Use of Diploid/Dominance in Genetic Network Synthesis

PhD Dissertation
University of Washington
Department of Electrical Engineering
Seattle, WA

F. Greene

Table of Contents

I. Introduction

This proposal is a result of research over the past two years, and whose purpose was to develop a design methodology for low-cost ultrasonic blood flow and tissue quantification using signal processing. My original desire was to improve feature extraction techniques for use in statistical pattern recognition, but was almost immediately redirected along the lines of efficient genetic search of network solution spaces. Over ten years of experience with Doppler flow measurement suggests that dynamic processing of the clinical signals involved can be done with interconnected functional elements such as delays, filters, and thresholds. Some details of the processing issues and reasons for using genetic search will follow. The point of this dissertation is to study and develop a specific method for synthesizing processing networks that aid in the use, interpretation, and diagnostic power of low-cost medical technology.

Motivation: Development of signal processing algorithms for low-cost Doppler

Non-invasive, diagnostic ultrasound utilizes both imaging and real-time Doppler blood flow measurement. This dissertation proposes a general purpose methodology for developing real-time ultrasonic Doppler flow measurement. These flow measurements make possible the real-time detection and discrimination of basic flow characteristics from audio Doppler flow signals. Complications arise from the varying signal character and quality that results from the normal and pathological variations in anatomy as well as limitations imposed by the "non-invasive" (non-surgical) constraint. As a result of this complexity, any devised methodology needs to be powerful and yet have a wide range of applicability.

Flow characterization can also be useful with 2D or 3D imaging of tissue movement, of which blood flow is one example. In particular, a precise mapping of flow event boundaries (which divide, e.g., flow near vessel boundaries and obstructions) must, to some degree, take into account the case by case variation of naturally occurring flow states. For example, highly reflective artifact from vessel walls and poor S/N conditions arising from tissue depth, probe movement, etc. must be accurately discriminated in order to just determine if flow exists, and if so then what *type* of flow exists. The "type" of flow can be broadly characterized as being either *laminar*, which is associated with normal vessels, or *turbulent*, which can be associated with disease.

Doppler signal dropout detection

Signal validity (i.e., the assurance that the signal is arising from center stream blood flow) can be assessed by thresholding spectral energy content. An improvement can be made by restricting the range of spectral energies to those positive frequencies between approximately 1 and 4 kHz (for a 5 MHz Doppler.) Further improvement can be made if a reliable fiducial point (e.g., the ECG R-wave) can provide knowledge of where the systolic and diastolic portions of the cardiac cycle occur, so that filter band limits and thresholds can be adjusted dynamically according to the heart cycle state. There are several factors affecting even such a basic determination, including Doppler beam to vessel angle, normal patient to patient variation, variation due to disease state, and variation with respect to the choice of vessel being examined. This is the reason for a powerful, yet widely applicable approach that can still make use of problem knowledge the designer may already have.

The required processing power increases rapidly with increasing 2 or 3D Doppler sample voxel resolution. If the boundaries of normal versus turbulent and valid versus non-existent flow can be deduced rapidly enough, a visual map of flow boundaries, and hence pathology such as arterial obstruction, can be provided, to the examiner. If the processing of samples is fast enough, the examination process becomes interactive, which is very important for searching patient anatomy and looking for cause and effect relationships from temporary, stress enduced changes.

Currently available digital data

A series of digitized Doppler data has been obtained, courtesy of the UW Department. of Surgery. These data consist of 15 or more seconds of 5 MHz, quadrature Doppler data. There are 20 cases from the common and internal carotid arteries of patients with varying atherosclerosis and encoded R-wave locations. There are also twenty fetal umbilical cases from a hand-held CW system. These data were taken with the system shown in Figure 1.1. This system has made it possible to FFT,

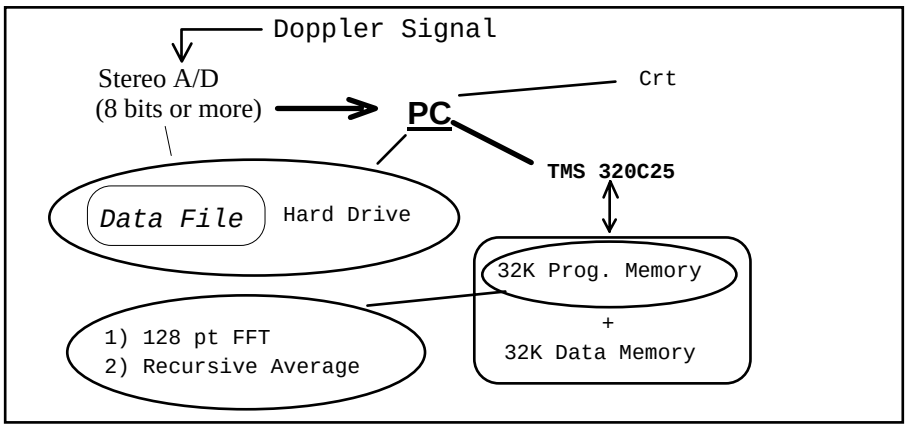


Figure 1.1. The system used to acquire clinical data base

recursively average each resulting frequency magnitude (with a bank of 1-pole digital filters), and display the results in real-time (100 Hz FFT rate.) The display separates positive and negative frequencies and plots an unprocessed estimate of the spectral mode (see figure 1.11.) The co-processor is programmed to calculate a 64 point complex FFT and recursively average all spectral magnitudes at a rate of 100 Hz, which is typical for medical equipment. The time domain data are stored as 8-bit complex pairs (from the I and Q channels of the quadrature output) which gives adequate dynamic range to represent Doppler data, largely because the signal has its DC component removed and because Doppler data are noisy, with there being*at best* 40 dB of S/N.

Two useful frequency estimates for recognizing diagnostic flow behavior patterns in normal and diseased arteries are the*mode* and *upper 9 dB* frequency contour of the signal 's magnitude spectrum. Figure 1.11 shows how these estimates are

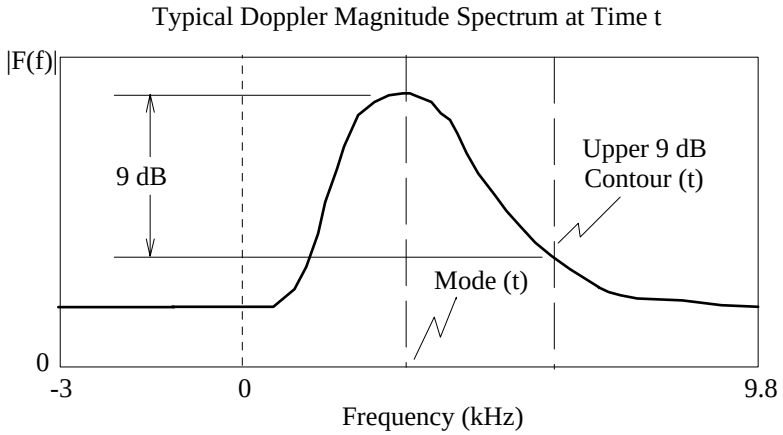


Figure 1.11 Definition of spectral mode and upper 9 dB contour.

defined. Reliable measurement of these quantities is complicated by the various signal states that can occur. For example, consider a Doppler signal produced by blood flow downstream from a tight vessel obstruction. Such a signal will have relatively wide bandwidth, and in some cases may have less than 9 dB of S/N. Such a situation much be reliably detected and dealt with. A rule based system developed by the author for this purpose uses a series of ad hoc decisions to determine which states are present and proceeds to alter the determination of the upper 9 dB level (and other parameters) accordingly (Figure 2.) These algorithms have two limitations: 1) the "rules" must be modified if they are to handle a wider range of disease states and 2) the rules are based on intuition, and extension of that methodology to the entire range of disease states, vessels, and patient variations would be time consuming and difficult to test and re-modify for that reason. The upper 9 dB contour is one of the measurements made by this system. The reasons for using this estimator, as opposed to the mean, mode, or median estimators, for example, are as follows:

1) The *upper* envelope produces essentially the same wave shapes (for various arteries, veins, and applicable disease states) as does the commonly used zero crosser frequency to voltage converter and 1st moment (mean) *when flow is*

laminar (i.e., narrow-band.) Zero crossers are still in wide use with many low cost medical systems and compatibility with existing or traditional techniques is an important feature;

2) If flow is disturbed (broad band), the *upper* envelope, or contour, can also reveal this very important condition in a visually obvious way. By contrast a zero crosser, mean or mode, in this case, produces an output that appears random or can even fall to zero;

3) The choice of *9 dB* is, in practice, as much S/N as the Doppler signal can be expected to have in cases of severe turbulence or difficult probe placement. The exact value of "9" corresponds to the 1/8th "power point" down from the signal maximum. Such a low S/N requirement makes it possible to utilize*most* of the clinically obtained data.

In summary, when compared to the upper (9 dB) envelope, the mean, mode, and median all fail in category (2) and are less reliable in showing the pulsatile behavior in (1).

The three main signal states and typical upper (and lower) 9 dB levels are shown in Figure 2. In addition, a possible

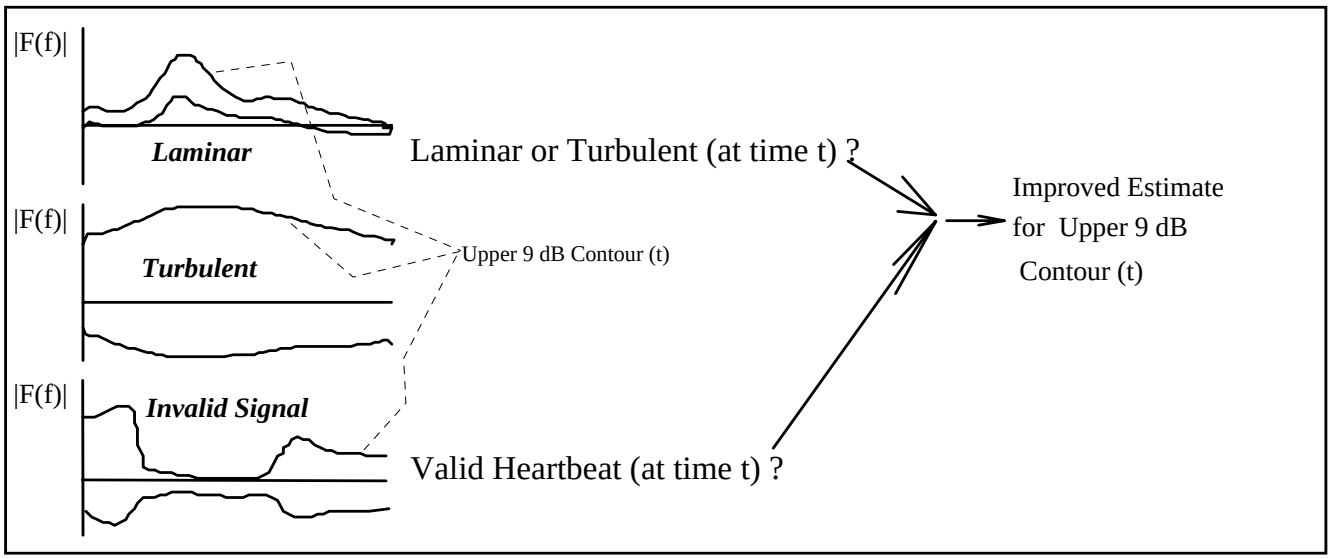


Figure 1.2. Basic Doppler signal flow states

improvement in the 9 dB estimate is suggested in this figure. That is, determination of signal validity and whether the signal is turbulent (wide band) or laminar can be used to qualify the signal for further analysis or modify calculation of the spectral mode as appropriate. The signal can become "invalid" if the Doppler sample volume moves relative to the actual region of flow. If this determination could be made automatically, data would can be automatically rejected until the signal is once again restored by re-positioning the probe. Turbulence is often an indication of disease. If turbulence is determined to be present (e.g., this can be done by counting the number of bins over a threshold amplitude as mentioned above), processing should be altered to find the first 9 dB crossing coming *down* in frequency from the maximum frequency limit. The latter step effectively eliminates artifact due to multiple crossings in the 9 dB frequency. In addition, the mode itself should be calculated somewhat differently in the case of turbulence since there may be multiple spectral peaks having very near or identical amplitudes. A similar problem arises due to fluid-mechanical *bruit noise*, which has the characteristic of being band-limited to between 100 and approximately 1.2 kHz. If "turbulence" is decided, then a bruit filter should be applied to eliminate a false spectral mode in this range. Other application of these ideas extends to deciding if the I and Q channels need reversal (which occurs when probe orientation is changed) and, in CW Dopplers, pre-filtering of the -1 to +1 kHz region to eliminate venous or other static clutter. The latter problem can be effectively addressed with a bank of frequency domain filters that subtract out the*lowest* amplitude frequency bin value (thereby approximating the time static value) seen over the previous 2 or so seconds.

