Complex Systems 4 (1990) 201–249

A Mathematical Theory of Generalization: Part II

David H. Wolpert
Theoretical Division, Center for Nonlinear Studies,
Los Alamos National Laboratory, Los Alamos, NM, 87545, USA

Abstract. The problem of how best to generalize from a given learning set of input–output examples is central to the fields of neural nets, statistics, approximation theory, and artificial intelligence. This series of papers investigates this problem from within an abstract and model-independent framework and then tests some of the resultant concepts in real-world situations. In this abstract framework a generalizer is completely specified by a certain countably infinite set of functions, so the mathematics of generalization becomes an investigation into candidate sets of criteria governing the behavior of that infinite set of functions. In the first paper of this series the foundations of this mathematics are spelled out and some relatively simple generalization criteria are investigated. Elsewhere the real-world generalizing of systems constructed with these generalization criteria in mind have been favorably compared to neural nets for several real generalization problems, including Sejnowski's problem of reading aloud. This leads to the conclusion that (current) neural nets in fact constitute a poor means of generalizing. In the second of this pair of papers other sets of criteria, more sophisticated than those criteria embodied in this first series of papers, are investigated. Generalizers meeting these more sophisticated criteria can readily be approximated on computers. Some of these approximations employ network structures built via an evolutionary process. A preliminary and favorable investigation into the generalization behavior of these approximations finishes the second paper of this series.

Introduction

This series of papers constitutes an investigation of generalization from a mathematical point of view. The investigation is behavior-driven as opposed to model-driven; the generalizing behavior is set and the set of models which exhibit this behavior deduced, as opposed to having the models be set and the generalizing behavior of the models deduced. In its being behavior-driven, the work in these papers is much more objective and far less ad hoc than

most of the other work on the issue of generalization, the vast majority of which is model-driven.

In the first paper of this series a mathematical framework for dealing with generalization in a model-independent manner was presented and the ramifications of certain generalization criteria were investigated within that framework. In this second paper of the series more criteria are investigated, and then an investigation into the real world behavior of systems meeting some of these sets of criteria is presented.

Although the material in this second paper is conceptually distinct from that of the first, the terminology and mathematics of the first paper is used in this second paper. Therefore, this second paper cannot be read profitably unless the first paper has already been skimmed (at least). Some of the more directly relevant definitions and criteria of the first paper are repeated here for the reader's convenience.

(0.1) An *m-dimensional generalizer* is a countably infinite set of continuous functions from a subset of $(\mathbf{R}^m \times \mathbf{R} \times \mathbf{R}^m)$ to $\mathbf{R}$, from a subset of $(\mathbf{R}^m \times \mathbf{R} \times \mathbf{R}^m \times \mathbf{R} \times \mathbf{R}^m)$ to $\mathbf{R}$, etc., where $\mathbf{R}$ denotes the space of real numbers and $\mathbf{R}^m$ denotes the Cartesian product of m such spaces. Notationally, an m-dimensional generalizer is a set of continuous functions $g\{i\}$ along with associated domains of definition, i being a nonzero natural number, and $g\{i\}$ being from $\mathbf{R}^{i(m+1)+m}$ to $\mathbf{R}$.

$\mathbf{R}^m$ is the input space, and $\mathbf{R}$ is the output space. The last $\mathbf{R}^m$ entry in the argument to a $g\{i\}$ is the question. The previous i $(\mathbf{R}^m \times \mathbf{R})$ entries are the learning set, which has order i. Each of the i $(\mathbf{R}^m \times \mathbf{R})$ spaces is a datum space, and a point in a datum space is a datum vector. Although in practice it is not uncommon to weaken (0.1) to allow discontinuous $g\{i\}$, unless it is explicitly stated otherwise it will always be assumed that this has not been done and that the requirement of continuity of the $g\{i\}$ is in force.

(0.2) Every $g\{i\}$ is invariant under permutation of the datum spaces.

(0.3) If, for any $g\{i\}$, the value of the question is the same as the value of an $\mathbf{R}^m$ entry in one of the datum spaces, the output of the function is the corresponding $\mathbf{R}$ entry from that datum space.

In other words, we are assuming noise-free data; this restriction can be eased in more sophisticated versions of the theory.

As a direct result of (0.3) we require that

(0.4) For any $g\{i\}$, if any two datum spaces have the same value for their $\mathbf{R}^m$ entries, then they must have the same values for their $\mathbf{R}$ entries.

(0.5) Unless i, the order of the learning set, exceeds m, the dimension of the generalizer, $g\{i\}$ is not defined. Even if $i > m$, $g\{i\}$ is not defined if the values of the $\mathbf{R}^m$ entries of the datum spaces all lie on the same m-dimensional hyperplane.

The next set of restrictions may or may not apply to a particular generalizer — they are ways of classifying generalizers according to their generalizing behavior.

(0.6) Every $g\{i\}$ is invariant under any rotation, parity, translation, or nonzero scaling transformation, applied simultaneously to the all the $\mathbf{R}^m$, including the question, or to all the $\mathbf{R}$, including the output.

Any generalizer obeying (0.6) is called a HERBIE. A "hyperplanar" HERBIE [5] is a HERBIE which generalizes by fitting the learning set with an input/output surface consisting of local hyperplanes.

(0.7) Any HERBIE obeys the following: .IP (0.7) For any i, if the values of the $\mathbf{R}$ entry of every datum space of a $g\{i\}$ of a continuous HERBIE are equal, the output is this value, regardless of the question.

A generalizer is "upwardly compatible" if

(0.8) For all $g\{i\}$, $i > m+1$, if the values of the entries of two datum spaces are identical, then the output is equal to the output of $g\{i-1\}$ working on the rest of the learning set and on one of the two identical datum spaces.

An upwardly compatible HERBIE is a "proper" generalizer. A generalizer necessarily meeting restrictions (0.1) through (0.8) except for (0.6) (and its consequence (0.7)) is known as a "semi-proper" generalizer. Semi-proper HERBIEs are proper generalizers. Other restrictions (e.g. output linearity) are presented and investigated in the first paper of this series. The ones just presented are those which are used the most in this, the second paper of the series.

1. Self-guessing

1.1 Introductory definitions

Just as the requirement that the generalizer be an output linear HERBIE is not enough to force unique generalization of a learning set, properness, by itself, is not enough to force a unique generalization of any learning set. For example, in addition to the standard npa generalizer described in appendix H of the first paper, an npa generalizer with the upper limit of integration being $3|\vec{r}_0|$ instead of $2|\vec{r}_0|$ is proper. Although it might turn out that some combination of the kinds of criteria presented in the first paper is enough to force uniqueness, certainly other generalizing criteria bear exploration as well. In that all the $g\{i\}$ are somehow supposed to refer to the same means of generalizing, in searching for these other criteria it is natural to look for criteria which are similar to the criteria of upward compatibility (0.8) in that they restrict the possible relationships between the different $g\{i\}$.

One of the more reasonable such additional criteria is the property of *self-guessing*. Intuitively, self-guessing is the requirement that, using the generalizer in question, the learning set must be self-consistent. If a subset of the learning set is fed to the generalizer, the generalizer must correctly guess the rest of the learning set. The "starfish" example should illustrate the point: Assume the learning set consists of points with 2 dimensions of input, and that the parent surface generating the learning set has output value 0 everywhere in the input plane, except for along the radial rays which have polar angle an integer multiple of 10 degrees, where the output value is 1. The parent surface is a starfish with 36 infinitesimally thin legs. Assume the learning set consists of many points all of which have polar angle less than 350 degrees, and the question has polar angle 350 degrees, so that the learning set is peppered amongst the first 35 legs and the the question is situated on what should be guessed as the 36th leg. Any of the conventional generalizers, most HERBIEs included, would not guess an output value of 1 for the question. However a human would likely notice that the simple rule of output a 1 if you are on a multiple of $\pi/18$ radians, 0 otherwise, would produce the entire learning set. The reason the human would have faith in this generalization is because if (s)he used it to generalize from a subset of the learning set (s)he would correctly guess the rest of the learning set. In other words, a human would come up with a generalizer — finding and then applying the angular periodicity of the learning set — which is self-guessing for this particular learning set. To winnow the set of generalizers down to one which is in any sense "optimal" for a given learning set, it is necessary to milk as much information from the learning set as possible, and self-guessing is designed specifically to perform such milking. There might be situations where self-guessing would not be the method used by a human, but it is hard to imagine situations where a human would deny the reasonableness of applying a generalizer to a particular learning set simply because it is self-guessing for that learning set.

In a crude sense, self-guessing can be viewed as a kind of "meta" version of the requirement of reproducing the learning set, i.e. the requirement of not contradicting those examples of the mapping which you already know to be true. You are given a learning set, and want to decide on a generalizer to apply to it. Self-guessing says choose that generalizer which agrees with those examples which you already know to be true, i.e. which makes a (learning set, question) $\rightarrow$ guess mapping in agreement with those examples of such a mapping already given. The examples of such a mapping already given are nothing less than the set of instances (proper subset of the full learning set, question from within the rest of the learning set) $\rightarrow$ guess of the (known) output corresponding to that question.

Self-guessing is not a completely new idea. For example, Samuel's prediction equalizing scheme for his checkerplayer program [1] can be viewed as a crude form of self-guessing. In addition, except for the fact that such systems cannot answer questions not contained in their full "learning set," autoassociaters of strings of data can be viewed as self-guessing one-dimensional

generalizers. Finally, statisticians have experimented with simplified versions of (weak) self-guessers under the name of "cross-validation" [2].

To define self-guessing in a rigorous manner, we start by defining any combination of a set of datum vectors, a question, and an output as a *lesson*. The *order* of the lesson is the number of datum spaces. One lesson is said to *generate* another lesson if the second lesson can be produced from the first by means of any sequence of a predefined set of lesson to lesson mappings. In particular, one lesson is said to *properly generate* another lesson if the second lesson can be produced from the first by means of any sequence of: the invariances of (0.2) and (0.6), replacing the question/output pair by the input and output values of one of the datum vectors as in equation (0.3), or the procedure outlined in (0.8) (used to generate either higher or lower order lessons). No lesson is allowed to generate another lesson if that second lesson violates equation (0.5). If one lesson properly generated from another lesson contradicts either equations (0.4) or (0.7) or by any other means results in a paradox, then either the generalizer is not proper or the original lesson must be removed from the generalizer's domain of definition. In general, when just the word "generate" is used, unless explicitly stated otherwise only the mappings implied by restrictions (0.2) through (0.5) will be assumed to apply.

A learning set is said to generate an *expanded* learning set by first providing the set of lessons called the *initial expansion*. The lessons of the initial expansion consist of all the questions and outputs necessitated by the generalizer's reproducing the learning set (see equation (0.3)), using the full learning set as the values for the datum vectors. If we are provided a learning set consisting of n distinct input–output pairs, the initial expansion consists of n nth order lessons. The *full expansion* of the learning set (or just *expansion* for short) is constructed by having each lesson in the initial expansion generate all the lessons it can, given the lesson-to-lesson mappings at hand. In particular, the *proper expansion* is the full expansion in which each lesson in the initial expansion properly generates all the lessons it can. Intuitively, every lesson in the proper expansion of a particular learning set is a consequence of the input–output pairs of that learning set and the requirements of properness. Note that the proper expansion of a learning set is *not* the same as the set of all lessons consistent with the starting learning set and the requirements of properness. For example, let the learning set have order n and consist of the n pairs (input(i), output(i)), $0 < i < n + 1$. Any proper generalizer will have $g\{n + 1\}$ take the learning set along with any arbitrary pair (input($n + 1$), output($n + 1$)) and the question input $(n + 1)$ to an answer output $(n + 1)$, by requirement (0.3). This is true even if the pair (input($n+1$), output($n+1$)) is not in the original learning set. However provided only with the original learning set, we do not know that the question input($n + 1$) should result in the answer output($n + 1$). So we do not include this new lesson of order $(n + 1)$ in our expansion of the learning set — it is not a consequence of the input/output pairs of the original learning set together with the requirements of properness.

We are now in a position to rigorously define the property of self-guessing:

(1.1) A self-guessing expansion of a learning set is one in which in addition to the other preset ways in which one lesson can generate another, new lessons are allowed to be formed from old ones by interchanging the ordered product (the question, guessed output to the question) of such an old lesson with the entries of any datum space in that old lesson. A particular generalizer is said to be self-guessing for a particular learning set if, for all i, $g\{i\}$ of that generalizer agrees with all lessons (of appropriate order) in the self-guessing expansion of that learning set.

Note that a set of lesson to lesson mappings must be given (usually implicitly) to fix the mappings making up the transition from initial expansion to full expansion and therefore to fix the meaning of "self-guessing." If all the $g\{i\}$ in a generalizer G agree with any lesson in a proper self-guessing expansion of a learning set, then G is said to be properly self-guessing for that learning set. For a generalizer known to be self-guessing for a particular learning set, the definitions of "generate" and "expansion of a learning set" are usually assumed to be modified in accordance with (1.1).

As an example of all this, taking $g\{2\}$ as the second function of a generalizer self-guessing for a particular learning set θ, then if $(b, B, c, C; a, A)$ is in the expansion of Θ, $g\{2\}(b, B, c, C; a) = A$, and it is also necessarily true that $g\{2\}(b, B, a, A; c) = C$ and $g\{2\}(a, A, c, C; b) = B$ (whether or not the pairs (a, A), (b, B) and (c, C) are in θ). For self-guessing generalizers, the ordered pair of the question and the output is on the exact same footing as any of the datum spaces. Given datum space interchange symmetry, we therefore can (and do) view a lesson as an unordered set of datum spaces. For example, the $g\{2\}$ above guesses the lesson (a, A, b, B, c, C). In contradistinction, functions in their full generality constitute ordered sets of datum spaces.

We refer to a generalizer as being properly self-guessing "for a (particular) learning set" because no generalizer can, in general, be properly self-guessing for all learning sets. For example, no one-dimensional generalizer can be self-guessing for both the learning set (a, A, b, B, c, C) and the learning set (a, A, b, B, c, D) where $C \neq D$. The first set forces $g\{3\}(a, A, b, B, c, C; c) = C$ due to the initial expansion of that set, and therefore, using self-guessing and upward compatibility, the lessons $g\{3\}\ (a, A, c, C, c, C; b) = B$, $g\{2\}(a, A, c, C; b) = B$, and $g\{2\}(a, A, b, B; c) = C$. (Note that G had to be one-dimensional for $g\{2\}$ to be defined.) Similarly, the lesson $g\{2\}(a, A, b, B; c) = D$ is in the second set's expansion. Since any generalizer's $g\{2\}$ must be single-valued, no generalizer can be properly self-guessing for both learning sets, which proves the supposition. Note that a generalizer properly self-guessing for a particular learning set need not be proper in general — it only has to obey (0.6) through (0.8) for the elements in the expansion of the learning set.

We can now recast restrictions (0.2) through (1.1):

Any lesson in a learning set guessable by a generalizer properly self-guessing for that learning set

(a) generates other lessons with which the generalizer must agree by rotating, translating, parity inverting and rescaling all the $\mathbb{R}^m$ values and/or all the $\mathbf{R}$ values (0.6), and therefore

(b) cannot have all the $\mathbf{R}$ values identical except for one, which is different (0.7 in conjunction with 1.1),

(c) cannot have unequal $\mathbf{R}$ values for any two datum spaces if the corresponding $\mathbb{R}^m$ values are identical (0.3, 0.4), and therefore

(d) generates other lessons by copying over the values for any one of the datum space pairs with the values for one of the other pairs (the lesson $g\{i\}(a, A, b, B, \ldots ; z) = Z$ generates the lesson $g\{i\}(a, A, b, B, \ldots ; a) = A$) and therefore

(e) assuming the number of datum spaces is at least $m + 2$, generates other lessons with one fewer datum spaces by removing the entries for any datum space (0.8), and

(f) generates other lessons with one more datum space by duplicating the entries for any datum space (0.8).

