1974

The Algorithm Selection Problem III - Approximation Theory Machinery

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
74-130

THE ALGORITHM SELECTION PROBLEM III --

APPROXIMATION THEORY MACHINERY

John R. Rice

Computer Science Department
Purdue University
West Lafayette, Indiana 47907

December, 1974

CSD-TR 130

THE ALGORITHM SELECTION PROBLEM III --

APPROXIMATION THEORY MACHINERY

FORMULATION AND STRUCTURE OF THE APPROXIMATION PROBLEM.

The purpose of this report is to analyze the algorithm selection problem within the framework of approximation theory. We will see that the principle questions of this problem can be formulated within the traditional framework of approximation theory. Even so, the answers to many of the questions require the development of very novel techniques and theories of approximation. More specifically then, our purpose is to systematically examine these questions, to indicate what light can be shed on them from the existing theory of approximation and to point out the new problems in approximation theory that are raised by the algorithm selection problem. Needless to say, we do not propose to solve these new problems in this report. The principle questions are divided into four groups:

1. Norms and approximation forms
2. Degree of convergence, complexity and robustness
3. Existence, uniqueness and characterization
4. Computation

The question of computation is deferred to another report.

For convenience and completeness, we summarize the abstract formulation of the algorithm selection problem as presented in [6]. We present the model which includes selection based on features, but which does not include variable performance measures. This latter aspect of the algorithm selection problem has interesting consequences which we mention at some points, but the main theme of the formulation is not affected by it.

The model is described by the schematic diagram in Figure 1.

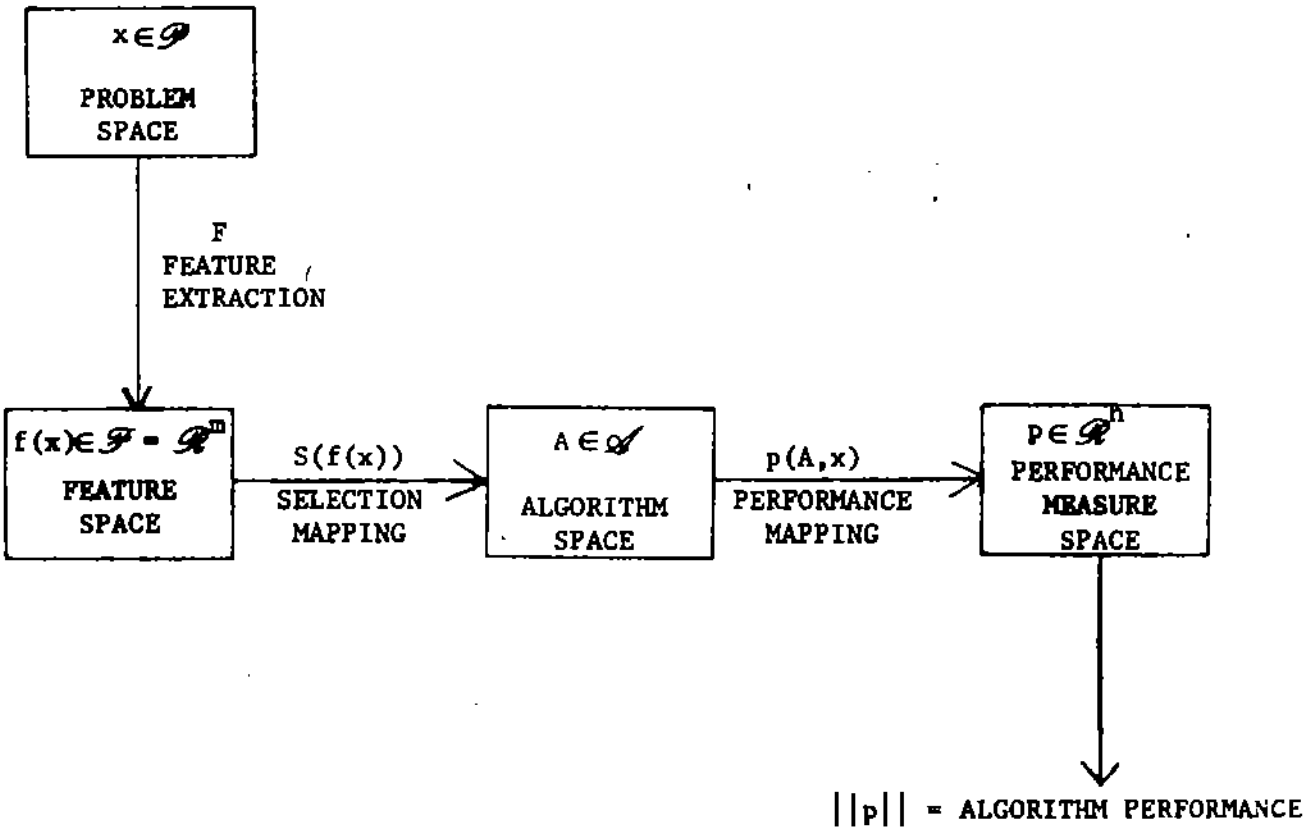


Figure 1. Schematic diagram of the abstract model for the algorithm selection problem.

Definitions for the abstract model in Figure 1.

$\mathcal{P}$ = Problem space or collection

x = Member of $\mathcal{P}$, problem to be solved

$\mathcal{F}$ = Feature space identified with $\mathcal{R}^m$ here to suggest it is simpler and of lower dimension than $\mathcal{P}$.

F = Mapping from $\mathcal{P}$ to $\mathcal{F}$ which associates features with problems.

$\mathcal{A}$ = Algorithm space or collection

A = Member of $\mathcal{A}$, algorithm applicable to problems from $\mathcal{P}$

S = Mapping from $\mathcal{P}$ to $\mathcal{A}$

$\mathcal{R}^n$ = n-dimensional real vector space of performance measures

p = Mapping from $\mathscr{A} \times \mathscr{P}$ to $\mathscr{R}^n$ determining performance measures

$||\ ||$ = Norm on $\mathscr{R}^n$ providing one number to evaluate an algorithm's performance on a particular problem.

Note that the selection mapping depends only on the features f(x) but yet the performance mapping still depends on the problem x. The introduction of features may be viewed as a way to systematize the introduction of problem subclasses in the model.

1. NORMS AND APPROXIMATION FORMS.

The question of norms enters in the final step from the algorithm performance space $\mathscr{R}^n$ to the single number which represents the algorithm performance. Since we have a norm on a standard n-dimensional vector space, the possibilities are well-known. The most common are of the form

$$||p|| = \left[\sum_{i=1}^{N} w_i p_i^r \right]^{1/r}$$

with typical values of r being 1, 2 or infinity (for the Tchebycheff or minimax norm). However, the nature of the selection problem is such that we can anticipate using non-standard norms. The reason is that the performance measures tend to include essentially incomparable variables, e.g.

p_1 = computer time used (measured in seconds)

p_2 = computer memory used (measured in words)

p_3 = complexity of setting up the computer run (measured in hours required by the programmer)

A plausible norm to use in such a context might be

$$||p|| = p_1 + \alpha(p_2)p_2 + \beta(p_3)p_3$$

where

$$\alpha(p_2) = \begin{cases} 0 & \text{for } p_2 \leq 10{,}000 \\ 10^{-5} & \text{for } 10{,}000 \leq p_2 \leq 20{,}000 \\ 2*10^{-5} & \text{for } 20{,}000 \leq p_2 \leq 30{,}000 \\ 7p_2*10^{-9} & \text{for } p_2 \geq 30{,}000 \end{cases}$$

and

$$\beta(p_3) = \begin{cases} 0 & \text{for } p_3 \leq .5 \\ 2 & \text{for } .5 \leq p_3 \leq 2 \\ p_3 & \text{for } p_3 \geq 2 \end{cases}$$

There are two observations, one positive and one negative, about such complicated norms that can be made based on current experience in approximation. The negative one is that they do complicate the theory sometimes and, more often, make the computations substantially more difficult. The positive one is that the choice of norm is normally a secondary effect compared to the choice of approximation form. That is, if one has a good choice of approximation form, one obtains a good approximation for any reasonable norm. This implies that one can, within reason, modify the norm used so as to simplify the analysis or computations. A significant corollary to this last observation is that one cannot compensate for a poor choice of approximation form by computing power or technical skill in analysis.

We now turn to the crucial question of approximation forms which we group into five classes:

a. discrete

b. linear

c. piecewise

d. general non-linear:
   standard mathematical
   separable
   abstract

e. tree and algorithm forms.

