar and Karp (1982) for recent results and a survey of earlier work.]

This idea of trading increased uncertainty for lower complexity does not always work. There are problems (Garey and Johnson, 1979) for which the complexity is unchanged (more precisely, the problem remains NP-complete) no matter how much we increase the uncertainty.

### 3.2.3 Probabilistic Algorithms

We briefly indicate the use of probabilistic algorithms to decrease complexity. [See Rabin (1976) for additional material.]

Randomization is introduced into the algorithm. If $\varphi_r$ is a random algorithm, we say that $\varphi_r$ solves the problem with confidence greater than $1 - \varepsilon$ if for every $f \in F$ the probability that $\varphi_r$ produces an incorrect solution is smaller than $\varepsilon$.

As Rabin observes, it may at first seem surprising that employing randomization decreases complexity. He gives two examples. The first is to find the nearest neighbors of $n$ points in $k$ dimensions. The second is to determine whether a number is prime. Solovay and Strassen (1977) give a different probabilistic algorithm for determining primality.

We briefly discuss primality. Given an integer $f$ we wish to determine if it is prime. Note that the information is complete and exact. Hence, in the class of all algorithms, there exist algorithms which solve the problem exactly, that is, which determine whether or not $f$ is prime. To decrease complexity we settle for an answer with uncertainty; that is, we sometimes get the wrong answer. However, the probability of a wrong answer is "small."

### 3.2.4 Iterative Solution of Large Linear Systems

The approximate solution of NP-complete problems and the use of randomized algorithms to reduce complexity are recent developments. A far earlier use of the idea that it might be possible to reduce cost by solving approximately may be found in the iterative solution of large linear algebraic systems.

Let the linear system be specified by $Ax = b$, where $A$ is an $n$ by $n$ matrix. "Direct" methods can be used which (neglecting round-off errors) solve the system exactly at a cost proportional to $n^3$. (Direct methods based on fast matrix multiplication are not used in computational practice.) The values of $n$ occurring in practice are so large that direct methods may take too much time or space. For large values of $n$ the matrix $A$ is usually sparse, i.e., only a few elements are nonzero. Systems of linear equations with sparse matrices are especially well suited for solution by iterative methods.

More precisely, suppose we want to find an $\varepsilon$-approximation, i.e., a vector $x$ such that $\|Ax - b\| < \varepsilon$, where $\|b\| = 1$ and $\varepsilon \in (0, 1)$. An $\varepsilon$-approximation can be computed using an iterative method based on partial information. Depending on the size of $\varepsilon$, the size of $n$, the information $N$, and the class $F$ to which $A$ belongs, it may be that an $\varepsilon$-approximation can be iteratively computed at substantially lower cost than the exact solution.

To be specific, let $F$ be the class of symmetric positive definite matrices whose condition number is bounded by $M$. Let the information be

$$N_k(A, b) = [b, Ab, \ldots, A^k b].$$

This is called Krylov information and is commonly used in the iterative solution of large linear systems. If $k < n$, the information is partial. It is easy to show that $N_k(A, b)$ can be computed with $k$ matrix-vector multiplications. If $A$ is sparse, one matrix-vector multiplication takes time proportional to $n$ instead of $n^2$, and $N_k(A, b)$ can be computed in time proportional to $kn$.

How many matrix-vector multiplications do we need to determine an $\varepsilon$-approximation? This problem was studied by Traub and Woźniakowski (1980b), who showed that for large $n$ (relative to $M$ and $1/\varepsilon$) we must perform $k$ matrix-vector multiplications where

$$k \cong (\sqrt{M}/2)\ln(2/\varepsilon).$$

The minimal cost of finding an $\varepsilon$-approximation, comp($\varepsilon$), is given by

$$\text{comp}(\varepsilon) \cong n(\sqrt{M}/2)\ln(2/\varepsilon).$$

This should be contrasted with the cost of a direct method. If the sparseness of $A$ is not utilized, then typically the cost of a direct method is proportional to $n^3$, which for large $n$ is much greater than $n(\sqrt{M}/2)\ln(2/\varepsilon)$.

There do exist efficient direct methods which utilize the sparseness of $A$ and whose cost is substantially less than $n^3$. These methods are usually heavily dependent on the structure of $A$ and cannot be as widely applied as iterative methods.

## 4. Nonadaptive Information and Parallel Computation

One achievement of the information-based approach is general results on when nonadaptive information is just as powerful as adaptive information. Indeed, it is the notion of the radius of information that permits us to pose this question in general. In this section we give some examples; general results are reported in Section 7.1. In Section 4.5 we show that on a parallel computer, the use of nonadaptive information can lead to linear speedup.

We indicate the difference between nonadaptive and adaptive information through an example (see Section 7.1 for a general formulation). Let the information be $n$ evaluations of $f$. Thus

$$N(f) = [f(t_1), \ldots, f(t_n)].$$

If the $t_i$ are independently chosen, we say this is *nonadaptive information*. Now, assume that $f$ is evaluated at $t_1$. Then $t_2$ is chosen, knowing $t_1$ and $f(t_1)$. After $f(t_2)$ is evaluated, $t_3$ is chosen, knowing $t_1$, $f(t_1)$, $t_2$, and $f(t_2)$. Generally, $t_i$ is chosen, knowing $t_1$, $f(t_1)$, ..., $t_{i-1}$, $f(t_{i-1})$. We call this *adaptive information*.

Because the radius of information $r(N)$ measures the intrinsic uncertainty if $N$ is used, we determine the power of information $N$ by considering $r(N)$. Let $N^{\text{a}}$ be optimal adaptive information. If there exists nonadaptive information $N^{\text{non}}$, so that $r(N^{\text{non}})$ is comparable to $r(N^{\text{a}})$, then we conclude that adaptive information is no more powerful than nonadaptive information. This is made precise in Section 7.1.

Adaptive algorithms based on adaptive information are widely used. We shall see that for certain problems, adaption cannot help. There are also problems for which adaptive information is exponentially better than nonadaptive information.

Our interest in the power of nonadaptive information is motivated by a number of considerations.

1. If the optimal information is nonadaptive, we have a natural decomposition for parallel computation. Because nonadaptive information mini-

mizes communication requirements, it is desirable for distributed computation.

2. If we know that the optimal information is nonadaptive, we can very significantly cut the search space when we seek optimal information. Nonadaptive information is much simpler and therefore much easier to analyze than adaptive information.

3. Because the structure of nonadaptive information is so much simpler than the structure of adaptive information, it is of intrinsic mathematical interest when nonadaptive information is just as powerful.

## 4.1 Example: Zero Finding for Functions Which Change Sign

We give an example where adaptive information is exponentially more powerful than nonadaptive information. Let $F$ be the class of continuous functions $f$ which change sign on $[0, 1]$. Therefore, $f$ vanishes at least once on the interval. Let $\alpha$ be any point where $f$ vanishes. Without loss of generality we can assume $f(0) < 0$, $f(1) > 0$. The information is values of $f$.

Given only that $f$ belongs to $F$, we can compute an $\varepsilon$-approximation provided $\varepsilon > \frac{1}{2}$. We simply take as our $\varepsilon$-approximation, $x = \frac{1}{2}$. If $\varepsilon \leq \frac{1}{2}$ we must have more information.

Compute $f(\frac{1}{2})$. If $f(\frac{1}{2}) > 0$, there is a zero on $(0, \frac{1}{2})$. If $f(\frac{1}{2}) < 0$, there is a zero on $(\frac{1}{2}, 1)$. If $f(\frac{1}{2}) = 0$, then $\alpha = \frac{1}{2}$. In all cases, whereas we originally knew $\alpha$ lay in an interval of length 1, we now know it lies in an interval of length $\frac{1}{2}$.

We will describe one more step of this process. Without loss of generality, assume there is a zero on $(0, \frac{1}{2})$. We are now in the same situation as we started except that we have evaluated $f$ at one point and we have halved the interval. We can now compute an $\varepsilon$-approximation provided $\varepsilon > \frac{1}{4}$. We take as our $\varepsilon$-approximation, $x = \frac{1}{4}$. If $\varepsilon \leq \frac{1}{4}$ we must have more information.

Compute $f(\frac{1}{4})$. If $f(\frac{1}{4}) > 0$, there is a zero on $(0, \frac{1}{4})$. If $f(\frac{1}{4}) < 0$, there is a zero on $(\frac{1}{4}, \frac{1}{2})$. If $f(\frac{1}{4}) = 0$, then $\alpha = \frac{1}{4}$. We have again halved the interval in which $\alpha$ lies.

The general pattern should now be clear. At each step we evaluate $f$ in the center of the interval where $\alpha$ is known to be. This information is called bisection information. It is clearly adaptive because we cannot decide where to sample next until we know the result of the previous sample. After sampling at $n$ points, we have reduced the size of the interval in which $\alpha$ lies to $2^{-n}$. If $\varepsilon > 2^{-(n+1)}$, we take our $\varepsilon$-approximation as the midpoint of the interval in which $\alpha$ is known to lie.

Thus bisection information using $n$ points cuts the uncertainty to $2^{-(n+1)}$. Furthermore, it is known that this is optimal adaptive information (see Sikorski, 1982).

How much can we reduce the uncertainty if we restrict ourselves to nonadaptive information? The optimal nonadaptive information is to sample $f$ at $n$ equispaced points (see Traub and Woźniakowski, 1980a, p. 166). Then $\alpha$ lies in an interval of length $1/(n + 1)$. If $\varepsilon > 1/(2(n + 1))$, then we take our $\varepsilon$-approximation as the midpoint of the interval in which $\alpha$ is known to lie.

Thus the optimal nonadaptive information cuts the uncertainty to $1/(2(n + 1))$. In contrast, the optimal adaptive information cuts the uncertainty to $2^{-(n+1)}$. Thus, for this problem (zero finding) and this set of functions (continuous functions which change sign on $[0, 1]$), adaptive information is exponentially better than nonadaptive information.

## 4.2 Example: Integration

In the previous section we saw an example where adaptive information was exponentially stronger than nonadaptive information. We now give an example where adaptive information is no stronger than nonadaptive information.

Recall the integration example of Section 2. We said that the optimal evaluation points were $t_i = (2i - 1)/2n$, $i = 1, 2, \ldots, n$. This information is nonadaptive. There exists *no* adaptive information which is superior.

This is a special case of a very general result. We will return to this in Section 7.1 when we discuss results of the general theory.

The fact that adaption does not help is counterintuitive. It might be expected that it is possible to sample to see where the integrand is changing rapidly and once such a region is identified to put more sample points there. The theory states that this intuition is fallacious.

Recall that the example of Section 2 is in a worst-case setting. It may be hoped that on the average, adaptation helps. Recent work shows adaptive information is not stronger, even on the average (see the discussion in Section 8).

## 4.3 Example: Zero Finding for Lipschitz Functions

In Section 4.1 we considered zero finding for functions which changed sign and found that adaptive information was exponentially more powerful than nonadaptive information. Our second example was integration, where adaption does not help. Now, integration is a linear operation.