In addition, since the $g\{i\}$ are single-valued, no expanded learning set can have two lessons identical in all values except for the output ($\mathbf{R}$) value of one of the datum spaces. Semi-proper self-guessing only necessitates (c) through (f).[1]

As an example of a generalizer semi-properly self-guessing for a particular learning set, consider the following one-dimensional generalizer working on a three point learning set θ. Without loss of generality, we assume that the three points are arranged in the order of increasing input component. Working in the input–output plane, let the polar angle (with respect to the positive direction along the input axis) of the line connecting the first two points in θ be α, and let the polar angle of the second two points be β. One generalizer, which is self-guessing for θ, operates by using one procedure to guess beyond the right-most element of a learning set, another procedure

[1]As an aside, note that if the definition of proper expansion of a learning set were modified to allow every lesson that is true for all proper generalizers consistent with the initial expansion, then it would always be possible to arrive at a contradiction within the definition of self-guessing. This is done by making a lesson which is two orders higher than the learning set and which contains the learning set as a proper subset, with the two extra datum vectors being identical with each other. Another lesson is then made in the same way, except that in its two extra identical datum vectors, the output components (but not the input components) differ from those in the first lesson. For all generalizers, both of the two new lessons are guessed by the generalizer due to requirement (0.3), and both are consistent with the original learning set. However if they were both included in the expansion of the learning set, it would now be possible to use (e) to arrive at a contradiction — no generalizer would be self-guessing for our (arbitrary) original learning set, since (e) would allow us to come to two equally valid but incompatible conclusions about how the generalizer guesses.

to guess beyond the left-most element, and a third procedure to guess the answers to questions in between. The line giving the guesses beyond the right-most edge of any learning set (whether or not it is θ) goes through that right-most point and extends to the right at a polar angle of β, the polar angle implicit in the rightmost pair of points in θ. To guess answers to questions to the left of that arbitrary learning set, extend a line leftward from the left-most point of the learning set at a polar angle of α, the polar angle implicit in the leftmost pairs of points in θ. It should be clear that fed the first two points of θ, this generalizer would guess the third, and fed the second two points it would guess the first. The generalizer's guessing for questions within the bounds of the learning set is constructed so that the first and third points of θ will guess the second point of θ. For a question falling within the bounds of any learning set (again, whether or not it is θ), let the point in the learning set immediately to the question's left be (a, A) and let the point immediately to its right be (b, B). Take the curve formed by linearly connecting the three dots of θ, translate it so that its left tip is coincident with (a, A) and then scale its input extent so that its right tip has input value b. Let the coordinates of the resultant right-hand tip be (b, C). Then add $(B-C)(x-a)/(b-a)$ to all points along this dot-connecting curve, where x is the input coordinate of the point in question. (This ensures that the right hand coordinates of this final version of the curve are (b, B) whereas the left-hand coordinates are still (a, A).) The answer to the question if found by reading off of this resultant curve. If fed the first and third points of θ this generalizer will guess the second. The generalizer is now defined for the whole of the input space, for any learning set of any cardinality. Taking any two points in θ as a learning set, this generalizer will guess the third point. Therefore it is properly self-guessing, for θ.

It is not true that for every learning set there exist a properly self-guessing generalizer; proper self-guessing is an over-restrictive criterion for generalizing. For example, any one-dimensional generalizer which is properly self-guessing for the expansion of the learning set $((0,0), (1,1), (2,2), (4,5))$ contains the lesson $((0,0), (1,1), (2,2))$. By means of the invariances of (a), this generates the lesson $((0,0), (2,1), (4,2))$ and then $((0,0), (2,2), (4,4))$. But from the original learning set we can generate the lesson $((0,0), (2,2), (4,5))$, resulting in a contradiction. In addition, no learning set of dimension m with a subset of m or more datum vectors all having the same output $(\mathbf{R})$ value and also having other datum vectors with differing output values generates an expansion guessable by a properly self-guessing generalizer. This is true due to restrictions (b) and (e).

Neither of these two cases poses any difficulty for generalizers which are either proper but not self-guessing, or are self-guessing but not bound by the invariances of (0.6). In addition, an npa generalizer, a proper generalizer, is most definitely not self-guessing for (almost) all learning sets. It seems that the invariances of (0.6) and the property of self-guessing are at best not very compatible restrictions.

This incompatibility between self-guessing and the invariances of (0.6) is further borne out by considering some of the natural extensions to the invariances defining proper generalizers. For example, one natural and important extension is to require that

(1.2) If any $g\{i\}$, operating over a given learning set, guesses any two distinct question-output pairs, A and B, then $g\{i+1\}$ operating over a learning set made up of the original learning set plus either A or B will guess the other question-output pair, either B or A respectively.

Consider the situation where $g\{2\}(x, y, x', y', x'') = y''$ and $g\{2\}(x, y, x', y', x''') = y'''$. Requirement (1.2) is based on the idea that adding the datum point (x'' gives y'') adds nothing new to the second lesson, since that is what would be predicted from the two datum points (x, y) and (x', y') anyway: $g\{3\}(x, y, x', y', x'', y'', x''') = y'''$. Requirement (1.2) can be viewed as a stronger version of upward compatibility — any generalizer obeying (1.2) is automatically upwardly compatible, but not vice versa. When combined with proper self-guessing however, by use of properties (a) through (f), and in particular through use of of the scaling and translating invariances, property (1.2) would generate a sequence of lessons which in general constitute arbitrarily wildly oscillating on the generalizer's part — certainly not very desirable behavior for a generalizer.[2] (As an aside, note also that local generalizers like the hyperplanar HERBIE of reference [5] will not obey (1.2), in general.)

Due to this lack of total compatibility, it is useful to shift tactics slightly and restrict our attention to semi-proper self-guessing. To keep the criterion of self-guessing, we are sacrificing some of the requirements going into the criterion of properness. The definitions of "generate" and "expansion of a learning set" are modified appropriately when talking about semi-proper

[2]For example, examine the input/output curve constructed by a one-dimensional generalizer starting from $g\{2\}(x, y, x', y', q) = z$, where $x' > x, y' > y$, and (q, z) lies on the perpendicular bisector of the line connecting (x, y) with (x', y'), and below that line connecting (x, y) and (x', y'). Making liberal use of (a) through (f) as well as (1.2), it is easy to show that $g\{2\}(x, y, x', y', Q)$, where Q lies in between x and x', is wildly oscillatory. Do this by first using scaling and translating invariances to pull the point (x, y) to on top of the point (q, z), keeping the point (x', y') fixed. This gives $g\{2\}(q, z, x', y', (\frac{q-x}{x'-x})(x' - q) + q) = y' - [(\frac{y'-z}{y'-y})(y' - z)]$, which with $g\{2\}(q, z, x', y', x) = y$ means (assuming (1.2)) $g\{3\}(q, z, x, y, x', y', (\frac{q-x}{x'-x})(x' - q) + q) = y' - [(\frac{y'-z}{y'-y})(y' - z)]$. This in turn means $g\{2\}(x, y, x', y', (\frac{q-x}{x'-x})(x' - q) + q) = y' - [(\frac{y'-z}{y'-y})(y' - z)]$. Repeat this procedure of scaling the outer points in so they are coincident with the inner points, and make liberal use of (1.2). In this way we can deduce the guessing of $g\{2\}(x, y, x', y', Q)$, Q being arbitrarily close to any point between x and x' by "zeroing in" on a. All of these guesses will lie below the line connecting (x, y) and (x', y'). However note that scaling and translating invariance means that $g\{2\}(x, y, x', y', q) = z$ implies that $g\{2\}(x, y, x', y', \frac{x+x'}{2} - [q - \frac{x+x'}{2}]) = \frac{y+y'}{2} + [\frac{y+y'}{2} - z]$. Again playing the game of scaling the outer points so that they are coincident with the inner points, we can again construct the guessing of $g\{2\}(x, y, x', y', Q)$, with Q being arbitrarily close to any point between x and x'. However all of these guesses will lie above the line connecting (x, y) and (x', y'), not below it. As promised, wildly oscillating behavior for $g\{2\}(x, y, x', y', Q)$.

self-guessing. The full expansion of a learning set made with semi-proper self-guessing obeys properties (c) through (f), but not necessarily (a) or (b). We now restrict our attention to semi-properly self-guessing generalizers rather than (fully) properly self-guessing generalizers when generating the expansions of learning sets.

1.2 Strong self-guessing

For generalizers semi-properly self-guessing for a particular learning set, there is a natural extension of the definition of "generate" which is helpful at fixing the relationships between the $g\{i\}$ of a given generalizer. The idea behind this extension is to replace the question-output pair of a given lesson with a new question-output pair where the question is arbitrary and where the output is the output the generalizer would guess for that (arbitrary) question. More precisely, this extension modifies the definition of "generate" to encompass, for a particular generalizer, taking a lesson, feeding all but one of its datum spaces into the generalizer as a learning set, and forming ALL the new lessons given by the resultant predictions of the generalizer. For example, if (a, A, b, B, c, C) is a lesson and we have a fixed (if unknown) generalizer, then we say that (a, A, b, B, c, C) generates the lessons $(a, A, b, B, x, g\{2\}(a, A, b, B))$ for all x. For this extension of the definition of "generate," the definition of "expansion of a learning set" is assumed to be modified accordingly. When it is desirable to draw attention to the fact that these new definitions are being used, such generation is said to be *strong* generation, and such an expansion is said to be a *strong* expansion. (When it is desired to draw attention to the fact that the old definitions are being used, the adjective *weak* will be employed.) A *strong self-guessing* expansion of a learning set θ, given a set of lesson to lesson mappings Γ, is the expansion formed by strongly generating lessons from θ by using Γ together with the operator of interchanging the ordered product (the question, guessed output to the question) with the entries of any datum space. The weak self-guessing expansion of a learning set is always a proper subset of the strong self-guessing expansion.

The generalizer being used is always implicit when talking about strong generation and/or expansion, of course. Without knowing that particular generalizer, we do not know the strong expansion of an associated learning set. Nonetheless, we can still set restrictions on that expansion, regardless of the particular generalizer that goes with it. To do this we first need to make a definition:

(1.3) Take any generalizer G along with a learning set, θ, and a set of lesson to lesson mappings Γ. Use the elements of Γ together with G to form the strong self-guessing expansion of θ. For this fixed (if unknown) generalizer G, if no lesson in the strong self-guessing expansion of θ disagrees with G, then we say that G is *strongly self-guessing* for θ.

For the rest of this paper, unless explicitly stated otherwise, whenever talking about about a generalizer G strongly self-guessing for a learning set θ, we'll assume that G is semi-proper. We'll also assume that the implicit set of lesson to lesson mappings, Γ, includes all the lessons of semi-properness (i.e. we'll assume that G is semi-properly strongly self-guessing for θ). In other words, we'll be assuming that the lesson-to-lesson mappings used to generate the expansion are those of properties (c) through (f), (1.1), and the strong generation of one lesson from another.

Assume G is a (semi-proper) generalizer, strongly (semi-properly) self-guessing for a learning set θ. Take any lesson β in the expansion (be it weak or strong) of θ, consisting of a learning set B along with the question-output pair (x, y). Assume another lesson β' in the expansion of θ consists of B along with the pair (x', y') (for strong self-guessing, all such lessons, for any x', will exist in the expansion of θ). From semi-proper self-guessing, (B, x, y, x, y) is in the expansion. Therefore, using the new definition of generate, the lesson $((B, x, y), x', z)$ is also in the expansion, where the unknown z is the answer to the question x'. But by simple self-guessing, $z = y'$. This means that property (1.2) is obeyed by all lessons in the expansion of θ. (It does not mean that G obeys (1.2) for any learning set.)

In Appendix A it is shown that

(1.4) If an m-dimensional semi-proper generalizer G is strongly semi-properly self-guessing for a learning set *theta*, there exists a continuous function f from $\mathbf{R}^m$ to $\mathbf{R}$ such that the following holds: Every set of datum space entries in θ lies on f, and if any $g\{i\}$ of G is provided a set of points from f as a learning set, it will guess f as the question-output function. Conversely, if there exists such a function f for a semi-proper generalizer G and a learning set θ, G is strongly semi-properly self-guessing for θ.

Property (1.4), related to restriction (2.12) of the first paper, is very interesting mathematically in that it amounts to restrictions on functions concerning their behavior when fed their own outputs as inputs. However it still is not enough to uniquely set the optimal generalization of an arbitrary learning set. To see this, define the learning set to be generalized as θ. Let $f(x)$ be any continuous function which passes through all the input–output pairs of θ. We will demonstrate a procedure for constructing a semi-proper generalizer G such that for G and θ, f is the function referred to in (1.4). Let ω be an order n learning set fed to G. Label the elements of ω by (x_i, y_i), where $1 \leq i \leq n$ for all i. Now make a list of all the pairs $(x_i, y_i - f(x_i))$. The function $h(x)$ is then given by the npa generalizer (see appendix H of the first paper of this series) taking all the points in this list as a learning set. $g\{n\}(\omega; x)$ is then given by $f(x) + h(x)$. If the elements of ω all lie on $f(x)$, $h(x)$ identically equals 0, and $g\{n\}(\omega; x) = f(x)$, thereby satisfying (1.4). To verify that the resultant G is semi-proper, first note that whether or not the elements of ω lie on $f(x)$, $g\{n\}(\omega; x)$ will reproduce all the pairs in ω. Upward compatibility and the rest of semi-properness is ensured by $h(x)$'s

being an npa generalizer. In that G's guessing is dependent on $f(x)$ and $f(x)$ is not uniquely defined, we see that the criteria of strong semi-proper self-guessing is indeed under-restrictive.

1.3 Criteria sets

As it turns out, no criteria exist which, when combined with semi-proper self-guessing (strong, weak, or otherwise) produces unique generalization of any learning set. To see this, first define a *criteria set* as a partitioning of the set of all generalizers (meeting restrictions 0.2 through 0.5) into two classes of generalizers, one of generalizers which "meet all the criteria" of the criteria set, and one of generalizers which do not. HERBIEs are a criteria set, as is the set of all proper generalizers, and as is the set of all semi-proper generalizers. The intersection of the criteria set of HERBIEs with the criteria set of semi-proper generalizers is the criteria set of proper generalizers.

Semi-proper self-guessing then is not a criteria set, but rather a mapping from any allowed learning set (i.e., any learning set meeting requirement 0.4) θ to the criteria set of all generalizers which are semi-properly self-guessing for θ (i.e., to the criteria set of all generalizers in agreement with the semi-proper (strong, weak, or otherwise) self-guessing expansion of θ). Different θ's have different sets of generalizers which are semi-properly self-guessing for them, and therefore have different criteria sets.

In this context the goal of finding criteria which when combined with semi-proper self-guessing produce unique generalization is to find a criteria set Ω narrow enough such that $\Omega \cap sg(\theta)$ is a single unique generalizer for any learning set θ ($sg(\theta)$ is the set of generalizers which are semi-properly self-guessing for θ). For any θ, this would give us a unique generalizer which is semi-properly self-guessing for θ. Unfortunately, as was mentioned, there is no such criteria set Ω. Loosely speaking, this is because if there were a criteria set Ω such that $\Omega \cap sg(\theta)$ existed for all learning sets θ and, for a given θ, was unique, then the rule of generalizing from any provided learning set B' with $\Omega \cap sg(\theta')$ would itself constitute a generalizer, and this generalizer would be self-guessing for all learning sets. However, as was shown earlier in this chapter, no generalizer can be self-guessing for any and all learning sets. Therefore no such unique intersection can exist.