In order to discuss these choices, we need to formulate more precisely the standard idea of approximation form as it currently exists in approximation theory. The form is to be used for the selection mapping $S(f(x)): \mathcal{F} \to \mathcal{A}$ and we visualize a parameter (or coefficient) space $\mathcal{C}$ plus a particular form of the mapping. To show explicitly the dependence of S on the coefficients, we may write $S(f(x),c)$ at times. Specific examples of the five classes of approximation forms are given below:

a. Discrete $S(f(x),1)$ = computer program #1

$S(f(x),2)$ = computer program #2

$S(f(x),3)$ = computer program #3

b. Linear $S(f(x),c) = c_1+c_2f_1+c_3f_1^2+c_4(f_1f_2)^2+c_5(f_2-f_3)^3+c_6/f_3$

Note that linear refers to the dependence on the coefficients c_i and not the features f_j.

c. Piecewise linear

$$S(f(x),c) = c_1+c_2f_1+c_3f_2+c_4f_1f_2+c_5/f_2 \quad \text{for } |f_1+f_2| \geq 2$$

$$= c_6+c_7f_1+c_8f_2+c_9f_1f_2+c_{10}f_1^2 \quad \text{for } |f_1+f_2| \leq 2 \text{ and } f_1 \leq f_2$$

$$= c_{11}+c_{12}f_1+c_{13}f_2+c_{14}\frac{f_1-f_2}{1+f_1+f_2} \quad \text{for } |f_1+f_2| \leq 2 \text{ and } f_1 \geq f_2$$

We see that the feature space is subdivided into pieces and $S(f(x),c)$ is defined linearly on each of the pieces.

d. Non-linear, Standard forms:

Rational: $$S(f(x),c) = \frac{c_1+c_2f_1+c_3f_2}{c_4+c_5(f_1+f_2)^2}$$

Exponential: $$S(f(x),c) = c_1e^{-c_2f_1}+c_3e^{-c_4f_2}+c_5e^{-c_6(f_1+f_2)^2}$$

Spline: $$S(f(x),c) = c_1+c_2f_1+c_3f_2+c_5(f_1-c_4)_+ +c_7(f_2-c_6)_+$$

$$\text{where } (f-c)_+ = \begin{cases} 0 & \text{for } f \leq c \\ f-c & \text{for } f \geq c \end{cases}$$

This is an example of variable pieces. If c_4 and c_6 were constants, then this would be piecewise linear.

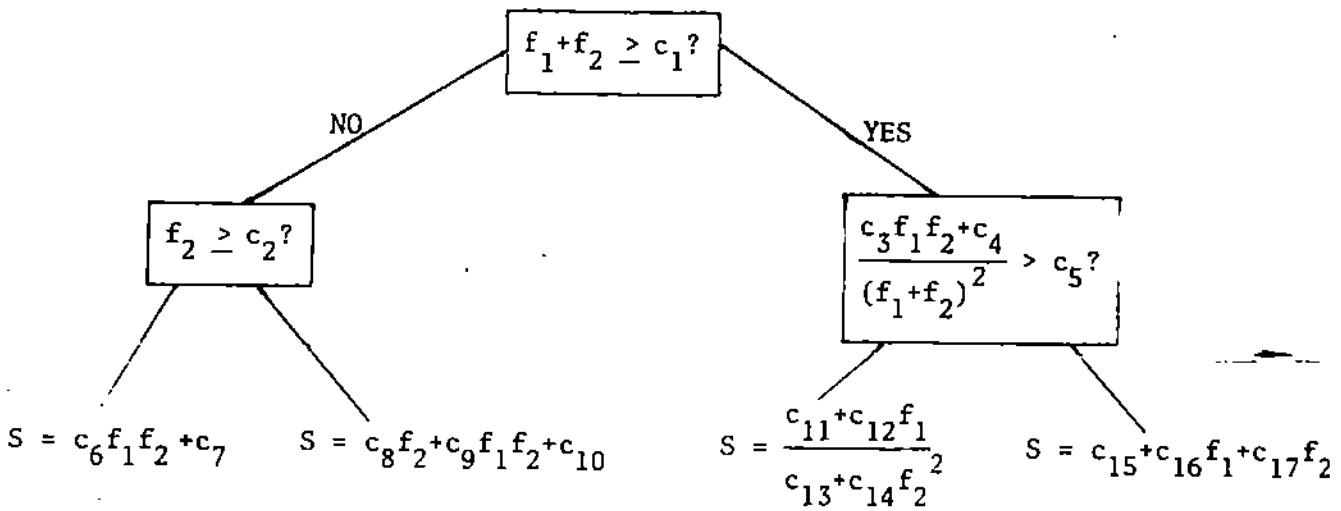
Non-linear, Separable:

$$S(f(x),c) = g_1(f_1,c_1,c_2)+g_2(f_2,c_3,c_4)+g_3(f_3,c_5,c_6)$$

The effects of the different features (and their associated coefficients) are completely independent of one another. The exponential example given just above is also of this form.

The abstract non-linear form is an arbitrary function of the features $f(x)$ and the coefficients c.

c. Tree and algorithm forms:




```
      FUNCTION  S(F,C)
      SUM=0
      DO 20  K=1, C(1)
20       SUM=SUM+C(K+1)*F(K)
      IF( F(1) > C(1) ) THEN SUM = SUM/( C(C(1)+1))+1 )
      PROD=1.
      IF( F(C(1)+2) < (C(C(1)+1)+F(2))/F(3) ) THEN PROD=F(1)*F(2)
      DO 40 K=1, C(C(1)+3)
40       PROD = ( F(K)+C(K))*PROD+C(C(1)+K+3 )
      S = C(1)*SUM+C(2)*PROD+C( C(1)+C( C(1)+3)+1 )*F(1)
```

The main thrust of approximation theory is for the case where the coefficients c are used to parameterize a relatively simple form (i.e. such as the linear, piecewise linear and non-linear forms). The distinguishing

characteristic of these cases is that the set of approximation forms can (at least locally) be identified with a manifold in some ordinary finite dimensional space. The approximation theory machinery is then used to obtain the best coefficients or parameters (again, at least locally) from this manifold.

One thus may conclude that there are three distinct situations as far as the applicability of approximation theory machinery. The first and most favorable situation is for the linear, piecewise linear and nonlinear approximation forms. Here the machinery may be applied essentially as it currently exists. This does not mean that all of these cases are already solved and all one has to do is to "copy" the solutions from somewhere. Rather, it means that these are the kinds of problems the machinery is supposed to handle and, if it is currently inadequate in some specific instance, it needs to be extended in the direction it is already headed.

The second situation is for the tree and algorithm forms. Here it seems that a major change in emphasis is required. The exact nature of the new machinery is certainly unclear and no doubt there are hidden difficulties which are not apparent from a casual inspection. However, it seems plausible that the general spirit of the approach and techniques may well be similar to that already existing. For example, the piecewise linear forms may be visualized as one of the simplest of the tree forms. The development and analysis for the piecewise forms (even for variable pieces) has progressed fairly smoothly over the past 10 years and the resulting body of results is very much of the flavor of the previously established linear and specialized non-linear theories. There were (and still are), of course, some difficult questions for the piecewise linear, but the prospects do not appear to be too bad for developing a useful body of approximation theory machinery for the tree and algorithm forms.

The third and least favorable situation is for the discrete forms. The standard mathematical approach results in stating that the problem is trivial in this case. One ascertains the best selection mapping by a finite enumeration. Unfortunately, the enumeration may well be over very large sets. Even 1000 elements (algorithms) are completely unmanageable in most instances and it is easy to find problems where there are millions of algorithms to be considered (at least in some abstract sense). It is not at all clear how algorithm selection procedures are to evolve in this situation and the development of such procedures is one of the foremost open questions in this entire area of study.

We close this section by repeating a fundamental observation: The most important single part of the successful solution of an approximation problem is the appropriate choice of the approximation form. Approximation theory machinery comes into play after this choice is made. Thus it is essential to have insight into both the problem and algorithm spaces and into the possible forms one might choose for the selection mappings.

2. CLASSIFICATION OF PROBLEMS, DEGREE OF CONVERGENCE, COMPLEXITY AND ROBUSTNESS.

This section has two distinct parts. First, we introduce the concept of classifying problems and second, we introduce three other concepts which are intimately related to ways of classifying problems. These three concepts -- degree of convergence, complexity and robustness -- are important for evaluating the overall value of various approximation forms for the algorithm selection problem.

2.1 Classification of Problems.

An important approach to obtaining insight into the nature of the problem space is to partition it into particular classes of problems. Ideally there is a representative member or property of each

class which is especially relevant to the selection of algorithms. The exact nature of the classification depends, of course, essentially on the specific problem space. Some typical examples include:

a. Numerical Quadrature: Compute $If = \int_a^b f(x)dx$

Class 1: Those f(x) which have continuous curvature

Class 2: Those f(x) which have 5 or fewer oscillations in [a,b]

Class 3: Those f(x) which are analytic

Mathematics has a highly developed classification system for functions (integrands f(x)) which provides literally dozens of classes relevant to numerical integration algorithms.

b. Scheduling a CPU in an operating system

Class 1: Batch processing multiprogramming, 1 CPU, 2 I/O channels and 1 disk

Class 2: Time sharing, 2 CPU's, 50 terminals

Class 3: Time sharing with a batch processing background, 2 CPU's, 50 terminals, saturation loading

We see that the problem classification has many independent variables giving a high dimensional problem space.

c. Scene analysis.

Class 1: One connected object, a line drawing with 50 or fewer lines

Class 2: Up to 10 objects, each composed of from 1 to 10 rectangles, triangles or circular arcs

Class 3: Unknown number of separated objects of one of 4 types; distinguishing properties are color, texture, size, position and orientation

It is easy to visualize thousands of particular types of scenes to analyze.