That is, the integral of a sum is the sum of the integrals. As we shall see in Section 7.1, adaption does not help for linear operations. Zero finding is not linear because the zeros of the sum of two functions are not the sum of the zeros. This suggests that perhaps adaption helps for nonlinear problems and does not help for linear problems. The following example shows that such a general result does not hold.

The problem is once more zero finding. Now let $F$ be the class of functions $f$ such that $f$ has a zero and such that

$$|f(x) - f(y)| \leq K|x - y|$$

for all real $x$ and $y$. This condition is called a Lipschitz condition and we therefore call $F$ a Lipschitz class. As in Section 4.1, we assume the information $N(f)$ consists of function values. Thus $N(f) = [f(t_1), f(t_2), \ldots, f(t_n)]$. Sukharev (1976) shows that adaption does not help. Furthermore, the optimal nonadaptive information is the values of $f$ at almost equispaced points. The radius of information of this optimal information is $\frac{1}{2}K/(n + 1)$. Hence an $\varepsilon$-approximation can be computed iff $K/2(n + 1) < \varepsilon$.

These results have been generalized to any number of dimensions by Sikorski (1983).

We review what we have learned from these examples. For the same problem, root finding, we have found that for one class of functions adaptive information is exponentially better than nonadaptive information, whereas for another class of functions adaption does not help. To date, there are no general results on when adaption helps for nonlinear problems. This should be contrasted with linear problems (see Section 7.1).

## 4.4 Example: Binary Search

We turn to a very different kind of example. Zero finding and integration are examples of continuous problems. We now give a discrete example.

Our example is an instance of binary search, popularly called the game of 20 Questions. Binary search models many important applications, including disease diagnosis and drug prescription.

We present a simple version as a game between players A and B. A thinks of an integer $\alpha$, where $1 \leq \alpha \leq m$. Player B tries to identify $\alpha$ by asking whether $\alpha$ belongs to certain subsets of the integers $\{1, 2, \ldots, m\}$. For each question, A answers "1" if $\alpha$ belongs to the subset, and "0" otherwise. The goal of the game is for B to identify $\alpha$ with the minimal number of questions. The information in this example is the sequence of zeros and ones which B gets in answer to his questions.

Assume for simplicity that $m = 2^n$. If B asks "bisection" questions which at each step halve the size of the set to which $\alpha$ belongs, then B can always identify $\alpha$ with $n = \log_2 m$ questions.

Bisection information is adaptive. Is there nonadaptive information which will also permit B to identify $\alpha$ with $\log_2 m$ questions? The answer is yes, and we will leave it as a small puzzle for the reader to see what pattern of questions B should ask.

Thus, for this example of binary search, adaption does not help. A general theorem is established in Traub *et al.* (1983, Chapter 4, Theorem 3.1) giving a condition under which adaption does not help. The statement of the general theorem requires more machinery than we want to introduce here. Suffice it to say that if the general theorem is used for the special setting of this example, it states that adaption does not help because any question on subsets is allowed.

There are close connections between zero finding for functions that change sign and for binary search. Yet, for zero finding, adaption helps exponentially whereas for binary search it does not help at all. The reader may want to consider why.

## 4.5 Parallel Computation and Nonadaptive Information

At the beginning of our discussion of nonadaptive information we pointed out that nonadaptive information is well suited to parallel computation because it provides a natural decomposition. Here we use a simple example to quantify this notion.

### *4.5.1 Parallel Speedup*

First we review the methodology for determining how much faster parallel computation is than sequential computation (see also Traub, 1974).

Let the minimal cost to compute an $\varepsilon$-approximation on a sequential computer be the ***sequential ε-complexity***, denoted by comp($\varepsilon$). We referred to this as $\varepsilon$-complexity, or problem complexity, in Section 2. We assume a parallel computer with $p$ processors which are identical and independent. Let the minimal cost to compute an $\varepsilon$-approximation on such a parallel computer be the ***parallel ε-complexity***, denoted by comp($\varepsilon$, $p$). Of course, comp($\varepsilon$, 1) = comp($\varepsilon$).

The speedup $R(\varepsilon, p)$ is defined as

$$R(\varepsilon, p) = \text{comp}(\varepsilon)/\text{comp}(\varepsilon, p). \tag{4.1}$$

We comment on this definition. Researchers sometimes misinterpret this measure and compare their favorite parallel algorithm with *some* sequential algorithm. To see what has been achieved by parallelism, the comparison must be with the $\varepsilon$-complexity, that is, with the cost of the optimal sequential algorithm.

The speedup $R(\varepsilon, p)$ measures how much parallelism can speed the solution of a problem. It is an algorithm-independent measure. It is easy to see that

$$R(\varepsilon, p) \le p. \tag{4.2}$$

Therefore, linear speedup is optimal. The speedup of some problems is only log $p$ or even a constant independent of $p$ (see Traub, 1974).

In Section 4.5.2 we shall see that under simple, nonrestrictive assumptions, the speedup of the integration problem studied in Section 2 is close to $p$.

### 4.5.2 An Example Where Parallel Speedup Is Close to Optimal

Recall that for the integration problem studied in Section 2 a lower bound on the $\varepsilon$-complexity is given by

$$\text{comp}(\varepsilon) \ge mc + m - 1 \tag{4.3}$$

where the number of function samples, $m$, is given by

$$m = \lfloor 1/4\varepsilon \rfloor + 1.$$

Our sequential model of computation is that every arithmetic operation costs unity and each evaluation of $f$ at a point costs $c$.

Recall that the optimal sequential algorithm is

$$\varphi(N(f)) = \frac{1}{m}\sum_{i=1}^{m} f\left(\frac{2i-1}{2m}\right). \tag{4.4}$$

We use this as a parallel algorithm. Assume that the cost of evaluating $f$ at a point is again $c$. That is, we do not use parallelism in the computation of $f$. Assume for simplicity that $p$, the number of processors, divides $m$. Then the cost of computing the $m$ values of $f$ is $cm/p$. The arithmetic operations to form $\varphi$ can be performed at cost $m/p + \lceil \log_2 p \rceil$. Hence the cost of computing $\varphi$ on our parallel computer is

$$\text{cost}(\varphi, p) = cm/p + m/p + \lceil \log_2 p \rceil. \tag{4.5}$$

Because comp$(\varepsilon, p)$ is the minimal cost of computing an $\varepsilon$-approximation, we conclude from Eqs. (4.3) and (4.5) that

$$R(\varepsilon, p) = \frac{\text{comp}(\varepsilon)}{\text{comp}(\varepsilon, p)} \ge \frac{mc + m - 1}{cm/p + m/p + \lceil \log_2 p \rceil}. \tag{4.6}$$

From Eqs. (4.2) and (4.6),

$$p \ge R(\varepsilon, p) \ge p\left[\frac{m(c+1) - 1}{m(c+1) + p\lceil \log_2 p \rceil}\right]. \tag{4.7}$$

Observe that if $mc$ is large compared to $p \log_2 p$, which is a very reasonable assumption in practice, then $R(\varepsilon, p) \simeq p$.

We conclude that the parallel speedup is close to linear; that is, it is close to optimal.

We have carried out this analysis for the special problem of integration and for information consisting of function evaluations. The same conclusion, that the parallel speedup is close to optimal, holds whenever the optimal information is nonadaptive and there is a linear optimal error algorithm. [See Section 7 and Traub and Woźniakowski (1980a, Chapters 2 and 3).]

## 5. Limitations of the Algorithm-Centered Approach

As we mentioned in Section 1, we believe that for problems with partial or approximate information the usual algorithm-centered approach can be supplemented, and sometimes replaced, by the information-centered approach.

There will, of course, be problems for which it is technically difficult to apply the information-centered approach and it will still be necessary to resort to the algorithm-centered approach. This will be particularly the case for complicated "real-world" models. With time, we expect the technical difficulties to be overcome for harder and harder problems. Today, algorithms are often obtained on the basis of ad hoc criteria. Using such criteria has several disadvantages: ad hoc criteria may not be very good; and, if ad hoc criteria are used, there is no idea of how far the algorithm is from optimal, and, in practical terms, how much money is being wasted.

We use one well-known algorithm to illustrate the limitations of the ad hoc approach.

### 5.1 Example of Ad Hoc Criteria: Gauss Quadrature

The family of Gauss quadrature methods is widely used in practice. The methods are derived under three assumptions:

1. Methods of the form $\varphi = \sum_{i=1}^{n} a_i f(t_i)$ are considered.
2. The information $f(t_1), \ldots, f(t_n)$ is nonadaptive.

3. The $2n$ parameters $a_1, \ldots, a_n, t_1, \ldots, t_n$ are chosen by the criterion that the error of integration should be zero if $f$ is a polynomial of degree, at most, $2n - 1$.

We discuss these three criteria. It is unlikely that the first two criteria will be found in a text; the assumptions are made implicitly.

The first criterion is that the algorithm is linear. Smolyak (1965) showed that under a fairly weak assumption regarding the class of integrands, it can be *concluded* that the optimal algorithm for integration is linear. The advantage of *proving* that the optimal algorithm is linear, rather than deciding to study only linear algorithms, is clear. If an ad hoc decision is made to study only linear algorithms, a much better algorithm which is not linear may be missed; if it is proved that an optimal algorithm must be found among the linear ones, that cannot happen.

The second ad hoc criterion is that the information is nonadaptive. Under the same assumption mentioned previously regarding the class of integrands, Bakhvalov (1971) *proved* that nothing is lost by considering only nonadaptive information.

The third ad hoc criterion is that polynomials of sufficiently low degree are exactly integrated. Is there any reason to think this is a good criterion if the integrand is not a polynomial? As will be shown in Section 5.2, we can pay an exponential penalty by using Gauss information rather than optimal information even for analytic integrands.

If the class of integrands is sufficiently "polynomial-like," then these criteria lead to good algorithms. Thus, if $F$ is the class of analytic functions with uniform bounded norm on the disk of radius $r$, then, for large $r$, Gauss nodes and Gauss quadrature formulas are nearly optimal (Barnhill, 1968; Larkin, 1970; Pinkus, 1975). Note that the restrictions on $F$ are severe. The integrands must be "almost" entire and uniformly bounded.

We have no particular quarrel with Gauss quadrature. We use it because it is widely known and because it is typical of the numerous algorithms obtained on the basis of ad hoc criteria. Indeed, there is a very beautiful mathematical theory involving families of orthogonal polynomials (Ralston and Rabinowitz, 1978) that is used to analyze Gauss quadrature. However, the elegant theory of orthogonal polynomials does not necessarily lead to good information and algorithms.

## 5.2 How Bad Can Gauss Information Be for Analytic Integrands?

The title of this section refers to Gauss information rather than to Gauss algorithms. We will show that Gauss information can be poor, and therefore *any* algorithm using Gauss information must be poor. (By Gauss information we mean integrand evaluations at the Gauss nodes.)

Let $F$ be the class of real functions on $[-1, 1]$ which can be analytically extended to the unit disk and whose extension is uniformly bounded in norm. Let $N^G$ denote Gauss information and let $N^*$ denote optimal information. Kowalski *et al.* (1983) show that

$$r(N^G) \sim 1/n^2,$$

whereas Bojanov (1974) has shown that

$$r(N^*) \sim e^{-c\sqrt{n}}, \qquad c > 0.$$

Thus, an exponential penalty is paid for using Gauss information rather than the optimal information.