To prove this statement more rigorously, first construct any allowed learning set $\theta' \supset \theta$. It is easy to see that $sg(\theta') \subset sg(\theta)$, since the semi-proper self-guessing expansion of $\theta \subset$ the semi-proper self-guessing expansion of θ'. Therefore $\Omega \cap sg(\theta') \subseteq \Omega \cap sg(\theta)$. Since by hypothesis $\Omega \cap sg(\theta')$ is nonempty (containing exactly one generalizer), and since $\Omega \cap sg(\theta)$ has only one element, in fact we must have that $\Omega \cap sg(\theta') = \Omega \cap sg(\theta)$ for all $\theta' \supset \theta$. But by definition of semi-proper self-guessing, all generalizers in $sg(\theta')$ reproduce θ', and therefore $\Omega \cap sg(\theta')$ reproduces θ'. In particular, if (x, y) is an element of θ' but not of θ, $\{\Omega \cap sg(\theta')\}(\theta; x) = y$. Since the value of y is completely free we can choose it to not equal the guess for question x $\{\Omega \cap sg(\theta)\}(\theta; x)$,

whatever this guess happens to be. Therefore we can pick $\theta' \supset \theta$ such that $\Omega \cap sg(\theta') \neq \Omega \cap sg(\theta)$, arriving at a contradiction. ∎

There are basically three other kinds of goals involving semi-proper self-guessing which can be used in place of the one just shown to admit of no solutions. The first of these is to not demand perfect self-guessing and have Ω narrow enough so that $\Omega \cap sg(\theta) = \emptyset$, the empty set, in general. In this case we would search for the generalizer obeying Ω which is closest to $sg(\theta)$ (i.e. which has the smallest self-guessing error for θ, for some appropriate definition of "error"). In fact, this is the goal that is usually strived for when self-guessing is used in practice. For example, one way to make use of the concept of self-guessing, which is used in chapter 3, is to pick a subset of the set of all generalizers, this subset being parameterized somehow, and then search for the parameter value(s) which minimize self-guessing error for the learning set in question. The subset of all generalizers could be a set of (parameterized) learning rules for teaching neural nets, for instance, and our goal would to find the particular learning rule for producing neural nets which is closest to self-guessing for a given learning set. The criteria set Ω in this case is simply the restriction to generalizers which are neural nets produced by one of the provided learning rules. The statistics technique of cross-validation [2], mentioned in chapter 1, is an unsophisticated version of this first goal.

As another example of a scheme based on this first goal, suppose that we have reason to believe that the input–output surface should be constructed from some set of basis functions. For example, we might know that that surface is periodic, and therefore we are interested in creating a guessing surface using a Fourier series. The simple-minded approach would be to fit the lowest n terms of the Fourier series to the n points of the learning set and use these n terms for the guessing. However there is no a priori reason to use the lowest terms of the basis set, especially since using other terms might give better self-guessing. Accordingly, an alternative scheme, which will be investigated in more detail in the future, chooses elements of the basis set so as to minimize the self-guessing error for guessing any single point in the learning set based on the predictions of $g\{n-1\}$ operating on the rest of the learning set. If the learning set is given by θ, then this scheme tries to find the $n-1$ basis functions $f_j(x)$ such that $\sum_{i=1}^{n}[g\{n-1\}(\theta-(x_i,y_i),x_i)-y_i]^2$ is minimized, where $g\{n-1\}$ is constructed by fitting the optimal $n-1$ $f_j(x)$ to the $n-1$ elements of the set $\theta-(x_i,y_i)$. The minimization can be done via steepest descent or any other scheme which is convenient. With self-guessing minimized this way, $g\{n\}$ can be fitting to the $n-1 f_j(x)$ along with the constant function, for example. If one does not have any a prior reason for picking a particular basis set, then the same idea can still be used, except that now the minimization is over a set of preselected candidate basis sets as well as over the $n-1$ $f_j(x)$ from each such set. A simple-minded version of this approach is elucidated at the beginning of chapter 3.

Note that it is possible to use generalization itself to search for a generalizer with minimal self-guessing error. Just build a generalizer which has as

its input space the specifications of a series of generalizers, and as its output space the associated self-guessing error for the learning set in question. For example, for a given learning set and set of basis functions, if we have a generalizer which we think guesses well the input/output surface taking (a set of indices delineating) $n-1$ functions from the set of the basis functions to the self-guessing error for those $n-1$ functions (for the given learning set), then by searching for input values for which this generalizer guesses a small output (i.e. by "inverting" this generalizer), we might be able to improve upon steepest descent schemes for finding the $n-1$ functions from the basis set which minimize the self-guessing error. (Steepest descent amounts to assuming that a local hyperplane fitter is a good generalizer for the problem.)

Alternatively, if we assume that we are in a region in which the $n-1$ (indexed) basis functions which minimize the self-guessing error are approximately linear functions of their index, then (if we make the admittedly gross assumption that the coefficients fitting the $f_j(x)$ to the subsets of the learning set are the same for all sets of $f_j(x)$) the self-guessing error is quadratic in those index values and we can use the optimization scheme of Hopfield and Tank [3] to find the $n-1$ basis functions which minimize the self-guessing error. For example, if the basis set is the set of monomials of order m, x^m, m indexes the basis set, and if we assume that m is small and x near 1 we can replace x^m with $1+m(x-1)$ so that (under the mentioned assumption) the self-guessing error is quadratic in m, and therefore we can build a Hopfield net to find the m which minimizes the self-guessing error. In the more realistic case where we make no assumptions about the coefficients fitting the $f_j(x)$ to the subsets of the learning set, we can still minimize the error as a function of the indices of the functions making up the basis set, but now we cannot use the scheme of Hopfield to accomplish this minimization.

A similar idea to generalizing a new generalizer using the tool of self-guessing is to use self-guessing itself to set the measure used for self-guessing error. We could take a subset of our full learning set and, using various measures of self-guessing error, build generalizers from that subset which we could then test on the rest of the learning set. In this way we could determine the "best" measure of self-guessing error. To generalize from the full learning set we would build a generalizer which has minimal self-guessing "error," where the measure used for determining that error is the one which, when used to build generalizers from the subset of the full learning set, guessed best the remainder of the learning set. Although it is probably easiest to just fix some standard measure of self-guessing error (like minimizing the sum of the squares of the errors) for the testing, the testing itself could be done according to the same measure of self-guessing error as was used to construct the generalizer, thereby imparting a sort of self-consistency to the whole process.

The second possible goal, similar to the first, is searching for the generalizer $\in sg(\theta)$ which lies closest to Ω, or, alternatively, finding a generalizer for whom the sum of the distances (suitably defined) to $sg(\theta)$ and to Ω is minimized. The third possible goal replaces Ω with a mapping from learning

sets to criteria sets, $\Omega(\theta)$, such that $\Omega(\theta) \cap sg(\theta)$ contains a single element. So long as $\theta' \supset \theta$ does not necessarily imply that $\Omega(\theta') \subseteq \Omega(\theta)$, we do not get the contradiction we did above. As yet this second possible goal has not been extensively explored. One example of a system which meets this second goal (though in a not particularly useful or interesting way) is provided by modifying the example given above of how to build a strongly self-guessing generalizer for any learning set. If we come up with a rule for constructing the function $f(x)$ from the learning set (e.g. use polynomial fitting), then $f(x)$ and therefore the whole generalizer is uniquely fixed by the learning set, even though the generalizer is strongly self-guessing for the learning set in question.

At this stage of these papers, in addition to laying out a preliminary scheme for categorizing generalizers, we have partially accomplished the goal of creating criteria for optimal generalization. One such set of criteria is any of the various permutations of the criteria in the previous paper. Another is to find generalizers (from a criteria set) which are semi-proper and as close to (strongly) self-guessing as is possible, for the learning set in question. We have answers to the question, "if you are presented with a learning set and have several generalizers to choose from to generalize from the set, and all factors like guessing speed and system overhead are equal, which generalizer do you use?" If we use this second choice of the criteria of semi-properness and (strong) self-guessing, then the task involved in optimally generalizing from a learning set is to find a mapping from the learning set, θ, to an equivalence class of generalizers, G, such that G is semi-proper and (almost) strongly self-guessing for θ. G is otherwise unrestricted, being set by a criteria set of external considerations (e.g. a desire to investigate learning algorithms for neural nets), except that every member of G agrees on the function $g\{\text{order of } \theta\}$, acting with θ as a learning set, taking questions to outputs. In other words, the only thing that concerns us about the elements of G is that they all be semi-proper, as close as possible to self-guessing, and unique in how they tell us to make guesses for novel inputs, given our learning set θ. In chapter 3 of this paper some problems that indicate that such concepts of approximate self-guessing are quite useful in real-world situations are explored.

Unfortunately, whereas the requirements of (semi-) properness, output linearity, and the like can usually be used to choose between competing generalizers, they do not seem to be sufficient to determine the unique optimal equivalence set of generalizers for any given learning set. They either under- or over-restrict that equivalence set. Self-guessing does not change this, and indeed, for exact self-guessing, we proved just above that any criteria set has to be either over- or under- restrictive. In trying to find a way out of this impasse, criteria in addition to (or perhaps instead of) those studied so far bear investigating. One such criterion, quite different from any of the restrictions discussed so far, is information compactification.

2. Information compactification

The criterion of information compactification is motivated by examining the criteria humans use in deciding between competing theories to explain a given body of data. The most obvious criterion, that the theory not disagree with any of the available data, is, of course, the same as the restriction on an generalizer that it reproduce the learning set. Another criterion used by humans is that the theory be self-consistent and able to explain certain proper subsets of the data as ramifications of the rest of the data. This essentially amounts to the criterion of self-guessing. Yet another criterion humans use is Occam's razor: of two competing theories, everything else being equal, pick the "simplest" theory, the one which is made up of the least amount of information. In other words, try to compactify the information of the body of facts as much as possible when recasting it as a physical theory.[3]

To investigate information compactification from the point of view of generalizers, first note that for a strongly self-guessing generalizer, all we are interested in is the mapping performed between the question and the output ($f(x)$ in equation (1.4)). This mapping will be the same for all elements of any expanded learning set. Define this mapping as consisting of two parts: a set of numbers called the *defining set*, and a *method* which maps the defining set to functions between questions and outputs. Via the method, the defining set uniquely specifies the question-output function (which is usually assumed to be from a subset of $\mathbf{R}^m$ to a subset of $\mathbf{R}$). The amount of information in this function is some appropriate measure of the size of the defining set. With M being a method and ϕ being a defining set, (M, ϕ) is taken to be the set of all pairs (question, output to the question guessed using M and ϕ). The goal of information compactification is to, from the collection of all (method, defining set) pairs meeting some set of criteria (like reproducing a learning set), find that pair which has the smallest possible information measure of the defining set. We assume that the allowable form(s) and elements of ϕ are implicit in M. Since a defining set will, in general, be an ordered set, $(M, (\phi_1 \cup \phi_2))$, for example, is not defined for $\phi_1 \neq \phi_2$.

A priori, it is not clear precisely what form the defining set should take and what kinds of numbers its elements should be. In the attempt to approximate information compactification and self-guessing via networks of HERBIEs discussed below, the defining set consists of a finite but variable number of real numbers (See reference [4] for discussion of a similar idea). The same is true for most neural net schemes. In that these real numbers, being approximated in a computer program, have limited precision, the defining set essentially consisted of a variable number of integers, and therefore the defining set could be further compressed into being a single integer.[4]

---

[3]Note that "information" here does not necessarily have any relationship with Shannon information, $-\sum_i \rho_i ln(\rho_i)$, as described in reference [9], for example. A closer analogue can be found in the definition of information used by Chaitin, described in reference [7].

[4]e.g., the (unique) product of the lowest prime numbers, each raised to the power of an element of the defining set. Insofar as the defining set in this example of HERBIE nets

To ensure that no information from a learning set is hidden in a method, thereby allowing us to reproduce the learning set using defining sets with arbitrarily (and meaninglessly) small amounts of information, it is required that for all methods, M, given any (finite) learning set θ, there exists a defining set ϕ such that the question-output function made from M and ϕ reproduces θ. In other words, any method must be able to reproduce any learning set, and therefore cannot have any data "hidden" in it pertinent to some particular learning set. Therefore we can write

(2.1) For all methods M, and for all finite learning sets θ, there exists a defining set ϕ such that $\theta \subset (M, \phi)$.

For (2.1) to be meaningful, it is assumed that there is some notion of the allowed learning sets which is independent of the method (e.g. "all elements of the learning set lie in the interval $[0.0, 1.0]$"). If this is not the case, then (2.1) is essentially a tautology.

Given any learning set θ, pick any method M' and then the associated defining set ϕ' such that $\theta \subset (M', \phi')$. A method is any mapping which takes a defining set to a function from $\mathbf{R}^m$ to $\mathbf{R}$ such that (2.1) is obeyed. Therefore we can construct a new method M which first performs a mapping on its defining set and then runs M' on that new defining set, so long as the pre-mapping is bijective. In particular, let the pre-mapping be the identity mapping, except that it sends a predetermined defining set ϕ to our set ϕ' and vice versa. Clearly, since M' satisfies (2.1), so does M. It is also immediate that $(M, \phi) = (M', \phi')$. We were given θ and ϕ and constructed the associated M which reproduces θ. We can therefore say that for all ϕ, and for all learning sets θ, there exists a method allowing ϕ's form such that θ is contained in (M, ϕ). The third variation of (2.1), "for all methods M and all defining sets ϕ allowed by M, there exists a learning set θ such that θ is contained in (M, ϕ) " is true by construction.

Sometimes the rules in (2.1) have to be amended so as to not run astray of cardinality arguments. For example, a defining set consisting of a finite number of integers can not obey (2.1) exactly for learning sets consisting of finite numbers of reals. In such cases the first rule in (2.1) is assumed to be modified to read "Over an arbitrarily large region, for all methods M, and for all learning sets θ, there exists a defining set ϕ such that a set of numbers which constitute an arbitrarily good (but not necessarily perfect) approximation to the elements of θ, is contained in (M, ϕ)." For these cases the second variation of (2.1) is modified in a similar way.

In general we say that

(2.2) The method M_1 = the method M_2 iff, for all ϕ, either $(M_1, \phi) = (M_2, \phi)$, or both (M_1, ϕ) and (M_2, ϕ) are not defined.

---

turns out to be an ordered set, the sizes of the prime numbers would have to be ordered in the same way as their powers, the elements of the defining set, are ordered.

Arithmetic operations over methods can now be defined quite easily, if desired. For example, we can say that $M' = M_1 + M_2$ iff, viewing (M, ϕ) as a function from $\mathbb{R}^m$ to $\mathbb{R}$, for all ϕ, either $(M', \phi) = (M_1, \phi) + (M_2, \phi)$, or all three (M', ϕ), (M_1, ϕ), and (M_2, ϕ) are not defined.

In analogy to (2.2) it turns out that

(2.3) If ϕ_1 and ϕ_2 are both finite sets of ordered reals, $\phi_1 = \phi_2$ iff, for all methods M which allow defining sets of the form of ϕ_1 and ϕ_2, either $(M, \phi_1) = (M, \phi_2)$ or both (M, ϕ_1) and (M, ϕ_2) are not defined.