The system must discriminate between laminar and turbulent flow conditions in order to get a reliable estimate of the 9 dB contour and other diagnostically useful parameters and detect invalid signal states. The method for doing this will rely on

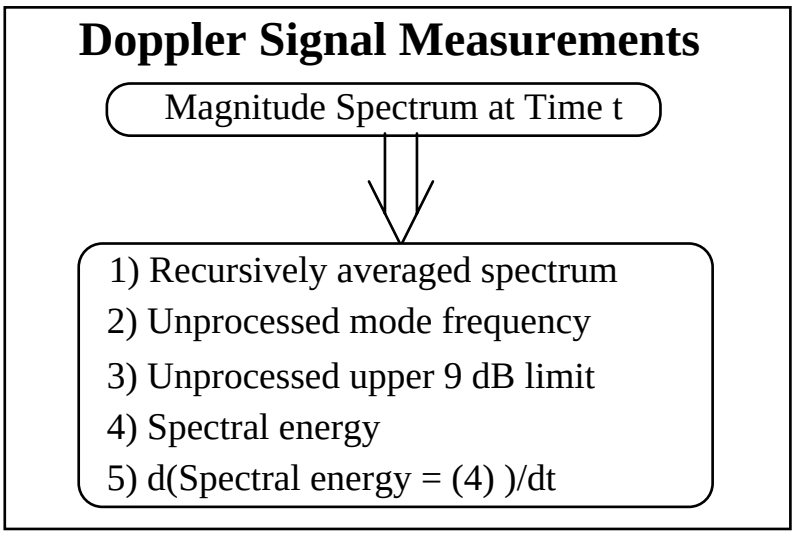


Figure 1.3 Basic spectral measurements used for flow state determination

the basic Doppler flow measurements listed in Figure 1.3. Measurements 1-5 are available for the methods about to be described (only measurement 1 is actually used in the test problem presented in "Results".) The overall approach is

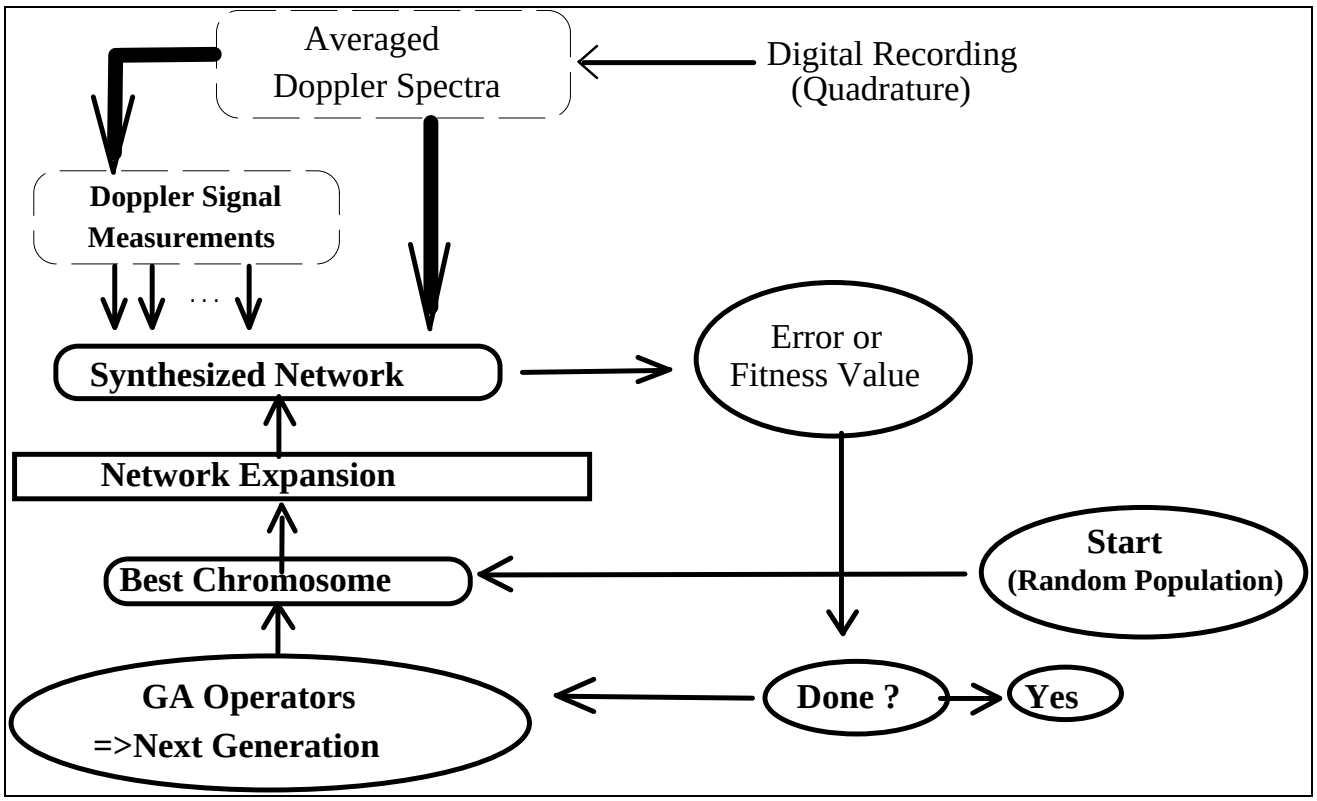


Figure 1.4. Overall approach to network synthesis, including use of the measurements of Figure 1.3.

summarized in Figure 1.4, including the use of genetic algorithms, which will now be discussed.

Use of Genetic Algorithms

A goal of this proposal is to define a signal processing development methodology that permits pre-specification of network design. Genetic algorithms ("GA's") are a probabilistic search technique that utilizes evolutionary mechanisms deduced from biology to locate optima or near optima in a relatively large search space. As will be shown in the *literature review* and as demonstrated in the *results* sections, GA's coupled with an appropriate ("back-codable") coding scheme provides a methodology that can complete or optimize a partially, or pre-specified, design.

Current interest in AI extends to neural networks, fuzzy logic, and genetic algorithms. One of the challenges inherent with neural network processing (and connectionist systems in general) is the complexity involved in selecting node characteristics and network architecture. This has resulted in a narrow range of architectures and often fairly problem specific results. Many network architectures are possible, but only a small subset of these are typically considered when making a design. A complex pattern recognition problem could make use of a variety of non-linear operations. In order to specify and systematically alter a complex network having a variety of node operators and architecture, it is logical to code the structure of the network to keep track of design building blocks, and then systematically alter the codings to search for a more optimal design. Making the search efficient may, of course, be the main difficulty. In nature, this process is accomplished by altering DNA codes through natural selection and what is broadly called evolution. The approach I am suggesting for this dissertation has been received extensive attention over the past two decades under the name: G*enetic algorithms* ("GA's").

One of the aspects of GA's that I would like to investigate is the idea of encoding the chromosome to represent both the architecture and internal parameters of connectionist systems. This will utilize an efficient encoding scheme together with associated procedures and algorithmic details in the GA itself. The basic idea is to represent a (connectionist) system by a carefully designed linear code sequence (like DNA) that has among its properties, the following:

1) Short length sub-codes that can be replicated with possible modification (via standard GA operators, to be discussed) for the purpose of optimizing or expanding on useful sub-designs

2) Useful with custom designed software tools that aid in "pre-specifying" a desired network structure so as to provide a good starting point on the design search.

3) Genetic operators that are especially suited to searching for near optimal connection weights. For example, instead of swapping weight values during genetic crossover, averaging them may be useful.

The next section will provide theoretical justification and background to the above statements, including the alleged power and broad applicability of GA's. This dissertation will largely concentrate on developing a GA approach for synthesizing signal processing network that are applicable to low-cost medical instrumentation. The test problem suite will include solution of a simple problem from Doppler signal processing as discussed above. In addition, this dissertation will explore a completely new approach to utilizing diploid/dominance in GA's. An overview of GA algorithmic details, e.g., crossover and mutation, population sizing, network structure and function specification (as defined by a GA chromosome -- see *methods*), and theory regarding the expected effects of diploid/dominance on GA performance in difficult problems.

II. Review of Literature

Survey of Doppler Signal Processing

Pattern recognition of Doppler flow velocity wave forms originated with Keller (1976) and Rutherford (1977) who developed statistical pattern recognition approaches that predicted the degree of carotid arterial blockage from blood flow measurements using Doppler ultrasound. These approaches utilized hand-measured parameters. A similar approach using digitally computed and selected measurements was described in Greene (1982). Methodology has also been described for *matching* the transfer functions across different manufacturers' Doppler system electronics. Such a procedure is needed, for example, when implementing signal pattern recognition with training results taken from a different transducer/electronics combination Greene (1989)(p. 64). Similar approaches have been taken with predicting gestation times from fetal umbilical artery Doppler data Maulik (1982) and Saini (1986).

As mentioned in the introduction, the latter (fetal) cases require an additional level of capability since no ECG is available as a fiducial, or registration, event. Results from my personal experience in the field suggests that a system consisting of a network of rules and simple digital filters, can be heuristically designed to not only locate the ECG from the Doppler signal in real time, but also may be capable of measuring spectral characteristics of the Doppler flow signal. Such measurements present a definite challenge for application across all disease states and S/N conditions. An interesting feature of these methods was their potential to increase the ease of use and accuracy with which hand-held, low-cost Doppler instruments can be put to use.

Survey of Genetic Search

This section describes terminology and some theory behind GA's. Many of the concepts presented have bearing on the application in question. Other concepts, such as population sizing vs. string length, are presented as current state of GA theory

even though they are not utilized as hard and fast design rules. Following the section entitled "schemata as processing elements" are selected applications that make use of the terms and concepts now to be discussed.

Terminology

Most of GA terminology is based on terms used in biology. This table summarizes the basic terms:

Natural	Genetic Algorithm
chromosome	**string or vector of binary or real nos.**
gene	**string or substring (used interchangeably with "chromosome")**
allele	**value of gene or string element**
locus	**string element index**
genotype	**genetic level coding (e.g., binary bits)**
phenotype	**(scalar) level of fitness $\propto$ 1 / [network error]**
schema/schemata (pl.)	**similarity template for sets of string elements**

Basic Genetic Algorithm

The following pseudo-code is presented in Fitzpatrick (1988) for a simple genetic algorithm. *P*(*t*) represents a population of chromosomes at generation (time) *t*. The function r*ecombine* contains the crossover operator that is explained in the following sections. *Select* picks chromosomes for propagation and recombination according to fitness and *evaluate* applies the fitness or objectivity criterion to the newly recombined population.

```
procedure genetic algorithm
begin
    t= 0;  initialize P(t);  evaluate P(t);
    while (not termination condition) do
    begin
        t= t+1;  select P(t) from P(t-1);  recombination and mutation;  evaluate P(t);
    end
end
```

Crossover

Crossover (sometimes referred to as *recombination*) of genetic material involves a limited exchange of string data between two parent strings to produce two children. The key feature of crossover is that a series of contiguous string elements are swapped ("uniform" crossover and real-encoded strings modify this procedure slightly, as will be discussed below.) The result is that relatively short string subsets, or *schemata*, are more likely to propagate to future generations than are longer subsets of equal fitness (this will also be further discussed.) Since these schema are selected according to their contribution to string fitness, they can commingle to create highly fit strings and rapidly locate the global optimum. In this sense, crossover gives GA's what is called a *global search* capability that greatly increases the ability to handle relatively complex and multi-modal fitness function behavior. Crossover was studied extensively by Holland (1975) in his analysis of genetic search. There can be either one or multiple sites at which crossover occurs. A more in depth analysis of crossover, as used in GA's, is given below.

Schemata as Processing Elements: The Effects of Crossover

This section provides mathematical explanations of how GA's efficiently utilize non-contiguous chromosome sub-strings, or schemata, to locate a near optimal coding. The GA operator that does most of this work is *crossover*, which provides a mechanism for recombining schemata and encourages the propagation of short highly fit schemata or *building blocks*. Much of the to be discussed theory has to do with how crossover is thought to operate. Other theory presented in this section attempts to identify conditions (e.g., population size) under which GA's will more likely succeed or fail.

Definitions

As mentioned in the introduction, the heart of genetic search involves random crossover of genetic material. A *schema* is a *similarity template*, H, of a chromosome that is encoded to represent "solutions" having varying fitness, f(H). A given schema defines the positions (e.g., bits), of a particular sub length of the chromosome that are of actual relevance in locating the global optimum. For example, in the schema H= *11*0**, the *'s are "don't care" positions in the schema. Its *length* is defined to be the total length of the schema including *'s, and is equal to 7. Its *defining length* is the length from the first non "*" to the last "*", and is equal to 4. Its *order*, o(H), is the number of non *'s, and is equal to 3. The number of schemata of a given order is $2^{o(H)}$. The number of schema in an l bit string is 2^l, since each position is either defined or an "*". Therefore, the number of schema in a population of n, l bit strings is somewhere between 2^l and $n2^l$, depending on the number of

duplicate schema in the population. Each schema functions can be considered to be a processing element and participates in representing a *phenotype* having a fitness value f(H).

Each schema may be viewed as a partial solution which is defined by its fixed positions. Schemata can be organized into "competition partitions", or sets of schemata that share the same fixed bit positions, but with differing values in the "don't care" positions. Every member of a population, which represents a complete solution, belongs to exactly one schema within each competition partition. The partial solutions within a competition partition compete, in effect, for representation in the population of complete solutions.

Values for f(H) can be estimated using the mean fitness of all chromosomes containing H. *Schema variance* is estimated similarly. By substituting 0's for all "don't cares" and 1's for all fixed schema positions, Walsh transforms have been used to analyze, among other things, signal to noise of competing schema partitions Rudnick (1991),Goldberg (1991),Goldberg (1989). Rudnick interprets schema variance as the force tending toward convergence, i.e.,*partition signal strength*. The *partition noise* is then defined as the sum of the schema variances for all competing schemata.