## 5.3 Is There a Relation between the Exactness Criterion and Optimal Algorithms?

The Gauss quadrature coefficients are chosen to exactly integrate all polynomials of degree, at most, $2n - 1$. Is there any relation between exactness and optimal algorithms?

The answer is negative. For example, the optimal algorithm discussed in Section 2.7,

$$\varphi = \frac{1}{n}\sum_{i=1}^{n} f\left(\frac{2i-1}{2n}\right),$$

turns out to be exact only for first-degree polynomials. Werschulz (1983) has shown there exists a class of integrands for which not even constants are exactly integrated by an algorithm which is optimal for that class.

## 6. An Abstract Model

We present an abstract model; the following sections are numbered and titled to correspond to those of Section 2. We emphasize that even though a numerical example was used in Section 2, the abstract formulation is not confined to such applications.

This is a normed worst-case model because uncertainty is measured by a norm. A model where uncertainty is measured without a norm is briefly mentioned in Section 8; average-case models are also discussed in Section 8. We limit ourselves to uncertainty caused by partial information. [The theory of approximate information and further development of a model where uncertainty is measured without a norm may be found in Traub *et al.* (1983).]

Although we will formulate an abstract model and present a number of important results, we will keep the presentation as simple as possible. For example, we will not cover rather deep connections with pure approximation theory [for which, see Traub and Woźniakowski (1980a, Chapter 2, Sect. 6; Chapter 3, Sect. 5; Chapter 7, Sect. 4)].

## 6.1 Problem Formulation

Let $S$ be a linear or nonlinear operator (mapping)

$$S: F \to G.$$

We wish to compute the element $S(f)$, $f \in F$. In general, we have to settle for an approximation to $S(f)$. We want to compute an element $x = x(f)$ such that

$$\|S(f) - x\| < \varepsilon \tag{6.1}$$

for some preassigned $\varepsilon > 0$, and $x = S(f)$ if $\varepsilon = 0$. We say $x$ is an *$\varepsilon$-approximation*. It measures the uncertainty in our knowledge of $S(f)$. We assume $F$ is a subset of a linear space $F'$ over the real or complex field and that $G$ is a normed linear space.

We must know something about $f$ to compute an $\varepsilon$-approximation. We assume we know the element $N(f)$ where

$$N: F \to H$$

is a linear or nonlinear operator (mapping). We say $S$ is the *solution operator*, $F$ is the class of *problem elements*, $N$ is the *information operator*, and $N(f)$ is the *information*.

The operators $S$ and $N$ and the abstract set $F$ are the basic concepts of our formulation. We call this the *SFN* model. We seek an $\varepsilon$-approximation for all $f \in F$. This normed worst-case model may be formulated as follows:

**Problem:** Compute an $\varepsilon$-approximation to $S(f)$, $\forall f \in F$.
**Information:** $N(f)$.

In Table I we relate these concepts to the integration example of Section 2. The left-hand column has the abstract concepts and on the right are the corresponding concepts for the integration example.

By special choices of $S$, $F$, and $N$, we specialize to areas which are major disciplines. Thus $S = I$ (the identity operator) is the approximation

TABLE I

PROBLEM FORMULATION CONCEPTS

| Abstract concept | Integration example |
|---|---|
| $f$ | An integrand |
| $S(f)$ | $S(f) = \int_0^1 f(t)\,dt$ |
| $F$ | Set of integrands such that $\lvert f^{r}(t)\rvert \le 1$ |
| $N(f)$ | $N(f) = [f(t_1), \dots, f(t_n)]$ |
| $\|S(f) - x\| < \varepsilon$ | $\lvert\int_0^1 f(t)dt - x\rvert < \varepsilon$ |

problem (optimal estimation). On the other hand, $N = I$ is typical of the problems treated in algebraic and combinatorial complexity.

### *6.1.1 Linear Problems*

The case that $S$ is a linear operator is of special interest for two reasons. For many applications, $S$ is linear. Examples include approximation (optimal estimation), integration (especially multivariate integration), interpolation (especially multivariate interpolation), and linear partial differential equations. Furthermore, the theory, not surprisingly, is far more developed for linear $S$. [See Micchelli and Rivlin (1977) and Traub and Woźniakowski (1980a, Chapters 1–6) for the worst case; see Traub *et al.* (1981), Wasilkowski and Woźniakowski (1982), and Woźniakowski (1982) for the average case.]

It is desirable to consider the case that the set $F$ is generated by a linear operator. Recall that $F$ is a subset of a linear space $F'$. We now add the assumption that $T$, $T: F' \to K$, where $T$ is linear and where $K$ is a linear normed space over the real or complex field. Assume

$$F = \{f \in F' \quad \text{and} \quad \|Tf\| \le 1\}. \tag{6.2}$$

We assume $S$, $S: F \to G$ is also linear. If these assumptions hold we say the *problem is linear.*

The definition of linear problem may seem artificial, but many problems are of this form. The integration example formulated in Section 2.1 is linear. Often, $Tf = f^{(r)}$. The assumption $\|Tf\| \le 1$ is for convenience; it is equivalent to the assumption that $\|Tf\|$ is uniformly bounded.

The important assumption is that $Tf$ exists. The quantity $\|Tf\|$ appears in the formula for radius of information. If $\|Tf\|$ is unbounded, the radius of information and therefore the uncertainty are also unbounded.

In Section 7.2 we will discuss whether linear problems have "linear optimal algorithms."

## 6.2 Radius of Information

There exists a quantity called the radius of information which measures the intrinsic uncertainty of solving a problem when information $N$ is available. The concept of radius of information is motivated by the following considerations.

Fix $f \in F$. We know $y = N(f)$. Assume that $N$ is a many-to-one operator, that is, $N(f)$ is *partial information*. Let $V(y)$ be the preimage set of $y$ in $F$. That is, it is the set of elements in $F$ indistinguishable under $N$ (see Fig. 1). Let $U(y)$ be the image set of $V(y)$ under $S$.

It should be clear that with the information $N(f)$ we cannot distinguish the element $S(f)$ among the elements $U(y)$. Hence the "size" of the set $U(y)$ is a measure of the intrinsic uncertainty due to $N$. A standard measure of the size of a set is the minimal radius of a "ball" which contains it. Let $\mathrm{rad}(U(y))$ denote the radius of $U(y)$. We now vary $f$ and define the radius of information (for the worst case) by

$$r(N) = \sup_{y \in N(F)} \mathrm{rad}(U(y)).$$

The preceding discussion serves as a sketch of the proof of

**Theorem 6.1** The information $N$ is strong enough to determine an $\varepsilon$-approximation, $\forall f \in F$, iff $r(N) < \varepsilon$. ■

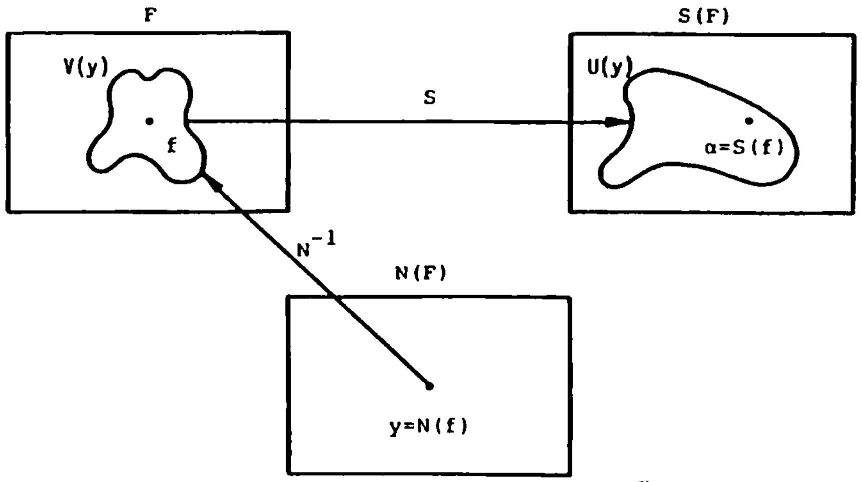

FIG. 1.

A formal definition of radius of information and proof of Theorem 6.1 may be found in Traub and Woźniakowski (1980a, Chapter 1, Definition 2.1, and Theorem 2.1).

Observe that the radius of information should be written $r(N, S, F)$. We write it simply as $r(N)$ for simplicity and because we will usually keep $S$ and $F$ fixed and study the radius as a function of information. Of course $S$ and $F$ also supply information [see Traub *et al.* (1981, Sect. 6) for a quantification].

The radius of information defined here bounds the uncertainty caused by partial information. The generalization that includes uncertainty caused by approximate information and restriction to a class of realizable algorithms may be found in Traub *et al.* (1983, Chapters 2 and 3, Appendix H).

## 6.3 Algorithms

An *idealized algorithm* (or simply, algorithm) is any mapping for computing an approximation knowing the information $N(f)$ and knowing that $f \in F$. Thus, an algorithm is any linear or nonlinear mapping $\varphi: N(F) \to G$.

## 6.4 Optimal Algorithms

The error of approximating $S(f)$ using the algorithm $\varphi$ is defined as

$$e(\varphi, f) = \|S(f) - \varphi(N(f))\|.$$

The *algorithm error* $e(\varphi)$ is the worse $e(\varphi, f)$ for all $f \in F$. That is,

$$e(\varphi) = \sup_{f \in F} e(\varphi, f).$$

The radius of information is a lower bound on the algorithm error. We have

**Theorem 6.2** For any algorithm $\varphi$ which uses the information $N(f)$

$$e(\varphi) \geq r(N).$$

Furthermore, this lower bound is best possible. ■

We say $\varphi$ is an *optimal error algorithm* (or simply an optimal algorithm) if

$$e(\varphi) = r(N).$$

We denote an optimal algorithm by $\varphi^*$. A second notion of optimality (optimal complexity algorithm) is defined in Section 6.11.

### 6.4.1 How to Generate Good Algorithms

We define two paradigms for generating optimal or near-optimal algorithms.

An *interpolatory algorithm*, $\varphi^I$, simply chooses any element of $U(y)$ (defined in Section 6.2) as an approximation to $S(f)$. An example of an interpolatory algorithm is given in Fig. 2. Which element of $U(y)$ should be chosen? The set $V(y)$ consists of all $f \in F$ indistinguishable from $f$ under $N$. Choose an element $\tilde{f}$ from $V(y)$ which is "simpler" than $f$. Then the interpolatory algorithm is $S(\tilde{f})$.

For example, if $F$ is some class of scalar functions, then $\tilde{f}$ can be chosen as a polynomial or piecewise polynomial $p$ such that $p \in F$ and $N(p) = N(f)$. Then the interpolatory algorithm is $S(p)$.