Proof: Trivially, if $\phi_1 = \phi_2$, then either $(M, \phi_1) = (M, \phi_2)$ or both are not defined, for all methods M. Now define M' to be the set of methods which preprocess a defining set ϕ_i into a new defining set ϕ_i' and then map the elements of ϕ_i' to a curve in $\mathbb{R} \times \mathbb{R}$ made up of straight lines which connect the points (1.0, first element of ϕ_i'), (2.0, second element of ϕ_i'), etc. The line is taken to have the value of the first element of ϕ_i' for all points to the left of 1.0, and the value of the last element of ϕ_i' for all elements to the right of $n.0$, n being the number of elements in ϕ_i'. Varying through M' is adding new and different points to the right of the elements of ϕ_i in the preprocessing, thereby creating ϕ_i'. M' is in agreement with (2.1), where θ consists of elements in $\mathbb{R} \times \mathbb{R}$. Furthermore, is contained within the set of all methods which allow defining sets of the form of ϕ_i. Now note that if ϕ_1 and ϕ_2 have the same number of elements, then if for any one of the $m' \in M'$ $(m', \phi_1) = (m', \phi_2)$, ϕ_1 must equal ϕ_2. If they have different numbers of elements, then it is possible that for a particular $m' \in M'$ $(m', \phi_1) = (m', \phi_2)$ (have ϕ_2 be ϕ_1 with its rightmost element repeated, for example). However, it is clearly not possible that this is the case for all $m' \in M'$. Therefore if $(m', \phi_1) = (m', \phi_2)$ for all $m' \in M'$, ϕ_1 and ϕ_2 must have the same number of elements and must actually be identical. Since $(M, \phi_1) = (M, \phi_2)$ (or both are not defined) for all M implies the same for all $M \in M'$, the proposition is proven. ∎

A rule similar to (2.3) holds if both ϕ_1 and ϕ_2 are unordered.

Along with the defining set and the method a measure is needed of the information contained in the defining set. For a defining set ϕ, such a measure is written $I(\phi)$. (Sometimes $I(\phi)$ is called the simplicity measure although in this paper we'll usually use the term information measure.) Since it is not in general clear what form the defining set should take, it is not clear what the information measure should be. As an example, one means of measuring information would be to have the method be a universal Turing machine used to build multitape Turning machines which calculate functions taking a string on one of the tapes as input with the final string lying on another of the tapes being interpreted as output. To allow these functions to be from reals to reals, it is sufficient to have the period be a symbol on both the input and output tapes and to add a delineated precision number to the beginning of the input tape such that when the answer on the output tape has precision exceeding this number, the Turing machine halts. With this system, any (halting) function from the reals to the reals can be calculated

by taking the limit as the precision number goes to infinity of the input tape / output tape function calculated by the Turing machine. The defining set in this system is the code read by the universal Turing machine, and the information measure is the number of states in the resultant Turing machine (see reference [32] of the first paper for investigation of a similar idea).

As another example, in the hyperplanar HERBIE networks discussed below, the defining set is a set of numbers specifying both the architecture of the network and the learning sets defining the individual hyperplanar HERBIEs (these learning sets not to be confused with that learning set θ used to teach the entire net). A hyperplanar HERBIE is a particular kind of HERBIE based on locally fitting subsets of the learning set with hyperplanes. For brevity, we will often use the term "HERBIE" when what is really meant is this particular kind of hyperplanar HERBIE. The context should make the meaning clear. (See reference [5].) Here the information of a defining set can be taken to be the number of elements in it. In this particular instance the same measure can be applied to the learning set fed to the net as a whole (we require that the entire network reproduces this learning set). Now examine the trivial case of a net consisting of a single HERBIE, so that the question-output function for the whole net is just the single HERBIE "generated" by the learning set used to teach the full net. Since the learning set of this HERBIE is the defining set, the information in the defining set equals the information in the learning set fed to the entire net. We can therefore state that for this information measure, in general $\min(I(\phi)) \leq I(\theta)$, where ϕ is a defining set and θ is the learning set. The minimum is taken over all defining sets which reproduce θ. As a trivial example of information compactification, if θ consists of $n \gg m$ elements all lying on a single m-dimensional hyperplane, the criterion of information compactification would (probably) pick a single HERBIE defined by a learning set consisting of $m+1$ of the elements of θ to generalize from θ. This is because in this case such a single $m+1$ element HERBIE is (probably) the HERBIE net with the lowest $I(\phi)$ which reproduces θ. (For all the elements of θ not lying on the same hyperplane, no $m+1$ element HERBIE would reproduce θ.)

Since the information measure is supposed to mesh smoothly with Occam's razor, it is usually assumed that the higher the value of $I(\phi)$, the greater the number of defining sets with information measure equal to $I(\phi)$, and therefore, given a method M, the higher the number of functions in (M, Φ), where Φ is the set of all defining sets with the same information measure as ϕ. In other words, given a fixed method M, the lower the information measure I, the more constrained the set of all functions producible from M by using M in conjunction with defining sets of information measure I. More functions can be created from a set of definings sets of high information measure than from a set of defining sets of low information measure (see reference [6]). Most, if not all, information measures considered in practice obey this restriction. Stating this restriction more rigorously, $I(\phi)$ is a monotonically increasing function of the number of defining sets with the same information measure as ϕ.

Formally speaking, a generalizer is a continuous map from learning sets θ to guessers, (M, ϕ). Since the guesser (M, ϕ) is required to obey (0.1) through (0.5) (and in particular (0.3)), the method M necessarily obeys (2.1) and its ramifications ((2.3) for example). Self-guessing is a further restriction on this map to ensure certain properties when it is run on subsets of θ. For strongly self-guessing maps, this restriction is particularly simple: any (large enough) subset of θ gets mapped to the same guesser, (M, ϕ). Given a learning set θ, the task for finding the optimal generalization is now amended to be finding the equivalence set of generalizers which is (semi-) proper and approximately (strongly) self-guessing for θ, uniquely defined for $g\{\text{order of } \theta\}$, and for which the guesser (M, ϕ) induced by the learning set on the function $g\{\text{order of } \theta\}$ has minimal $I(\phi)$.[5]

In point of fact, a generalizer itself is a kind of method; it takes defining sets (i.e., learning sets) to mappers from questions to outputs. As such one can view the search for a minimal defining set ϕ which will reproduce a learning set θ as being equivalent to trying to find a minimal learning set (ϕ) which, when fed through the generalizer, will guess the original learning set θ. This, of course, is quite similar to the task of self-guessing in its various forms. Note, however, that this relationship between self-guessing and information compactification also means that all the limitations on self-guessing discussed in the last chapter, and in particular the limitations presented in the discussion of criteria sets, have applicability to schemes involving information compactification as well.

Finally, another way to exploit information compactification is to view a generalizer itself as being a method along with a defining set (rather than just a method), and thereby cast the task of optimally generalizing θ as that of finding the generalizer, (M, ϕ) which has minimal $I(\phi)$ and which is (semi-)proper and approximately (strongly) self-guessing for θ.

The difficulty in all these uses of information compactification lies in specifying an a priori most reasonable information measure. Unfortunately, we seem to need to specify an a priori most reasonable method to go along with this a priori most reasonable information measure. Consider, for example, the case where only the information measure is fixed and given and we are trying to minimize the information measure of a defining set which, in conjunction with a method, constitutes the guessing surface going through the

[5]Note that there is a natural meshing between the concepts of self-guessing and information compactification in that self-guessing means that a subset of θ might be able to replace the full θ, and this in turn might result in a lower $I(\phi)$. After all, if you can guess a full learning set from a subset of it, then you have effectively compacted the information needed to reproduce the full learning set, assuming your information measure of a set is simply the cardinality of that set. For example, if the input space were m-dimensional all the points in the learning set lay on a hyperplane, and if M were a hyperplanar HERBIE (see two paragraphs ago) working on points provided by ϕ, with $I(\phi)$ being the number of such points, the ϕ with the minimal information would consist of only m points, and linear fitting to those points would suffice to reproduce the entire learning set hyperplane. This corresponds exactly with the fact that linear fitting is self-guessing for this learning set of points on a hyperplane, since this means that a subset of that learning set can be used to generate the entire input/output surface.

points of the learning set. In this scenario, very slight modifications to the method can result in completely different predictions. Using method M, if (M, ϕ) and (M, ϕ') both reproduce the learning set and $I(\phi) < I(\phi')$, then we would be led to fit the points of the learning set with the function (M, ϕ). Now let M' be a new method identical with M except that if the defining set is ϕ, M' first replaces it with ϕ' and then performs the same mapping M does, and if the defining set is ϕ', M' first replaces it with ϕ and then performs the same mapping M does. The information measure has not been changed, but now we'd be led to generalize with $(M', \phi) = (M, \phi')$, getting the exact opposite results from before. If we had some way to pick between M and M' we would not have this difficulty; barring such a way to choose, we can not derive unique generalization of our learning set using only the criterion of information compactification.

Things are a bit better if it is the method which is fixed and given, since then it is fairly reasonable to take the cardinality of the defining set as its information measure. However note that in no sense is one justified in assuming such a choice of measure is a priori "most" reasonable. For example, it is trivial to modify any method M very slightly into a new method M' which takes only cardinality 1 defining sets and operates by mapping any such single number into a set of numbers on which it then runs the method M. In this way an information measure based on the cardinality of the defining set loses all ability to distinguish amongst the various defining sets. Note also that whatever the method, it is always possible to modify the information measure slightly by interchanging the information values associated with any two defining sets (this modification does not run astray of the requirement that information measure scales with the number of defining sets having such a measure). Such a modification, slight as it is, can be used to justify the choice of *any* of the defining sets which (along with the fixed method) reproduce the learning set. In general then, it seems that unless both the method and information measure are engraved in stone, the criteria of information compactification simply is not capable (either by itself or in concert with some of the other criteria of generalization theory) of determining how to generalize from a learning set. Yet if one is to fix a method and information measure, it should be clear why such a choice is definitely the best one to make. Otherwise, there is no reason at all to expect the resultant generalization to be optimal.

It is interesting to note that despite these problems in justifying the choice of method and information measure, most of the generalizers (which aren't neural nets) that have been explored to date have tried to use information compactification rather than any of the criteria in chapter 1 of the last paper for choosing between different generalizing schemes. For example, one kind of information measure that has come under investigation is the number of high-level language statements necessary to reproduce the learning set (see reference [49] of the first paper of this series). Similarly the work at minimizing the algorithmic information complexity of a TM that reproduces a given learning set [7] uses essentially the same information measure as in the

TM example given above. As a final example, even Rissanen's "minimum description length principle" [8] can be viewed as an attempt to exploit the concept of information compactification. (Rissanen's scheme can also (perhaps more naturally) be viewed as a crude attempt to minimize a kind of self-guessing error[6]).

Unfortunately, none of the information measures associated with these schemes is clearly "a priori most reasonable." As a result, as indicated above, all of these schemes are essentially ad hoc, despite their apparent reasonableness.

Many attempts have been made to formulate a comprehensive analysis of the mathematics of self-guessing and information compactification. Usually they have entailed modeling these concepts as something better understood:

1. Information theory and statistical mechanics have been explored with an eye to better understanding how to measure information and how to then minimize it.

2. Information compactification has also been explored through expressing a guesser as a polynomial of polynomials, and then trying to minimize the total number of coefficients necessary to guess a given learning set.

3. For self-guessing, one approach that has been investigated views generalizers as surfaces in $\mathbf{R}^n$ evolving in time according to functions which give the height of the surface at a given point at a given moment of time as a functional of the heights everywhere else at the preceding moment. This is essentially a generalized network structure; neural nets are such structures which confirm to a rather narrowly proscribed set of allowed evolving functions. In the discrete limit, this approach has led to the investigation of Markov chains as models of generalizers.

[6]That information compactification should be somewhat equivalent to self-guessing should be no surprise, given that, as discussed previously, the reverse is true. In Rissanen's scheme the learning set is assumed to be one-dimensional, and the output components of the learning set are called "observations," with the (one-dimensional) inputs being viewed as the successive "times" of the observations. Rissanen's scheme for picking a (conditional, probabilistic) estimator to predict the next element of a series of observations from a set of previous observations can be viewed, roughly, as simply picking the generalizer with the smallest self-guessing error for the learning set of those previous observations. For Rissanen the "self-guessing error" for a particular element of the learning set, given a (distinct) subset of the learning set from which to generalize, is essentially just the negative of the logarithm of the conditional estimate of that particular element of the learning set, given all previous elements of the learning set from which to build the conditional estimate. In trying to find the estimator which minimizes the sum of these negative logarithms (up to an additive penalty term measuring the number of parameters of the estimator) Rissanen is simply trying to find the generalizer which minimizes this measure of self-guessing error. In the terminology of the section of the self-guessing chapter in this paper which dealt with criteria sets, Rissanen is simply following the first of the three possible goals involving self-guessing, namely restricting himself to a criteria set Ω and then finding the element of Ω which lies closest to $sg(\phi)$, where ϕ is the learning set in question. Finally, note that since, in practice, Ω is set by restricting oneself to some arbitrary (and small) set of estimators, Rissanen's scheme is, essentially, ad hoc. It is model-driven, and not purely generalization criteria-driven, despite appearances to the contrary.

In the continuous limit, it has led to the investigation of certain kinds of integral equations and of field theoretic path integral formalisms as models of generalizers.

4. Another approach investigated was to expand each $g\{i\}$, perhaps in a Taylor series, and then try to relate the expansion coefficients of the $g\{i\}$'s making up an generalizer.

5. Other approaches are assorted variational principles, like those which determine the shape of a sheet of stiff material forced to go through certain points in space,[7] or maximization of (Shannon) missing information [9] subject to various constraints (the probability distribution ρ_i in the $-\sum_i \rho_i ln(\rho_i)$ being over possible input–output functions).

6. Others are taking the inputs and outputs to be from spaces with higher cardinality than the real numbers (i.e. taking them to be functions) and then perhaps applying distribution theory [10] or perhaps finding the (function-valued) coefficients of a Taylor expansion which reproduces the learning set.

7. Other lines of attack have been investigating input–output curves created by having both the input and the output determined by low-order differential equations in a common parameter, and viewing the learning set as a probability distribution over an input–output space, rather than as a set of vectors, thereby obviating the need for an infinite number of $g\{i\}$ (in this scheme all questions are answered according to a single diffeomorphism mapping the learning set probability distribution surface to a new probability distribution surface which gives the output guessed for any given input).

8. Yet another approach that has been investigated has been trying to find an (appropriate) continuous model of computation, the hope being that solving extremal problems with such a model would be easier than solving them with (discrete) Turing Machines.

9. Along the same lines, various attempts were made to construct a meaningful notion of a function with a fractional number of arguments, so that the criterion of upward compatibility could in some sense be made continuous rather than referring to $g\{i\}$ with i's differing by an integer amount.

10. Finally, a preliminary investigation has been made into a scheme involving having a point move through an appropriate space, perhaps chaotically, until it alights within a predetermined boundary in that space. The starting position of the point's journey is the input to the generalizer, and the ending position within the boundary is the output. The scheme for turning this into a generalizer would be to alter

[7] This is essentially the same approach as regularization theory (see reference [32] of the first paper).

the parameters controlling the jumping of the point so that it can both reproduce a given learning set and approximate self-guessing for that learning set.

So far, none of these purely analytical approaches to the problem of specifying unique generalization has been entirely successful. As a result, a fair amount of work has been done outside of such an analytic context by exploring the reasonableness of the criteria and concepts of generalization theory as solvers of real generalization problems. This work consists of computer runs of structures which approximate solutions to various of the criteria of generalization theory. In particular, the criteria of semi-proper self-guessing and information compactification have been approximated. Although research continues at trying to find analytical solutions to generalization theory, these computer runs have exhibited some quite encouraging generalization behavior, and are therefore interesting in their own right.