The idea of problem classification is simple, but important. Most algorithms are developed for a particular class of problems even though the class is never explicitly defined. Thus the performance of algorithms is unlikely to be understood without some idea of the problem class associated with their development.

It is particularly common to attempt a classification system which goes from easy to hard. Thus one visualizes a nested set of problems where the innermost set consists of very easy problems and the largest set consists of very hard ones. Unfortunately, it is not always easy to make such a classification (at least in a reasonable way) for complex problem spaces. One is lacking the insight to know in all circumstances just what makes a problem hard or easy.

2.2 Degree of Convergence. The idea of degree of convergence comes from considering a sequence of approximation forms and asking: How much better do these forms do as one goes further out in the sequence? A standard example would be for computing Log x by polynomials of degree $0,1,2,3,\ldots,N,\ldots$. We assume that for each approximation from the sequence we have the best coefficients possible.

In the present context, our ultimate objective is to choose the best algorithm for every problem. If we let $A^*(x)$ be the best algorithm for problem x and let $A_N(x)$ be the algorithm chosen by the best coefficients for the N-th approximation form, then the question is: How does

$$\epsilon_N(x) = ||p(A^*(x))|| - ||p(A_N(x))||$$

behave as N gets big? Does it go to zero for every x? Suppose we set

$$\epsilon_N = \max_{x \in \mathcal{P}} \epsilon_N(x) \quad ,$$

does ϵ_N go to zero fast, slow or at all? The answer to these questions is called the degree of convergence for the problem space $\mathcal{P}$ and the sequence of approximation forms.

In standard mathematical situations this idea is well-developed and the degree of convergence is known for many cases. In the standard case the problem is to evaluate a function f(x) and the best algorithm A*(x) is taken to be the exact value of f(x). The measure of performance of an algorithm A that produces an approximation a(x) is taken to be $|f(x) - a(x)|$. Thus, for computing sin(x) for $x \in \mathcal{P} = [0,\pi/2]$ we know that polynomial forms give $\epsilon_N \sim KN^{-N}$ for some constant K. In this case ϵ_N goes to zero extremely fast. If one replaces sin(x) by ABS(x-1), then $\epsilon_N \sim KN^{-1}$ which is not very fast at all.

The analogy with approximately evaluating a function can be carried further, but theoretical information about the degree of convergence is limited to "mathematical" functions. That is, functions defined in a mathematical context where one knows a variety of properties. We can say, however, that really fast convergence using simple forms (i.e. polynomials and similar linear forms) requires that the function involved be very well-behaved. By well-behaved we mean smooth (no jumps or discontinuities of any kind, including in derivatives) and of a consistent global nature (i.e. if it oscillates one place, it oscillates everywhere; if it is flat one place, it is flat everywhere). A large proportion (at least 50%) of the "functions" that arise naturally in the real world are not well-behaved in this sense.

2.3 Complexity. A fashionable idea related to degree of convergence is complexity. Thus the complexity of a function is some intrinsic measure of how hard it is to compute the function. The idea extends directly to solving problems by noting that solving a problem is equivalent to computing the value of the function which gives the solution of the problem.

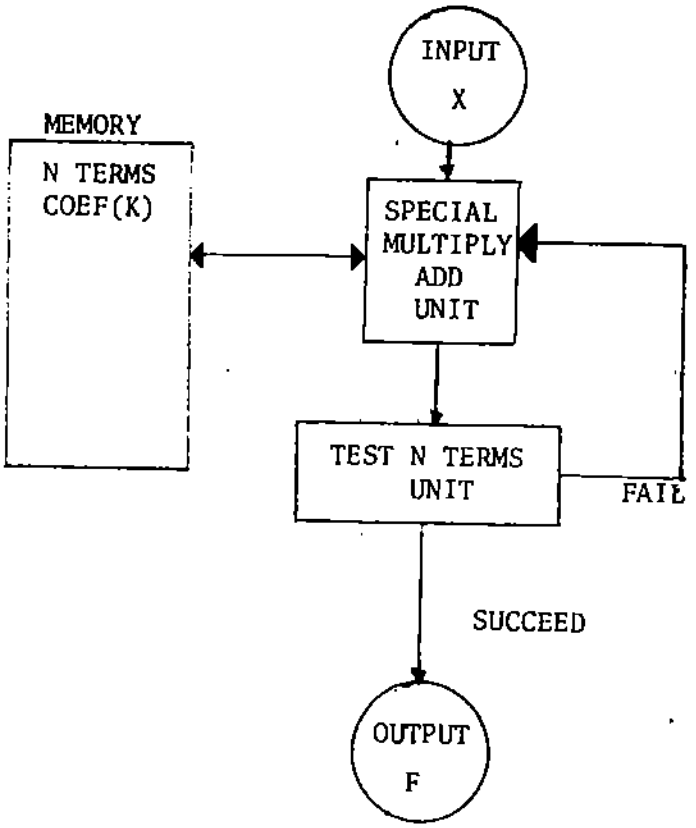
In actually measuring complexity, one does several things:

A. Introduce some measure of the work involved in a computation. Typical examples are: number of arithmetic operations, number of multiplies, execution time of a real program on a particular real computer, length of Fortran program needed, number of steps in a Turing machine computation.

B. Assume that one considers the most efficient scheme. There is no limit on how badly one can evaluate a function, complexity is measured with methods of optimal efficiency.

C. Restrict the kinds of steps in the algorithms used for the computation. For example, polynomial approximation excludes division so 1/x may be difficult to compute, but if division were allowed then this would be a very easy function. Similarly $|x-.5|$ is very easy if ABS is allowed or if a test and branch operation is allowed.

A uniform way to impose the above conditions on the complexity question is to say that the function is to be evaluated by a particular machine or, essentially equivalent, by one of a particular class of programs for a general purpose machine. We illustrate this approach for polynomials:



(a) Polynomial evaluation machine

```
      READ X
      F=COEF(0)
      DO 10 K=1,NTERMS
        F=F+X*COEF(K)
10    CONTINUE
      PRINT F
```

(b) Polynomial evaluation program

Figure 2. Polynomial evaluation via machine or program. The special MULTIPLY/ADD unit and TEST unit of the machine are such that they can only and automatically do execute the program on the right.

The advantage of the idea of complexity over that of the degree of convergence is that much greater generality is achieved. Degree of convergence can be normally interpreted as complexity using a very specialized machine. For example, a machine which can only add and multiply but which can be programmed to do this in more or less arbitrary sequence and with arbitrary operands is considerably more versatile than the polynomial evaluation machine shown in Figure 2. It could, for example, evaluate the function X^{1024} in 10 operations rather than the 1024 required for the

strictly limited polynomial evaluation machine. This added generality also makes it possible to place the standard mathematical approximation forms into the same framework as the piecewise forms and the tree or algorithm forms. One merely adds or changes a piece of "hardware" on the machine.

The disadvantage of the idea of complexity is that its generality makes it very difficult to obtain specific results. Current research is very intensive and yet concentrated on rather simple problems as seen in Table 1.

Computation	Work or Complexity of		
	Standard Method	Optimal	Best Known
Add two N-digit integers	N	N	N
Multiply two N-digit integers	N^2	?	$N \log^2 N$
Evaluate polynomial degree N	N multiplies	?	[N/2]+2 multiplies
Median of list of length N	N log N	N	N
Multiply two N x N matrices	N^3	?	$N^{2.7---}$

Table 1. Summary of complexity results for some common computations

These problems are orders of magnitude simpler than the typical situation that arises in the algorithm selection problem. Thus there is little hope for the near future that we will obtain optimal algorithms for most of these problems (except possibly from very limited subclasses of algorithms).

In spite of the low probability of obtaining precise results about complexity in the algorithm selection problem, there are three good reasons to consider the idea. First, it provides the proper framework within which to contemplate the problem. Second, the results for simple problems show that the standard ways of doing things are often not optimal or even anywhere close to best. Third, the high degree of complexity in "real" problems indicates that simple-minded approaches are unlikely to do well and even sophisticated approaches will often fall very short of optimal. Indeed, it

is likely that further theoretical developments in the area will indicate that it is essentially impossible to obtain the optimal algorithms for many real problems.

2.4 Robustness. Robustness is a technically precise term in Statistics which relates the quality of statistical estimates in extreme situations. Thus an estimation procedure is robust of its quality degrades gracefully as the situation becomes more and more extreme. We do not attempt to define this concept precisely here but it is quite useful in considering the selection of algorithms. It is a common phenomena for algorithms to do very well on a certain class of "easy" problems and to do increasingly less well as one moves away from these easy problems. A robust algorithm then is one whose performance degrades slowly as one moves away from the problems for which it was designed. Since the problem space is so large and so poorly understood in many real situations, this quality can be extremely important. There is a reasonable probability that one will face a problem with a completely unforeseen combination of attributes which invalidate some of the "working assumptions" used in the development of the algorithm. The worst situation is, of course, an algorithm which fails completely and quietly as soon as one moves away from the ideal problems.