Although this seems like a very crude process, any interpolatory algorithm (in the worst-case normed linear space setting described in Section 6.1) is within at most a factor of two of having optimal error! Let $\varphi^I(N(f))$ be any interpolatory algorithm. We have

**Theorem 6.3**

$$e(\varphi^I) \le 2r(N). \quad \blacksquare$$

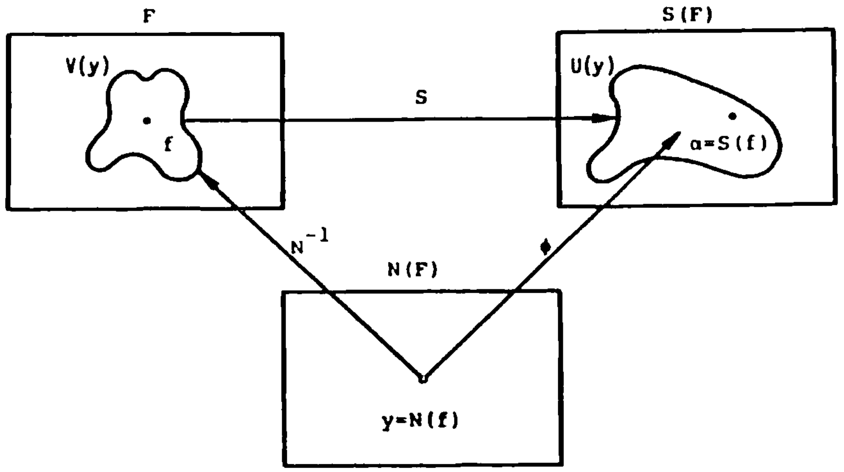

FIG. 2.

A *central algorithm* $\varphi^c$ chooses a "center" of the set $U(y)$, if it exists, as an approximation to $S(f)$. (Not every set has a center.) A central algorithm is always optimal. That is, we have

**Theorem 6.4**

$$e(\varphi^c) = r(N). \quad \square$$

Formal definitions of interpolatory and central algorithms and proofs of the two theorems may be found in Traub and Woźniakowski (1980a, Chapter 1, Sect. 2).

Note that because $r(N)$ is a sharp lower bound on the error of any algorithm, interpolatory and central algorithms provide tight upper bounds.

If it is desired to actually construct a good algorithm for a particular $S$, $F$, $N$, it is possible to proceed as follows: (1) the definitions of interpolatory and central algorithms provide simple paradigms for generating good algorithms, and a decision is made about which of these to use; (2) apply the paradigm for a particular $S$, $F$, $N$.

Although the first step is conceptually easy, the second may be hard. Depending on $S$, $F$, and $N$, it can be technically difficult to obtain the radius of information and the optimal algorithm. Furthermore, these quantities vary if any of $S$, $F$, or $N$ change. For fixed $S$, $F$, $N$, the optimal algorithm need be obtained only once. Therefore the analysis may be viewed as a precomputing cost. An interpolatory algorithm is often far easier to construct than a central algorithm.

Many examples of the radius of information and optimal algorithms may be found in Traub and Woźniakowski (1980a, Chapters 6 and 8) and in Traub *et al.* (1983, Chapter 6).

We emphasize that Theorems 6.3 and 6.4 hold for the normed worst-case setting. If uncertainty is not measured by a norm, an interpolatory algorithm might be useless (see Traub *et al.*, 1983, Appendix A). In this more general setting, we introduce "interior" and "midpoint" algorithms and show that they have good error properties.

### 6.4.2 Spline Algorithms

We briefly discuss *spline algorithms* which have many desirable properties.

Let the class $F$ be given by

$$F = \{f: \|Tf\| \le 1\}, \tag{6.3}$$

where $T$ is a linear operator. Let $\bar{f} \in F$, be indistinguishable from $f$ under $N$, and have minimal $T$-norm, that is,

$$\|T\bar{f}\| = \min\{\|Tg\|: N(g) = N(f)\}. \tag{6.4}$$

A *spline algorithm*, $\phi^s$, is defined by $\phi^s = S(\bar{f})$.

We mention some of the desirable properties of spline algorithms. They are interpolatory and therefore close to optimal. If the norm of Eqs. (6.3) and (6.4) is a Hilbert norm, $S$ a linear operator, and $T(\ker N)$ is closed, then the spline algorithm is a linear central algorithm. (See Section 6.5 for the definition of linear algorithm.) Traub and Woźniakowski (1980a, Chapter 4) develop other useful properties of spline algorithms for the worst-case model. Spline algorithms also enjoy optimality properties for the average case (see Traub *et al.*, 1981; Wasilkowski and Woźniakowski, 1982).

## 6.5 Linear Algorithms

We say an algorithm $\varphi$ is a *linear algorithm* if

$$\varphi(N(f)) = \sum_{i=1}^{n} L_i(f)g_i,$$

where the $g_i$ are elements of $G$ and where $N(f) = [L_1(f), \ldots, L_n(f)]$.

Linear algorithms are easy to implement. Because $g_1, \ldots, g_n$ are independent of $f$, they can be precomputed. Given the $g_i$, we perform at most $n$ multiplications of elements from $G$ by a scalar and $n - 1$ additions of elements from $G$. Given the information $N(f)$, the cost of forming $\varphi(N(f))$ is linear in $n$ and usually small with respect to the complexity of computing $N(f)$. Traub and Woźniakowski (1980a, Chapter 5) present a discussion of the complexity of linear algorithms.

It is desirable to use an optimal algorithm which is linear. A linear optimal algorithm always enjoys close to optimal complexity (see Section 6.11). When does a linear optimal algorithm exist? It might be hoped that linear problems always enjoy linear optimal algorithms, but this turns out not to be the case (see Section 7.2 for further material).

Another desirable property of linear optimal algorithms is that if they use optimal nonadaptive information, they enjoy close to optimal parallel speedup. An example was given in Section 4.5.1.

## 6.6 Optimal Information

We have assumed that the information operator $N$ is fixed. We now vary $N$ and pose the problem of optimal information.

We confine ourselves to *linear* information. Arbitrary nonlinear information operators are usually too powerful and all problems become trivial (see Traub and Woźniakowski, 1980a, Chapter 7).

Without loss of generality we say $N$ is a *linear information operator* if $N = [L_1, \ldots, L_n]$, where the $L_i$, $i = 1, \ldots, n$, are linearly independent linear functionals. We call $n$ the *cardinality* of $N$. To stress dependence on cardinality we sometimes write $N_n$.

Note that in the integration example we chose $L_i(f) = f(t_i)$. The cardinality was the number of sample points. Linear information is very commonly used in practice. It generalizes information such as evaluation of functions and derivatives.

Fix $n$. Let $N_n^*$ be such that

$$r(N_n^*) = \inf_{N_n} r(N_n).$$

Then we say that $N_n^*$ is *optimal information of cardinality n* (or simply, optimal information). The infimum is taken over all linear information of cardinality $n$. Sometimes the infimum is taken over a restricted class of linear information of cardinality $n$. For example, for the integration example, we restrict ourselves to information consisting of function evaluations.

## 6.7 Example

An illustration of these concepts can be found in Section 2.7.

## 6.8 Is the Information Strong Enough?

So far, we have fixed $n$, the cardinality of information. Recall that we want to compute an $\varepsilon$-approximation. It may turn out, even using optimal information, that $n$ is not large enough.

We therefore vary $n$ and ask what is the smallest $n$ such that the information is strong enough to determine an $\varepsilon$-approximation. Recall that we can determine an $\varepsilon$-approximation, $\forall f \in F$, iff $r(N) < \varepsilon$. As before, let $N_n^*$ denote the optimal information of cardinality $n$. Define the *$\varepsilon$-cardinality number* as

$$m(\varepsilon) = \begin{cases} \min\{n: r(N_n^*) < \varepsilon\} & \text{for } \varepsilon > 0, \\ \min\{n: r(N_n^*) = 0\} & \text{for } \varepsilon = 0, \end{cases}$$

with the convention $\min(\emptyset) = \infty$. In words, we consider the optimal information of cardinality $n$ and then vary $n$, seeking the optimal information of smallest cardinality whose radius of information is less than $\varepsilon$.

Can it happen that $r(N_n^*) = \infty$ for all finite $n$? This would imply that no $\varepsilon$-approximation is possible for any finite $\varepsilon$, no matter how large, using optimal linear information of finite cardinality. The answer is yes, even if $S$ is linear (see Traub and Woźniakowski, 1980a, Chapter 2, Sect. 3).

## 6.9 Computational Complexity

If $r(N) < \varepsilon$, an $\varepsilon$-approximation can be determined and we can ask what is the computational complexity. First we must define our model of computation.

### 6.9.1 Model of Computation

We indicate our model of computation, which consists of a set of primitive operations, permissible information, and permissible algorithms.

1. Let $p$ be a *primitive* operation. Examples of primitive operations include arithmetic operations, comparisons, taking the maximum of $n$ numbers, and the evaluation of a radical, an integral, a linear functional, or a nonlinear functional. Let comp$(p)$ be the cost of $p$. We assume that comp$(p)$ is finite. Suppose that $P$ is a given collection of primitives. The choice of $P$ and comp$(p)$, $p \in P$, are arbitrary and can depend on the particular problem being solved.
2. Let $N$ be an information operator. We say $N$ is *permissible* with respect to $P$ if there exists a program using a finite number of primitive operations from $P$ which computes $N(f)$ for all $f \in F$. If $N(f)$ requires the evaluation of primitives $p_1, p_2, \ldots, p_k$, then

$$\text{comp}(N(f)) = \sum_{i=1}^{k} \text{comp}(p_i).$$

We call comp$(N(f))$ the *information complexity* of computing $N(f)$.

3. Let $\varphi$ be an algorithm which uses the permissible information $N$. To evaluate $\varphi(N(f))$ we

    compute $y = N(f)$,
    compute $\varphi(y)$.

The complexity of computing $y$ is given by (2). We say that $\varphi$ is *permissible* with respect to $P$ if there exists a program using a finite number of primitive operations from $P$ which computes $\varphi(y)$ for all $y = N(f), f \in F$. Let comp$(\varphi(y))$ be the *combinatory complexity* of computing $\varphi(y)$. Thus, if $\varphi(y)$ requires the evaluation of primitives $q_1, \ldots, q_j$, then

$$\text{comp}(\varphi(y)) = \sum_{i=1}^{j} \text{comp}(q_i).$$

In the model of computation in Section 2.9, the primitive operations are function evaluation and the four arithmetic operations. In the algorithm $\varphi^*(N^*(f))$ in Section 2.10, $k = j = m$. Therefore the information complexity is $mc$ and the combinatory complexity is $m$.

### 6.9.2 Definition of $\varepsilon$-Complexity

Suppose that $r(N) < \varepsilon$ for a permissible $N$ and let $\Phi(\varepsilon)$ denote the class of permissible algorithms that use $N$ and for which $e(\varphi) < \varepsilon$. Assume that $\Phi(\varepsilon)$ is not empty. Let $\varphi \in \Phi(\varepsilon)$. Then the *algorithm complexity* of $\varphi$ is defined by

$$\text{comp}(\varphi) = \sup_{f \in F}[\text{comp}(N(f)) + \text{comp}(\varphi(N(f)))].$$

We define the *$\varepsilon$-complexity for the information $N$* as

$$\text{comp}(N, \varepsilon) = \inf\{\text{comp}(\varphi) : \varphi \in \Phi(\varepsilon)\}$$

with the convention that $\inf\{\emptyset\} = +\infty$.