Schemata Interpreted: Hyperplanes in Solution Space
This section shows how the concept of schema helps to understand the mechanism behind genetic search., following an example from Whitley (1993). The concept, schema in genetic search , is generally attributed to (Holland, 1975). As an example of schemata with defining length of 3, consider the function in figure 2.1, which is to be searched for its

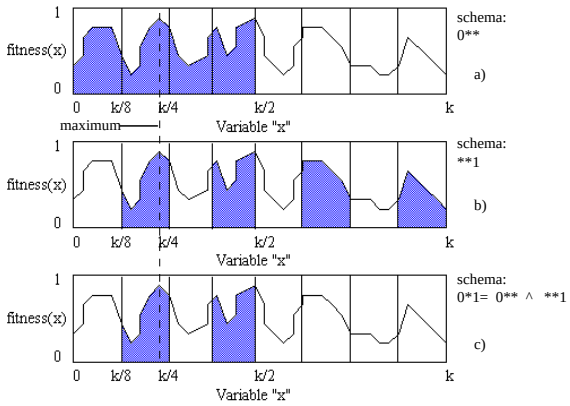


Figure 2.1. Examples of schemata as hyperplanes partitioning a search space. The shaded areas represent the portions of the search space defined by the fixed schema bits.

single maximum. The search has a resolution of 3 bits (e.g., the population of 3-bit strings). The intersection of the top two order-1 schemata shown in (a) and (b) is an order-2 schema as shown in (c). A population consisting of a variety of such schema provides a variety of hyperplane (sub-spaces) and makes it possible to locate a global optimum in the indicated multimodal fitness function regardless of the starting point. Many differenthyperplanes are evaluated in a manner that Holland terms *implicitly parallel* Holland (1975) (pg. 74). Whitley explains this term as follows:

> "... it is the cumulative effects of evaluating a population of points that provides statistical information about any particular subset of hyperplanes. *Implicit parallelism* implies that many hyperplane competitions are simultaneously solved in parallel. Because genetic algorithms operate on populations of strings one can track the proportional representation in the population over time when fitness based selection is combined with crossover to produce offspring from existing strings in the population."

Crossover of genetic material favors shorter length schema over longer ones (assuming approximately equal schema fitnesses), since a relatively long schema is more likely to be disrupted by crossover) As a result, we can expect short length *building blocks* will be located and subsequently combined with increasing frequency (these claims will be further discussed.) As stated in Goldberg (1989a).(pg. 41): "... by working with ... building blocks ... we have reduced the complexity of our problem -- instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings.

Optimal Trial Allocation
In Holland (1975), an explanation for how a highly efficient search for sub strings that leads to an optimal or near optimal solution is presented. Holland considers a simplified, hypothetical population having only two schemata. The problem, then,

is to find out which schema is the better of the two, by observing phenotypic fitness. This is identical to a two-armed slot machine where we wish to determine the arm with the better payoff, by observing trial outcomes, and then after some "optimal" number of trials dedicate our trials to the "best" arm. The situation is modeled by writing the *expected loss* for any allocation of n trials to schema 2 and N-n trials to schema 1. We also are given the mean and variance of the *payoffs* for the two schemata: $\mu_1, \mu_2, \sigma_1, \sigma_2$. The idea is that loss will occur due to 1) wasted trials during the experimental phase that *could* have been allocated to the best schema (arm) and 2) a mistaken decision about which schema (arm) is actually best. The resulting expression for loss is:

$L(N, n) = |\mu_1 - \mu_2| \bullet [(N-n)q(n) + n(1-q(n))]$

where: N is the total number of trials, n is the number of trials during the initial experimental phase, and q(n) is the probability that the worst schema (arm) is actually determined to be the best arm during the experimental phase.

By approximating q(n) as the tail of a normal distribution:

$$q(n) \cong \sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2\pi n}} \frac{e^{-x^2/2}}{x \bullet (\mu_1 - \mu_2)} \qquad (2.11)$$

and setting the derivative of L(N,n) to zero, Holland provides mathematics to show that if n^* trials are allocated to what is actually the *worst* schema (arm), then the number of trials that should be *optimally* allocated to the better arm (schema) is bounded by:

$$n_{\text{optimal}} = N - n^* \cong N \le \sqrt{8\pi b^4 \ln N^2} \bullet e^{n^*/2b^2}, \text{where } :b = \sigma_1 / (\mu_1 - \mu_2), \qquad (2.12)$$

In other words, an optimal plan for allocating trials will give slightly more than an exponentially increasing number to the best schema (arm). This approach assumes knowledge of trial outcomes before they occur, and as such does not represent a realizable scheme, but does provide a descriptive bound on performance that an ideal approach would achieve. As will be shown below, fitness proportionate reproduction allocates an exponentially increasing number of schema with each new generation, which is very nearly the behavior of eq. 2.12.

Each schema may be viewed as a partial solution which is defined by its fixed positions. Schemata can be organized into "competition partitions", or sets of schemata that share the same fixed bit positions, but with differing values in those positions. Every member of a population, which represents a *complete* solution belongs to exactly one schema within each competition partition (the competing partitions can also be thought of as spanning a subspace of the space defined by entire string). The partial solutions within a competition partition compete, in effect, for representation in the population of complete solutions.

The Schema Theorem

A fundamental theorem of genetic algorithms was originally presented in Holland (1975), called the *schema theorem*. Discussions of this result are also given in Goldberg (1989a) and Whitley (1993). The end result is an equation that gives a *lower bound on the expected number of schema* at the next generation, t+1:

$$m(H, t+1) \ge m(H, t) \cdot \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1} - o(h)\, p_m \right]$$

where: p_c = probabiilty of crossover of genetic material (at a single site)

p_m = probability of mutation (altering a single isolated bit)

δ(H) = total number bits from the first to last defined (fixed) bits in H (H's *defining length*) (2.41)

$o(H)$ = number of defined (fixed) bits in schema H (H's *order*)

$\bar{f}$ = average fitness of all strings in the population

$f(H)$ = average of fitness of the strings that contain schema H

The schema equation is based on the reproductive growth rate of schema, which is defined as: $m(H, t+1) \equiv m(H, t) \cdot \frac{f(H)}{\bar{f}}$. The rightmost terms in "[ ]" brackets of (2.41) accounts for loss in the number of schema due to crossover and mutation, respectively. The term $p_c \frac{\delta(H)}{l-1}$ gives the joint probability of a crossover event occurring *and*

of that event occurring within the length of the schema defined by its fixed bits. The result is an upper bound on schema disruption due to crossover, since it is still possible, though unlikely, that the material being crossed over is identical in the two parents. This latter term is especially important since it takes into account the fact that shorter length schemata are more likely to survive and thereby propagate to future generations. The term $o(h)\,p_m$ is the joint probability of a mutation event occurring *and* of the event occurring on one of the fixed bits in the schema (thereby disrupting the schema.)

A most important conclusion from the schema theorem is that it predicts exponential growth/decay for schema of above/below average fitness. To see this, recognize that the basic growth rate

$$m(H,t+1) = m(H,t)\cdot\frac{f(H)}{\bar{f}}\bigg|_{f(H)=\bar{f}+c\bar{f}} = m(H,t)\cdot(1+c)\ , \qquad (2.42)$$

can be re-written as:

$$\Rightarrow m(H,t) = m(H,0)\cdot(1+c)^t \qquad (2.43)$$

where c designates a (hypothetical) constant amount of above/below average fitness for every generation. In this sense, the fitness reproduction as used in genetic algorithms provides a near optimal allocation of trials to the observed best schema, at least as described by 2.12. Eq. 2.43 also provides the basis for predicting growth rates of various reproductive schemes, as will be described later.

Expected Number of Schema in a Population

For the purpose of determining adequate population size, one figure of merit is the expected number of schema in a population of binary strings Goldberg (1989b) . Assuming the probabilities that any particular bit is 1 or 0 are equal ($p_1 = p_0 = 1/2$), the probability of a match in all fixed positions in a schema of order 'i" is $(1/2)^i$. The complementary probability of no matches in a given string is then $1 - (1/2)^i$, and the joint (assuming independence) probability of no matches in a population of n strings is then $[1 - (1/2)^i]^n$. The complementary probability of at least one match in the population is then $1 - [1-(1/2)^i]^n$. The total number of possible *different* schemata of order i is 2^i, and in a string of length *l* there are $\binom{l}{i}$ sets of such schemata. The expected number of schemata using the above assumptions are then:

$$S(n,l) = \sum_{i=0}^{l}\binom{l}{i}2^i\left\{1-\left[1-\left(\frac{1}{2}\right)^i\right]^n\right\}. \quad (2.21)$$

Taking just the first term within the brackets, we have,

$$\sum_{i=0}^{l}\binom{l}{i}2^i\{1\} = \sum_{i=0}^{l}\binom{l}{i}2^i 1^{l-j} = (2+1)^l = 3^l\ .$$ Using the approximation $[1-x]^n = \sum_{j=0}^{n}\binom{n}{j}x^j \cong e^{-n/x}$, for sufficiently large n (greater than approximately 2^l), the resulting asymptotic expression for expected schema in a large population of strings is:

$$S(n,l) \cong 3^l - 2^l e^{-n/2^l}. \quad (2.22)$$

To get a small population asymptotic expression for *S*( ) (2.21) is re-written so that we sum over "n".
First, the term $[1-(1/2)^i]^n$ is expanded (using the binomial theorem) into a summation:

$$S(n,l) = 3^l - \sum_{j=0}^{n}\binom{n}{j}(-1)^j\left[1+\left(\frac{1}{2}\right)^{j-1}\right]^l.$$

By summing over n instead of *l*, as was done above, we can also obtain an asymptotic expression for the number of expected schema in small populations.

$$S(n,l) = 3^l - \sum_{i=0}^{l}\binom{l}{i}2^i\left\{\sum_{j=0}^{n}\binom{n}{j}1^{n-j}(-1/2)^{ji}\right\}$$

$$= 3^l - \sum_{i=0}^{l}\binom{l}{i}2^i\left\{\sum_{j=0}^{n}\binom{n}{j}(-1)^j(1/2)^{ji}\right\}, \text{ reversing the summation order :}$$

$$= 3^l - \left\{ \sum_{j=0}^{n} \binom{n}{j} (-1)^j \sum_{i=0}^{l} \binom{l}{i} 1^{l-i} 2^i (1/2)^{ji} \right\}$$

$$= 3^l - \sum_{j=0}^{n} \binom{n}{j} (-1)^j \left[\sum_{i=0}^{l} \binom{l}{i} 1^{l-i} (1/2)^{(j-1)i} \right], \text{ applying the binomial theorem :}$$

$$= 3^l - \sum_{j=0}^{n} \binom{n}{j} (-1)^j \left[1 + \left(\frac{1}{2} \right)^{j-1} \right]^l, \text{ taking the first two terms as an aproximation :}$$

$$\cong 3^l - 3^l + n2^l - 2^l = (n-1)2^l \cong n2^l.$$

Goldberg makes these results more useful in Goldberg (1985), by arguing that the schema contained within a string are useful *only* when complimented by the same or different schemata in other strings, and should not themselves be counted. He therefore proposes the "effective" number of schemata as: $n_{es} = n_s - 2^l$; where n_s is given by eq. 2.21. He then does a fibonacci search Avriel (1976) to find the optimal population size (maximum of $n_{es}(n)$ for a various string lengths. The results roughly agree when compared to *average* optimal populations found for an independent test suite of GA problems DeJong (1975), DeJong (1980), Grefenstette (1986). For a typical binary string with $20 < l < 30$, Goldbergs's findings Goldberg (1985) suggested a population in the range of roughly 100 to 300.

Genetic Drift and Diffusion Theory

Since GA's rely on populations from which candidates are drawn for mating and recombination, the process has a discrete nature to it. If the populations are sufficiently small, there results a phenomenon caused by sampling error called *genetic drift*. This situation, in the absence of a mechanism such as adequate mutation rate, can cause the population to be overtaken by a particular allele value prior to locating the optimum, resulting in failure to locate the global optimum. This section analyzes genetic drift using diffusion theory. Some insight into appropriate mutation rate for a given population size (or vice versa) result from this analysis.

[Inman, 1993 #59] describes genetic drift as follows:

> "If a coin is tossed 100 times, then on average it will be heads 50 times, but it is unlikely to be exactly 50. The same holds if 100 random [coins] are selected *with replacement* from [a population consisting of exactly] 50 heads and 50 tails [without flipping any]. In the [latter, selection] case, repetition of the process will on average give the same result as on the previous, but the variance allows significant change in the average [or *drift*] over time. If at any stage the selection resulted in all heads (or tails), then future change would be impossible."

One of the consequence of genetic drift analysis is that if the mutation rate is too small for the population size and fitness pressure, then each allele (string position) in the population will eventually converge to one (binary) value or the other. Intermediate states will lend excessive weight to the allele proportions in the initial (random) population making successful convergence to the global optimum dependent on the makeup of the initial population.

Analysis of genetic drift using diffusion calculus has a history extending back decades in diploid populations, as described in Roughgarden (1979) in detail. The idea is to treat a (stochastic) sequence of populations, at a specific chromosome locus, as a Markov chain which is approximated by a diffusion equation which equates change in density at a location, x, to the spatial derivative of the flow rate:

$$\frac{\partial}{\partial t} \rho(x,t) = -\frac{\partial}{\partial x} J(x,t) \quad (2.51)$$

The density, ρ, corresponds to the proportion of populations (considered as separate trials) that have a value of, x, at the relevant string locus, at time (generation) t. $J(x,t)$ contains the effects of mutation, selection, and *diffusion* (note that crossover is not a factor, since it doesn't introduce any *new* allele values). Specifically:

$$J(x,t) = M(x)\rho(x,t) - \frac{1}{2} \frac{\partial}{\partial x} V(x)\rho(x,t), \quad (2.52)$$

where $M(x)\Delta t$ is the average distance traveled from a point x under force of mutation *and* selective pressure and $V(x)\Delta t$ is the variance of the (random) distances traveled from a point x. What is desired is the steady-state distribution of $\hat{\rho}(x)$ in those situations where it exists, which implies that $J(x) = 0$ so that from (2.51) we have:

$M(x)\rho(x,t) = \frac{1}{2}\frac{\partial}{\partial x}V(x)\rho(x,t)$, which after a change in variables results in

$$\hat{\rho}(x) = \frac{c}{V(x)}\exp\left[2\int\frac{M(x)}{V(x)}dx\right]; c \text{ chosen so that } \int_0^1\hat{\rho}(x)dx = 1 \qquad (2.53).$$

For a haploid population, [Inman, 1993 #59] shows how to introduce mutation and selection pressures and that for one generation, the variance in Δx is x(1-x) (where x is the *proportion* of 1's at the locus of interest). The result is:

$$\hat{\rho}(x) = \frac{c(1+sx)^{2N}}{\left[x(1-x)\right]^{1-2p_mN}};\ where:\ s \equiv \frac{f_1 - f_0}{f_0} \equiv selective\ pressure\ for\ a\ "1" \qquad (2.54)$$

and, for a mutation rate (p_m) of zero gives: $\hat{\rho}(x) = \frac{c(1+sx)^{2N}}{\left[x(1-x)\right]}$, which is a "U" shaped curve that reflects the tendency for such a situation to result in an almost immediate convergence to either a 0 or a 1 at the locus of interest. We might conclude that, in general, we should have $N \cdot p_m >> 1/2$ to prevent significant genetic drift from occurring (by making the denominator of eq. 2.54 much less dependent on x). Roughgarden (1979) works out this result for the diploid case, resulting in $N \cdot p_m >> 1/4$. The inequalities are affected somewhat, of course, by the amount of selective pressure, *s*. In conclusion too little mutation or too small population size (or too much selective pressure) can result in premature convergence and excessive dependence on the specific make-up of the initial population in less extreme cases.