Consider the simple example of estimating the wealth of the "typical" student in a classroom. One has three candidate algorithms for the estimate: the average wealth, the medium wealth and the mid-range wealth. In a "normal" situation these algorithms produce similar estimates, any one of which is satisfactory. A difficulty occurs with Howard Hughes III (wealth of $625 million) or John D. Rockefeller V (wealth of $398 million). The mid-range now produces ridiculous estimates like $200 or $300 million and the average is not much better with estimates like $20 or $30 million. The

median estimate is, however, essentially unaffected by the pressure of such a wealthy person and thus is a very robust algorithm for this problem. While the average is more robust than the mid-range, it is not very satisfactory in extreme situations.

While robustness of an algorithm is a quality of a different and more nebulous nature than things like efficiency, it is nevertheless a very desireable one. It is related to concepts like complexity and degree of convergence because it too varies over the problem space and a precise definition of robustness would involve some classification of the problems into "easy, medium, hard, harder" classes. Note that robustness is relevant to even a single algorithm while the other two concepts intrinsically involve classes or sequences of algorithms.

Finally we note that robustness is frequently difficult to identify or measure. In some situations one can achieve robustness with very simple algorithms. In others it seems that robustness requires a complex algorithm that has numerous tests for special situations and cases.

3. SURVEY OF APPROXIMATION FORM ATTRIBUTES.

This section presents a survey of the general attributes of five important types of approximation forms. Of necessity we speak in generalities and thus there is a real danger that a casual reader is misled. The statements we make about attributes apply "usually" or "commonly". Realistic specific situations exist which exhibit behaviors exactly opposite the usual one. We have already noted that the most crucial decision in the algorithm selection problem is that of the approximation form. Ideally, this process goes as follows: one is intimately familiar with the problem space and with a large variety of approximation forms. One weighs the various advantages and disadvantages of the forms as they interact with the special features of

the problem space. Perhaps some simple experimentation is made. Finally a choice of form for the algorithm selection mapping is made which achieves a good balance with the overall objectives.

Thus one can visualize this section as a primer on the choice of approximation forms. Unfortunately, it is only an elementary primer and there is no substitute for detailed experience with a variety of real situations.

3.1 Discrete Forms. One might tend to dismiss this case as "degenerate". After all, if one is merely to select the best one of three or eleven algorithms, there seems to be little need for any elaborate machinery about approximation forms. We do not imply that how to identify the best will be easy, rather we say that concepts like complexity, degree of convergence, etc. will not play a role. This reaction is appropriate in many cases. However, sometimes there are some very interesting and challenging features of these forms.

The principle feature is that the finite number of algorithm is either in fact or in concept a very large set. Even though we may have selected just three algorithms, we often visualize that these are representative samples from a very much larger set. Recall from the discussion of the numerical quadrature problem that there may well be tens of millions of algorithms of even a rather restricted nature. Thus in the mind's eye there is almost a continuum of algorithms even though we may in fact be examining only three of them. One of the major weaknesses of modern mathematical machinery is in its ability to handle problems involving very large finite sets. The emphasis has been on developing tools to handle problems with infinite sets (e.g. the continuum) and one frequently draws a complete blank when faced with a finite set of, say, 10^{123} elements.

We are really saying that the proper way to consider discrete forms is

as a discretization of a continuum. One then applies some intuitive ideas about continuous forms (such as presented later in this section) and hopefully obtains satisfactory results.

Unfortunately, we cannot continue a meaningful discussion here along these lines because we have no knowledge of the possible continuum behind the discrete set.

We conclude by recalling that robustness is a property of individual algorithms and thus immediately relevant to discrete forms. It could be evaluated for each algorithm in the discrete set. However, if the set is large, then this is impractical. In this latter case, one probably must attempt to transfer information about robustness from some underlying continuum.

3.2 Linear Forms. There are so many obviously nice things about linear forms that we might tend to concentrate too much on what is bad about them; or we might tend to ignore anything bad about them. Some of these nice things are:

They are simple and efficient to use.

They are the easiest to analyze (by far).

They are easy to understand and visualize intuitively.

They are often extremely successful in achieving good approximation. These observations imply that we should give these forms first consideration and that we should try other things only after we are fairly sure that some linear form does not suffice.

The bad thing about these forms comes from the following experimentally observed fact: Many real world processes are not linear or anywhere close to being linear. In particular, we would like to emphasize that: Most of the world processes are not a linear combination of simple, standard mathematical

entities. Since these facts are experimental rather than theoretical, we cannot prove them here. Indeed, certain theoretical results (e.g. the Weirstrass Theorem) are frequently used to support just the opposite conclusion (e.g. one can use polynomials for everything).

Let us illustrate the situation by a trivial example: Our problem space $\mathcal{P}$ has just one attribute of consequence and we call it x (which identifies the problem with a real number that measure this attribute). Our algorithm space $\mathcal{A}$ is likewise simple with no attribute which we call A. Suppose that x and A range between 0 andl and suppose the best algorithm is for A = .27 if $x < .41$, A = .82 if $.41 \leq x \leq .8$ and is A = .73 for $x > .8$. The best or optimal algorithm selection mapping is then as shown in Figure 3 (left).

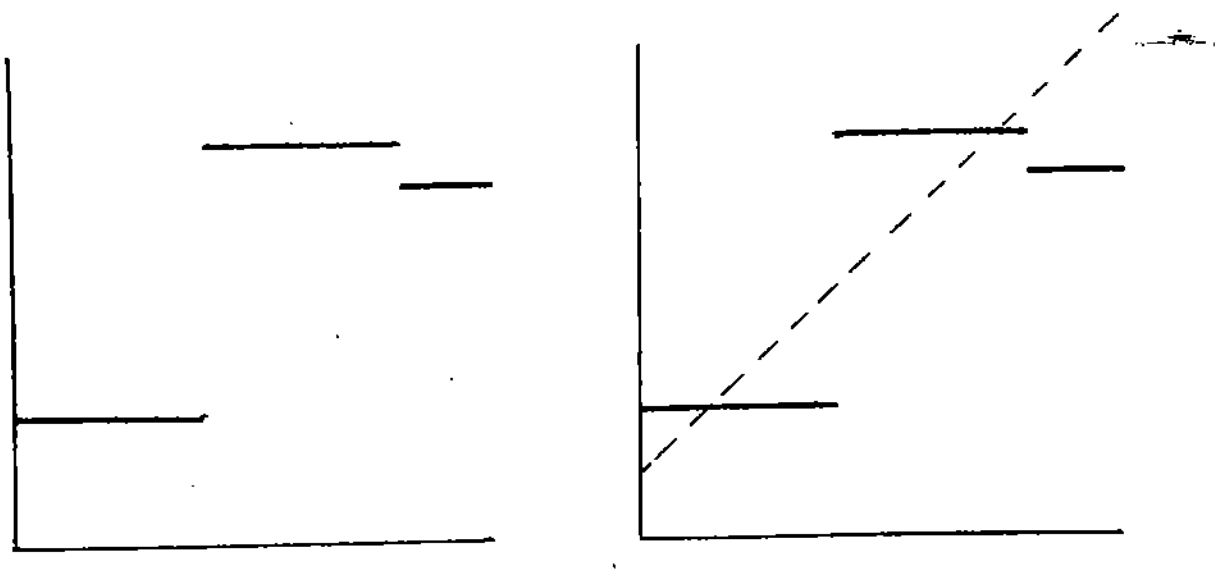


Figure 3. (left) Graphical representation of the optimal algorithm selection mapping for a simplified example. (right) The optimal plus the best linear algorithm selection mapping.

If we attempt a linear form then we would have $A = \alpha + \beta x$ where α and β are coefficients to be determined. The optimal values α^* and β^* for these coefficients give a mapping shown as the dashed line in Figure 3. This mapping is clearly not very close to being optimal.

Once this completely linear form is recognized as inadequate, one then tends to proceed on to something more flexible. A natural idea is to use polynomials, e.g.

$$A = \alpha_1+\alpha_2 x+\alpha_3 x^2+\alpha_4 x^3+\ldots+\alpha_N x^{N-1} .$$

If one carries this out for N=4 (cubic polynomials) and N=20, one can expect results such as shown in Figure 4 (provided one has been very careful in the computations). It is hard to argue that either one of these selection mappings is a good approximation to the optimal one. Note that in both cases that the polynomials are truncated at either A=0 or at A=1 in order to avoid obtaining non-existant algorithms for some values of x.