3. Processors, nets, and applied generalization theory

3.1 Applications of generalization theory which do not use networks

The first structures based on generalization theory which were investigated in real world situations were hyperplanar HERBIEs (see reference [5]). The hyperplanar HERBIE, unlike backpropagated neural nets, is very close to being a proper generalizer (discontinuities being how it falls short of full properness). As explained in reference [5] there are many other advantages to this HERBIE as well. Accordingly, better generalization would be expected of such a HERBIE, and in all the tests reported in reference [5] of the guess-the-function-I'm-thinking-of type, the hyperplanar HERBIE has indeed beaten neural nets, sometimes by a wide margin. A simple hyperplanar HERBIE, however, employs neither self-guessing nor information compactification.

One could, for an arbitrary learning set, take all generalizers, including HERBIEs and neural nets, see which one achieves greatest information compactification and is most closely self-guessing for the learning set, and use that generalizer to generalize. This is essentially the tack taken in reference [11] in which the task of reading English text aloud is attacked by building generalizers via the technique of self-guessing. The reasonableness of this approach is evident in the ease with which it leads to generalizers far superior to NETtalk, the generalizer Sejnowski and Rosenberg constructed for the same task via the technique of backpropagation [12]. Not only did this approach beat NETtalk in terms of ease of use, speed of construction, error rate for reproducing the learning set, and simplicity and manipulability of operation, it far outperformed NETtalk in terms of generalization error on testing sets.

Another example of the efficacy of this self-guessing approach is provided by the the following toy problem. Take the first eight frequencies 2π, π, $2\pi/2$, $2\pi/4$, etc. You are told that a (random) linear combination of one-

dimensional sine waves with these frequencies has been used to construct a parent surface. Five points are randomly chosen from this parent surface and presented to you as a learning set. You are then given novel inputs as questions and are asked to predict what the corresponding output on the parent surface is for these questions. Since five is less than eight, you cannot just fit the eight frequencies to the points of the learning set. How can you generalize in an intelligent manner?

One way to generalize intelligently is to exploit self-guessing to decide which five out of the eight total frequencies to fit to the points of the learning set. (See the discussion of criteria sets in the last chapter — what is being done here is a form of the first of the three realizable self-guessing goals mentioned in that discussion.) Choose all possible sets of four points from the five of the learning set. For each such quadruplet fit a set of four of the eight (known) frequencies to those points, and measure the square of the error of the resultant four-term linear combination in guessing the remaining fifth point. Then sum up these errors over all the quadruplets. This gives us a total guessing error for those four frequencies, for the learning set provided. Follow this procedure for all sets of four of the eight frequencies. Then rank all these sets of four frequencies according to this total self-guessing error. Set aside the four frequencies in the best performing set. Then find the remaining frequency which has the smallest average self-guessing error, where the average is over all four-frequency sets in which that frequency appears. This gives us a total of five frequencies which are in some sense best at self-guessing. Using them we can fit all five points of the learning set. It is this resultant five-term linear combination which is then used to make guesses for new points. In effect, this procedure is a way of constructing an LMMG (see the first paper in this series) which is approximately self-guessing.

In figures 1 and 2 the generalizing efficacy of choosing the 5 frequencies in this manner is compared to the efficacy of choosing the 5 frequencies in a random manner. 10,000 trials (approximately) were conducted in each of which eight amplitudes for the eight frequencies were randomly chosen (from between -1 and $+1$), a 5-point learning set and a 20-point testing set were randomly chosen, and then the self-guessing scheme elucidated in the previous paragraph was used to construct a generalizer from the 5-point learning set.

Figures 1 and 2 are made up of two histograms. In figure 1, each bin represents the number of 5-frequency sets (out of a total of 56) that generalize better than the set chosen by self-guessing for a given trial. If self-guessing chose what were effectively random frequencies, then the histogram would be symmetric about bin 27. The more biased toward the lower bins the histogram is, the better self-guessing is performing as a technique for choosing a generalizer of the 5-point learning set. In figure 2, the bins represent ranges for the difference between the generalizing error for the frequency set picked out by self-guessing and the average such error over all frequency sets. Only the central part of the histogram is given in this figure, because the full histogram has exceptionally long tails. (Indeed, the average across

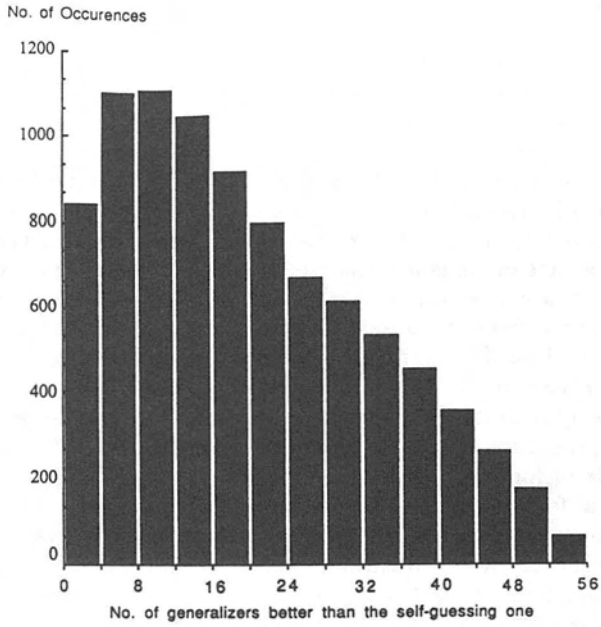

Figure 1: This histogram has as its bins the number of 5-frequency generalizers (out of 56 altogether) which had smaller guessing error than the 5-frequency generalizer chosen via self-guessing. 8948 trials went into making up this histogram. The binning is done in groups of four. For example, having two generalizers better than the self-guessing one and having three generalizers better would both contribute to the left-most, 0–3 bin. The guessing errors were determined via random 20-element testing sets. The advantage of choosing the (approximately) self-guessing generalizer is reflected in the fact that this histogram is centered about a value smaller than 56/2.

the histogram (i.e., the average over the 10,000 trials) of the error difference is on the order of 10^{16}, 10^{11} times the range of the histogram beyond the right edge of that portion of the histogram reproduced in figure 2.)

The results of figures 1 and 2 are fairly unambiguous. Self-guessing has helped substantially in finding which of the five frequencies to use for generalizing. For this problem, self-guessing is clearly a technique helpful in trying to pass the guess-what-function-I'm-thinking-of test.

Nonetheless, there are tasks for which all the usual generalizers, including HERBIEs, approximate self-guessers, and all the rest, fail miserably this guess-what-function-I'm-thinking-of test, and indeed, it is not too difficult

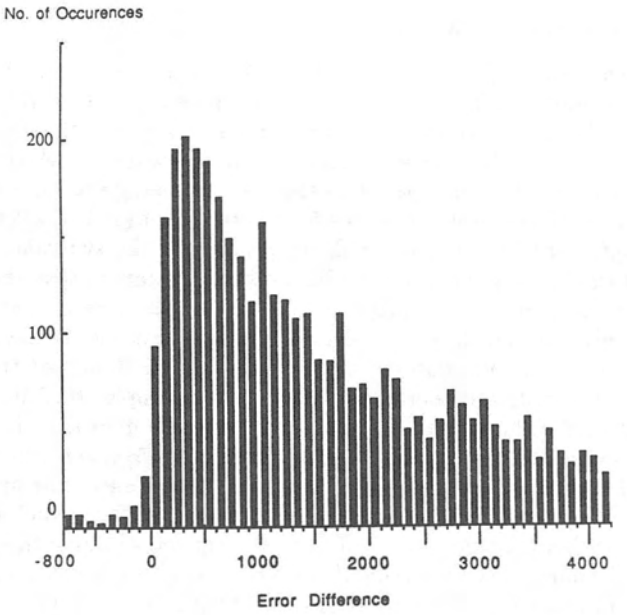

Figure 2: This histogram has as its bins the average guessing error of all 56 5-frequency generalizers, minus the guessing error of the 5-frequency generalizer built via self-guessing. The guessing error is determined by means of a random 20-element testing set. 8948 separate trials went into making this histogram. Here the advantage of choosing the (approximately) self-guessing generalizer is reflected in the fact that this histogram is weighted toward the positive bins.

to construct such tasks. By analyzing such tasks with an eye to a more systematic exploitation of self-guessing and information compactification, it has proven possible to create structures which exhibit richer generalization behavior than the structures investigated in [5] and [11], and which also provide insights into how to further exploit generalization theory in the real world.

This chapter provides an initial exploration of some such structures. Note that in investigating these structures, and indeed in investigating any real-world generalizers, the goal of model-independence is of necessity being compromised somewhat. We are no longer conducting the research in a purely generalization behavior driven manner. However, in that the structures presented in this chapter are at least based on the concepts of a model-independent conceptualization of generalization, they are certainly less ad

hoc (and therefore better tools for research) than the alternative generalizing structures (like neural nets) which are often encountered in the literature.

3.2 Processors vs. mappers

One problem which gives both HERBIEs and neural nets difficulties is the scanning problem. This problem has as its parent function the mapping which takes both a sequence of numbers and an offset into that sequence as input, and outputs the first nonzero entry in the sequence which is to the right of the offset. For example, with the first slot being the value of the offset, the input $(1,0,0,5,0,0,6)$ gives 5 as output and $(4,0,0,5,0,0,6)$ gives 6 as output. Another similarly difficult problem is the switching problem: according to the value of the first dimension of the input, take the value of one or the other of the remaining dimensions for the output. For example, $(1,3,4,5)$ gives 3 and $(2,3,4,5)$ gives 4.[8] A final example is the find-the-0 problem: given n input coordinates, one of which is a 0, output the number of the input coordinate containing the 0. For example, $(6,7,0,2,9)$ gives an output of 3. The hyperplanar HERBIE actually does surprisingly well, with the switching problem at least (see reference [5]), but will always be hampered by certain properties of these types of problems. The hyperplanar HERBIE is a purely local generalizer — a parent surface which cannot be predicted without looking at all of the learning set will give this HERBIE problems.[9] Difficulties will occur whenever a Euclidean metric does not correctly break up the space into sections of similar input–output mapping (as when there is an ordinal relationship between successive input components), giving the surface a discontinuous nature. Simple interpolations, or in general any scheme which tries to create nonvolatile surfaces, just is not going to key in on the tricks defining these kinds of surfaces.

One alternative generalizer which is better suited to these problems is a feedback network of hyperplanar HERBIEs (or just "HERBIEs" for short) which uses output flagging.[10] By having the number of iterations of the generalizing structure vary with the input, and by having data fed back through the structure, such a net amounts to many different conventional generalizers (i.e., many different feedforward nets), each run on a different

[8] As an aside, note that the technique of dividing up the input space into subsets each of which are self-guessing for hyperplanar HERBIEs (see end of reference [5]) would solve this problem perfectly. The elements of each subset would be those points of the learning set which all share the same value for the first dimension of the input. The resultant surface generated by any of these subsets would be a single hyperplane. For example, the subset with first input dimension = 1 would output the value of dimension 2, regardless of all other dimensions. This is a hyperplane, and, it so happens, the "correct" surface according to the guess-the-function-I'm-thinking-of criterion.

[9] It is precisely this global information that self-guessing is designed to exploit.

[10] "Output flagging" means that the output of the net is the value on a certain output line at the iteration for which the value of an output-flagging line meets some pre-set criterion. The number of iterations for which an output-flagged net runs before giving its answer will, in general, vary with the input. In practice, the output-flagging line is often the same as the output line, and the output criterion is that the value on this line exceed a pre-set threshold.

section of the input space. If the output flagging is done properly, one would expect a high degree of information compactification in such a network. This kind of structure will be generically referred to as a *processor*, as opposed to a *mapper*, which uses a fixed number of iterations to perform its calculations. An array matching all possible inputs to the associated outputs is an example of a mapper, whereas any computer program which polls conditionals and then branches accordingly, perhaps to an earlier part of the program (e.g., a Turing Machine), is a processor. It is always possible to cast a (terminating) processor as a mapper, and vice versa (i.e., the total recursive functions can be generated by Turing Machines). However, it is usually easier and more (informationally) compact to do any computation with a processor rather than a mapper. Indeed, close to 100% of all computer programs ever written have been processors rather than mappers.

An example of a a feedback, output-flagged net which solves the scanning problem, yet consists of only three (hyperplanar) HERBIEs, is shown in figure 3. Only the first dimension of the input, the offset value, is fed in to the bottom of the net. The rest of the input, the sequence of numbers to be scanned, is entered via an environmental HERBIE.[11] The value coming out of the counter HERBIE increases by 1 from one iteration to the next. This then gets added to the input in the summing HERBIE to feed points successively 1 more to the right of the starting offset into the environment HERBIE. For the whole net, output is flagged when the last HERBIE gives a nonzero output. Therefore, the whole net is only flagged after the summing HERBIE has scanned up to an input which causes the environment HERBIE to output a nonzero.[12] The last HERBIE is actually superfluous for this perfect solution, but will be useful later in approximating self-guessing solutions to the scanning problem.

Note the pronounced similarity between the net in figure 3 and the flowchart a human would use to solve the scanning problem. Indeed, the

[11]The "environment HERBIE" is simply a compact way of introducing the sequence of numbers into the network. It is a pre-set one-dimensional HERBIE with data vectors constructed so that if an integer is fed to it as input, it gives the value in the corresponding slot in the number sequence as output. For example, if the number sequence to be scanned is $(0, 0, 4, 0, 0, 6)$, then the numbers 1, 2, 4, or 5 fed into the environment HERBIE will produce an output of 0, whereas feeding in 3 will give a 4 and feeding in 6 will give a 6. Given that in reality a HERBIE takes real numbers as inputs, not integers, the environment HERBIE is built so that for example any real between 3.0 and 3.99 will give an output of 4.0 and any real between 6.0 and 6.99 will give an output of 6.0. In general, the environment HERBIE should not be viewed as on the same footing as the rest of the net. If a new number sequence is desired, the old environment is pulled out, and a new one representing the new sequence is slotted in. The rest of the net remains unchanged.

[12]Note that use of a environment HERBIE to input a sequence of numbers allows for extreme compactification of the number of input dimensions to the net. It also renders the size of the environment (the number of elements in the number sequence) arbitrary and irrelevant. In contrast, feeding the number sequence in to the net via a set of input nodes hard-wires in the size of the environment. As a result, any such net which solves the scanning problem for an environment of one size is essentially useless when trying to solve the problem for an environment of a different size, in marked contrast to the situation here.