Deceptive Fitness Functions

Deception refers to a situation where low-order schemata of relatively high fitness can lead the search *away* from the global optimum. Such a condition can arise because of non-monotonicity or multi-modality in the fitness function. It can also be affected by the way the string is encoded, as shown below in the section on real-encoded strings. Any problem requiring a GA approach will have some deception involved, and attempts have been made to characterize GA test problems in terms of the amount of deceptiveness they exhibit..

Deception can sometimes be made less deleterious, but not completely eliminated, by reducing the fitness pressure or increasing the mutation rate. Whitley (1991) states that most problems involve some deception, since in general we would not expect that *all* competing hyperplanes defined by a particular schema will be leading towards the global optimum. A *fully* deceptive problem is one in which *all* hyperlanes of lower order schema lead away from the global optimum. He then provides a proof that for a function to be fully deceptive either the deceptive attractor or some string that differs from the deceptive attractor by 1 bit must be a local optimum in Hamming space. As a result we can say that a fully deceptive attractor will be the complement of the global optimum in Hamming space

We can also define deceptive GA problems of order-N, corresponding to the order of the hyperplane competition at which one or more relevant *lower* order hyperplane competitions guide the search away from the global optimum. A *deceptive attractor* is a hyperplane of order N other than the global optimum. A *fully deceptive* problem is one in which all relevant lower order hyperplanes lead toward a deceptive attractor. Liepins and Vose Liepins (1991) constructed fully and partially deceptive GA test problems.

Schema Processing Efficiency and Implicit Parallelism

For a hypothetical binary string of length *l*, where any position can be either a 0, 1, or "don't care" symbol, the number of possible schema is 3^l. If we consider an actual (instantiated) population, a particular string in that population will actually have a potential for 2^l schema, since any position will be either be fixed (at either a 1 or a 0), or it will be "don't care" symbol. In a population of n strings, then, and depending on the actual diversity of this (instantiated) population, there may be *as many* as $n2^l$ schema.

Short "Proof" of implicit parallelism: That $o(N^3)$ schema are processed per GA generation:

The following is presented in Fitzpatrick (1988) and gives an *indication* of how schema processing is $O[N^3]$, where N is the population size.

Given:

A population P of N randomly valued binary strings of length L.

To Show:

At least N^3 schema are allocated trials per generation according to:

$$M(H,t+1) = \frac{\mu(H,t)}{\mu(P,t)} M(H,t)$$

where: $P(t)$ = current population of strings (2.31)

$\mu(H,t)$ = avg. fitness of strings that are in both P(t) and H

$\mu(P,t)$ = avg. fitness of strings in P(t)

Proof:

Consider only schemata that have at least r representations in P. Let the schema order be k= $\log_2$ (N/r). For any choice of k positions, there are 2^k unique schemata defined at those k positions. Each of these are represented by r chromosomes in P. Therefore, the number of distinct schemata with r representatives in P is at least:

$$M_r = 2^k \bullet \binom{L}{k}. \quad (2.32)$$

Since k= log(N/r), 2^k= N/r, and N= 2^kr. Assume we have L > 64 and $2^6 \le N \le 2^{20}$ for an typical problem, and it is reasonable to require that r > 7. If r = 8, then $3 \le k \le 17$. By inspection of (2.32) over this range, we find $M_r|_{k=3;L=64} > 3*10^5$, $M_r|_{k=17;L=64} > 10^{20}$, and $N^3 = 2.6*10^5$. Therefore, we have $M_r > N^3$. Notice that this counts only schemata of *exactly* order k. The sum of all schemata from order 1 to m that are processed is given by $\sum_{i=1}^{m-1} 2^i \binom{L}{i}$. A more general proof of the O[n^3] claim to schema processing efficiency is given in the Appendix.

Strings with Cardinality Greater Than Two

Genetic algorithm researchers commonly utilize binary strings for two reasons: 1) Binary vectors are simpler and the associated mathematics involved in studying GA behavior is tractable; 2) The desire to maximize the number of schema being processed, with the assumption being that with no underlying information being unavailable, the most information is available for recombination. In Antonisse (1989), this latter notion that bitwise recombination results in an inherent increase in schema processing efficiency is brought into question.

The use of binary strings can be traced to Holland (1975) where the "*", or "don't care" operator is introduced to identify schema bit positions that can take any value. Antonisse (1989) (pg. 88) points out that the "... don't care symbol can be construed as denoting the set of strings sharing a *subset* of possible values." Comparisons are drawn in terms of the number of *states* that can be represented by a string of cardinality *k* elements, and if string sizes are adjusted so strings of different cardinality have a comparable number of states. Since the number of schema in an uninstantiated *binary* string is 2^l, and this formulation was also applied to higher cardinality strings, it appears that the binary string will always have more schema than its representationally equivalent string of higher cardinality.

Antonisse, as previously mentioned, asserted that the "don't care" symbol represents all possible similarity *subsets* (as opposed to singletons) of cardinality values. This assertion radically increases the number of schemata being processed, even when string length is shortened according to the increase in cardinality. A number of papers have referenced Antonisse without seriously contesting this idea, including the following:

- Michalewicz (1992) presents real crossover and mutation operators and extensive examples and guidance for their use in function optimization. These operators can be summarized as follows:

Mutation Operator (mutation is non-uniform, meaning that it varies with generation)

$$\text{if } v \text{ is chromosome, and } v_k \text{ is the element selected for mutation :}$$

$$v'_k = \begin{cases} v_k + D(t, 1 - v_k) \text{ if a random binary digit is 0} \\ v_k - D(t, v_k) \text{ if a random binary digit is 1} \end{cases}$$

$$\textit{where}: v'_k = \text{new element value after mutation,}$$

$$D(t, y) = y \cdot \left(1 - r^{(1-g)^b}\right), \qquad 2.41$$

$$r = \text{random number}[0..1], \; g = \text{convergence factor}[0..1]$$

$$b = \text{degree of non-uniformity (e.g., b = 5)}$$

$$\textit{Net effect}: \text{as g} \to 1, \text{ mutation rate is reduced : typically g} \propto \text{t}$$

Crossover Operator (crossover also varies with generation)

$$\textit{if } s_v^t \textit{ and } s_w^t \textit{ are crossed (at generation } t \textit{) :}$$

$$\textit{the resulting offspring are :}$$

$$s_v^{t+1} = \left\rangle v_1, \ldots, v_k, w_{k+1} \cdot a + v_{k+1} \cdot (1-a), \ldots, w_m \cdot a + v_m \cdot (1-a) \right\langle \in S \qquad 2.42$$

$$s_w^{t+1} = \left\rangle w_1, \ldots, w_k, v_{k+1} \cdot a + w_{k+1} \cdot (1-a), \ldots, v_m \cdot a + w_m \cdot (1-a) \right\langle \in S$$

$$\textit{where}: S = \textit{convex solution space}$$

The above mutation operator is used here as described in the methods and results section. The above crossover operator is modified slightly in light of more recent research as follows:

- Radcliffe (1991), introduced a more general interpretation of schema/schemata call "forma/formae". This permits a more formal interpretation of Michalewicz's real crossover operator, which he terms a *respectful* operator, in that the offspring produced (either by muataion or crossover) are members of the space spanned by the two parent chromosomes.
- Eshelman (1992), extended the real-parameter crossover operator of Michalewics, that is *not* "respectful", to one that permits offspring within the convex hull of the space spanned by the parents. With such an operator, *any* point in the search space is reachable *even* if the original population has limited allele representation.

The following argument in Goldberg (1991b), summarizes his argument *against* using higher cardinality strings :

> Assume there are *k* schemata per string position *plus 1* for the "don't care" symbol. Each [cardinality *k*] position requires $\log_2 k$ bits to be represented and so there are therefore $n_s = (k+1)^{1/\log_2 k}$ schemata *per bit* in the string. Taking $\frac{\mathrm{d}\ \log_2(\mathrm{n_s})}{\mathrm{dk}} = \frac{k \ln k - (k+1)\ln(k+1)}{k(k+1)\ln^2 k}$ and noting that $k\ln(k)$ monotonically increases requires that the derivative be negative for all *k*, and that therefore *smaller cardinality strings have more schemata per bit*.

The latter argument assumes that efficient use of binary machine bits and the resulting ability to specify a crossover point between any two such bits is of primary importance. One can also argue that if memory *objects* having more than one bit can be processed with high efficiency, they can be treated more like binary bits. For example, hardware floating point processors make efficient use of a groups of bits as a single object representing a real number. Such an object cannot be arbitrarily subjected to crossover/recombination, when made available for bit-wise crossover, without highly non-linear consequences. Put another way, a crossover *or* mutation event taking place *within* a real number would result in epistatic behavior, since random juxtaposing parts of exponents and mantissas would not have a known useful result. Goldberg very objectively points out in the same article the following reasons ***in favor*** of using high cardinality strings:

- *Elimination of Hamming cliff deception:* If a binary encoded string element is subjected to crossover, non-monotonically increasing hamming distance (with changing fitness), in a particular application's encoding, may require multiple, simultaneous bit changes to occur while in the process of searching for a global optimum. In this scenario, a single bit change due to crossover/mutation, when several are actually necessary, ends up directing the search towards a sub-optimum. Such single bit changes require repeated, independent mutations, with a correspondingly small joint probability of occurrence. Use of Gray codes might seem helpful, but appears to actually not be the case in light of an

analysis using Walsh functions in Goldberg (1989). Such *deceptive* behavior is the subject of much analysis The point of my argument here is that the use of high cardinality alphabets eliminates the possibility of disruption within a parameter, and may justify its use from a purely practical standpoint.

- Real-coded GA's permit the use of mutation operators that perturb the current solution a little about the current value. Binary GA's, on the other hand, implement mutation by randomly flipping individual chromosome bits without regard to any possible linear relationships in the problem encoding. As a result, real-coded GA's can *hillclimb* local optima in the underlying decision space at varying rates. This capability is exploited with both mutation and crossover and is further discussed in the *methods* section.

The Genitor GA Algorithm

This algorithm is a public domain GA implementation that was used, with modifications and additions, in this proposal (as described in the *methods* and *results* sections.) Genitor is capable of producing strong fitness pressure with a single string evaluation per generation, largely a result of always and only deleting the worst individual. No function evaluations are wasted on children that have not been subjected to crossover, since in effect, $p_C = 1$, apparently increasing efficiency (since no evaluations are "wasted" on children that have only been subjected to mutation.) A likely disadvantage, as will be shown, is that the propagation of highly fit individuals does not taper off logistically with time as does the standard GA algorithm. The result may be an increased tendency for *premature convergence* to a sub-optimum result. Such a condition occurs when a relatively high fitness pressure, for example, causes a sub-optimal solution to completely take over the population. Since diversity is lost, normal crossover cannot generate new string combinations, and mutation will usually result in less fit progeny that rapidly get deleted from the population Diploid/dominance, it can be argued, reduces the required mutation rate to maintain alleles of relatively low fitness, as will be shown later in this and the *results* section. Since an approach to diploid/dominance is being studied as part of this dissertation, the high growth rate provided by Genitor is a reasonable choice for this dissertation. Another consideration in choosing Genitor is that the code was readily ported from UNIX to PC-DOS, and was also easily converted to C++ (one other package evaluated was not nearly so portable.)

The Genitor algorithm utilizes *overlapping* (sometimes also referred to as *static*) populations, which are characterized by the survival of one or more individuals from one generation to the next. This means that, except for the offspring, all members of any previous population survive, unmodified, to the present population. Rank based selection is used, which as analyzed below, removes the actual fitness value from the selection *process*. The (single) offspring created with each generation is inserted into the ranked population according to its fitness, and all strings below it are moved down one position with the worst being deleted. The result is a high degree of selective pressure, as will now be shown.