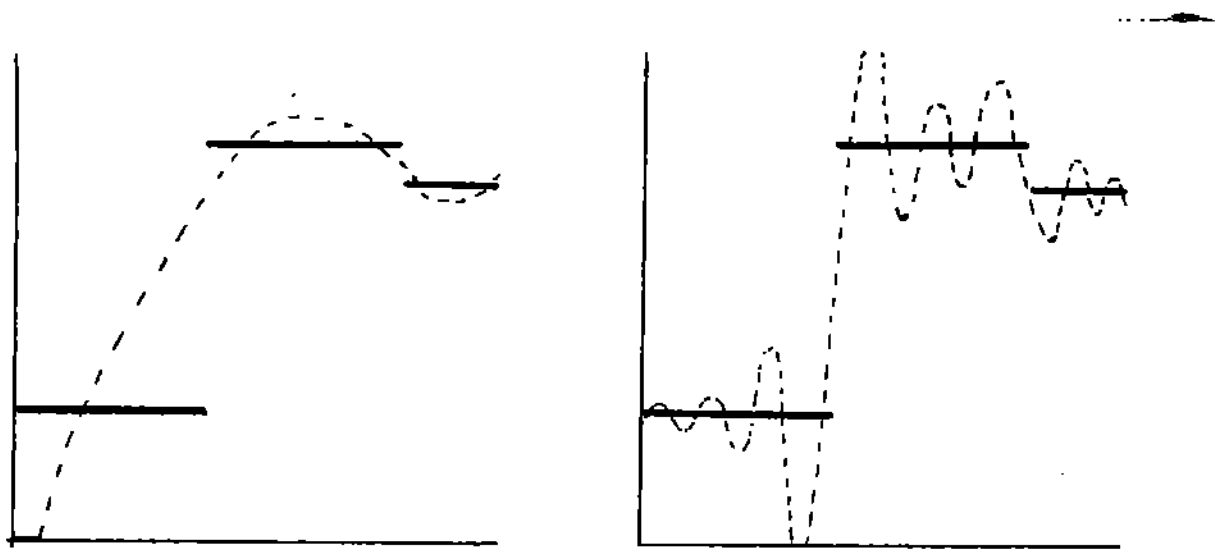


Figure 4. Graphical representation of the optimal plus the best cubic (left) and best 20th degree (right) polynomial selection mappings.

Can one hope to do much better by choosing something besides polynomials? One frequently sees Fourier Series (sines and cosines), exponentials, Bessel functions, etc., etc. None of these give noticeably better approximations. There is, of course, a way to obtain excellent results by a linear form: $A = \alpha + \beta\omega(x)$. One merely chooses $\omega(x)$ to be the optimal selection mapping and then we find $\alpha^*=0$ and $\beta^*=1$ gives a perfect approximation.

This last observation shows the impossibility of making universal judgements about linear forms. If you choose linear combinations of the right things, then the linear forms can do very well indeed. In practice though, one is usually limited to just a few possibilities and one has very little information about the optimal mapping. Note that a typical real problem has 5 to 15 dimensions in each of x and A variables. One is not likely to hit upon the optimal mapping as one of the things to include in the linear mapping.

We now attempt to motivate the above conclusions from the point of view of degree of convergence and complexity. For standard mathematical situations there are numerous results about how the error of polynomial and similar functions behave as the number of terms increases. The phenomena of Figure 4 shows very slow convergence, or poor degree of convergence. Of course, if the optimal selection mapping has a jump as seen in Figure 3, there will always be a large error at that jump. We also see that the large error at the jump induces large errors everywhere.

If the optimal mapping is continuous but has breaks in the slope, then it is known that the degree of convergence for N-terms is like 1/N. That means that if 1 term gives a unit error, then 10 terms give a .1 error, 100 terms give .01 error, etc. This is a very bad situation even for the

simplest case of a 1-dimensional problem space. Higher dimensions compound this difficulty enormously. Thus if several of these breaks occur in a K-dimensional problem space, then the error behaves like $1/\sqrt[K]{N}$ where N is again the number of terms. For K=5, if 1 term gives a unit error then we would expect to need about 32 terms for 1/2 unit error, 1000 terms for 1/4 unit error and 100,000 for .1 error. For K=10, the corresponding numbers are 1,000, 1,000,000 and 10^{10}, respectively for errors of 1/2, 1/4 and .1. Clearly polynomials and related functions are hopeless in such situations except for the crudest of approximations to the optimal selection mapping.

How often can one expect the problem space to produce selection mappings with these troublesome properties? Experimental evidence with phenomena from physics and engineering problems indicates more than 50% of these functions are unsuitable for polynomials and other standard linear mathematical forms. This includes Fourier Series which are currently widely used in engineering situations where they cannot possibly give accurate results. There is an intuitive reason why one should expect this. Many physical phenomena have several domains where different factors completely dominate the behavior. As one goes from one domain to another there is a kind of discontinuity in behavior even if there is no sharp break in the slope. These discontinuities affect the degree of convergence directly and, expecially for low accuracies, lead to a very excessive number of turns being required. Recall that polynomials, Fourier Series, etc. have the property that their global behavior is completely determined by their behavior on an arbitrarily small domain. This property is not present in many real world situations and is another intuitive reason for doubting the general applicability of the standard mathematical forms.

One must admit that the above arguments are taken from simplified and

specialized situations. The extrapolation to all kinds of algorithm selection problems is very tenuous indeed. Yet, we conjecture that things get worse rather than better as one gets away from these situations into a broad range of real world problems.

3.3 Piecewise Linear Forms. In simple terms, we break up the problem domain into pieces and use separate linear forms on each piece. The motivation is to circumvent the difficulties described in the preceding discussion. In many cases the most crucial step is to determine the appropriate pieces and yet these forms assume that they are fixed and given by some a priori process. In these cases we in fact have a two stage process: the first is an intuitive-hopefully realistic, partition of the problem domain into separate pieces. The second is the application of mathematical techniques to obtain the best coefficients for each of the linear pieces. Note that there are often some interconnections between the pieces (for example, broken lines are piecewise linear functions of one variable which join up continuously) which give rise to mathematical problems which are non-standard but still linear (and hence usually tractible).

It is difficult to draw general conclusions about this approach because of the vagueness of the process for determining the pieces. Indeed if the pieces are poorly chosen or too big, then one can have all the difficulties mentioned with the traditional linear forms. On the other hand, there are the following hopeful facts about this approach:

(i) Sometimes one does have good enough intuition to determine the pieces so that a very significant improvement is made. Sometimes only a very few pieces are required for this improvement to happen.

(ii) Sometimes the problem domain is small enough that one can

break it up into more or less equal pieces that are small enough to obtain good results and yet still not obtain an intractible number of pieces.

(iii) There are theoretical results (admittedly again from the narrow context of approximating functions of one variable) which indicate that if the best selection of pieces is made, then there are fantastic improvements possible. That is so that the degree of convergence may change from something like $\frac{1}{\sqrt{N}}$ to $\frac{1}{N^3}$ where N is the number of coefficients involved. If one piece gives accuracy 1, then these convergence rates indicate that about 10,000 or 5, respectively, coefficients are needed to give an accuracy of .01 in the determination of the best selection mapping. Such an improvement obviously changes the entire nature of the problem.

We conclude that piecewise linear forms merit separate consideration for three reasons:

A. They are non-standard in mathematical/scientific analysis and might be overlooked if lumped into a larger class.

B. Once the difficult determination of pieces is made, then more or less standard machinery can be used in further analysis and computation.

C. They have been very useful in a variety of difficult situations and, while they are not a panecea, there is reason to believe that they will continue to be so.

3.4 General Nonlinear Forms. It is not very profitable to discuss such forms in the abstract. These forms include everything, including the best possible selection mapping, and thus one can do perfectly with them. Thus we must

really be concerned with various specific classes of nonlinear forms. The literature on approximation theory contains a considerable development of a variety of such classes. A partial list of these with simple examples is:

Rational Functions: $$\frac{c_1+c_2x+c_3x^2}{c_4+c_5x}$$

Exponential/Trigonometric Functions: $c_1e^{c_2x} + c_3e^{c_4x} + c_5\cos(c_6x)$

Piecewise Polynomials: $$\begin{cases} c_1+c_2x+c_3x^2 & \text{for } -\infty < x \leq c_4 \\ c_5+c_6x+c_7x^2+c_8x^3 & \text{for } c_4 \leq x \leq c_9 \\ c_{10}+c_{11}x+c_{12}x^2 & \text{for } c_9 \leq x < \infty \end{cases}$$

Unisolvent Functions: The set of all conic sections in the place.

Varisolvent Functions: A general class of non-linear forms which includes the rationals, exponentials, etc.

There are several general statements that one can make about these forms:

(i) A considerable (or even very extensive) amount of analysis has been made of the theory of approximations

(ii) In those cases where degree of convergence results are available (e.g. piecewise polynomials and rationals), they imply that these special forms are much more capable of approximating a wide variety of behaviors. For example, both rationals and piecewise polynomials can do very well at approximating a jump discontinuity or a behavior like $\sqrt{x}$ or $1/\sqrt{x}$.

(iii) The computational effort required to obtain best (or even very good) coefficients of these forms can be substantial. The development of computational methods is more difficult than for linear forms. However, it is practical to carry out these computations in

a variety of cases.

Thus one expects (and observes) these forms to be useful in a variety of situations. The key to success is to analyze one's particular situation sufficiently to obtain general knowledge of the required behavior of the selection mapping. One then chooses that nonlinear form which possesses this behavior and for which one can handle the analytical and computational difficulties.

In conclusion, the determination of the proper non-linear form is still somewhat of an art and there is no algorithm for making the choice. On the other hand, the degree of convergence and complexity results for rational functions and piecewise polynomials show that they have great flexibility and are likely to do well in most situations. Doing well might not be good enough. In real problems the dimensionalities are high and needing five coefficients per dimension implies that 5^n coefficients are required for an n-diemnsional feature (or problem) space. With n=2 this is a modest 25 coefficients, but n=10 would then require almost 10 million coefficients. This 10 million may be considered doing well compared to the 6 decillion coefficients of another approach, but in either case one cannot use the forms.