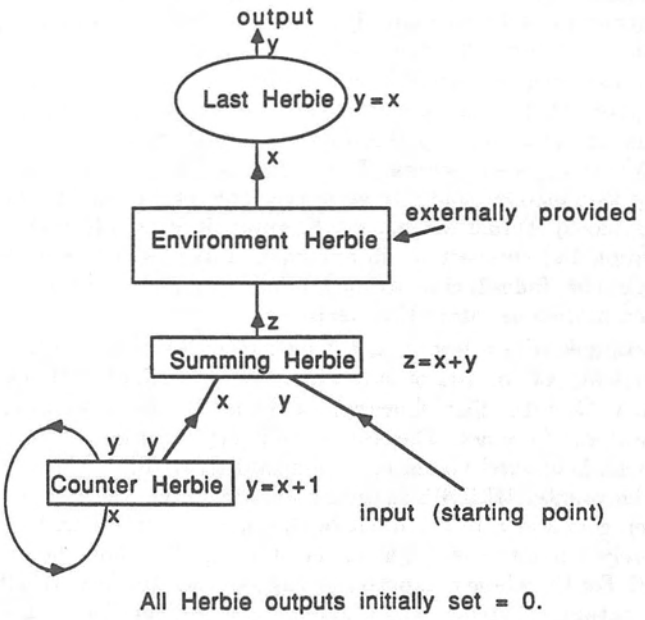

Figure 3: A network of one-dimensional HERBIEs which solves the scanning problem perfectly. The mapping performed by each HERBIE as well as its purpose is indicated along with the communication lines for the entire network. Note the feedback in the network connectivity.

simplicity of the human's flowchart is directly reflected in figure 3. Also note that it is very difficult (to say the least) to build a conventional, feedforward, fixed-number-of-iterations net which solves the scanning problem at all, never mind one which solves it using a net of only several nodes. This is true whether the nodes of the net are HERBIEs, "neurons," or anything else. Even if one introduces the number sequence via an environment HERBIE (rather than as numbers fed, along with the offset, to the input nodes of the feedforward net), there still is not any architecture nearly as simple as that of figure 3 that solves the scanning problem for feedforward, fixed iteration nets. What is more, for neural nets the integration of the environment as a HERBIE has a rather sizable drawback: it is not clear that there exists any finite feedforward neural net which can perform the scanning task for any and all environmental HERBIEs. This despite the fact that any partial recursive function can be calculated by a neural net. (Without feedback and output flagging, different environmental HERBIEs require network

structures feeding into them that perform different mappings of the input. However, the net has no way of knowing what environment is in there, and therefore has no way to modify its mapping accordingly.) Note also that feedback, output-flagged nets are a lot more realistic from a neurobiological point of view than feedforward nets with a fixed number of iterations. In addition, features like internal memory and self-modifying behavior are more readily modeled with feedback, output-flagged nets (by using structures like the counter HERBIE in figure 3, for example) than with conventional neural nets. Finally, note that feedback, output-flagged nets allow for a lot more information compactification than feedforward nets.

It is important to realize, however, that any single net, feedback and output-flagged or not, is not by itself a generalizer. A single net is incapable of reproducing an arbitrary learning set, and therefore it is necessary to have some structure which can modify the net to have it reproduce the provided learning set. For conventional, fixed-number-of-iterations, feedforward neural nets, the additional structure necessary to have a generalizer can be the technique of backpropagation. Given a learning set, the backpropagation technique produces a net which (approximately) reproduces that learning set. In this way, equation (2.1) is satisfied. Unfortunately, backpropagation does not work for feedback, output-flagged nets and rarely achieves perfect reproduction of the learning set anyway. For network structures which are not purely feedforward, Hopfield's procedure for building associative memories might be used to give us a generalizer, except that associative memories are ill suited to serving as generalizers due to the restrictions on their possible outputs. (What is more, Hopfield's procedure will work only so long as the learning set meets certain very restrictive criteria.)

For a feedback, output-flagged net, with no restrictions on the learning set, a novel technique is needed to provide us with a generalizer. One such technique is to run the net's output, when flagged, through a last, postprocessing HERBIE. The output of that last HERBIE one iteration after output is flagged is taken to be the output of the entire processor. Given any learning set, the data vectors of the last HERBIE can be set so that the entire processor, including that post-processing HERBIE, reproduces the learning set, thereby satisfying requirement (2.1) and giving us a generalizer (which is also a processor).

Note that this new generalizer lacks some of the practical advantages of a single HERBIE. This new generalizer will, in general, not satisfy the criteria defining a HERBIE. In addition, generalization is no longer completely overt and under the researcher's control (though certainly more so than with conventional neural nets, even feedforward ones). On the other hand, it is now possible to build a self-guessing generalizer. Our generalizer, which already possesses a fair degree of information compactification, can be modified to be approximately self-guessing as well. To see this, note that if you vary the architecture of the HERBIE net feeding into the last HERBIE, you are effectively altering the generalizer. While reproduction of the learning set is always achieved (and ensured) by setting the last HERBIE, variation of the

net feeding into that last HERBIE varies the guessing for points not in the learning set. In this way it is possible to vary the architecture until we find one which, for a given learning set, is close to self-guessing (i.e., is such that if we set the last HERBIE by running a subset of the learning set through the net, the entire resulting processor guesses well for the remainder of the learning set). In an abstract sense, we vary the generalizer until, if fed in a subset of the whole learning set, the generalizer guesses the remainder of the learning set. As an example, note that the architecture of figure 3 run with the last HERBIE determined by the learning set will, in setting that last HERBIE to reproduce any subset of a scanning problem learning set, cause that last HERBIE to be the identity mapping, as is indicated in figure 3. As a result, used with the bottom three HERBIEs of the net in figure 3, this generalizer is perfectly self-guessing for any scanning problem learning set. In addition, that last HERBIE, being the line $y = x$, needs only two data vectors to be completely specified, regardless of the size of the learning set. Therefore, this processor also achieves a high degree of information compactification.[13] Indeed, even with a fairly small learning set of solutions to the scanning problem, it is clear that no other net of any kind will be self-guessing and also have the degree of information compactification of the net in figure 3. Therefore, using those criteria, to generalize from the learning set we would be led to pick the net in figure 3, a net which just happens to be perfect according to the criterion of guess-the-function-I'm-thinking-of as well; this net generalizes perfectly.

3.3 Evolution vs. learning

It is one thing to deal with problems straightforward enough so that simple nets like that of figure 3 can solve them well, but building such a net by hand for arbitrary learning sets can be a prohibitively difficult task. As a result, the crux of this whole procedure is how the "variation" finding an approximately self-guessing HERBIE net is done. Even just when setting the last HERBIE, both the HERBIE net feeding in to the last HERBIE and the entire net including the last HERBIE act as processors, with output flagged only when a communication line achieves a certain condition. The number of iterations needed to run these processors to get output is not known beforehand and will change as the net itself changes. As a result, backpropagation will not help us to find a net that, fed into another HERBIE, forms a self-guessing structure for a particular learning set.[14]

[13]Note that in general, unless the last HERBIE turns out to be describable with fewer data vectors than are in the learning set for the entire processor, a single HERBIE necessarily needs less information to reproduce a given learning set than a net of HERBIEs feeding into a last HERBIE. The hope is that for a self-guessing processor net the last HERBIE will usually turn out to be particularly simple and therefore require few data vectors. In any case, even if this processor net scheme is worse at information compactification than single HERBIEs for a particular learning set, unlike single HERBIEs it can approximate self-guessing for that learning set.

[14]Note, however, that if the output is taken to occur after a fixed number of iterations (as opposed to via output flagging), then we effectively have a purely feedforward net (see

To understand what procedure should be used to direct the variation it is first necessary to digress and note that the proper paradigm for all current net research (including research on conventional neural nets) is evolution, NOT "learning." Learning is when a fixed system responds to external inputs by modifying an information storage structure. The stored information is easily erasable — the learning can be done again, quickly, superceding the original lesson. The system controlling the learning cannot be changed, however. A computer, for example, is a fixed system which learns by modifying its memory. Similarly, short-term learning by the human brain (i.e., short-term memory) is so quick that the physical architecture of the brain presumably remains unchanged throughout the learning process. For long-term memory, the fixed system consists of the rules of neurobiochemistry, with the physical architecture of the brain now being the storage device. Biologically speaking, learning refers to the ability of an organism to store and modify extra-genetic information in an easy, repeatable, temporally continuous manner. In evolution, on the other hand, there is no fixed system in the organism modifying an information storage structure. Rather, it is the fixed system itself that is modified. Evolution deals with changing the hard-wired behavior and instincts of an organism. In evolution, it is genetic, not extra-genetic, information which is modified, and it is done so in a highly noncontinuous manner (once per organism, at conception). In addition, in evolution it is highly nontrivial to find the modification to the stored information (the genotypic change) which will result in the new desired behavior (the phenotypic change). This is in contrast to learning, where the modification the fixed system applies to the information storage structure is practically instantaneous and highly accurate.

By definition, then, a (supervised) network by itself does not learn. It does not even have a modifiable information storage structure, never mind the ability to perform such a modification. The best that can be said is that a net, in conjunction with an associated (fixed) system which modifies the net to behave in accord with an externally provided learning session, can be viewed as a very cumbersome memory. This is the limited sense in which a Hopfield net can be said to "learn." Things become even more forced with schemes like backpropagation, in which the the tweaking of the weights is so involved that the "learning" cannot be continuous. What is more, such net-altering schemes clearly have little if anything to do with how a real brain learns. They can make biological sense, but only if they are viewed as forming their nets once and only once, at conception.

---

reference [23] in the first paper of this series). If we then choose the surface-fitting of all the HERBIEs in the net to be analytically differentiable functions of their data vectors, we could use the technique of error backpropagation to modify those data vectors until they resulted in an approximately self-guessing net. What is more, by judicious choice of the surface-fitting functions it should be possible to improve upon conventional backpropagation by numerically solving the simultaneous nonlinear equations for the minima of the error surface as opposed to (as in conventional backpropagation) using gradient-descent to find those minima. This line of research has not been actively pursued since the nets these schemes would build are mappers rather than processors.

It is apparent, therefore, that backpropagation should be viewed as an evolutionary scheme for modifying a hard-wired brain (a net) so that it exhibits certain behavior. It is not a scheme for storing information in a creature's brain. Consequently, although with current net technology there is no strong reason to be wed to the notions of neurobiology,[15] there *are* strong reasons to be wed to the notions of evolutionary biology. Once this has been realized, many new issues based in evolutionary biology are raised:

1. How does evolution work so well, and can we exploit the same techniques?

2. For example, how should you implement a trick of evolution's (like sex) amongst networks or generalizers?[16]

[15]In creating devices that think, nature was severely handicapped in the range of tools it could use. It had to use slow, inaccurate, biological systems evolutionarily derivable from the systems already occurring in early, primitive organisms. Modern man does not have these restrictions (or, viewed from a different perspective, he does not have this option). It is quite probable that, in addition to the processes employed by the human brain, there are other, perhaps strongly preferable schemes for empowering machines to think. Perhaps the strongest argument against over-reliance on neurobiology, however, lies in the simple fact that we know so little of it. Imagine taking a supercomputer, tearing it apart, and doing some investigating so that you learn that electric current runs through it, and that it is made primarily of metals and paints, with trace quantities of certain other elements (silicon for example). Then, you build a pile of metals and paints, send some electric current through it, and hope to read off somewhere on the pile's surface the value of π, calculated to the millionth digit. The analogy is obvious. It is hard to defend the view that current neuronal nets, being such vastly simplified versions of the real thing, will somehow magically share with real brains the ability to reason.

[16]One way to do it is to simply average the numbers defining the network. (The average of two networks that are very different is likely to result in one with poor performance, but this is akin to the unviability of matings of animals of different species.) However, note that parthenogenetic mutational evolution essentially amounts to variations randomly distributed in genotype space about the parent. If sexual reproduction were a simple averaging of two points in genotype space, it would not seem to have much of an advantage over a parthenogenetic process as a strategy for producing offspring with a genotype corresponding to a superior phenotype. There is no a priori reason why the phenotype of a creature with genotype exactly midway between two parents should be any better than the phenotype of a creature with genotype a small random step away from one of those parents. Nor would a novel beneficial trait disseminate any quicker through a species which uses sexual reproduction. The dissemination rate is, in general, determined by how advantageous the new trait is and how quickly offspring with that trait can be produced. These factors are independent of which reproduction strategy the species uses. As it turns out, evolutionary biologists are not really sure precisely why sex works so well (see review starting on page 214 of March 17, 1988 *Nature*). In a crude sense, the advantage to sex in nature seems to lie in the fact that certain components of the position in genotype space (i.e., certain genes) correspond very precisely to certain associated phenotypic traits and are combined in reproduction not via averaging, but rather on an either-all-one-or-all-the-other basis. This is the familiar process of crossover from Mendelian genetics. It allows two animals, each of which contains one novel favorable trait not shared by the other, to produce offspring containing both favorable traits, in full. Parthenogenetic mutation could not do this, and in a simple averaging scheme those favorable traits would be watered down as they disseminated through the rest of the population. Unfortunately, it is not obvious how best to go about implementing Mendelian sex amongst networks.

3. Given that there is no developmental stage to networks, are tricks like neoteny and suicide cells at all helpful in evolving networks?

4. What about the tricks involved in population biology and mating strategies, and in general all the ways in which distinct animals can affect one another's performance?

5. Is there any way to enhance the degeneracy effects of multiple genotypes coding for the same phenotype (convergent evolution)?

6. Should we develop some sort of embryogenesis for networks, perhaps as a means of information compactification? If the mapping from genotype to phenotype is smooth, you can use "iterative focusing" and evolve a population of nets, focusing in on the desired phenotype by always evolving off of the genotype of the net exhibiting the phenotype closest to the desired one. Indeed, this is precisely what evolution does. But if the genotype–phenotype mapping is particularly volatile and singular so that closeness in phenotype does not indicate closeness in genotype, this procedure should not be particularly helpful. Therefore, if we do develop an embryogenesis for nets, in that the genotype being varied will no longer be the architecture of the net but rather the information coding for the construction of that architecture, will not we just make even more obscure the mapping from the genotype to the phenotype, and therefore hinder the process of iterative focusing?

7. Taking a different slant, for all of evolution's amazing ability to find a correct genotype, it is operating under a number of constraints that do not apply to computer runs. Can we come up with schemes which allow us to beat natural evolution? For example, nature makes no use of negative performance information. Can we?[17]

8. Is there any way to get around genetic evolution's massive parallelism advantage? For that matter, just how real is that advantage, given that the vast majority of individuals in a generation are not involved in the evolutionary growth of their species, and given that we are interested in large changes, analogous to punctuated equilibria, not small-scale phenotypic find-tuning?

[17]In nature, if a certain genotype performs particularly badly, the information that it does so will be lost to future generations — the gene will die out. Only information concerning good genotypes is sent to succeeding generations. However, cultural evolution, for example, does make use of negative performance information. If a society tries on a certain form which turns out to be particularly unsuccessful, the members of that society (and others) remember this failure and try to avoid such forms in the future. Similarly, although in their naive form they are practically useless due to the highly convoluted nature of the genotype–phenotype surface, steepest descent and hyperparaboloid fitting make use of negative information in their construction of succeeding generations.

All of these questions are very open-ended. Nonetheless, simple-minded answers to some of them can and have been readily incorporated into the scheme for varying the architecture of the net feeding in to the last HERBIE. However, instead of criteria concerning the viability of an organism in its environment, the goal directing the evolution was self-guessing. The goal was to evolve an organism not simply to behave in a certain way in a certain limited set of situations, but to also behave reasonably in situations beyond those which directly forced its evolution. (Interestingly, if by using evolution we could achieve these goals perfectly and the nets so produced were quite complicated, we might be in the odd situation of having built a machine which thinks without understanding how it does so.)

3.4 Evolving processor nets

Because the scanning problem is so difficult for conventional generalizers to solve, most of the work using evolution to vary the architecture of a HERBIE net has used it as a test case. The initial work along these lines used the connectivity of figure 3 with the data vectors specifying the bottom two HERBIEs of the net being variable. The program was given a fixed environment and a learning set consisting of inputs to the whole net (i.e., offsets into the environment) along with the corresponding outputs (i.e., the first nonzero elements in the environment to the right of the offset). Then, the data vectors of the first two HERBIEs were varied to try to make a net which is self-guessing for this learning set, i.e. to try to make a net which, when used with a subset of the total learning set to set the last HERBIE, sets it in such a way that the resultant structure guesses the other points in the learning set.