To define $\varepsilon$-complexity we must first specify a class of permissible information, $\Psi$. An example will show why this is necessary. We wish to approximate $\int_0^1 f(t)\,dt$. If all linear functionals were permissible information, the integral could be exactly computed with one piece of information. For this problem it is natural to assume that only function evaluations (or its derivatives) are permissible.

For a given class of permissible information operators we define the $\varepsilon$-complexity in the class $\Psi$ as

$$\text{comp}(\Psi, \varepsilon) = \inf\{\text{comp}(N, \varepsilon) : N \in \Psi\}.$$

Note that all the concepts presented here depend on the solution operator $S$. The $\varepsilon$-complexity in the class $\Psi$ might be denoted by comp$(\Psi, S, \varepsilon)$ rather than by comp$(\Psi, \varepsilon)$. Because $S$ is fixed, it is omitted for simplicity.

If the class $\Psi$ is understood from the context, we can write comp$(\varepsilon)$ instead of comp$(\Psi, \varepsilon)$. We call comp$(\varepsilon)$ the $\varepsilon$-complexity. We sometimes refer to $\varepsilon$-complexity as problem complexity or computational complexity.

## 6.10 Example

An example of computing upper and lower bounds on $\varepsilon$-complexity may be found in Section 2.10.

## 6.11 Optimal Complexity Algorithms

We say $\varphi^{**}$ is an *optimal complexity algorithm* in the class $\Psi$ iff $\varphi^{**}$ uses an information operator $N$ from $\Psi$ and

$$\operatorname{comp}(\varphi^{**}) = \operatorname{comp}(\varepsilon).$$

Note that we have two major notions of optimality: optimal error algorithm (which we have been calling optimal algorithm) and optimal complexity algorithm.

### *6.11.1 When Is an Optimal Error Algorithm Nearly an Optimal Complexity Algorithm?*

We discuss the relation between an optimal error algorithm and an optimal complexity algorithm. Suppose there exists an optimal error algorithm whose combinatory complexity is small relative to the information complexity. Such an algorithm is close to being an optimal complexity algorithm.

This is a very favorable situation because the algorithm minimizes both error and cost. This suggests the question: For which solution operators $S$ does there exist an optimal error algorithm with small combinatory complexity?

Fortunately, for many practical problems $S$, we can find an optimal error algorithm with small combinatory complexity. For instance, for many linear $S$ (although not for all; see Section 7.2) there exists a linear optimal error algorithm. It is easy to show that a linear optimal error algorithm is always close to an optimal complexity algorithm (see Traub and Woźniakowski, 1980a, Chapter 5).

For some nonlinear $S$ we can also find an optimal error algorithm with small combinatory complexity. An example is provided by the zero-finding problem for functions which change sign. The bisection algorithm is an optimal error algorithm with constant combinatory complexity, that is, the combinatory complexity is independent of the cardinality of information (see Traub and Woźniakowski, 1980a, Chapter 8, Sect. 3).

What is the relation between these notions? Because the combinatory complexity of an optimal error algorithm may be very high, there is, in general, no relation. In the following section we will present conditions under which an algorithm that minimizes error must come close to minimizing cost.

## 7. Some Results

In this section we give some results in the model introduced in Section 6. In Sections 7.1–7.3 we state three types of results which enable us to

cut the search space for optimal information and optimal algorithms: (1) Is *nonadaptive* information optimal? (2) Is a *linear* algorithm optimal? (3) Is a *special* type of information optimal?

We emphasize that we cut the search space by proving that the optimal information or algorithm must be of a particular, simple form. This should be contrasted with cutting the search space heuristically, as in artificial intelligence.

In Section 7.1 we give conditions under which nonadaptive linear information is optimal in the class of adaptive linear information. The significance of such results is that, when we seek optimal information, we can confine our attention to nonadaptive information. Because the structure of nonadaptive information is far simpler than that of adaptive information, this is most advantageous.

In Section 7.2 we give conditions under which a linear algorithm is optimal in the class of all algorithms; in Section 7.3 we give examples where function evaluations are optimal in the class of adaptive linear information.

We move to a different theme in Section 7.4. Intuitively, the smoother a problem, the lower the complexity for its solution. We discuss what has been established to date.

In the concluding section we quote a result from mathematical economics and a result regarding locally convergent iterations, each of which indicates that some $n^2$ scalar pieces of information are required to solve nonlinear equations in $n$ dimensions.

## 7.1 Can Adaption Help?

In Section 4 we discussed why it is of interest to know whether nonadaptive information is as powerful as adaptive information. We gave examples where adaption is much more powerful, as well as examples where adaption does not buy you anything. Here we present some general results regarding this question.

### *7.1.1 Some Concepts*

First we must define some concepts. We begin by defining nonadaptive and adaptive information for the case of linear information operators. If $N^{\text{non}}(f) = [L_1(f), \ldots, L_n(f)]$, where $L_1, \ldots, L_n$ are $n$ independently given linear functionals, then $N^{\text{non}}(f)$ is *nonadaptive linear information*, and we write $N^{\text{non}}$ to denote a nonadaptive information operator.

If $N^{\text{a}}(f) = [L_1(f), L_2(f; y_1), \ldots, L_n(f; y_1, \ldots, y_{n-1})]$, where $L_i$ depends linearly on its first argument and $y_i = L_i(f; y_1, \ldots, y_{i-1})$, then $N^{\text{a}}(f)$ is *adaptive linear information*. Note that adaptive information can use any function of any previously computed functionals to determine the next functional.

### 7.1.2 *Two Theorems*

**Theorem 7.1** For any linear problem and any adaptive linear information $N^a$ of cardinality $n$, there exists a nonadaptive information operator, $N^{non}$, of cardinality at most $n$, such that

$$r(N^a) \geq \tfrac{1}{2} r(N^{non}). \quad \blacksquare \tag{7.1}$$

[See Gal and Micchelli (1980) and Traub and Woźniakowski (1980a, Chapter 2, p. 48).]

We discuss the implications of this theorem. Because the radius of information, $r(N)$, is a sharp lower bound on the error of any algorithm using $N$, we measure the power of the information operator $N$ through $r(N)$. That is, if $r(N)$ is smaller, then $N$ is more powerful. The theorem tells us that adaption, at best, can reduce the radius by one-half and this is independent of $n$. For many linear problems we have the stronger result that

$$r(N^a) = r(N^{non}). \tag{7.2}$$

Theorem 7.1 is a special case of a more general result (see Traub *et al.* 1983, Chapter 4, Theorem 3.2).

A second theorem giving a condition under which nonadaptive information is just as powerful as adaptive information may be found in Traub *et al.* (1983, Chapter 4, Theorem 3.1). The fact that adaption does not help for binary search (discussed in Section 4.4) is a special case of this theorem.

## 7.2 Does a Linear Optimal Algorithm Exist?

As we discussed in Section 6.11.1, a linear optimal algorithm is always close to being an optimal complexity algorithm. Therefore, linear optimal algorithms are very desirable. If the problem is linear, must an optimal linear algorithm exist?

The answer is negative. C. A. Micchelli (private communication, 1978) has constructed an example of a linear problem for which no linear optimal algorithm exists (see Traub and Woźniakowski, 1980a, Chapter 3, Example 4.1). To get his counterexample, Micchelli uses a nonstandard norm. We know of no linear problem arising in practice which does not have a linear optimal algorithm.

We state three theorems which give positive results.

**Theorem 7.2** Assume the problem is linear and that $S$ is a real linear functional. Let $N$ be a real linear operator. Then there exists a linear optimal algorithm. $\blacksquare$

This result is due to Smolyak (1965). We state it here in our terminology.

Next we consider the case where $S$ is any linear operator. Then we have

**Theorem 7.3** Let the problem be linear. Let $N$ be a linear information operator of finite cardinality. Then there exists a linear algorithm $\varphi(N(f))$ such that $r(N) \leq e(\varphi(N(f))) \leq cr(N)$, where $c$ depends only on $N(\ker T)$. $\blacksquare$

**Theorem 7.4** In addition, let the range of $T$ be a Hilbert space and let $T(\ker N)$ be closed. Then $c = 1$ in the previous theorem and *$\varphi$ is a linear optimal algorithm.* $\blacksquare$

The proofs for the theorems may be found in Traub and Woźniakowski (1980a, Chapter 3, Theorems 4.1 and 4.2). These are constructive and indicate how the linear algorithms are obtained.

## 7.3 Is a Certain Type of Information Optimal?

We can sometimes cut our search space by proving that a certain type of information must be optimal. We provide two illustrations selected from recent research.

The first illustration is provided by the zero-finding problem already discussed in Section 4.1, for which we now discuss a further result. Recall that the problem is to find an $\varepsilon$-approximation to a zero of the nonlinear function $f$ where $f$ is continuous on $[0, 1]$ and $f(0) < 0$, $f(1) > 0$. We assumed that the type of information available is $n$ evaluations of $f$ and concluded that, relative to this type of information, bisection information (which is, of course, adaptive) is optimal. The radius of optimal information is $2^{-(n+1)}$ and the bisection algorithm is optimal.

Suppose now that we assume only that we can use adaptive linear information as defined in Section 7.1. What is the optimal information in this very large class?

In Traub and Woźniakowski (1980a, p. 170) we conjectured that the optimal adaptive linear information is just function evaluations. This conjecture was established by Sikorski (1982). Thus any algorithm using any linear adaptive information must have error of at least $2^{-(n+1)}$. Proving such a result is difficult, and Sikorski uses a very ingenious argument.

The second illustration is provided by the problem of approximating the inverse function, $f^{-1}$. Let $F$ be the set of all monotonically increasing functions on $[0, 1]$. We seek an $\varepsilon$-approximation to $f^{-1}$ in the supremum norm.

Wasilkowski (1982) proves that if $N^a$ is arbitrary adaptive linear information, then

$$r(N^a) \geq 1/(4(n + 1)).$$

Let

$$N^*(f) = [f(t_1), \ldots, f(t_n)], \qquad t_i = i/(n + 1).$$

Note that $N^*$ uses *only function values* at equispaced points and is nonadaptive. Yet

$$r(N^*) = 1/(2(n + 1)).$$

Thus nonadaptive function evaluations are almost optimal in the class of adaptive linear information.

It is interesting to contrast the inverse function and zero-finding results. Zero finding can be stated as evaluating $f^{-1}(0)$. Thus, it is a special case of computing $f^{-1}$. However, optimality results are in sharp contrast. For zero finding, adaptive information is exponentially stronger than nonadaptive information. For the function inverse, nonadaptive information is nearly optimal. Furthermore, the inverse function problem is exponentially harder than the zero-finding problem.

## 7.4 Do Smoother Problems Have Lower Complexity?

Intuitively, smoother problems have lower complexity. We quantify this for a particular problem and briefly indicate what has been established in general.