3.5 Tree and Algorithm Forms. These forms are most intriguing because they promise so much and have the mystery of the unknown. Perhaps it is a case of the grass being greener on the other side of the fence. These forms may have difficulties and disadvantages which are not apparent now but which may limit their usefulness much more than one hopes.

The primary basis for their promise is their flexibility and potential for complexity. They certainly should complement the more traditional mathematical forms. Their flexibility and complexity might be the limitation on their application. Computational methods for good coefficients of

traditional forms have taken many years to develop and even now can be quite demanding. It may well be that the computation of good coefficients will severely restrict the usefulness of these forms for many years.

The piecewise linear forms are an example of a simple tree form and their success bodes well for other cases. Computational techniques and theoretical analysis for these forms is progressing steadily and we can look for them to enter into the "standard and routine" category before long. This development should serve as a useful guide for other simple tree and algorithmic forms. Still, we are very far removed from the time when we can select as our approximation form a 72 line Fortran program and then compute the best "coefficient values" (Fortran statements) for a particular application.

In summary, we have very little hard information about these forms, but they appear to hold great promise and to provide a great challenge for theoreticians and practicioners.

3.6 An Error to Avoid. Occasionally one observes the following situation develop:

(i) A real world problem is considered

(ii) A crude model is made of it. This model perhaps has some undertermined coefficients or is to be manipulated to obtain predictions about the real world problem's solution.

(iii) A huge effort is spent in obtaining accurate coefficients or predictions based on the model.

In the specific instance at hand, the real world problem is the algorithm selection mapping, the model is the approximation form selected and the effort is in determining the coefficients of this form. The error that one can make is in believing that finding the best coefficients of the selection mapping will result in good selections. In many cases there is no reason

to believe that the best coefficients will give good selections. One is particularly susceptible to making this error when using simple linear forms for the selection mapping. One may refer to Figure 3 for an illustration for this situation.

4. EXISTENCE, UNIQUENESS AND CHARACTERIZATION.

This section presents an intuitive summary of three principal topics of approximation theory. The algorithm selection problem presents some new open questions in these topics and some of these are indicated. There is more emphasis on summarizing the theory of approximations than on the implications for the algorithm selection problem.

4.1 The Existence Question. In concrete situations one rarely worries about the existence of best selection algorithms (even though one continually worries about the existence of good ones). Yet, from time to time this question sheds important light on practical questions. Parameterization plays an important role here, one is continually identifying algorithms by means of a set of coefficients or parameters. The question of existence of a best algorithm then becomes a question of the existence of a best set of coefficients. In the simplest cases (e.g., linear forms) the coefficients are just sets of real numbers and the question is readily reduced to a problem about sets of real numbers. One then attempts to show that:

a. infinite coefficients cannot be best

b. the algorithms depend continuously on the coefficients

It then follows from standard mathematical arguments that a best set of coefficients exists.

This line of reasoning may fail a various points for nonlinear approximation forms. The failure is usually because of some weakness in the

parameterization. A key point to remember is distinguish carefully between an approximation form and the particular set of coefficients used to parameterize it. Consider the two simple examples:

$$S(f,c) = c_1+f/c_2$$

$$S(f,c) = c_1+e^{-c_2 f}$$

In both of these cases $c_2 = +\infty$ corresponds to a constant and hence a perfectly reasonable function. In the first example this is due to a silly parameterization, one should have $c_1+c_2 f$ instead. It is sometimes not so easy to see such silliness in more complex examples. The second example presents a more delicate situation, there is no familiar mathematical way to rewrite this form so that the difficulty disappears. One can, however, obtain a perfectly satisfactory parameterization by taking c_1 and c_2 to be the values of S(f,c) at f=0 and f=1, respectively. However, there is now no nice way to express S(f,c) explicitly in terms of c_1 and c_2.

True non-existence is fairly common for non-linear forms and discrete sets. The standard example is

$$S(f,c) = \frac{c_1}{1+c_2 f^2} \qquad f \in \{-1,0,1\}$$

Thus the feature f can take on only one of three possible values and we choose to give S the form of the reciprocal of a quadratic polynomial. Suppose now that the best selection (of all possible forms and problems) is 1 if f=0 and 0 if f ≠ 1. Consider the case where $c_1=1$; we have

$$S(0,c) = \frac{1}{1+0*f^2} = 1$$

$$S(-1,c) = S(+1,c) = \frac{1}{1+c_2}$$

We can make $S(\pm 1,c)$ as close to zero as we want by making c_2 large, however, if we set $c_2=\infty$, then $S(0,c)$ is ruined. The difficulty in this example is an essential one. There is no way to reparameterize $S(f,c)$ so as to obtain the best selection, yet we can come as close to it as we please.

Study of the existence question occasionally leads one to realize that the approximation form chosen must be extended in some way. A simple mathematical example of this occurs for the two exponential form

$$S(f,c) = c_1 e^{c_2 f} + c_3 e^{c_4 f}$$

Let $c_2 = (1+\epsilon)c_4$ and expand the first term in a Taylors series after factoring out $c_1 e^{c_4 f}$ to obtain

$$c_1 e^{c_4 f}\left[1+\epsilon c_4 f+ \frac{(\epsilon c_4 f)^2}{2!} + \frac{(\epsilon c_4 f)^3}{3!} + \ldots\right] + c_3 e^{c_4 f}$$

This may be rewritten as

$$e^{c_4 f}[c_1+c_3+c_1\epsilon c_4 f+c_1(\epsilon c_4 f)^2/2+\ldots]$$

Now let $c_1=-c_3$, $c_1=\alpha/\epsilon$ and then let ϵ go to zero. The result is

$$\alpha f e^{c_4 f}$$

and we see that this form with two exponentials also contains a function of completely different mathematical form. However, the plot of fe^f and neighboring curves in Figure 5 shows that there is nothing exceptional about $S(f,c)$ near this curve. Even so, the coefficients are $c_1 = +\infty$, $c_3 = -\infty$, $c_2 = c_4$ with

$$(c_4-c_2)*c_1 = \alpha$$

There is a singularity in the parameterization near this curve, much as there is a singularity at the north and south poles for the geographic coordinates parameterization of the globe.

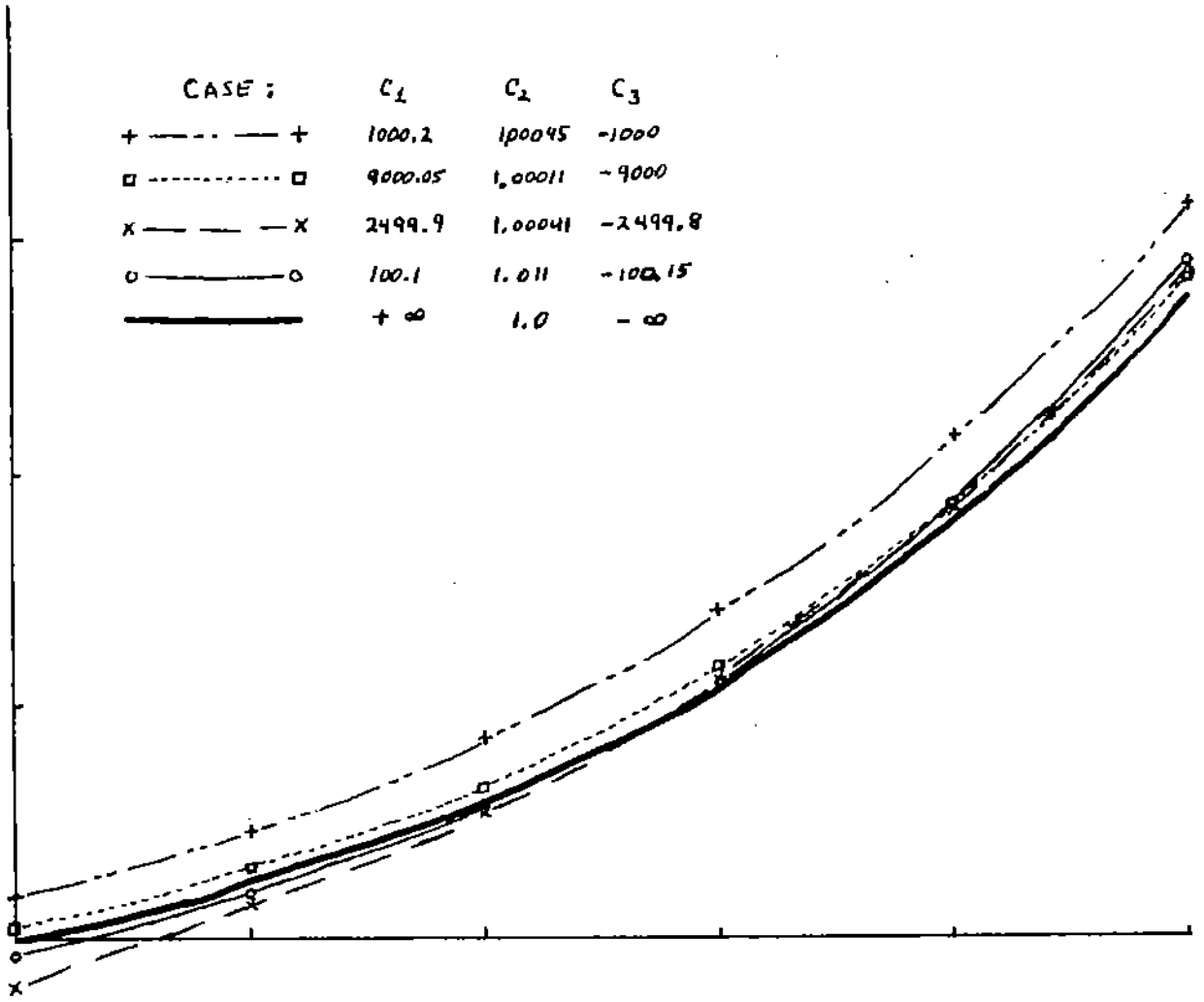


Figure 5. The curve fe^f and nearby curves of the form $c_1e^{c_2f} + c_3e^f$ with various values of c_1, c_2, and c_3.