Propagation rates for two GA implementations: Non-overlapping rank based selection and Genitor

Goldberg (1991a) provides an in-depth analysis of several GA implementations, including Genitor. Goldberg therein compares GA algorithms on the basis of the propagation rate of individuals having fitness in excess of a some arbitrary class boundary (i.e., the *best* individuals.) Such an analysis defines the proportion of such "highly fit" individuals as:

$$P_{t+1} = \beta(P_t), \quad (2.71)$$

where t is the generation number and β is a *cumulative assignment function*, defined by:

$$\beta(x) = \int_0^x \alpha(\eta)d\eta;\ \int_0^1 \alpha(x)dx = 1; \alpha(x) > 0 \quad (2.714)$$

The range of x is [0,1] designating the most fit as x=0 and the least fit as x=1. Fitness decreases monotonically from 0 to 1. If P is the proportion of individuals with fitness better than or equal to some particular fitness level, *f*, then the proportion of individuals assigned to the same fitness range in the next generation is as stated above. The following *linear* assignment function is used for the experiments in this proposal: $\alpha(x) = c - 2(c-1)x$ (where: c=constant). This satisfies the above requirements (2.714), and results in $\beta(x) = cx - (c-1)x^2$, which when substituted into eq. 2.71 gives a difference equation for the generational GA as:

$$P_{t+1} = P_t[c - (c-1)P_t] \quad (2.715).$$

For the Genitor, *overlapping* population algorithm, Goldberg observes that exactly one individual is assigned into the fixed size population every Genitor generation, or what he refers to as an *iteration*. We can thereby analyze the growth of

individuals in blocks of *n* iterations above a given fitness level by defining a new index: τ= nt (n= population size). The proportion of individuals assigned in the next block, or generation, at or greater than a given fitness is then:

$$P_{\tau+1} = P_\tau + \frac{P_\tau[c-(c-1)P_\tau]}{n}$$

The arguments for this last result are that: 1) the population model is "overlapping", i.e., not all individuals die out prior to the succeeding generation, 2) only the *worst* individual does in fact die, and 3) the rest of the population is identical to the previous except for the single individual that was born. From this result we can derive an analytic expression for fitness proportion at a given generation t= τ/n given the fitness pressure, "c" (which corresponds to the "b" parameter mentioned in the "Results" section) and the initial proportion (as defined above) P_0.

Subtracting P_τ from both sides gives the following difference equation

$$\Delta P_\tau = P_\tau[c-(c-1)P_\tau]/n \qquad (2.72)$$

which can be approximated by the derivative, $\frac{dp}{dt}$

$$\Rightarrow \frac{n\cdot dp}{P[c-(c-1)P]} = d\tau$$

Expanding using partial fractions gives: $\frac{A}{P} + \frac{B}{c-(c-1)P} = \frac{n}{P[c-(c-1)P]}$

$$P=0 \Rightarrow A = \frac{n}{c-(c-1)P} = n/c; P = c/(c-1) \Rightarrow B = n\frac{c-(c-1)P}{P[c-(c-1)P]} = n/P = n(c-1)/c$$

$$\Rightarrow d\tau = n\cdot dp[\frac{1}{cP} + \frac{c-1}{c[c-(c-1)P]}].$$ Integrating and using t = τ/n, we get:

$$ct = \ln P\Big|_{P_0}^{P} + \int_{P_0}^{P}\frac{c-1}{c-(c-1)P}dP = \ln P - \ln P_0 + \int_{P_0}^{P}\frac{c-1}{c-(c-1)P}dP = \ln P - \ln P_0 + \int_{P_0}^{P}\frac{1}{\frac{c}{c-1}-P}dP =$$

$$\ln P - \ln P_0 + \ln(u-P)\Big|_{P_0}^{P}; \text{with}: u = \frac{c}{c-1} \Rightarrow e^{ct} = \ln\left(\frac{P(u-P_0)}{P_0(u-P)}\right) = v$$

let $P' = \frac{c-1}{c}P_0$, and solving for P : (= P_t)

$$P_t = \frac{vP_0}{(1-P')(1+\frac{vuP'}{u-uP'})} = \frac{vP_0}{(1-P')(1+\frac{vP'}{1-P'})} = \frac{vP_0}{1-P'+vP'} = \frac{cvP_0}{c-(c-1)P_0+(c-1)P_0v}$$

$$= \frac{cvP_0}{c+(c-1)P_0[v-1]} = \frac{cP_0e^{ct}}{c+(c-1)P_0(e^{ct}-1)} \quad (2.73)$$

Eq. 2.73 is logistic in form and as such asymptotes to a constant proportion for either extreme in t (t = the generation count.) However, if we take the limit as t goes to infinity we get: $P_t \rightarrow c/(c-1)$ and, since $1 < c \leq 2$, this limit can be no less than 2. Since $0 \leq P_t \leq 1$, the actual behavior of the Genitor growth is to "crash" into its upper limit of P=1. Such a characteristic may contribute, as Goldberg suggests Goldberg (1991a) pg. 84, to premature convergence in the Genitor algorithm to an extent that, seemingly at least, could make it more likely to fail than *generational* algorithms GA's (i.e., where the entire population

is replaced every generation.) Since I wish to consider the possible effects on premature convergence of using a Diploid population, a large selective pressure resulting in high allele loss is not necessarily undesirable. At any rate results described in the next section using Genitor, both for network synthesis and diploid/dominance, are encouraging.

More traditional generational GA's *will* show a true logistic form. The difference equation for the latter, in comparison to 2.72, is $\Delta P_t = P_t[c\text{-}1 - (c\text{-}1)P_t]$, resulting in $P_t = \frac{P_0 e^{(c-1)t}}{1 + P_0(e^{(c-1)t} - 1)}$, which as $t \to \infty$, slowly asymptotes to the value 1.0. As a result we might expect that the takeover rate of highly fit individuals in later generations will be slower, and that the likelihood of premature convergence is lessened, at the cost of decreased efficiency.

Uniform Crossover
If crossover is not expected or desired to produce increasingly fit building blocks, then we can use "uniform" crossover, which randomly selects bits and then exchanges the individual bits. Such a scheme has the property of being more disruptive to schema in general, and to short length schema in comparison to multiple point crossover. If no building block accumulation is expected to be possible, this method should produce a more rapid and thorough search and avoid to a greater extent premature convergence.

CHC Algorithm Eshelman (1991)
This is a modified GA algorithm that utilizes: 1) static generational selection, 2) uniform crossover, and 3) incest prevention to reduce the likelihood of premature convergence (to a local sub-optimum). The authors note that item (2) was replaced with (ordinary) two-point crossover with resulting improvement in performance for the "Liepins-Vose" deceptive problem, as discussed above. This modification makes CHC more similar to Genitor with the exception of (3). Incest prevention is implemented by comparing the Hamming distance of two prospective parents, and only accepting them if that value is greater than the H= (string length)/4. H gets decremented as the population converges. This algorithm is used in the multiplierless digital filter application (below) and was recommended by Whitley in a private communication as a good choice for reducing premature convergence. The source code for CHC is proprietary, however, and not available making it unusable for this dissertation.

Parallel GA's
Parallel processors (MISD) can be utilized to great advantage to evaluate populations of strings in a GA since the fitness function for each is identical and there is no inter-processor communication required during fitness evaluation. Another use of parallel processors (MIMD) is in evaluation of *multiple populations* that locate potentially different highly fit individuals that are subsequently re-introduced to each other Whitley (1993). "N" MIMD processors could be put to use with polyploid populations (see below for some examples) to evaluate N homologue fitnesses.

Example Applications from the Literature

- *VLSI design*
  Martin (1993) used a single cross point GA to optimize data/execution flow-graphs for VLSI implementation. A table of all available node/operators in the repertoire of the design is constructed that lists the execution time, silicon area, and power consumption for each operator (or gate.) The problem is defined by a flow graph for the digital circuit to be made into an IC including all processing steps and node characteristics, and an overall time constraint for one cycle of operation of the circuit is specified. This problem is clearly combinatorial and can get very large for typical IC designs. The parallel bit-width of each node/operator and the time delay from the last of the operator's predecessors in the node graph to finish are both encoded as successive fields in the chromosome. The fitness function itself gives better values to individuals using less silicon area while still satisfying the timing requirements. An important feature of the fitness function is that rather than simply rejecting an individual if it violates the timing constraint, the fitness is gradually penalized according to the square of the excess time required to execute.

  Successful results, which in some cases could not be manually improved upon, were reported for two test cases: A 16 operator arithmetic flow graph Park (1985) and an elliptic filter Kung (1985). The authors indicate they *did* examine differing probabilities of crossover (in addition to mutation and population size) to verify that a GA was in fact being done.

- ***Stack digital filter design***
  Stack filters are a generalization of median filtering. The signal is decomposed into a set of binary signals using a set of associated analog thresholds and a sliding window. Boolean functions of the signals can efficiently produce a set of filters, for each threshold, whose outputs can then be stacked to produce the filter output. If the filters stack, then the

output can be reconstructed without using addition by simply finding the threshold in the outputs at which a one to zero transition occurs. [Chu, 1989 #93] showed how to design stacked filters using a simple genetic algorithm with up to 7 taps. [Oakley, 1994 #92] has subsequently presented results using *genetic programming* that were superior to both median filtering and FIR filtering. Genetic programming is discussed below under *network synthesis*. Oakley states that the genetic programming stacked filter design is presently in use in a laser-Doppler medical instrument.

- *Multiplierless digital filter design*
  Design of digital FIR filters having power of two coefficients (i.e., +/-2^{-n}, or zero) is possible, as presented in [Schaffer, 1993 #94]. Four Gray coded bits are used to encode each filter coefficient in the range of +/-1 in logarithmic steps, including the value zero. The tap length and cascade structure are preset for each GA run.. Fitness evaluation consists of a 1) range violation term (rv_i) that indicates (linearly) the degree to which the ith frequency response bin is out of range, and 2) either a LMS term which sums weighted $(rv_i)^2$ terms, *or*, simply takes the maximum of the weighted rv_i terms. Several alternative algorithms including binary hill climbers and simulated annealing were reportedly tried and the authors state that they do not compare to the CHC genetic algorithm Eshelman (1991). One experiment resulted in the presence of 50×10^6 local optima in the search space for a 31-tap low pass filter design.

- *ART network development*
  Caudell (1992) describes the use of adaptive resonance theory to cluster two-dimensional objects according to shape. In order to enhance the accuracy of the resulting category boundaries, artificially generated data are created using a modified GA. The fitness function used is designed to encourage the creation of samples near the category boundaries. Credit for the idea is given to work described by Hwang (1990), in which generalization in a multilayer perceptron classifier is considerably improved by augmenting the training set with samples near the decision boundaries. This example represents a somewhat different use of GA's, having the purpose of creating pools of candidate individuals that are similar in fitness yet different in their genotype.

- *Population size vs. sampling tradeoff in a noisy signal*: *Analysis and application involving Image registration*
  An analysis of the effects of trading off measurement sampling vs. population size is presented in Fitzpatrick (1988). The idea is to average n *samples* of a random process, which is used in chromosome (fitness) evaluation, to see if there can be a tradeoff with population size. The authors show that a possible increase in overall processing efficiency is possible and that optimum sampling rates may exist. Such a situation occurs, for example, when noisy measurements are to be the basis for evaluating a GA population. The authors show an example of utilizing this in an image registration problem using a digitized angiogram pair in which the average pixel difference is measured over a 100X100 square.. The four corners of a proposed region of interest, which is to be used in registering the two images, are encoded as 2D vectors in each chromosome. A 2D linear transform is then used to map interior points in the trapezoid. The resulting distorted 2nd image is subtracted from the first to obtain the chromosome's fitness, using the absolute differences between the pixels of the two images. Since this action is obviously somewhat time intensive in itself, and we are dealing with noisy data, there is ample reason to average pixel differences prior to GA evaluation in order to allow for either a larger population size or more generations given a fixed evaluation time. By optimizing this tradeoff, increased GA efficiency results. The authors give a basis for this phenomenon which will now be summarized.

  The authors submit that for an arbitrary hyperplane *H*, comprising a set of chromosomes $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x}_{|\mathbf{H}|}\}$, a random sampling of *r* of these x_i results in a random variable *E* having mean $\mu_E = \mu$ and variance $\sigma_E = \sigma / \sqrt{r}$, where μ and σ are the mean and standard deviation of a random variable, $R(\mathbf{x})$, whose mean is equal to the true fitness function $f(\mathbf{x})$. The sample standard deviation for $R(\mathbf{x})$ is $\sigma_S = \sigma(\mathbf{x}) / \sqrt{n}$. The variance in *n* samples of $R(\mathbf{x})$ is:

$$\sigma_n^2(\mathbf{x_i}) = \frac{\sigma^2(\mathbf{x_i})}{n} \Rightarrow \frac{\partial \left\langle \sigma_n^2(\mathbf{x_i}) \right\rangle_H}{\partial n} = \frac{-\left\langle \sigma^2(\mathbf{x_i}) \right\rangle_H}{n^2} \quad (2.81)$$

  By expressing sampled mean fitness as the true mean fitness plus a zero mean random variable, $\eta(\mathbf{x}_i)$, we get: $\eta(\mathbf{x}_j) = \mu(\mathbf{x}_i) - \mu$. A similar expression holds for each individual sample: Let $p_j(\mathbf{x_i})$ be the j^{th} sample taken from the random variable $R(\mathbf{x_i})$ in the estimation of $f(\mathbf{x_i})$, then $\eta_j(\mathbf{x}_j) = p_j(\mathbf{x}_i) - \mu(\mathbf{x},)$. The variance of the average fitness of the n samples taken from r x_i's is then shown to be:

$$\sigma_S^2 = \left\langle \left\langle \left[\frac{1}{r}\sum_{i=1}^{r}\left[\frac{1}{n}\sum_{j=1}^{n}(p_j(\mathbf{x_i})) - \mu \right] \right]^2 \right\rangle_R \right\rangle_H$$

recognizing that : $p_j(\mathbf{x_i}) = \mu(\mathbf{x_i}) + \eta(\mathbf{x_i})$

$$= \frac{1}{r^2 n^2} \left\langle \left\langle \left[\sum_{i=1}^{r}\left[n\mu(\mathbf{x_i}) - n\mu + \sum_{j=1}^{n}\eta_j(\mathbf{x_i}) \right] \right]^2 \right\rangle_R \right\rangle_H$$

assuming that the various "noise" (η) terms are uncorrelated it follows that :

$$\sigma_S^2 = \frac{\sigma^2}{r} + \frac{1}{r \cdot n}\left\langle \sigma^2(\mathbf{x_i}) \right\rangle_H \Rightarrow \frac{\partial \sigma_S^2}{\partial n} = \frac{-1}{r \cdot n^2}\left\langle \sigma^2(\mathbf{x_i}) \right\rangle_H \quad (2.82)$$

note that: $\lim_{n\to\infty}\left(\sigma_S^2\right) = \sigma_E^2 = \sigma^2 / r$

where:

H is an arbitrary hyperplane or schema

E is the average fitness of r, randomly chosen $\mathbf{x_i}$'s from an arbitrary schema H

S is the average fitness of n samples taken from each of r, $\mathbf{x_i}$'s in H

$\langle \ldots \rangle_H$ denotes the mean over all sets of r chromosomes in H

$\langle \ldots \rangle_R$ denotes the mean over all sets of n samples taken from R($\mathbf{x_i}$)

$\sigma =$ schema standard deviation of the random variable R($\mathbf{x_i}$)

$\sigma_n^2 \equiv$ schema fitness variance in n samples taken directly from R

$\sigma_S^2 \equiv$ schema fitness variance of n samples taken from r, $\mathbf{x_i}$'s, randomly chosen from H

$\sigma_E^2 \equiv$ fitness variance of r, $\mathbf{x_i}$'s, randomly chosen from H

$n \equiv$ number of samples

$r \equiv$ number of chromosomes randomly chosen from H: r $\propto$ N

N is the population size

The total number of evaluations taken to estimate the performance of schema H is r•n. Eq. (2.82) therefore suggests that the μ= fitness estimate of H can be improved without additional sampling cost by increasing r (i.e., the population size) and decreasing n so that r•n= constant. We can also consider the effect of decreasing the number of samples, n, without varying the population size:

from (2.82):

$$\frac{\partial \sigma_S^2}{\partial n} = \frac{\partial}{\partial n}\left[\frac{1}{r}\sigma^2 + \frac{1}{rn}\left\langle \sigma^2(x_i) \right\rangle_H \right] = \frac{-1}{rn^2}\left\langle \sigma^2(x_i) \right\rangle_H$$

and from (2.81): $\frac{\partial \left\langle \sigma_n^2(x_i) \right\rangle_H}{\partial n} = -\frac{1}{n^2}\left\langle \sigma^2(x_i) \right\rangle_H$

giving: $$\frac{\partial \sigma_S^2}{\partial n} = \frac{1}{r} \frac{\partial \left\langle \sigma_n^2 \left(x_i \right) \right\rangle_H}{\partial n} \qquad (2.83)$$

σ_S can be viewed as the accuracy of the estimated fitness of schema H. (2.83) states that as the number of samples decreases, the estimated schema performance decreases at a fraction 1/r (proportional to 1/N) of the rate of decrease in the performance of entire strings. In other words, as the number n of data samples used per fitness evaluation increases, the accuracy of the estimated performance of the schema H increases at a rate 1/r times the rate of increase of the average accuracy for evaluating individual chromosomes. Since we assume r is proportional to N, this suggests that there may be an optimum tradeoff between the sampling of "training" data and population size.

By defining the computational cost, β, per sample taken during the evaluation of each chromosome and the cost per evaluation due to the GA itself as α, the authors demonstrate this effect for problems having various ratios of α/β. The effect is seen as a trade off between population size and sampling ($N \bullet n$ is kept constant). Each string represents a solution that is evaluated using n samples from the training data. The tradeoff is in deciding whether to expend computational energy in using more of the available samples in the training data versus sampling and using the freed up computational time to evaluate more strings.

In conclusion the authors state:

> "Since the quality of the search performed by genetic algorithms depends on the quality of its estimates of the performance of [schemata], rather than the evaluation of a particular individual [chromosomes], [eq. 2.83] suggests that genetic algorithms can be expected to perform well for problems requiring partial evaluation of candidate solutions."

This result has significance for the stochastic biological data that is the subject of this dissertation since sampling is inherent in the digitization, FFT rates, etc. For example, a logical result would be to sample more sparsely in early generations and only analyze more thoroughly in later populations when solutions presumably emerge.

Network Synthesis

According to [Schaffer, 1992 #28], GA's have been combined with neural or connectionist networks in two different ways: 1) as an aid before or after training a neural network (e.g., feature selection as in Brill (1992), or in refinement of unsupervised learning, as in Caudell (1992) , above), and 2) to determine the architecture, connection weights or both of the network itself. This proposal will concern itself primarily with the latter approach, for which various methods have been developed.

Network encoding methods

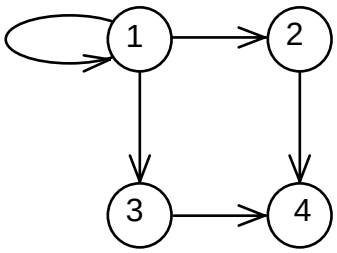
The main challenge in evolving neural networks is in how to represent the structure of a network and all possible combinations of connections and still have a manageable string length. For example, many early approaches mapped a connectivity matrix for a network with n nodes as follows:

Connectivity Matrix

The chromosome is mapped in row major order to a matrix Miller (1989). For example, given that the number of nodes is 4, we can map the binary chromosome, 1110000100010000, to a connectivity matrix as:

from/to	1	2	3	4
1	X	X	X	
2				X
3				X
4				

which then defines the (recursive) network:



Such a chromosomal representation makes it possible to specify recursive feedback connections. If crossover is used indiscriminately, however, there will be a tendency to randomize the beginnings and ends of rows. More importantly, this algorithm fails to provide an efficient chromosome encoding scheme, since N network nodes will in general require N^2 string elements.

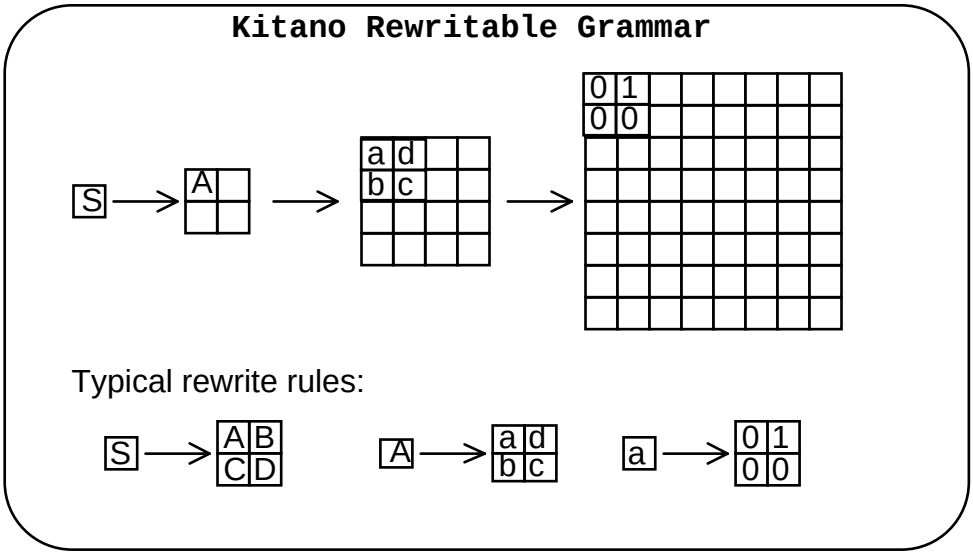
An additional problem in applying this approach to even smaller networks, is the lack of consistency in how close node specifications on the chromosome are with how close they are (connection-wise) in the network. In general, close positions on the chromosome should be closely related in the phenotype (a characteristic given the name *linkage* Whitley (1993).) In this way, swapping of genetic material will be able to swap and rearrange sub-networks within an existing design with the result always being a functional network This issue is referred to as the *structural/functional* problem Whitley (1990), Gruau (1992).

Targeted layers

Harp (1989) proposed a chromosome encoding technique that provides perceptron nodes with a target *vector* that specifies a distance and orientation in the network space to a connective *area* having a topology specified by an associated chromosome element. The connections are then made by matching closest nodes to the target area. The network is evaluated using backprop learning . This approach was used to successfully evolve XOR networks of various configurations. No attempt was made to utilize non-linear elements or recursive signal paths. More importantly the approach cannot guarantee formation of a completely connected network (e.g., a completed path between network input and output) without some added mechanism and also *requires* allocation of chromosome elements to specify node targets. The former results in wasted population strings (since they must be marked as invalid, and only with great difficulty could one differentiate various levels of validity.) The latter allocation requirement limits network expansiveness (complexity for a given string size) according to Gruau (1993). The structure also does not recombine intact sub-networks as a result of crossover, but will in fact break up subnets arbitrarily. These problems can be alleviated using grammar rewrite approaches, which will now be discussed.

Use of a Rewritable "Expansive" Grammar

Kitano (1990) proposed a grammar that decodes specifications in the chromosome according to a fixed set of *rewrite rules*, using 2X2 matrices, that are encoded in a fixed portion of each chromosome. This "grammar encoding method" vastly increases the expressive power of the encoding and is a clear improvement in efficiency



over the previous methods, which Kitano refers to as *direct encoding*, since each connection in the network has a one-to-one correspondence to a specification in the chromosome string. Grammar encoding still requires, however, that an mXm matrix be developed for a network of n neurons, where m is the smallest power of two bigger than n. A grammar produced by the GA may have many rules that rewrite the same character, or that are not even used. A small fraction of the coded rules are actually used, which further reduces efficiency. The method does not have an inherent ability to repeat sub networks although Kitano mentions the possibility of defining rewrite rules to do this.

L-Systems and Cell Division

This approach analyzes automata, known as *L-Systems*, that capture the reproductive behavior of cells in the early state of an organism's development (*morphogenesis*) Lindenmayer (1990). A context *sensitive* grammar is actually required, in general, in order to represent rules that are sensitive to generation number or the characteristics of the cell in question's neighbors, proximity to a boundary or surface, etc. The methods usually used, and those described below, constrain the grammar to being context free for simplicity. In addition to being context-free, however, L-systems permit simultaneous, as opposed to ordered, rewrites. An example of a *deterministic*, context sensitive, L-system grammar is:

productions: $a \rightarrow ab$, $b \rightarrow a$
start symbol, or *axiom*: b
resulting derivation:

b
a
ab
aba
abaab
abaababa
abaababaabaab
etc.

"Parallel" rewrites occur between the 3rd and 4th steps, where "ab" is replaced by "aba". Only slightly more complex rewrite rules are needed to generate 2D and 3D structures that are similar in appearance to plants and trees.

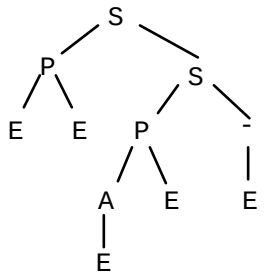
Cellular Encoding

Gruau (1993) has extensively demonstrated the use of "L-System" grammars that creates expansive networks with 1-bit connection weights. Instead of rewriting a matrix as done by Kitano, this grammar rewrites cells, so that architectures of considerable complexity can be described by a relatively short length chromosome. Cell attributes are inherited from the parent cell in a manner intended to mimic mutt-cellular development of complex organisms. The basic operators are listed in the table below, specifying such events as "cell divide", which simultaneously produces two daughter cells connected either in parallel or serially. In addition a branching or *repeat* mechanism is used to encode and promote the use of recursive grammars that encourage the formation of repeated sub networks. Gruaua named this method *Cellular Encoding*.

For example, consider the chromosome string: S P E E S P A E -E, whose symbols are defined by:

Symbol	Action	"Arity"
S	Serial Divide	2
P	Parallel Divide	2
A	Threshold set to "1"	1
E	End	0
-	Set Input Weight to -1	1
R	Repeat from Root n Times	1

resulting in: S(P(EE)S(P(A(E)E)-E)



As the resulting grammar tree is read depth first, cells divide or get attributes modified according to string codes and a neural net is formed of very high complexity (with respect to the number of string positions.) If a "repeat" symbol ("R") is encountered, the network recursively generates even more complex structures by repeatedly expanding the sub-tree defined from the root down to the current node a number of times, as specified by an adjacent string position. Gruau used this technique, with a modified Genitor algorithm (specifically retaining the "delete the worst" feature mentioned above), to successfully develop a number of Boolean networks including a 21-bit parity function Gruau (1993)(pg. 73.)

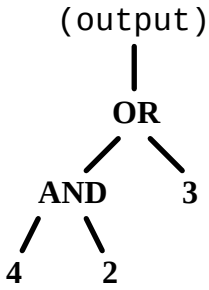
A drawback with this approach is the difficulty of *backcoding*, which refers to any method by which a chromosome is instantiated for a desired network. For the design of a signal processing network, in particular, a initial idea of what the desired network should look like or consist of may be known. This knowledge could conceivably be taken advantage of by *seeding* the initial population with a small percentage of backcoded chromosomes (further discussed in *methods* .)

***Genetic Programming* (*"GP"*)**

Koza (1993) has extensively developed and tested a grammar encoding method that generates a rooted tree with ordered branches. Each node in the tree is an operator out of a *function set* that usually has a pre-defined arity (number of children) of one to three. The arity-2 "functions" consist typically of AND, OR, ADD, MULTIPLY, etc. An example of an arity-three function/node controls execution using an IF THEN ELSE comparison of one input to select one of the remaining two inputs. Single arity functions for many GP problems are NOT, MINUS, but can be also SIN, COS, etc. The *terminal* set specifies arity-0 nodes that are tree leafs that function typically act as input variable or constant numeric values. Koza specifies the use of a minimal function (and terminal) set, which is intended to minimize search space complexity. Numerous successful applications of GP are described in Koza (1993), as well as elsewhere, including inverse, integral, and differential problem solving (the latter use a function set {+,-,*,/,%,SIN,COS,EXP,LOG}, discovery of trigonometric identities, various parity functions, etc.