The varying of the data vectors was done in such a way as to try to make use of iterative focusing, multi-partner sex, and negative performance information. Specifically, each generation consisted of 24 creatures (nets). In the first generation, all the creatures were purely random. In succeeding generations, the first 3 of the 24 were always the best 3 from the previous generation. The next 11 in each generation were parthenogenetic mutations of varying intensity off of these best 3, most of the mutations being off of the best of the triplet. The next creature was a (reciprocal) self-guessing error weighted average off of those nets making up the triplet along with the two best performing of the following 11, and a little bit of noise. (It was in manufacturing this creature that sex was implemented.) There was then another creature made in the same way, except that the worst performing of the 5 parents were taken to provide negative performance information; the averaging was performed so as to move the genotype specifying the architecture away from them. Finally, the last 8 creatures of any generation were purely random. After creating all 24 creatures, the squares of their error as

self-guessers were measured,[18] and the 3 creatures with the smallest error were fed on to the next generation. Then the whole process was repeated.

To refer to this as an "iterative focusing" evolutionary strategy means that the random mutations will (it is hoped) first find a crude approximation to the desired creature, then the coarse parthenogenetic mutations off of this crude approximation will find a better approximation, and finally the fine parthenogenetic mutations off of this better approximation will find an even better one. (Sex and the like is an aid to iterative focussing rather than a part of it.) If the genotype/phenotype surface (i.e., the net/net performance surface) is completely random and discontinuous, then attempted iterative focusing should do the same as purely random guessing. If for some reason the surface is perverse enough to have bad phenotypes clumped (in genotype space) around good phenotypes, then simple random guessing would actually do better than attempts at iterative focusing.

It is more likely, however, that the surface has good phenotypes clumped around good phenotypes, and for such a surface iterative focusing will perform much better than random guessing. For example, assume that the surface is such that the total volume in genotype space with corresponding fitness of phenotype better than some positive real value k (i.e., with fitness error $< k$) is roughly proportional to k. This amounts to assuming that there are few points with very good phenotypes. Now take any point λ with fitness a factor of m times better than the average fitness of points, and label by Λ the genotype hypersphere centered on that point having a volume m times smaller than the entire space. Then it is also assumed that the percentage of points in Λ which have phenotype fitness at least a further factor of m better than the fitness of the λ point is approximately $1/m$. (This assumption amounts to a very crude model of good phenotypes clumping in genotype space.) Aside from these two conditions, the surface is assumed to be random. Now assume that we are trying to find a genotype corresponding to a phenotype n times better than the average fitness. The probability of having found it through an exclusively random search will exceed x after trying $\log_{(1-1/n)}(1-x)$ creatures. Using the identity $ln(1-x) = -(x + \frac{x^2}{2} + \ldots)$, this number of creatures can be rewritten as

$$-\frac{ln(1-x)}{1/n + 1/(2n^2) + \ldots}.$$

For n large, this is approximately equal to $-nln(1-x)$. Now assume we want to find a genotype with phenotype n^2 better than average. With a random search, we have just seen it will take on the order of $n^2 ln(2)$ tries to find such a creature with a better than 50% probability. In contrast, if we

[18]Self-guessing error was approximated in various ways. As an example, for a learning set of 100 data vectors, self-guessing error was often approximated as the sum of 4 separate error numbers. Each one of those 4 error numbers was based on a net taught with a (different) subset of 75 of the 100 total data vectors, and was calculated by summing the squares of the errors the net had in guessing the output values of the remaining 25 data vectors.

used iterative focusing to first find a creature n times better than average and then a creature n times better than that, it would take only on the order of $-2nln(1-\sqrt{.5})$ tries to find such a creature with a better than 50% probability, $-nln(1-\sqrt{.5})$ for each phase of the search. (x is $\sqrt{.5}$ so that the probability of finding the first creature times the probability of finding the second is $1/2$.) It is this increase in search speed which is the reason for using iterative focusing as opposed to a purely random search.[19]

There is one major drawback to the evolving scheme as given, shared with real biological evolution. Typically, if you build a random creature whose genotype is only coarsely equivalent to that of a creature with very good performance, it takes many generations of sex and mutations off of that random creature before the genotype of its descendants focuses in on the genotype of the creature with very good performance. Even if the random creature contains some novel feature which is a great advance over the other creatures of its generation, unless it is allowed time to evolve to make full use of that feature and to weed out any other negative features it might have, in general it will not be able to make a contribution to the gene pool of the species. Its tree of descendants is likely to die off rather quickly, since it can be expected that the initial random creature will have more negative features than positive ones, relative to the currently best performing members of its species, despite its potential to focus in on a creature with very good performance. This effect amount to getting stuck in a local minimum — some starting genotype will be worked into the best form it can have, and from then on no revolutionarily novel genotype can break in to the inner circle of genotypes which pass from one generation to the next. To mitigate against this effect, in addition to passing the three best performing creatures from one generation to the next, a record was continually being updated of which of the purely random creatures had the best performance. Then, when the performance of the top three started to asymptote, the best of the purely random creatures to date was used as the seed for a whole new succession of mutations and sex. When the performance of that new sequence of generations starts to asymptote, the performance of its best creature was compared to the performance of the best creature in the original sequence of generations. The better performing of these two was then sent on to the next generation, and the whole process was repeated. In this way it was hoped that the evolving would not get caught in any local minima.

3.5 Results

In exploring these structures some interesting heuristic properties were found concerning the genotype–phenotype surface, the surface taking as input the data vectors of the HERBIEs making up the net which feeds into the last HERBIE, and giving as output the resulting errors at self-guessing based on a single, preset learning set. The first such property was that the addition of elements to a learning set, being the addition of restrictions to the gen-

[19] For an interesting discussion of other evolutionary strategies, see reference [13].

eralizer, cuts down on the degeneracy of generalizers which are self-guessing for the learning set. This results in the exclusion of generalizers which had previously had a small self-guessing error from the set of degenerate generalizers, since with the expanded learning set they now have a large self-guessing error. Therefore, in many situations, adding new elements to the learning set actually increases the average self-guessing error, even if it results in a better guesser of the parent surface generating the learning set.

Another interesting property, one with further-reaching consequences, is that the genotype-phenotype surface is usually highly convoluted. The sexually produced creatures, being slight modifications of the best performing creatures, often had very good performance, even beating out their parents on occasion. But since the resulting variations off of the best performing genotypes were kept small, they rarely resulted in creatures with substantially superior performance to that of their parents. Due to the highly volatile nature of the surface, when the variations were forced to be larger, they tended to result in creatures with performance essentially the same as that of creatures with random genotype. This behavior was found to be true with the parthenogenetic mutations as well. The advantages of iterative focusing (as opposed to making each new generation consist entirely of novel random genotypes) were not nearly as strong in practice as had been hoped. Indeed, the genotype–phenotype surface was found to be so volatile that hyperplanar and hyperparaboloidal fitting and descent schemes were essentially a waste of computer time. As a result, they were not used in any but the initial experiments.

The reasons for this volatility of the genotype–phenotype surface are not hard to find. There exists a natural notion of an equivalence set over the genotype–phenotype surface: two nets are equivalent if it takes the same number of iterations for each of them to answer a question used in evaluating the self-guessing error of the net (including the questions used to set the last HERBIE), for every question in the given learning set. In other words, they are equivalent if it takes the same number of iterations to figure out each one's output value on the genotype-phenotype surface. In general, one would expect a relatively smooth surface over the domain of such an equivalence set. For example, if each HERBIE in the net consists of just a single hyperplane, then over an equivalence set the self-guessing errors will just be polynomial functions of the data vectors of the HERBIEs in the net. Unfortunately, in general the equivalence sets are severely disjointed, appearing in patches over the domain of the genotype–phenotype surface. At the boundaries between equivalence sets the surface will be discontinuous, so the surface is very convoluted in general, with many discontinuities in it. It will appear like a set of polynomial surfaces interlaced together. In general, the higher the number of elements in the learning set, the higher the number of equivalence sets, and the more convoluted the genotype–phenotype surface.

Results of a typical run for a net evolved to self-guess the scanning problem are indicated in figure 4. To speed things up, the environment HERBIE was amended so that the nonzero blocks in the environment had a width of 8

rather than 1, and so that there was a slight downward slope of the environment HERBIE's output over all regions where that output is nonzero. This last was so that the self-guessing error could distinguish between an input to the environment HERBIE of 4.0 and one of 11.99, for example. It was hoped that together these modifications would make the evolution find and then focus in on a correct solution more readily. Note that the slight downward slope means that a perfect net, i.e. one made up of the HERBIEs shown in figure 3, will not guess a constant number over any interval of inputs corresponding to 0's in the environment. Because the scanning is done in jumps of 1.0 (the increment of the counter), the slopes in the environment HERBIE will be reflected in any such interval, even for a "perfect" guesser.

Reproduction of the learning set is perfect, of course, but what is truly remarkable is how good the guessing is for points outside of the learning set. Note that strictly speaking, since the input to the generalizer is the input alone, generalization just refers to how well the net guesses for the environment with which it was taught. In figure 5, however, the original environment has been replaced with a new one. Even here, the guessing is excellent in the sense that the behavior is very close to that which the researcher had in mind when making up the learning set. Interestingly enough, the bottom two HERBIEs in the evolved nets always differed greatly from those of figure 3. Convergent evolution had come into play. A point worth emphasizing in all this is that the learning set consists of only 8 input–output pairs, far less than the number of pairs that would be needed by backpropagation schemes to achieve similar performance (assuming they even could).

In addition to these results in which the architecture was preset to that indicated in figure 3, other runs were done in which the connectivity of the whole net was varied along with the data vectors defining the individual HERBIEs (see figure 6). The procedure for varying the connectivity was fairly complex, especially when mutating off of one net's connectivity to create another net's connectivity (as opposed to creating a new net's connectivity randomly, from scratch). Since the nets also now contained 10 individual HERBIEs which usually had up to about half a dozen input dimensions each, the running speed of the evolution was quite a bit slower, so there were fewer generations in a typical run. Since there are now many more degrees of freedom in the creatures, the generalization, though still good, is not as good as that shown in figure 5. Running the evolution for a greater number of generations might remove this disparity, although considerations of information compactification might militate against such a convergence of performance.

In addition to these results for the scanning problem, several runs have been made of 100 generations each for the find-the-0 problem with 5 input dimensions. Again, there were 10 HERBIEs in the net, most containing about 4 or 5 input dimensions, and the learning set consisted of 20 input–output pairs. No environment HERBIE was used in the nets — the nets had 5 input lines. A single hyperplanar HERBIE of 5 input dimensions was also taught with the same learning set. Then a testing set consisting of 20 different pairs was constructed and run through these two structures. The

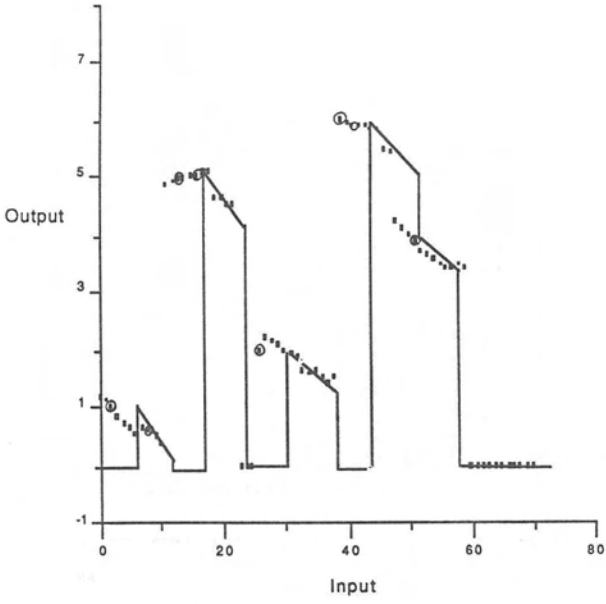

Figure 4: The line shows the mapping of the environmental HERBIE. The dots indicate the guesses of a network (i.e., the input–output mapping of that network), one of whose nodes was this environmental HERBIE. (The connectivity of the network was that of figure 3.) The network was made by evolving networks throught 1500 generations, where the data vectors defining the individual (hyperplanar HERBIE) nodes served as the genotype. Self-guessing was the error measure for the evolution, using an 8 point learning set (indicated by the circles in the figure) which was derived from the given environmental HERBIE. A network with perfect generalization would output the first nonzero value in the environmental HERBIE lying a positive integer to the right of the input.

rms average error on this testing set was three times smaller for the evolved nets than for the single HERBIE. This despite the fact that, unlike with the scanning problem, no perfect hand-crafted net solution to the switching problem has been found, and there is no reason to believe that one exists which uses only ten HERBIEs.

The generalization exhibited in these runs was quite impressive and seems to confirm the real-world applicability of the criterion of self-guessing. It should not be surprising that these systems exhibit generalization behavior superior to that which could be expected from the vast majority of conven-

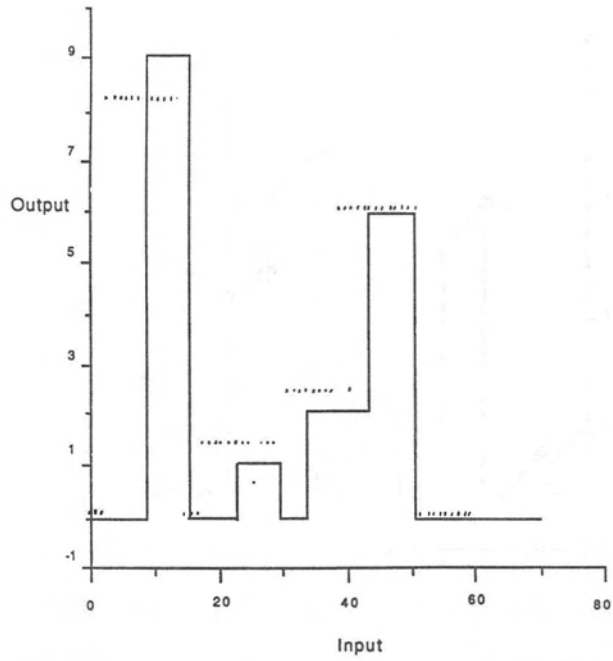

Figure 5: The line shows the mapping of an environmental HERBIE, different from the one in figure 4. The dots indicate the guesses of the net of figure 4 run through this new environmental HERBIE. Although this net was evolved using the environmental HERBIE of figure 4, it still guesses well using the novel environmental HERBIE indicated here.

tional generalizing schemes (e.g., backpropagation). After all, in their using self-guessing the systems explored here are attempting to address the issue of generalization directly, something conventional schemes do not even pretend to do. In addition to their generalization efficacy, however, these systems also have a number of other advantages over conventional generalizing schemes. Some of the more obvious of these are (1) they reproduce the learning set exactly and with ease, (2) they exploit environmental HERBIEs, which allows one to cut down drastically on the number of input coordinates, and which also allows the construction of systems for which varying the size of the input environment is a trivial operation (contrast this to the situation with conventional net structures in which the environment is fed in through input lines, so that if the size of the environment were to change the entire net would have to be changed as well), (3) they are relatively well suited to the role of research tools, since it is relatively easy to trace through their

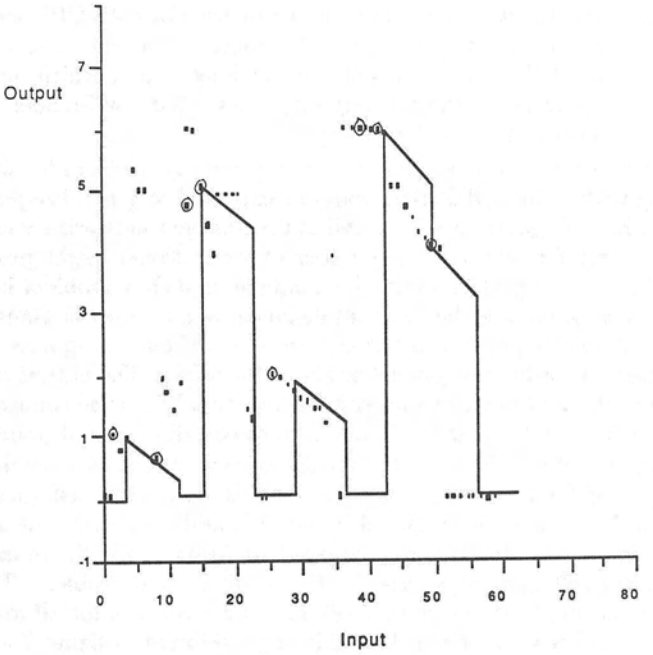

Figure 6: This figure represents the exact same position as in figure 4, except that the guessing shown is of a different net, one in which there were ten (hyperplanar) HERBIEs and whose connectivity was allowed to vary along with the individual HERBIEs. The self-guessing error was $\tilde{4}$ times worse than that of the net in figure 4, and since that net had so many fewer HERBIEs, the information compactification for the figure 4 net is better as well. Hence, the figure 4 net generalizes better than this one.

behavior (recall the discussion at the beginning of this section), and (4) as has been mentioned, through structures like the counter HERBIE they are well suited to modeling internal memory and "learning," in the proper sense of the term elucidated previously.