A variation of this phenomenon occurs with the piecewise forms. Consider piecewise linear forms (broken lines) with variable break points. Figure 6 shows two things that can happen when the break points come together. On the left we see that two of them can converge so that the result is a step function with a jump discontinuity. On the right we see that four break points can converge so that an isolated peak (a "delta" function) of arbitrarily small base and large height is obtained.

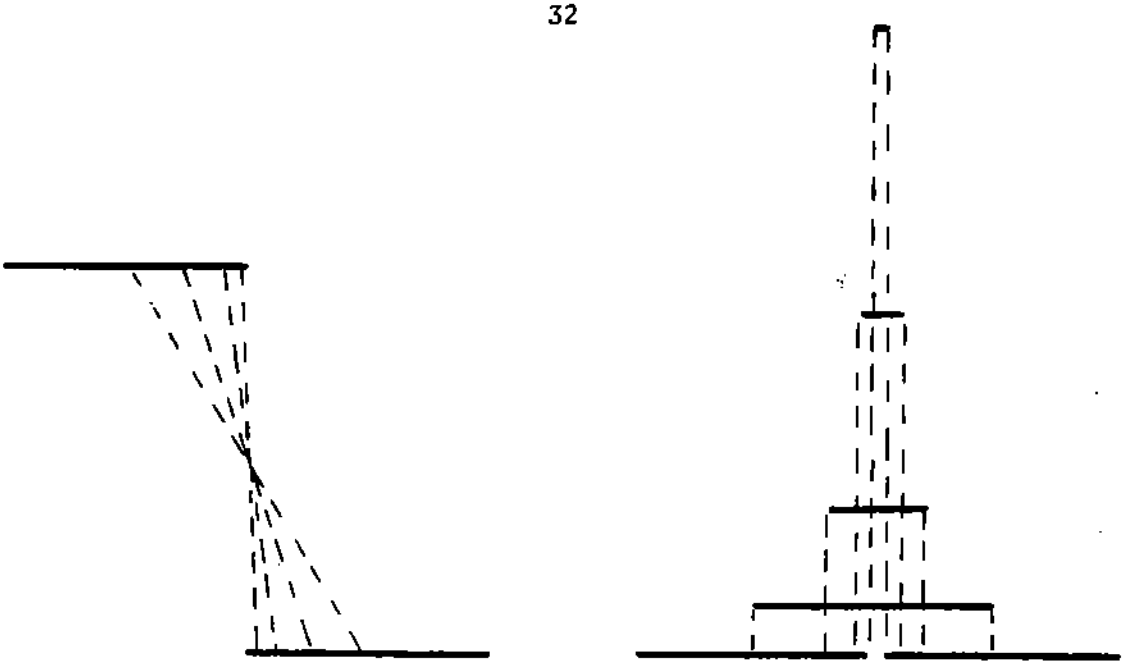


Figure 6. Two ways that non-linear break points in a broken line form can introduce new forms: a jump discontinuity (left) and a "delta" function (right).

Study of the existence question can have implications for computations in the following way. If either non-existance or the need for extending the definition are discovered, then one can expect computational difficulties. For example, if one is using the two exponential form $c_1e^{c_2f} + c_3e^{c_4f}$ and the best approximation is fe^f (or nearly so), then the computations become extremely ill-conditioned and normally collapse in a blizzard of highly magnified round-off errors.

So far we have discussed only classical mathematical forms, and we expect the same phenomena to occur for the tree and algorithm forms. A very interesting open question is whether other phenomena may occur.

4.2 The Uniqueness Question. One is usually not interested in this question per se, any best (or good) approximation will do. However, its study, like that of existence, can give insight into computational difficulties that may arise.

Global uniqueness is a rare property except for linear problems. This fact is intuitively illustrated by the simple problem of finding the closest point on a curve (a class of algorithms) from a given point (the optimal algorithm). This is illustrated in Figure 7.

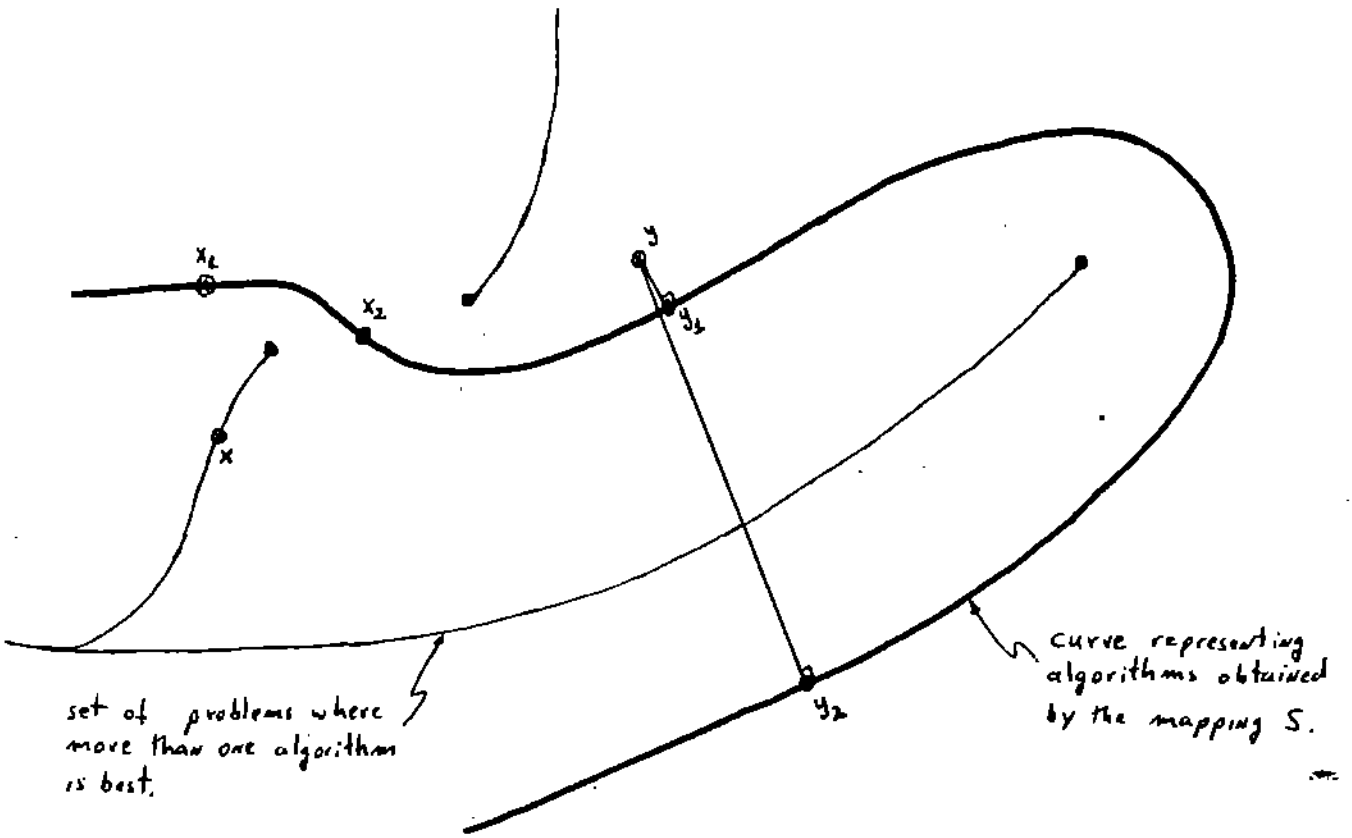


Figure 7. Illustration of non-uniqueness of best approximation for a nonlinear problem. In a linear problem, the curve would be a straight line and every point would have a unique closest point on the line.

Two other properties of the uniqueness question are illustrated by Figure 7. First is that almost all points have a unique best approximation even if a few do not. Second, we see that when there is more than one best approximation, they tend to be reasonably separated from one another. The point x, for example, has best approximations x_1 and x_2. Finally, the point y illustrates the most difficult situation where even though the

closest point (y_1) is uniquely determined, there is another point (y_2) much further away which is locally best and unique. That is to say, there is no point close to y_2 which is closer to y than y_2 is.