Koza also has developed methodology for evolving not only networks (which he refers to as *programs* because of their ability to implement conditional execution, loops, etc.) but also for evolving what he refers to as *functions.* These functions can be called by the main "program" repeatedly as a typical program does. He also has devised a method for evolving parameter values. It is easily *possible* for a network to be described with no more chromosome space than is required to represent each of the node types, for example:

The string: OR(AND(4)(2)3) would be decoded into the following network:



In addition to being structurally efficient, this method is also efficient from the standpoint of coding with C++. In Koza's implementation, LISP "s" expressions are instantiated and in fact are what the chromosome strings are made of. As a result, there are no chromosome *strings* that get decoded, but rather the members of the population's expressions get swapped and exchanged during crossover. Parameter values evolve by simply accumulating parameters nodes from one generation to the next. Methods for implementing genetic programming in C++ have also been developed, as described in Keith (1993), and in fact the instantiation of nodes using inherited class characteristics has proven to be very efficient execution-wise as well as easy to manage from a software maintenance standpoint.

Diploid/Dominance

Genetic algorithm research has largely involved the use of single chromosome *haploid* individuals in a population of individuals which are subjected to a variety of selection and crossover operations. In biology, however, higher organisms (including even single cell eukaryotes) utilize a *diploid* chromosome structure. Diploidy is so common that several investigators including Holland (1975) and Goldberg (1989a) have investigated its utility with genetic algorithms.

Diploid/dominance expression in biology suggests that two homologous alleles are paired and, through some mechanism, the dominant gene is phenotypically expressed. If, and only if, both genes are *recessive* then the recessive phenotype is expressed.

Hollstien Triallelic Approach to Diploid/Dominance

In Hollstien (1971), a GA was used to implement the above logic by increasing the cardinality of an ordinarily binary chromosome from 2 to 3. Each string position was specifically treated as a gene as far as pairing of alleles is concerned. Hollstien then added a "triallelic" third value ("2") to the usually two-valued ("0" or "1"), at each binary position. If a "2" occurs, it is phenotypically interpreted as a "1". Likewise for a "1", but only if it is paired with a "1". This approach made it possible to study dominance in multiple allele chromosomes (every bit position in the string is a separate gene with

Triallelic, Single locus, diploid dominance map

		Gene 2		
		0	1	2
Gene 1	0	0	0	1
	1	0	1	1
	2	1	1	1

Legend: 2=>dominant "1", 1=> recessive "1", 0=> "0"

Example:
Homologue #1: 01121; Homologue #2: 10201.
=> Expressed genotype (for the purpose of fitness evaluation) would be: 00111.

Figure 2.1. Triallelic dominance map that the effective allele value at a given diploid string position.

corresponding allele value), and is represented in *dominance map* shown in Figure 2.1. Results with this approach were mixed, perhaps due to the random dominance shifting methodology utilized, and diploid/dominance since Hollstein's work appears to have received little attention in the literature with the exception about to be described.

Non-Stationary Fitness

Smith (1992) used the Hollstein triallelic mechanism to study the ability of diploid/dominance to improve GA performance with changing global optima. The fitness function they utilized is known as a "non-stationary 0-1 knapsack problem," which has a search space of 2^L, for an L-bit chromosome. The 0-1 knapsack problem is defined as follows:

$$\text{maximize} \quad \sum_{i=1}^{L} v_i x_i = f(\mathbf{x}) \qquad (2.86a)$$

$$\text{given} \quad \sum_{i=1}^{L} w_i x_i \le W \qquad (2.86b)$$

where: $x_i \in [0,1]$ are the L chromosome bits

v_i and w_i are real, pre-defined weights

This function seeks to maximize the inner product of **v** with the binary chromosome, **x**, imposing the constraint (2.86b) that the inner product of **w** with the chromosome must be less than a fixed constant W. The above authors then modified the knapsack problem, in order to induce a *shift* in the fitness environment, by periodically switching the value of W between two different values every 15 generations. Results were definitely improved over Hollstein's stationary model, however, some instability is still evident in the results Smith (1992)(p280), Goldberg (1989a)(p156). The first experiments described in the *results* section of this proposal use an identical 0-1 knapsack problem, thereby providing direct comparison with a previous experiment having moderate problem difficulty.

Some biologists assert that diploid/dominance can provide protection against low-fitness allele loss. "The preservation in diploid populations of alleles that may be selectively disadvantageous under some conditions provides a source of evolutionary variability not available to haploids Novikoff (1970)." If a similar environment repeatedly arises, diploidy could presumably enable the species to rapidly re-express its former fitness for that environment. A shift in environment, or non-stationarity, was introduced into Smith's GA experiment by periodically switching the value of "W" in order to see if diploid/dominance had such an effect with GA's. His conclusions suggest the possibility that such might be the case. More importantly, Smith provides direct insight into: 1) how the use of diploidy can maintain a relatively high proportion of recessive alleles in a population without resorting to high levels of mutation, and 2) lower bounds on recessive allele retention (from one generation to the next). His analysis specifically examines item (2) for 2:1 and infinity:1 dominant to recessive fitness ratio. These results will now be summarized.

An explanation of how diploid dominance could permit a lower mutation rate while maintaining a constant (small) proportion of recessive alleles was published in Smith (1992). These results are an extension of conclusions presented in Holland (1975)(pg. 114). Convergence, for a given single allele, is analyzed while the proportion of that allele reaches a small, steady-state value P_{ss}, for a given mutation rate p_m and ε, the expected *change* in allele proportion due to fitness selection. Smith first derives p_m as a function of P_{ss} and ε, for the usual haploid GA. The result is then compared to the same relationship for a diploid (with dominance) GA, where a recessive allele is expressed only if paired with another recessive. The following is an expanded summary of Smith's result with some expansion to the diploid analysis. These results assume that recessive and dominance, for a given allele, are defined from one population to the next using a *dominance shift* operator that functions in a similar manner to pointwise mutation (except that instead of modifying allele values, the dominant/recessive label for each allele is what gets randomly modified.) The following analysis assumes that a given allele is either completely dominant or completely recessive:

For a haploid population:

For a specific binary allele value at time t, denote the expected proportion in the population for that allele by P^t. The expected increase due to selection, alone, is then: $-\varepsilon P^t$:

where: $\varepsilon =$ the expected change in P due to fitness-proportionate selection

($\varepsilon > 1 \Rightarrow$ allele value is favored for selection)

The expected increase due to mutating the opposite value to the one in question is:

$p_m(1-P^t)$; *where*: p_m = *mutation rate*

The expected increase due to mutating the given value is: $p_m P^t$. Combining selection and mutation gives:

$$P^{t+1} = (1-\varepsilon) P^t + p_m (1 - P^t) - p_m P^t \qquad (2.91)$$

Now let $P^{t+1} = P^t \equiv P_{ss} = steady\text{-}state\ proportion\ of\ recessive\ alleles.$

Then: $P_{ss} = (1-\varepsilon) P_{ss} + p_m (1 - P_{ss}) - p_m P_{ss}$

$$\Rightarrow -2 p_m P_{ss} + p_m = \varepsilon P_{ss}$$

$$\Rightarrow p_m = \frac{\varepsilon P_{ss}}{1 - 2 P_{ss}} \qquad (2.92)$$

For a diploid population:

let: $P^t = P_r \equiv$ proportion of *recessive* allele at time t, and

$P_d \equiv$ proportion of *dominant* allele at time t.

to find: $P_r^{t+1} \equiv$ prop. of the recessive allele at time $t+1$

since selection only affects recessive alleles when they are paired (homozygous recessive),

the increase in recessive alleles due to selection is: $-2\varepsilon P_t^2$ (2.93)

(the factor of 2 is due to losing 2 recessive alleles every time a recessive pair is deleted)

the diploid mutation term is given by: $2 p_m (P_d - P_r) = 2 p_m ((1 - P_r) - P_r) = 2 p_m (1 - 2P)$ (2.94)

combining (2.93) and (2.94) gives the diploid difference equation: $P^{t+1} = (1 - 2\varepsilon P^t) P^t + 2 p_m (1 - 2 P^t)$ (2.95)

now let $P^{t+1} = P^t = P_{ss} \equiv$ steady-state recessive allele proportion $P_{ss} = (1 - 2\varepsilon P_{ss}) P_{ss} + 2 p_m (1 - 2 P_{ss})$

$$\Rightarrow p_m = \frac{\varepsilon P_{ss}^2}{1 - 2 P_{ss}} \qquad (2.96)$$

Equation (2.96) shows that the p_m (mutation rate) necessary to maintain the level of the allele at proportion P_{ss} is proportional to P_{ss}^2, a potentially much lower rate than for the haploid situation. In addition, Smith shows that the retention of recessive alleles from one generation to the next has a greater upper bound for the diploid case (than the haploid).

Sub-fitness Dominance Map

GA Implementation

The approach to diploid/dominance used for this dissertation is inspired by zoology and genetics, and this basis will now be described. The actual algorithm and its performance will be described in the methods and results sections.

In contrast to the Hollstein approach, this dissertation considers the diploid chromosome as two chromosomes with one gene each, for simplicity, and one corresponding allele value each. The allele values are defined to be the fitness values of the two strings. These two *sub* fitness values are then utilized as if they actually represent an intermediate phenotype level between genotype and phenotype. This process, where two homologous alleles seemingly compete prior to manifestation in the observed phenotype, will be referred to as *sub-phenotype* interaction. This approach is problem independent and can be extended to polyploid (greater than two homologues per individual), as will be show in the methods section.

The two sub-phenotypes are obtained by evaluating both chromosomes using the fitness function that is to be maximized. Dominance is implemented by mapping the two sub-phenotype fitness values to the scalar fitness value which will then be used for selection. The mapping function is referred to as the *dominance map* or *dominance function*, and the fitnesses of the individual haploid chromosomes (homologues), is referred to as *sub-fitness* Greene (1994). "Sub-fitness" dominance, as used here, is similar to *partial* or *incomplete* dominance in genetics, as summarized in [Stansfield, 1983 #98], pg. 219.

Utilization of "Sub-fitness" Diploid/Dominance in Biology

In biology, it has been postulated Muller (1950) that homologous genes can interact in a non linear fashion to produce their associated phenotype. If the dominant form of an allele produces enough of the enzyme needed to catalyze the reaction for, e.g., eye coloration (Muller, pg. 179) then a homozygous combination of dominants (i.e., both alleles are dominant) will produce only a barely noticeable increase over the heterozygous combination. This idea is summarized in Crow (1983) as follows. "...that genes produce enzymes provides an explanation of dominance." [Consider that C is the dominant allele and c is the recessive form of that allele] "Since only very small amounts of enzymes are needed to

catalyze chemical reactions, one gene usually produces more than enough enzyme to convert all the substrate into product.... Since one C allele is sufficient, the genotypes CC and Cc both have the same amount of pigment even though CC produces twice as much enzyme as Cc." One consequence of the dominance mechanism occurring at a level between the string level (genotype) and fitness level (phenotype) is that homologous genes need not be explicitly linked (e.g. through gene position), they simply compete through the similarity of their intermediate (enzyme) phenotype.

III. Methods

This section describes in detail the approaches used in the *Results* section, and some comments on extensions for future work.

A. Genitor

The Genitor public domain software (described in the previous section) is used because of its apparent efficiency, fairly wide usage, and ease of implementation and modification. Only one evaluation is made per (haploid) generation. Genitor is well written and was easily ported from Unix to DOS. It is characterized by 1) rank based selection of individuals for mating, 2) deleting the worst individual at each generational cycle, and 3) carrying all but the worst individual forward to create the new generation. Further justification of using Genitor was provided in the previous section.

B. Grammar Tree Gene to Network Mapping

- *Overview*

All string values are constrained to be in the range [-1.0, 1.0] both by the initialization procedure (which places random numbers into all string positions) and is maintained by the crossover and mutation operators. Checks are done during crossover and mutation to see if any string values are outside this range, and an immediate error is flagged if so. To re-map parameters to various range(s) or dimension(s) for a node operator, the node's constructor method (i.e., associated instantiation and initiliazation function) provides one or more function arguments. For example, the *delay* node scales the associated string value linearly by 100 giving a range of 0 to 99 (after mapping the allele from [-1,1] to [0,1]). Such scaling factors are provided as parameters to the C++ constructors for each particular node so that specification and modification are easy. In some cases where emphasis may be desired on a certain sub-range of the overall parameter range, a node may be "defined" more than once in the *decodeNode* function with differing scaling factors specified for multiple instances of the parameter in question. This latter approach is intended to increase search efficiency for function parameters having an asymmetric distribution.

- *Similarity to Koza*

The proposed approach to genetic network synthesis uses a chromosome to network mapping that follows the approach taken by Koza (1993). This method has been followed by a number of others, as exemplified by the section on "Genetic Programming" in ICGA-93 (1993). In order to encourage the formation of viable networks, the crossover operator used here (see below for details) works in conjunction with a global *GeneMarker* register that records the string positions of valid nodes every time the nextNode function (below) is called. Since all nodes produce a single, floating point output and expect floating point inputs, all nodes are compatible in this respect.

- *Differences with Koza's implementation of Genetic Programming: The nextNode function*

Instead of building and exchanging Lisp s-expressions, my approach uses a traditional GA with populations of fixed length floating point arrays, or strings. A string decoder evaluates string elements from left to right. The reading head (pointer to a given string element) matches the string value to one of the node operator types in a pre-defined *function set*, each member of which has a pre-defined arity (number of children), as well as other characteristics (see below for details). String decoding is handled by the *nextnode* procedure, which:

- tests for end of string;
- tests the current tree branch to see if it is longer than *MaxDepth*= 4;
- if either of the above two conditions are true, the branch is *terminated* by attaching a *terminal* node from the *terminal set* and unwinding the current recursion, after which processing proceeds;
- records the current string position in a the global G*eneMarker* register as a point where crossover is allowed;
- maps the current string value into one of the available node-operators in the "function set", defined below.