One drawback which became apparent in conducting these runs however is that the amount of time necessary to do the evolving was often quite large. With the program written in *C*, and with no concerted effort to optimize the code for speed, the nets whose behavior is indicated in figures 3 through 5 each took on the order of a day of VAX 750 cpu time to make. Since the potential uses of such net evolving lies in generalizing from a predetermined learning set, such time requirements are not so disappointing as they would be if, for example, we were trying to build a structure for real-time learning.

Also note that the reason so much CPU time is required is because we are trying to approximate self-guessing. Each individual creature in the evolution reproduces the learning set perfectly and takes only fractions of a CPU second to build. This should be compared to backpropagated neural nets, which often take tens of CPU minutes to build (if not longer) and which do not even achieve reproduction of the learning set, never mind any broader kind of generalization criterion like self-guessing.

Another caution which must be raised is that for any problem for which surface-fitting with a single HERBIE would be expected to generalize particularly well, using self-guessing with a full output-flagged network structure (and then evolving for only a finite number of generations) might provide little if any increase in performance. An example of such a problem is the hill-climbing task of finding the minimal distance over a series of Gaussian hills. The input to this problem is the y coordinates of two end-points with fixed and preset x coordinates symmetric about the origin. The output is the y coordinate of the mid-point of the path with minimal distance connecting the two end-points. A Gaussian hill lives in between the two end-points at $(0,0)$. Several nets of metric-based HERBIEs were evolved so as to minimize self-guessing error for a learning set built for this task. The self-guessing and generalization errors are compared in figure 7, and are clearly not at all well correlated. (It is instructive to compare this figure to the figure in reference [11] where self-guessing is used for the reading aloud problem. There the samples of generalization error vs. self-guessing error are, for all intents and purposes, perfectly colinear.) The difficulty exhibited in figure 7 arises from the fact that the generalization using purely local information is particularly good in this case, since the task is not really dependent on global characteristics of the parent surface and since that parent surface is particularly smooth. Indeed, the right-most point in figure 7 was the performance of a single metric-based HERBIE (see the first paper) with no evolving at all. Although self-guessing should be helpful at this problem if used to vary amongst single HERBIEs with different kinds of surface-fitting, when the varying is amongst output-flagged networks of metric-based HERBIEs the discontinuities inherent in such nets make the criterion of self-guessing, for this type of problem, essentially useless. It is extremely difficulty to get generalization behavior better than that of a single, nonevolved, metric-based HERBIE by evolving networks of such HERBIEs so as to minimize their self-guessing error. Interesting enough, however, an individual hyperplanar HERBIE has a guessing error on the testing set of 7,550,000, whereas a net of hyperplanar HERBIEs evolved so as to minimize self-guessing error has a guessing error on the testing set of only 25! For hyperplanar HERBIEs, which due to their volatile and discontinuous nature are not well suited to this path-finding task, evolving self-guessing nets provides an astronomical improvement in guessing ability.

In general, these results, like those from earlier in this chapter, should not be viewed as a conclusive measure of the efficacy of the concepts of generalization theory, evolving network structures, and the like. Rather, they

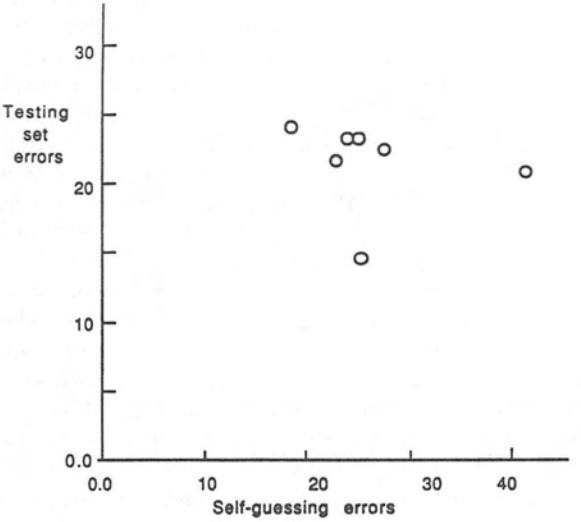

Figure 7: This figure shows that there is little if any correlation between self-guessing and testing set error for the problem of guessing the mid-point of the shortest path across a gaussian hill. The horizontal axis gives the error in arbitrary units on the second half of a 100 element training set for seven different generalizers all taught with the first half of the training set. The vertical axis gives the error in (different) arbitrary units for guessing the 100 elements of a testing set when taught with the full training set.

should be viewed as an initial exploration of these concepts. Much more testing remains to be done before any definite conclusions can be reached about where, when and how much these concepts can be expected to prove useful. Nonetheless it can be concluded that there are situations where they are extremely useful, the NETtalk case of reference [11] being perhaps the most clear example.

4. Concluding remarks on neural nets and generalization

Of the usual reasons for neural net research, perhaps the most compelling is that neural nets generalize easily and naturally. However, no work has been done up to now which rigorously investigates whether or not neural nets generalize well, and if so what properties of them cause such adept

generalization. Clearly, if neural nets are to meet their promise such work is critically important.

The hyperplanar HERBIE was actually developed before the generalization theory of these papers, and was meant to serve as a simple benchmark of generalization. When used to measure the generalization efficacy of neural nets [5], the most obvious interpretation of the results is that the performance of this HERBIE actually indicates that neural nets as presently constituted perform quite poorly. One could, of course, interpret the results of [5] (along with [11] and [12]) differently, saying that neural nets generalize well, and therefore that the hyperplanar HERBIE, self-guessers, and the like generalize exceptionally well. However there is little justification for such an interpretation, given that the hyperplanar HERBIE, for example, make no attempt to view the problem of generalization from an abstract point of view, and given that all of these systems are far from being fully model-independent. When saying that these systems are good generalizers all that is really meant is that they have tested well when presented with problems whose "correct" generalizations are known beforehand to the researcher. Clearly, this is an unsatisfactorally ad hoc way of dealing with the highly complex and vital issue of generalization. Of course, it may turn out that there is no (consistent) abstract way of dealing with generalization, and that there is no way to measure generalization more rigorously than via test problems. In this case the interpretation that hyperplanar HERBIE and the various techniques investigated in the previous chapters constitute particularly good generalizers would be easier to accept. However, even if it turns out that no consistent mathematics of generalization can be created, neural nets would still be left in the lurch — they have to beat or at least equal the techniques and systems presented in this series of papers and in references [5] and [11] if they are to ever serve as the generalizers of choice.

To try to address the problem of whether the systems presented here really are exceptional generalizers, it is natural to try to examine things in light of an abstract mathematics of generalization. Unfortunately, the problem of finding satisfacotry abstract criteria for the generalization from a learning set is a very difficult one, and one which has not previously been fully addressed. Section 2 of the first paper of this series and sections 1 and 2 of this paper constitute only a very preliminary venture into the mathematics of generalization. Nonetheless, this is a vitally important problem, not just for network research and tests between generalizers, but for AI as a whole. Its solution would provide a very flexible and widely applicable inference engine. Unlike conventional AI techniques, this engine would not be dependent upon a laborious, overly literal and problem-specific listing of the logic needed to carry out a given application. Rather, with appropriate supporting programs, one engine would fit all applications.

As a first attempt to attack this problem, generalization theory, involving the concepts of generalizers, (semi-) properness, (strong) self-guessing, information compactification and the like has been explored. Generalization theory as presently constituted does not provide us with a full set of

generalization criteria (in the sense of giving us a mapping from learning sets to "optimal" generalizers of those learning sets), but the criteria it does provide have proven very useful, both as classifiers of generalizers and as pointers to new type of generalizers. Computer-run approximations to some of these criteria have been carried out, quite successfully, for the scanning and switching problems, two problems which are extremely difficult for conventional techniques like backpropagated neural nets or hyperplanar HERBIEs. The zeitgeist of biological evolution arises naturally in such a context, providing an interesting alternative viewpoint to network construction. Although the results of these runs constitute only a preliminary investigation, it seems that the criteria used in the current version of generalization theory are quite powerful, especially in comparison to the conventional techniques for generalizing. At a minimum, these results are promising enough that there no longer seems to be any justification whatsoever for conducting neural net research without taking into account the concerns of generalization theory.

Appendix A. Proof of (1.4)

We have a learning set θ and a semi-proper generalizer G which is (semi-properly) strongly self-guessing for θ. Since we are dealing with (semi-proper) self-guessing expansions, we can view any given lesson in the strong expansion of θ as in the discussion in the paragraph following (1.1), i.e. as an unordered collection of datum spaces, with no datum space being delineated as a special "question-output" pair. From this viewpoint, the set of operators taking any one lesson in the strong expansion of θ to another is identical to the set of all sequences of the operators {remove a datum space from the old lesson to generate the new one (so long as this does not violate (0.5))} and append a new datum space to the old lesson to get the new one, where the new datum space is a question-output pair predicted by G, using the old lesson as a learning set. (Note that these two operators are each other's inverse.) For example, the lesson-to-lesson mapping of property (f) is application to a lesson of this second operator, where the question being used is the input component of one of the lesson's datum spaces. Therefore, in discussing the relationships of the lessons in the expansion of θ, it is sufficient to treat them as being related through a sequence of these two operators.

Now take any lesson C from this expansion of θ. Distinguish those datum spaces of C whose entries are one of the original $(a_i, A_i) \in \theta$ from those which are variable (i.e., dependent on the particulars of G). By the aforementioned identity of sets of operators, any lesson will split up this way. First, examine the case where the number of datum space entries in C which are one of the original (a_i, A_i) exceeds the dimension of the generalizer and where those datum space entries do not all lie on the same m-dimensional hyperplane. Label the elements not one of the (a_i, A_i) as (x_1, X_1), (x_2, X_2), etc. Generate a new lesson by removing from C all datum spaces (x_i, X_i) except for a particular datum space (x_j, X_j). Also generate (θ, x_j, y). By hypothesis, there are enough datum spaces in C identical to one of those in θ to allow

self-guessing to force y to equal X_j. This is true for any j. Therefore, no lesson (θ, u, v) will have u one of datum space inputs in the lesson C unless v is the corresponding datum space output. This is true for any lesson C in the expansion of θ having enough of its datum spaces identical to one of the ones which make up θ.

To take care of the case where C does not have sufficient overlap with θ, list the sequence of lessons formed by the operations which made C by $c(1), c(2), \ldots c(n)$. $c(1) = \theta$, and $c(n) = C$. Now form the lessons $d(1), d(2)$, $\ldots d(n)$ using the exact same sequence of operations, except that if the operation taking $c(i) \rightarrow c(i+1)$ does not increase the order of the lesson, replace this operation by the null operation: $d(i+1) = d(i)$. In other words, the $d(i)$ are formed the same way as the $c(i)$ except for the presence of a filter allowing only order-increasing operations. (Note that for any order-increasing operation from the kth step to the $k+1$th, the input component of the new datum space is the same for both $c(k+1)$ and $d(k+1)$, but since $c(k)$ and $d(k)$ might be different, the output components of the two new datum spaces guessed by G (using $c(k)$ and $d(k)$ respectively as the learning sets) might differ.) Now the order of any $c(i)$, $O(c(i))$, exceeds m, the elements of any $c(i)$ do not all lie on the same m-dimensional hyperplane, and $O(d(i)) \geq O(c(i))$ for all i. By definition, for $i = 1$, $g\{O(c(i))\}(c(i), q) = g\{O(d(i))\}(d(i), q)$ for any question q. If this equality were to hold for all $i <$ some particular value l, then $c(i) \subseteq d(i)$ for all $i \leq l$, since for all order-increasing operations from iteration $i < l$ to iteration $i + 1$ we would be adding the same datum space to both $c(i)$ and $d(i)$. Then, since $O(c(i)) > m$, by the self-guessing argument in the preceding paragraph there would be sufficient overlap between $c(l)$ and $d(l)$ to force $g\{O(c(l))\}(c(l), q) = g\{O(d(l))\}(d(l), q)$ for any q. This completes an inductive proof that for any i, $g\{O(c(i))\}(c(i), q) = g\{O(d(i))\}(d(i), q)$, and therefore that any two lessons $(d(n), u, v)$ and (C, u, v') must have $v = v'$. By self-guessing, we then get that any two lessons (θ, u, v) and (C, u, v') must have $v = v'$. Consequently, no two lessons in the expansion of θ can disagree on the output, v, corresponding to a given question, u, since they have to agree with the (single-valued) lesson (θ, u, v).

Let $f(u)$ be the function taking u to v via the lesson (θ, u, v). Clearly $f(u)$ reproduces the learning set θ. Also, the lesson C can be composed of any (sufficiently large) set of pairs $(u, f(u))$, and as we have just shown G running off of C must guess the question-output function f. This concludes the proof of the first part of (1.4). The second part of (1.4) follows directly from the definition of strong self-guessing. ∎

References

[1] Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, **3** (1959) 210–229.

[2] B. Efron, "Computers and the theory of statistics: Thinking the unthinkable," *SIAM Review*, **21** (1979) 460–480.

[3] J.J. Hopfield and D.W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, **52** (1985) 141–152.

[4] J. Denker, et al., "Large automatic learning, rule extraction, and generalization," *Complex Systems*, **1** (1987) 877–922.

[5] D. Wolpert, "A benchmark for how well neural nets generalize," *Biological Cybernetics*, **61** (1989) 303–315.

[6] J. Pearl, "On the connection between the complexity and credibility of inferred models," *International Journal of General Systems*, **4** (1978) 255–264.

[7] G. Chaitin, "Randomness and mathematical proof," *Scientific American*, **232** (1975) 47–52.

[8] J. Rissanen, "Stochastic complexity and modeling," *The Annals of Statistics*, **14** (1986) 1080–1100.

[9] C.E. Shannon, *The Mathematical Theory of Communication* (University of Illinois Press, Urbana, 1949).

[10] A.H. Zemanian, *Distribution Theory and Transform Analysis: An Introduction to Generalized Functions, with Applications* (McGraw-Hill, 1965).

[11] D. Wolpert, "Using a mathematical theory of generalization to build a generalizer superior to NETtalk," *Neural Networks*, to appear.

[12] T.J. Sejnowski and C.W. Rosenberg, *NETtalk: A Parallel Network that Learns to Read Aloud*, Johns Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01, 1986.

[13] Muhlebein, et al., "Evolution algorithms in combinatorial optimization," *Parallel Computing*, **7**(4) (1988) 65–86.