There are enormous computational implications of the phenomena illustrated in Figure 7. First, and somewhat less important, one can expect trouble at those points where two or more closest points are close together. This occurs near the three ends of the "lines of non-uniqueness" in Figure 7. More important is the fact that computational schemes are almost always local in nature and thus might well locate y_2 as the closest point to y. Further, such schemes usually give no inkling that there might be a point much closer to y. Note that this unfortunate situation occurs when we find a bad approximation (y_2 is far from y) and our limited experience in these matters does support the hope that "good" locally best approximations are likely to be global best approximations.

4.3 The Characterization Question. A characterization theorem gives some property of a best approximation which characterizes it, i.e., which allows us to distinguish it from other approximations. An elementary approach to the question goes as follows: If we have a best approximation S(F,C*) with best coefficients C*, then we have minimized something, namely our measure of performance $||p(S(F,C),F)||$. At minima we have derivatives equal to zero. Therefore, a characteristic property comes from the equations that result from evaluating the derivative of the measure of performance and setting it equal to zero.

The application of this approach is straight forward in many instances, for example, the derivation of the normal equations for least squares approximations. In other instances, the characteristic conditions might

appear to be completely unrelated to this approach. However, there usually is a direct relationship. For example, the conditions for optimality in linear programming problems is obtained this way modulo the changes necessary to include "differentiation" at the corners of multi-dimensional polyhedra. As an example, we derive the classical alternation theorem of minimax approximation using this elementary approach. Assume we want to approximate f(x) by S(c,t) with coefficients $c = c_1, c_2, \ldots, c_n$ so that

$$\max_t |f(t) - S(c,t)| = \text{minimum}$$

Then we want

$$\frac{\partial}{\partial c_j} \max_t |f(t) - S(c,t)| = 0 \qquad j = 1,2,\ldots,n$$

Now, the maximum only occurs at the extrema of $|f-S|$ and if we denote them by t_i^*, i=1,2,3,... we have

$$\frac{\partial}{\partial c_j} |f(t) - S(c,t)|_{\text{at } t_i^*} = 0 \qquad \begin{matrix} j = 1,2,\ldots,n \\ i = 1,2,3,\ldots \end{matrix}$$

We now differentiate off the absolute value sign to get

$$\text{sign}|f - S|_{\text{at } t_i^*} \frac{\partial}{\partial c_j} S(c,t)_{\text{at } t_i^*} = 0 \qquad \begin{matrix} j = 1,2,\ldots,n \\ i = 1,2,3,\ldots \end{matrix}$$

If S(c,t) is linear i.e., $S(c,t) = \sum_{j=1}^{n} c_j \phi_j(t)$ then we have

$$\text{(1)} \quad \text{sign}\,|f - S|\,\phi_j(t)_{\text{at } t_i^*} = 0 \qquad \begin{matrix} j = 1,2,\ldots,n \\ i = 1,2,3,\ldots \end{matrix}$$

That this is a variation of the alternation theorem is seen as follows (for the case of polynomial approximation, $\phi_j(t) = t^{j-1}$). First note that there must be at least n extrema t_i^* because otherwise we could find a

polynomial S(d,t) of degree n-1 so that

$$(2)\quad S(d,t_i^*) = \text{sign}\,|f(t_i^*) - S(c,t_i^*)| \qquad i = 1,2,3,\ldots,k\leq n$$

which contradicts the preceding relationship (1). More generally, the extrema t_i^* must occur with a combination of signs so that it is impossible to achieve (2) with any choice of coefficients d. Thus, using elementary properties of polynomials, one finds that there must be a set of extrema t_i^* so that

$$\text{sign}\,|f - S|_{\text{at } t_i^*} = (-1)^i \text{ or } (-1)^{i+1} \qquad i = 1,2,\ldots,n+1$$

This is the classical alternation property that characterizes best minimax approximations.

The main point made is that almost all characterization conditions come from setting derivatives equal to zero even though in some cases it may look much different because of special situations or because the conditions have been manipulated after equating the derivatives to zero.

The implication for computation is that they also are based on finding coefficients where the derivative is zero. In many situations the key to an effective computational procedure is to find a proper interpretation of the derivative in the problem at hand. These procedures are generally iterative in nature (unless one is lucky) and share many of the computational properties of similar methods of elementary numerical analysis (e.g., Newton's method, secant method, bisection, fixed point iteration). Unfortunately, these shared properties are not that attractive in high dimensional problems. That is that some of them are slow to converge or computationally expensive or difficult to initialize for convergence. Some methods may have all three of these unattractive properties in certain cases.

5. CONCLUSIONS.

One objective of this paper is to explore the applicability of approximation theory to the algorithm selection problem. We conclude that there is an intimate relationship here and that approximation theory forms an appropriate base upon which to develop a theory of algorithm selection methods. We also conclude that approximation theory currently lacks much of the necessary machinery for the algorithm selection problem. There is a need to develop new results for and apply known techniques to these new circumstances. The final section of this paper is somewhat of an appendix which lists 15 specific open problems and questions in this area.

We note that there is a close relationship between the algorithm selection problem and general optimization theory. This is not surprising since the approximation problem is a special form of the optimization problem. We have not attempted to detail this relationship here, but one may refer to [8] where the relationship between non-linear approximation and optimization is explored.

We conclude that most realistic algorithm selection problems are of moderate to high dimensionality and thus one should expect them to be quite complex. One consequence of this is that most straight forward approaches (even well-conceived ones) are likely to lead to enormous computations for the best selection. Indeed, the results of Rabin [5] suggest that this complexity precludes the determination of the best selection in many important cases.

Finally, we reiterate the observation that the single most important part of the solution of a selection problem is the appropriate choice of the form for the selection mapping. It is here that theories give the least guidance and where the art of problem solving is most crucial.

6. OPEN QUESTIONS AND PROBLEMS.

We list 15 questions that are given or suggested by the developments of this paper.

1. What is the relationship between tree forms and piecewise linear forms? Can all tree forms be made equivalent to some piecewise form, linear or non-linear?
2. What are the algorithm forms for the standard mathematical forms? Do they suggest useful simple classes of algorithm forms? See [2, Chapter 4] for algorithm forms for some polynomial and rational forms.
3. Determine specific classes of tree forms where the current machinery of non-linear approximation is applicable.
4. Develop some general approaches (or methods) to classifying problems within a problem space. This is related to the next problem.
5. Develop an abstract machinery for analyzing optimal features. Such a machinery might well combine the theoretical ideas of n-widths and/or enthropy [3] with the intuitive ideas of performance profiles given earlier [7].
6. What is the nature of the dependence of the degree of convergence on the dimensionality of the problem? Some results are known for polynomial approximation to multivariate functions. Are these typical of what one should expect in general?
7. What is the nature of the dependence of complexity on the dimensionality of the problem? Can results of 6. above be translated directly into statements about complexity?
8. Obtain more precise information about the nature of real world functions? The generalities used in this report were obtained by selecting a large

number of empirically determined functions from [1] and then observing how effective polynomial approximation is. Are the results of this experiment representative of other contexts? Can more precise information about the properties of such classes be obtained?

9. Determine the computational complexity of the following specific problems. For simplicity, one may use one evaluation of $f(x)$ as the unit of computation and ignore all other work.

   (a) Approximation to $f(x)$ via interpolation by polynomials. Assume various kinds of smoothness for $f(x)$.

   (b) Least squares approximation to $f(x)$ on $[0,1]$ by polynomials. Assume various kinds of smoothness for $f(x)$.

   (c) Evaluate $\int_0^1 f(x)dx$. This is closely related to the least squares problem.

10. Formulate a more precise and general concept of robustness.

11. Develop useful mechanisms to embed certain classes of discrete forms into continuous ones. This is particularly relevant for non-standard mathematical forms.

12. Develop techniques to partition high dimensional problem sets into subsets where good linear approximations are possible. A particular instance would be to develop adaptive algorithms for piecewise linear (no continuity) approximations in high dimensions. See [4] for some work in one dimension.

13. Develop existence theorems for various classes of tree form approximations. Do the difficulties of coalesced knots that occur in spline approximation have an analogy in general tree forms?

14. What are the relationships between best algorithm selection and the results in automata theory about computability and computational complexity?

15. Is there anyway to "differentiate" the tree form so as to obtain a local characterization theorem?

REFERENCES

1. Handbook of Chemistry and Physics, Handbook Publishers Inc., Sandusky, Ohio, 1960.

2. Hart, John F. et al., Computer Approximations, John Wiley, New York, 1968.

3. Lorentz, G. G., Approximation of Functions, Holt, Rinehart and Winston, New York, 1966.

4. Pavlidis, T. Waveform segmentation through functional approximation, IEEE Trans. C-22 (1973), 689-697.

5. Rabin, M. O., Theoretical impediments to artificial intelligence. Proc. IFIP '74, North-Holland, Amsterdam, 1974, pp. 615-619.

6. Rice, John R., The algorithm selection problem - Abstract models, CSD-TR 116, Purdue University, May, 1974, 18 pages.

7. _____, The algorithm selection problem II - Two concrete problems, CSD-TR 117, Purdue University, May, 1974, 19 pages.

8. _____, Minimization and techniques in nonlinear approximation. Studies in Numerical Analysis, Vol. 2 (1970), 80-98.