### *7.4.1 An Example*

In Section 2 we considered scalar integration for the class of integrands for which $|f'(t)| \leq 1$, $t \in [0, 1]$. It would be of interest to consider the integration problem for smoother functions.

More precisely, we seek an $\varepsilon$-approximation to $\int_0^1 f(t)\,dt$ using the information $N(f) = [f(t_1), \ldots, f(t_n)]$. Let $F$ be the set of periodic functions with period one such that $|f^{(r)}(t)| \leq 1$, $t \in [0, 1]$. [This definition of $F$ serves our present purpose; for a precise definition, see Traub and Woźniakowski (1980a, pp. 90, 109).]

This problem has been studied by Motornyj (1974) and we report his results using our terminology. The optimal information is

$$N^*(f) = \left[f\left(\frac{1}{2n}\right), \ldots, f\left(\frac{2i-1}{2n}\right), \ldots, f\left(\frac{2n-1}{2n}\right)\right]. \tag{7.3}$$

The radius of optimal information is

$$r(N^*) = K_r/(2\pi n)^r. \tag{7.4}$$

Here $K_r$ is the $r$th Favard constant defined as

$$K_r = 4/\pi \sum_{i=0}^{\infty} (-1)^{i(r+1)}/(2i + 1)^{r+1}. \tag{7.5}$$

It is known that $K_1 = \pi/2$, $K_2 = \pi^2/8$, $K_3 = \pi^3/24$, and that $1 = K_0 < K_2 < \cdots < 4/\pi < \cdots < K_3 < K_1 = \pi/2$. Thus the smallest number of integrand samples, $m$, such that we can compute an $\varepsilon$-approximation is

$$m = \left\lfloor\left(\frac{K_r}{(2\pi)^r}\frac{1}{\varepsilon}\right)^{1/r}\right\rfloor + 1. \tag{7.6}$$

Furthermore, the averaging algorithm

$$\varphi^*(N^*(f)) = \frac{1}{n}\sum_{i=1}^{n} f\left(\frac{2i-1}{2n}\right) \tag{7.7}$$

is an optimal algorithm using optimal information (relative to information of the form $N(f) = [f(t_1), \ldots, f(t_n)]$).

Assume each arithmetic operation costs unity and each function evaluation costs $c$. Then the $\varepsilon$-complexity, comp($\varepsilon$), is given by

$$\mathrm{comp}(\varepsilon) = (c + 1)\left(\left\lfloor\left(\frac{K_r}{(2\pi)^r}\frac{1}{\varepsilon}\right)^{1/r}\right\rfloor + 1\right) + a, \tag{7.8}$$

where $a = -1$ or $0$. Thus $\varphi^*(N^*(f))$ is an almost optimal complexity algorithm.

We discuss these results. Observe the Knuthian nature of Eqs. (7.4) and (7.5). We do not have just an order-of-magnitude result for the radius of optimal information; we know the constant.

The optimal information and the optimal algorithm using optimal information, given by Eqs. (7.3) and (7.7), are extraordinarily simple. Indeed, the optimal algorithm is just the averaging algorithm at $n$ equispaced points. We used this example precisely because the answers are so simple.

The approximation of integrals where the set of integrands is generated by a linear restriction operator $T$ always enjoys *nonadaptive* optimal information and *linear* optimal algorithms (as defined in Section 6.1.1). However, the formulas are not always simple (see, for example, Traub and Woźniakowski, 1980a, Chapter 6, Sect. 4).

Note how well we know the problem complexity. It has exactly one of two possible values an integer apart. The tight lower and upper complexity bounds are typical when there is a linear optimal algorithm. [See Traub and Woźniakowski (1980a, Chapter 3) for a general investigation of when a linear optimal algorithm exists.]

Observe that if $r = 0$, then Eq. (7.4) shows that the radius of optimal information is unity. This implies that no $\varepsilon$-approximation exists, for any $\varepsilon$ no greater than unity, for the class of integrands which are periodic, continuous, and bounded, no matter how large the number of function samples!

Finally, we want to discuss an important implication of Eq. (7.8). It shows quantitatively how $\varepsilon$-complexity decreases with the smoothness of the class of integrands which is measured by the parameter $r$. In particular, we see that complexity decreases as smoothness increases.

Observe that an argument based on the simple observation that the class of functions with smoothness $r + 1$ is contained in the class of functions with smoothness $r$ fails because the class of functions such that $|f^{(r+1)}| \le 1$ is not contained in the class for which $|f^{(r)}| \le 1$.

Does complexity generally decrease as smoothness increases? We discuss this in the next section.

### *7.4.2 An Open Question and a Partial Result*

It seems intuitive that more regular problems should be easier to solve and should therefore enjoy lower $\varepsilon$-complexity. Traub and Woźniakowski (1980a, p. 147) asked whether this is true in general.

A partial answer is provided by Werschulz (1982a). He answers the question affirmatively in the case that $S$ is linear and that regularity is measured by a Sobolev norm or seminorm.

### 7.5 An Example From Mathematical Economics

The examples we have used, such as integration, zero finding, and binary search, are drawn from numerical analysis and computer science. We want to provide the reader with an example of an information result from mathematical economics. We do not define the concepts used in this section, but refer the interested reader to the papers cited below.

Saari and Simon (1978) consider how much information is required for a price mechanism to converge to an economic equilibrium. They formulate this as obtaining a solution of a certain nonlinear system in $n$ dimensions. Usually, the nonlinear systems are solved by an iteration which uses $f$ and $f'$ (or its approximation) at each step. In $n$ dimensions this requires the evaluation of the $n$ components of $f$ and the $n^2$ components of $f'$. The economic interpretation of knowing $f'$ is that it is necessary to know how changes in demand for the $j$th commodity affect changes in the price of the $k$th commodity, where $j$ and $k$ range from 1 to $n$. We quote Saari and Simon (1978):

> For practical problems, this is a staggering amount of information. Consequently, the natural question is whether there exist effective mechanisms with a more modest dependence on information content. ... We investigate this question in this paper, and our results show that the information required cannot be relaxed by any significant amount.

Price mechanisms studied by Saari and Simon correspond to iterations which are not necessarily locally convergent. The problem of what information is required by locally convergent iterations for the solution of nonlinear equations has been studied in many papers [see Traub and Woźniakowski (1980a, Part B)]. Here we report a result of Traub and Woźniakowski (1976, Lemma 4.3 and Theorem 4.2) which is of the same flavor as the result of Saari and Simon.

We wish to approximate a simple zero of $f$ where $f$ is a nonlinear operator, $f\colon D \subset B_1 \to B_2$, where $B_1$ and $B_2$ are Banach spaces of dimension $n$, $1 \le n \le +\infty$. We have

**Theorem 10.1** Any locally convergent one-point iteration that uses linear information requires at least the evaluation of $f$ and $f'$. ■

Thus, for both effective price mechanisms and locally convergent one-point iterations, roughly $n^2$ scalar pieces of information must be used. This gives a measure of the inherent difficulty of solving nonlinear equations.

## 8. Other Models

So far we have made three major model assumptions: (1) uncertainty is measured by a norm, (2) the model is worst case, and (3) information is exact. We briefly discuss models where these assumptions are not made.

## 8.1 Measuring Uncertainty without a Norm

The normed setting we have used so far is appropriate for some contin uous problems such as integration, approximation, and computing a extremum of a continuous function. There are, however, simple continu ous problems which cannot be formulated in the normed model presente in Section 6. For example, find $x$ such that $|f(x)| < \varepsilon$. [See Werschul (1982b) for a discussion of this example.] Furthermore, there are discret problems where uncertainty is not measured by a norm.

Traub *et al.* (1983) show how uncertainty can be measured without norm. The model uses abstract sets and the solution operator is assume to satisfy two "nonrestrictive" axioms. This formulation permits a syn thesis between the study of discrete and continuous problems.

## 8.2 Average-Case Models

The model we have discussed so far is worst case. That is, we guaran tee an $\varepsilon$-approximation for every element of $F$. Worst-case models ar sometimes too pessimistic and we therefore study the average case.

Average-case analysis is far more difficult than worst-case analysis This is because integration with respect to a measure is far harder t analyze than the supremum operation. This is especially the case becaus the set $F$ usually lies in an infinite-dimensional space and the analysi therefore requires rather heavy mathematical machinery such as measur theory in infinite-dimensional spaces.

A general study of the optimal reduction of uncertainty for an averag case model was initiated by Traub *et al.* (1981). We briefly summarize th results.

The setting is linear problems on a finite-dimensional space equippe with a weighted Lebesgue measure. An average-case model is specifie and general notions of radius of information, optimal algorithm, and opt mal information are introduced. Among the results obtained are (1) th same algorithm is optimal in the worst and average cases, (2) the sam information is optimal in the worst and average cases, and (3) adaptiv information is not more powerful than nonadaptive information.

We discuss these results. The first two conclusions are favorable to th user because the same algorithm with the same information minimiz both the worst and average error. This is the spline algorithm (see Sectio 6.4.2). As we saw earlier, adaptive information does not help for the wor case. Many researchers believe this is true only in the worst-case settin The last conclusion states the counterintuitive result that adaption do not help even on the average.

How does the average radius of information, $r^{\text{avg}}(N)$, compare with t worst-case radius of information $r(N)$? It is possible that $r^{\text{avg}}(N) \ll r(N$

However, for "reasonable" measures and "typical" problems the two are comparable (see Traub *et al.*, 1981, Sect. 6).

The infinite-dimensional case is under investigation. Wasilkowski and Woźniakowski (1982) obtain optimal algorithms and optimal information for linear problems in infinite-dimensional Hilbert spaces. They show that for any measure, a spline algorithm is optimal among linear algorithms. The spline algorithm is defined in terms of the covariance operator of the measure. If the measure is "orthogonally invariant," then the spline algorithm is optimal among all algorithms. Orthogonal invariance means that the measure of a set is invariant under certain linear mappings. Examples of orthogonally invariant measures include Gaussian measures and, for the finite-dimensional case, weighted Lebesgue measures. Under the assumption of orthogonal invariance of the measure, Woźniakowski (1982) has shown that adaption does not help even on the average.

## 8.3 Approximate Information

In Section 3 we listed three reasons why problems cannot be solved exactly. In our discussions we have confined ourselves to just one of these causes of uncertainty, partial information.

Approximate information is a very important cause of uncertainty. Optimal algorithms and optimal information for approximate information are studied in Traub *et al.* (1983).

In many application areas the information is stochastic. Of particular importance is the average case with stochastic information, and this will be one of the focuses of future research.

## 8.4 Asymptotic Models

We motivate our interest in asymptotic models with the following example. We seek to approximate $\int_0^1 f(t)\,dt$ knowing the information

$$N(f) = [f(t_1), \ldots, f(t_n)].$$

Let $F$ be the set of integrands such that $f''(t)$ is continuous on $[0, 1]$. Unlike the approach developed in Section 2, we do not assume a bound on $f''(t)$.

It is not difficult to show that $r(N) = \infty$ for any finite $n$. Therefore it is impossible to compute an $\varepsilon$-approximation for any finite $\varepsilon$.

Despite the infinite radius of information, we can proceed as follows. Let

$$N_n(f) = \left[f(0), f\left(\frac{1}{n-1}\right), f\left(\frac{2}{n-1}\right), \ldots, f(1)\right].$$

We choose as our algorithm the composite trapezoidal rule

$$\varphi_n(N_n(f)) = \tfrac{1}{2}h(f(0) + f(1)) + h\sum_{i=1}^{n-2} f\left(\frac{i}{n-1}\right).$$

Then the error is given by

$$e_n(\varphi, f) = \int_0^1 f(t)\,dt - \varphi_n(N_n(f)) = f''(\theta_n)/(12(n-1)^2), \qquad \theta_n \in (0, 1)$$

This error is of the form

$$e_n(\varphi, f) = h(f, n)g(n),$$

where $h(f, n) = f''(\theta_n)$, $g(n) = 1/(12(n-1)^2)$. Observe that if we var) over all integrands in $F$, then $h(f, n)$ is unbounded for any $n$. This is wl the radius of information is infinite in the worst-case model. However, f any fixed $f$, the sequence $\{\varphi_n(N_n(f))\}$ converges to $\int_0^1 f(t)\,dt$ and the spe of convergence is proportional to $n^{-2}$.

The way that this algorithm is used in practice is that the sequence approximations is terminated with some finite $n$, which depends on according to some "termination criteria." The user is not sure that : $\varepsilon$-approximation is computed whenever the termination criterion is sati fied. Thus, the user is gambling that for his $f$ he will be lucky.

If the user knows $|f''(t)| \le L$, $t \in [0, 1]$, then he can guarantee : $\varepsilon$-approximation by choosing $n$ such that

$$L/(12(n-1)^2) \le \varepsilon.$$

This is, however, equivalent to the worst-case model.

Traub and Woźniakowski (1980a, Chapter 10, Sect. 5) present : asymptotic model which is an abstraction of the example present here. Some interesting results on the asymptotic case have been obtain by J. M. Trojan (private communication, 1982). The current state of o knowledge indicates that for linear problems, the asymptotic case is "bad" as the worst case! Further research should be done on this mod which is very important in practice. The average asymptotic model shou also be investigated.

## 9. Comments Regarding the Information-Centered Approach

Our information-centered approach has stimulated many commen and questions. Some of these have been very perceptive and have infl enced our work. We have also received some comments which we rega as missing the point, but feel they deserve a thoughtful response.

One frequent comment is that in real-world problems, the user does not know how to choose $F$. This is true; dealing with this case is extremely important and will be the area of much future work (see Section 10.2.1).

It has been pointed out that in many applications, the information is not exactly known. Furthermore, the available information may come from various sources, be of varying quality, and may even be contradictory. This is true, and the models will be extended to deal with this.

One scientist complained that it took as much work to specialize the general theory to his application as to just solve his problem from scratch. This addresses the fundamental intellectual claims of science. Most scientists would agree that scientific progress is made by showing that diverse phenomena can be uniformly explained. A sound general theory exhibits structure which is invisible to someone looking at a particular problem. It also suggests entirely new questions and approaches. For example, *some* diseases could be cured before the germ theory of disease transformed medicine, but many more could be cured after that understanding was achieved.

Because of our emphasis on information, people have asked whether information theory contains our results. It does not. Indeed, Shannon and Weaver called what they did the mathematical theory of communication, which is very descriptive of their work. One of us (JFT) benefitted from a discussion with Robert Gallager. Our conclusion was that although there is some overlap between our work and information theory, the subject matter and methodology are very different. Traub *et al.* (1983, Sect. 6.9) show how an example from information theory can be formulated in our framework.

We have been told that what we want to achieve with the notion of radius of information has been accomplished by the notion of Shannon entropy. The notion of Shannon entropy has been found useful in many applications, but it is not the same as our fundamental invariant, radius of information. Traub *et al.* (1983, Sect. 6.9) give an example where Shannon entropy is shown to be related to our notion of average cardinality number.

Many people have expressed surprise that adaption does not help for linear problems. There is widespread belief that adaption helps for problems such as quadrature. Furthermore, there is much current research on algorithms using adaptive information.

Why are people's beliefs that adaption helps at such variance with our theorems that adaption does not help for either the worst or average case? There are a number of possible explanations. First, our results may not be applicable. For example, one of our results is that adaption does not help for linear problems. If a problem is linear, then $F$ is convex and balanced (see Traub and Woźniakowski (1980a, p. 32). If someone wishes to solve a problem where $S$ is linear but $F$ is not convex or not balanced, then the results obtained so far do not apply and adaption may help.

Furthermore, the evidence cited in support of adaptive quadrature is obtained by tests run for particular integrands. As Einstein noted in a very different context, "It is the theory which decides what we can observe." Once the theory has shown that it is the structure of a class of integrands which determines whether adaption helps, this can be verified by testing.

Our theory has been called too hard. Many theories which were initially considered difficult are no longer so regarded. We believe the information-centered approach is not difficult, just new. Indeed, the information-centered approach permits vast simplifications.

We have been asked if we are serious about algorithms. We are very serious. We want to create real algorithms used to solve real problems on real computers. The comment refers to the fact that we do not work within a formal model of computation, such as the Turing-machine model. We refer the reader to Section 3.1.3 for a discussion of the relation between our notions of idealized and realizable algorithms and other notions of algorithms.

One scientist commented that the optimal algorithm is never wanted. We believe this comment was made in the mistaken belief that an optimal algorithm need be complicated. Although that may be true for some areas, we hope we have convinced the reader that the optimal algorithm is often simple. Indeed, one of the contributions of this theory is to ascertain when the optimal algorithm is simple, and Sections 7.1–7.3 are devoted to this topic.

Finally, a scientist said to us, "I never solve problems for a class of matrices, just for a single matrix. Therefore your notion of the class $F$ is irrelevant to me." The person who made this comment was referring to his experience with matrix eigenvalue problems, but our response applies generally. We believe it is fairly uncommon for someone to be interested in only one matrix (or, more generally, in a single $f$), although there are circumstances when this is so. After all, we expect to solve these problems on a computer. Our program will have to work not just for one matrix, but for a variety of matrices.

## 10. Where Are We and Where Are We Going?

We discuss the history and nature of ε-complexity and indicate some of the directions for future work.

### 10.1 ε-Complexity

We refer to the information-centered approach for dealing optimally with uncertainty as ε-complexity. We will give a brief history of the field, discuss whether ε-complexity is a new discipline, and discuss alternate names.

#### *10.1.1 A Very Brief History*

We indicate the work which we believe initiated research in analytic complexity, iterative complexity, and ε-complexity.

The pioneering work on analytic complexity was done by Kiefer, Sard, and Nikolskij around 1950. Regrettably, Kiefer and Sard both passed away recently.

Kiefer (1953) showed that if function evaluations are used, then Fibonacci search is optimal in searching for the maximum of a unimodal function. Professor Kiefer has informed us that this work was done as a master's thesis at the Massachusetts Institute of Technology in 1948, but was published only later with the encouragement of J. Wolfowitz.

Sard (1949) studied optimal algorithms for quadrature which use function evaluations at fixed points and discussed extending his results to the approximation of linear functionals. Independently, Nikolskij (1950) posed the same problem and permitted the points of evaluation to be optimally chosen. Sard and Nikolskij *assumed* that the algorithms were linear.

Iteration complexity had its inception in the work of Traub (1961, 1964). Iterative algorithms are classified by the information they use. Theorems are obtained and conjectures are proposed on the maximal order of iterative algorithms for solving scalar nonlinear equations. Such maximal order results are needed to obtain lower bounds on complexity.

The work on analytic and iterative complexity was brought together for the first time in the research monograph of Traub and Woźniakowski (1980a). This monograph includes a brief history and an annotated bibliography with over 300 of the most important "core" papers and books.

The general study of ε-complexity has been initiated by Traub *et al.* (1983).

#### *10.1.2 Is ε-Complexity a New Discipline?*

Only time will provide an answer to this question. The program of ε-complexity is ambitious: a general theory for dealing optimally with uncertainty. Many of the concepts and points of view are novel. We regard the information-centered point of view as fundamental and powerful.

How is $\varepsilon$-complexity related to other disciplines? It has been heavily influenced by computational complexity, the mathematical theory of approximation, applied mathematics, and numerical analysis. Because much of $\varepsilon$-complexity deals with infinite-dimensional problems, the techniques and language of functional analysis are heavily used. In the average-case setting, the tools of measure theory in infinite-dimensional spaces are utilized.

As we have repeatedly stressed, most mathematically formulated problems can be solved only with uncertainty. Therefore, applications can be found everywhere. A partial list of applications that have been studied and plans for the investigation of new applications may be found in Section 10.2.2.

### 10.1.3 What's in a Name?

The name we have suggested is *ε-complexity*. Other names could also be used. Some people prefer to use *optimal algorithm theory* (see, for example, Belforte *et al.*, 1982). We often refer to the information-centered approach, which suggests that *information-centered theory* should be the name of the field.

An ideal name would be information theory. That, unfortunately, has been used to denote something else.

## 10.2 Future Work

Although much has been accomplished, a vast amount remains to be done. There are numerous open problems ranging from very theoretical to applied. We indicate just a few of these in the following.

### 10.2.1 Future Theoretical Research

We briefly indicate some directions for future theoretical research.

*Average-Case Models.* This has been discussed in some detail in Section 8.2. We list it here without further comment.

*F Not Known.* In numerous applications, the user does not know the class of problem elements. We give a simple example. Let $f$ represent the temperature distribution of the atmosphere as a function of the distance above the earth's surface. The smoothness of the class would depend on whether there was a temperature inversion.

Dealing with the case that $F$ is not known is essential if we are to extend our methods to the solution of certain important problem areas.

*N Not Known.* In numerous applications, $N$ is not known exactly. For example, $N$ might be stochastic. We plan to extend our models to deal with this case.

*Optimal Information and Optimal Algorithms for Particular S,F,N.* We have provided concepts and general theorems. Applying our methodology to particular instances of $S$, $F$, and $N$, can be technically difficult (see Traub and Woźniakowski, 1980a, for numerous examples). This must be done if the theory is to be widely applied.

### 10.2.2 Future Applications Research

Numerous applications have already been studied. Areas for which results are reported in Traub and Woźniakowski (1980a) and Traub *et al.* (1983) are integration, interpolation, large linear systems, linear functionals, linear partial differential equations, nonlinear equations, optimal recovery, optimization, and polynomial zeros. We have also illustrated our theory by examples from algebraic coding theory, binary search, continuous binary search, database security, decision theory, and information theory.

These applications are for problems arising in scientific computation and computer science. There are numerous areas and disciplines dealing with uncertainty which we plan to investigate using the information-centered approach. We give some very brief examples in the following.

*Remote Sensing.* Numerous important applications involve remote sensing. Examples include seismology, remote sensing of the atmosphere, and tomography (see, for example, Twomey, 1977). We hope to answer questions such as: What are the best measurements? What is the best way to combine these measurements? What is the minimal number of measurements to guarantee a good answer in either a worst or average case?

One of the remote-sensing areas we plan to investigate is seismology. In particular, we will apply our techniques to the Backus–Gilbert theory [for which, see Backus and Gilbert (1970) and Burridge (1974–1975)].

*Estimation, Prediction, Control.* A group at the Politecnico di Torino has been using the information-centered approach to solve problems in estimation (Belforte *et al.*, 1982). They also report excellent results in prediction (G. Milanese, private communication, 1982). We anticipate that this will be a very active area for future research.

*Distributed Computation.* As we observed earlier, even problems capable of exact solution on a uniprocessor will be solved only under uncertainty in the distributed environments of the future because complete, exact information on the current state of the distributed system will not be available.

We plan to model distributed systems. In particular, we wish to investigate distributed databases.

*Signal Recovery and Processing.* This very important area seems well suited to our techniques.

*Statistics.* This is a huge discipline which deals with uncertainty. We wish to understand how it is related to our work. We are hopeful that we can pose and solve problems of interest to statisticians.

A first step has been taken by Kadane and Wasilkowski (1983), who investigate relations between our average-case model and optimal decisions and experiments in Bayesian statistics.

## ACKNOWLEDGMENTS

It gives us pleasure to acknowledge our debts. We want to thank Z. Galil, R. M. Karp, E. Packel, K. Sikorski, S. Stolfo, G. W. Wasilkowski, and A. G. Werschulz for reading the manuscript. We benefitted greatly by their comments. We are indebted to K. March for typing the manuscript and to K. McLaughlin and B. Borske for preparing the revised manuscript.

This research was supported in part by the National Science Foundation under Grant MCS-7823676. J. F. Traub wishes to acknowledge the support of the Advanced Research Projects Agency under contract N00039-82-C-0427.

## REFERENCES

For additional references, see Traub and Woźniakowski (1980a, Part C), who provide an annotated bibliography of some 325 of the most important papers and books.

Backus, G., and Gilbert, F. (1970). Uniqueness in the inversion of inaccurate gross earth data. *Philos. Soc. Trans. London* **266**, 1970.

Bakhvalov, N. S. (1971). On the optimality of linear methods for operator approximation in convex classes of functions. *Zh. Vychisl. Mat. Mat. Fiz.* **11**, 1014–1018, *USSR Comput. Math. Math. Phys. (Engl. Transl.)* **11**, 244–249 (1971).

Barnhill, R. E. (1968). Asymptotic properties of minimum norm and optimal quadratures. *Numer. Math.* **12**, 389–393.

Belforte, G., Milanese, M., and Tempo, R. (1982). "Optimal Algorithm Theory and Estimation with Unknown but Bounded Error." Dipartimento Di Automatica-Informatica Report, Politecnico Di Torino, Italy.

Bojanov, B. D. (1974). Best quadrature formula for a certain class of analytic functions. *Zastosow. Mat.* **14**, 441–447.

Burridge, R. (1974–1975). "Some Mathematical Topics in Seismology," Sci. Rep. Courant Institute of Mathematics, New York University, New York.

Gal, S., and Micchelli, C. A. (1980). Optimal sequential and non-sequential procedures for evaluating a functional. *Appl. Anal.* **10**, 105–120.

Garey, M. R., and Johnson, D. S. (1979). "Computers and Intractability." Freeman, San Francisco, California.

Kacewicz, B. (1976a). An integral-interpolation iterative method for the solution of scalar equations. *Numer. Math.* **26**, 355–365.

Kacewicz, B. (1976b). The use of integrals in the solution of nonlinear equations in N dimensions. *In* "Analytic Computational Complexity" (J. F. Traub, ed.), pp. 127–141. Academic Press, New York.

Kacewicz, B. (1979). Integrands with a kernel in the solution of nonlinear equations in N dimensions. *J. Assoc. Comput. Mach.* **26**, 239–249.

Kadane, J., and Wasilkowski, G. W. (1983). Average case $\varepsilon$-complexity: A Bayesian view. *Valencia Int. Meet. Bayesian Stat., 2nd, 1983.*

Karmarkar, N., and Karp, R. M. (1982). An efficient approximation scheme for the one-dimensional bin-packing problem. *23rd Annu. Symp. Found. Comput. Sci.* pp. 312–320.

Kiefer, J. (1953). Sequential minimax search for a maximum. *Proc. Am. Math. Soc.* **4**, 502–505.

Kowalski, M. A., Werschulz, A. G., and Woźniakowski, H. (1983). "Is Gauss Quadrature Optimal for Analytic Functions?" Rep. Division of Science and Mathematics, Fordham University and Dept. of Computer Science, Columbia University, New York.

Kuhn, H. W., Wang, Z., and Xu, S. (1983). "On the Cost of Computing Roots of Polynomials," Rep. Dept. of Mathematics, Princeton University, Princeton, New Jersey.

Larkin, F. M. (1970). Optimal approximation in Hilbert spaces with reproducing kernel functions. *Math. Comput.* **24**, 911–921.

McCorduck, P. (1979). "Machines Who Think." Freeman, San Francisco, California.

Micchelli, C. A., and Rivlin, T. J. (1977). A survey of optimal recovery. *In* "Optimal Estimation in Approximation Theory" (C. A. Micchelli and T. J. Rivlin, eds.), pp. 1–54. Plenum, New York.

Motornyj, V. P. (1974). On the best quadrature formulae of the form $\Sigma_{i=1}^{n} p_i f(x_i)$ for some classes of periodic differentiable functions. *Dokl. Akad. Nauk SSSR, Ser. Math.* **38**, 583–614.

Murota, K. (1982). Global convergence of a modified Newton iteration for algebraic equations. *SIAM J. Numer. Anal.* **19**, 793–799.

Nikolskij, S. M. (1950). On the problem of approximation estimate by quadrature formulae. *Usp. Mat. Nauk* **5**, 165–177.

Pinkus, A. (1975). Asymptotic minimum norm quadrature formulae. *Numer. Math.* **24**, 163–175.

Rabin, M. O. (1976). Probabilistic algorithms. *In* "Algorithms and Complexity: New Directions and Recent Results" (J. F. Traub ed.), pp. 21–39. Academic Press, New York.

Ralston, A., and Rabinowitz, P. (1978). "A First Course in Numerical Analysis." McGraw-Hill, New York.

Rivest, R. L., Meyer, A. R., Kleitman, D. J., Winklmann, K., and Spencer, J. (1980). Coping with errors in binary search procedures. *J. Comput. Syst. Sci.* **20**, 396–404.

Saari, D. G., and Simon, C. P. (1978). Effective price mechanisms. *Econometrica* **46**, 1097–1125.

Sard, A. (1949). Best approximate integration formulas; best approximation formulas. *Am. J. Math.* **71**, 80–91.

Schönhage, A. (1982). "The Fundamental Theorem of Algebra in Terms of Computational Complexity," Rep. Mathematisches Institut, Universität Tübingen.

Shub, M., and Smale, S. (1982a). "Computational Complexity On the Geometry of Polynomials and a Theory of Cost: Part I," Rep. Center for Pure and Applied Mathematics, University of California, Berkeley.

Shub, M., and Smale, S. (1982b). On the average cost of solving polynomial equations. *Proc. Rio Conf. Dyn. Syst., 1982.*

Sikorski, K. (1982). Bisection is optimal. *Numer. Math.* **40**, 111–117.

Sikorski, K. (1983). "Optimal Solution of Nonlinear Equations Satisfying a Lipschitz Condition," Rep. Dept. of Computer Science, Columbia University, New York. (To be published in *Numer. Math.*)

Smale, S. (1981). The fundamental theorem of algebra and complexity theory. *Bull. Am. Math. Soc.* **4**, 1–36.

Smolyak, S. A. (1965). On optimal restoration of functions and functionals of them. Candidate Dissertation, Moscow State University (in Russian).

Solovay, R., and Strassen, V. (1977). A fast Monte-Carlo test for primality. *SIAM J. Comput.* **6**, 84–85; erratum, **7**, 118 (1978).

Sukharev, A. G. (1976). Optimal search for a zero of function satisfying Lipschitz's condition. *Zh. Vychisl. Mat. Mat. Fiz.* **16**, 20–30; Optimal search for the roots of a function satisfying a Lipschitz condition. *USSR Comput. Math. Math. Phys. (Engl. Transl.)* **16**, 17–26 (1976).

Traub, J. F. (1961). On functional iteration and the calculation of roots. *Prepr. Pap., Natl. ACM Conf., 16th,* 1961, Sess. 5A-1, pp. 1–4.

Traub, J. F. (1964). "Iterative Methods for the Solution of Equations." Prentice-Hall, Engelwood Cliffs, New Jersey. (Reissued 1982. Chelsea, Bronx, New York.)

Traub, J. F. (1974). Parallel algorithms and parallel computational complexity. *Inf. Process.* **74**, 685–687.

Traub, J. F. (1978). The influence of algorithms and heuristics on mathematics, science, and education. *Proc. Anniv. Symp., I.R.I.A., 10th, 1978.*

Traub, J. F., and Woźniakowski, H. (1976). Optimal linear information for the solution of nonlinear equations. *In* "Algorithms and Complexity: New Directions and Recent Results" (J. F. Traub, ed.), pp. 103–119. Academic Press, New York.

Traub, J. F., and Woźniakowski, H. (1980a). "A General Theory of Optimal Algorithms." Academic Press, New York.

Traub, J. F., and Woźniakowski, H. (1980b). "On the Optimal Solution of Large Linear Systems," Rep. Dept. of Computer Science, Columbia University, 1980. (To be published in *J. Assoc. Comput. Mach.*)

Traub, J. F., Wasilkowski, G. W., and Woźniakowski, H. (1981). Average case optimality for linear problems. *Theor. Comput. Sci.* (to be published).

Traub, J. F., Wasilkowski, G. W., and Woźniakowski, H. (1983). "Information, Uncertainty, Complexity." Addison-Wesley, Reading, Massachusetts.

Twomey, S. (1977). "Introduction to the Mathematics of Inversion in Remote Sensing and Indirect Measurements." Elsevier, Amsterdam.

Wasilkowski, G. W. (1982). Inverse function problem. Rep. Dept. of Computer Science, Columbia University, New York. (To be published in *J. Inf. Process. Cybernetics—EIK.*)

Wasilkowski, G. W., and Woźniakowski, H. (1982). "Average Case Optimal Algorithms in Hilbert Spaces," Rep. Dept. of Computer Science, Columbia University, New York.

Werschulz, A. G. (1982a). Does increased regularity lower complexity. Rep. Division of Science and Mathematics, Fordham University and Dept. of Computer Science, Columbia University. (To be published in *Math. Comput.*)

Werschulz, A. G. (1982b). Measuring uncertainty without a norm. Rep. Division of Science and Mathematics, Fordham University and Dept. of Computer Science, Columbia University. (To be published in *Aequationes Math.*)

Werschulz, A. G. (1983). "Counterexamples in Optimal Quadrature," Rep. Division of Science and Mathematics, Fordham University and Dept. of Computer Science, Columbia University, New York.

Woźniakowski, H. (1982). "Can Adaption Help On the Average?" Rep. Dept. of Computer Science, Columbia University, New York.