Upon decoding, each node is instantiated as a C++ object. The node instantiation process does two things: 1) Read successive function parameters as sequential string values and 2) recursively calls the "nextnode" procedure to begin instantiation of each child of the node (up to its pre-defined arity), which in general consists of a sub-tree of nodes.

Having decoded the chromosome into a tree, or network, the nodes are evaluated depth-first with a single call to the *eval* function of the root node. Similarly, the network is deleted (to reclaim the memory) with a single function call to delete the root node, which then automatically deletes the entire tree through the nodes' destructors.

In order to develop specific numeric values, Koza uses what he calls an "ephemeral random constant" generator that accumulates randomly generated numeric values which are subsequently retained and accumulated from generation to generation within s-expressions. Such an approach is Lamarkian in that it involves if not modification and subsequent inheritance of phenotypes and therefore is somewhat at odds with biology. The approach used here uses the actual real string values that evolve using standard (non-Lamarkian) mutation and crossover operators as described below.

- *C++ Class Hierarchy*

  C++ provides for inheritance of class data and methods from parent objects or *classes*. My GA implementation uses a single level of inheritance. A *base class* provides common characteristics for all node operators.

  1) Base Class
  - Data: Provides storage for arity count, pointers to children, node name, *targeter* flag that indicates if the node is to be interpreted as one which can target another node, and node name
  - Methods: Node depth control, default initialization of above variables.
  - C++ Implementation (sample code):

```
#define MaxArgs 5
#define RETURN_TYPE float
int treeDepth= 0; // global variable to keep track of how deep each branch is

// ***************************************************************************************
class nodeBase  //  <---------- this is the "base" class that is common to all node/operators
{
  public: virtual (RETURN_TYPE) Eval();  // "virtual" methods can be overloaded by ancestor class (below)
          nodeBase() {arity= targeter= 0; depth= treeDepth++;  outputVal= 0;}
  private:  int arity, targeter, depth, outputVal;
            RETURN_TYPE *Arg[MaxArgs];
};
```

  2) Child Class
  - Data: Specify structures used for ufnction implementation
  - Methods: Code to implement node/operator functionality
  - C++ Implementation (sample code):

```
// ***************************************************************************************
// typical "ancestor" class that inherits properties of the above base class
class AddNode : public nodeBase
{ // example of "sub-class" that defines a node/operator to add "N" subnets
  public:
  AddNode(int N)  // constructor -- initialize node
  {
   if (N > MaxArgs) {N= MaxArgs; /* flag node definition error */; }
   arity= N;
   //
   // <--- place holder: for decoding any node parameters directly from the chromosome
   //
  }
  (RETURN_TYPE) Eval()  // eval member function -- contains actual operator code
  { // in this case, simply add up evaluated output of arity number of subnets
   RETURN_TYPE r= 0; for(i=0; i<arity; i++) r+= Arg[i]->Eval();
   return outputVal= r; // store return value in "outputVal" register and also retun as functional value
  }
  ~AddNode() {for(i=0; i<arity; i++) delete Arg[i]; } // destructor, to reclaim memory when all done
  private: int i;  RETURN_TYPE *Arg[2];
}
```

In summary, each string element that does not map to a node parameter decodes to a node/operator C++ *object* that *encapsulates* its own *Eval()* handler. There are two types of node/operators: Functions and terminals, which is the strategy followed by Koza. The approach taken here can in some cases utilize a node operator to be both a function *and* a terminal (e.g., the *targeter* and *mdvi* operators defined below.) I have also made no attempt to restrict the function set to the minimum number needed to solve the specific problem under consideration, although doing this for a particular problem (a routine part of Koza's GP problem definition) would of reduce the search space and therefore simplify the specific problem under consideration. The function and terminal sets are defined as follows:.

- ***Function set (organized by category)***
  - Boolean: arity= 2
    - OR(x,y)= Max(x,y); AND(x,y)= Min(x,y); NOT(x)= -x; NOR(x,y)= NOT(OR(x,y); NAND(x,y)= NOT(AND(x,y)
  - Arithmetic
    - ADDW: weights sum of inputs, each input is a separate chromosome value. (assigned arities= 2 or 3)
    - MUL: multiplies "safely" the input by a chromosome value scaled by an assignable range -- if result is out of machine floating point range, the value is clipped to prevent overflow ("safe multiply"). (arity= 1)
    - ADDC adds a string defined value multiplied by pre-defined constant to input. (arity= 1)
  - Signal Processing
    - One-shot (OS): Produces one evaluation-cycle long pulse if threshold reached. (arity= 1)
    - One-pole filter (IR1): Chromosome value determines filter time-constant. (arity= 1)
    - Adaptive, two-coefficient, one-pole filter (IRB): Selects one of two feedback paths depending on thesholded 2nd input. Output is filtered output of 1st input. Each filter has its time-constant independently assigned from the chromosome Very useful with wave form event determination. (arity= 2)

- ***Terminal Set (i.e., leaf nodes)***
  - MDVI: Takes vector input (e.g., magnitude spectra) and returns mode between specified limits. Offset and width of the (band) limits are supplied by the chromosome: arity= 0 (terminal node)
  - TAR See description below -- clips values similar to "MUL" function, arity= 0
  - IN: Input from one of up to ten or so chromosome specified input scalars: arity= 0
  - NULL: Returns a value of 0,, except for "OR" parent nodes, for which a value of "1" is returned: arity= 0

The "TAR" node operator causes one additional string position, per node, to be read to obtain a relative network depth or *offset*. After the network is instantiated, a second instantiation pass is made during which all such *targeter* nodes in the network are identified and assigned an absolute node (x,y) address from which they "target" their input. The base class's "outputVal" register (see above code) is utilized to obtain the last outputted value from the targeted node. This capability augments the network connections defined by relative string position and provides the capability for any node to connect directly to any other node.

C; Crossover and Mutation Operators

The algorithms for the crossover and mutation operators are described in the *literature survey* section, and were originally devised by Michalewicz (1992). Additional details are as follows:

- Crossover

Arithmetic crossover is implemented by sweeping the blending *"a"* value between .5 and 1.5, in a triangular wave (with period of approximately. 1800 generations) so that the solution space, S , is not strictly convex, as mentioned in section II. In other words, all possible allele values can be reached through crossover, alone, even if the initial population doesn't contain the exact values needed. The definition of arithmetic crossover is.

if s_v^t *and* s_w^t *are crossed* (*at generation* t):

the resulting offspring are:

$$s_v^{t+1} = \left\langle v_1, \ldots, v_k, w_{k+1} \cdot a + v_{k+1} \cdot (1-a), \ldots, w_m \cdot a + v_m \cdot (1-a) \right\rangle \in S \qquad 3.2$$

$$s_w^{t+1} = \left\langle w_1, \ldots, w_k, v_{k+1} \cdot a + w_{k+1} \cdot (1-a), \ldots, v_m \cdot a + w_m \cdot (1-a) \right\rangle \in S$$

- Mutation

As with most GA's, each element in a string selected for mating is mutated according to p_m, which is the *mutation rate*. If an element is thereby selected, the mutation operator either increases or decreases the string element value with equal probability. An exponential weighting function is then applied that encourages a result near the original (parent's) value to an

extent, Δ(t,y) that is controlled by the current generation index. The idea is to allow greater variation or *spread* in possible new allele values compared to the results of mutation in earlier stages. This "spread" then decreases as the generation count increases. I have modified this slightly so that:

Δ(t,y)= y*(1-r**(1-T/GC**bb) (pg. 80 of Michalewicz); where T= t mod GC

As t periodically approaches ["GC modulo N] generations (N typically 1000), the mutation spread is reduced.

- Gene markers

  When sites are selected for crossover, the closest existing values in the GeneMarker register are used to select the actual sites for the first parent. The second parent has no such restriction. This is similar to what Koza and Gruau do and produces a notable improvement in GA efficiency (both the rate of problem solution and success rate) when compared to totally random selection of crossover points.

D. Use of Diploid/Dominance

Diploid/dominance as implemented here is defined by eq. 3.80:

$f = \mathrm{Max}(f_1, f_2)$ (3.80)

where: f_i is the fitness (or "sub fitness") of the i^{th} homologue, and f is the resulting fitness for the individual.

The *sub-fitness* dominance mapping scheme proposed here establishes partial dominance on the basis of the *relative* allele fitness between each homologue, and is also described in Greene (1994). One result of this approach is that no mutation-like dominance shift operator is involved, as with the previously described Hollstein triallelic approach. Gamete (or haploid homologue) selection is done prior to chromosome crossover, which is the opposite of biology. This is done to increase algorithm efficiency, since no individuals get crossed without their gametes getting used. By contrast, the possible number of *new* crossover pairs per (biological) mating event are M^2, where M= # homologues, and two are subsequently selected to form the diploid offspring. An attractive property of eq. 3.80 is that it retains the original units or dimensions of the f_i fitness functions.

Extension to the Smith/Holland calculations Smith (1992), Holland (1975).
By assigning *absolute* states of recessive or dominant to each homologue, the following definitions suggest a reduction in the required p_m rate similar to that predicted by Smith as follows:

Define a recessive allele as one with subfitness $f_i < f_a = \bar{f}$, and a dominant allele as one with subfitness $f_i \geq f_a = \bar{f}$.

where:

f_i = sub - fitness of the i^{th} homologue: $i \in [1,2]$

Then:

$$f_{r_e} = P_r^t \cdot f_r + P_d^t \cdot f_d = P_r^t \left(\frac{\int_0^{f_\alpha} f \cdot p^t(f)\,df}{\int_0^{f_\alpha} f \cdot df} \right) + P_d^t \left(\frac{\int_{f_\alpha}^{\infty} f \cdot p^t(f)\,dx}{\int_{f_\alpha}^{\infty} f \cdot df} \right)$$

where:

f_{r_e} = fitness of the average recessive allele considering the affects of its homologue (which may be recessive or dominant)

f_r = average subfitness of a recessive allele

f_d = average subfitness of a dominant allele

$p^t(f)$ = single (haploid) allele fitness density function at generation t

P_r^t = proportion of alleles defined as recessive

P_d^t = proportion of alleles defined as dominant.

This approach provides the definitions needed for eqs. 2.93-2.96. By examining the derivation eq. 2.93, the necessary p_m for a polyploid population can be seen to be on the order of P_{ss}^{M}, *where*: M= number of chromosome homologues. The exact algorithm (including the gamete and recombination operations) used here for implementing polyploid dominance mapping is described in Greene (1994).

The definitions needed to analyze recessive allele proportion changes with diploid versus haploid can also be defined as follows:

Let $f_\alpha = \overline{f}$,

where : $\overline{f}$ = population average subfitness (over *all* homologues)

and since $P_d^t = 1 - P_r^t$,

where :

f_r = subfitness of the average recessive allele

f_d = subfitness of the average dominant allele

we have, $f_{r_e} = P_r^t f_r + (1 - P_r^t) f_d$. We can also write the average population fitness as : $\overline{f} = f_r \left(P_r^t\right)^2 + f_d \left[1 - \left(P_r^t\right)^2\right]$

Further Extension to the Smith/Holland calculations: Rate of change in recessive allele proportions

The expected change in recessive allele proportion for a diploid population Smith (1992), using the above definitions can now be derived as follows:

$$P_r^{t+1} = P_r^t K\left[\frac{f_{re}}{\bar{f}}\right] = P_r^t K\left[\frac{f_r P_r^t + f_d\left(1 - P_r^t\right)}{f_r (P_r^t)^2 + f_d\left(1 - (P_r^t)^2\right)}\right] = P_r^t K\left[\frac{P_r^t + c\left(1 - P_r^t\right)}{(P_r^t)^2 + c\left(1 - (P_r^t)^2\right)}\right]; c = \frac{f_d}{f_r} \quad (3.81)$$

and *K* is a constant that accounts for loss due to mutation. For a haploid population, the result is:

$$P_r^{t+1} = P_r^t K\left[\frac{f_{re}}{\bar{f}}\right] = P_r^t K\left[\frac{f_r}{f_r P_r^t + f_d\left(1 - P_r^t\right)}\right] = P_r^t K\left[\frac{P_r^t}{P_r^t + c\left(1 - P_r^t\right)}\right]; where c = \frac{f_d}{f_r} \quad (3.82)$$

By evaluating eqs. 3.81 and 3.82 for various values of P_r, Smith shows that the rate of change of recessive alleles is substantially slower for diploid than haploid populations with c=2. As c goes to infinity for a diploid population, eq. 3.81 can be written as $P^{t+1} = \frac{KP^t}{1 + P^t}$, which by induction can be written as:

$$P_t = \frac{KP_0}{1 + tP_0} \quad (3.83)$$

where $P_0 = P_t$ at t=0 (the initial proportion of recessives). In comparison to eq. 3.83, letting c go to infinity in eq. 3.82 results in $P_t = 0$. This result is true for all values $P_0 > 0$, further supporting the view that diploid dominance retains even *extremely* recessive alleles from one generation to the next.

Fitness switching with the R-Wave problem

Diploid/dominance is implemented together with environment switching, as described in the literature survey, for the Doppler network synthesis test problem described below. This has been done by defining one environment as the complete fitness function, which consists of a phase portion (f_t) and an energy portion (f_e.) This function is:

$f_1(x) = f_t + f_e$; (3.20)

where: f_t= avg. over N heartbeats of abs(true time of R-WAVE - network output's N highest maxima)

f_a= abs(difference in energy between true R-WAVE and network output).

The second or alternate environment is a simplified version of $f_1(x)$ that consists only of f_2= fe + .02. f_2 never goes completely to zero and so it can never indicate a solution, which occurs with Genitor when an individual's objective value in reaches zero. Reducing f_2 corresponds to finding a function with the right energy, which in the tests done corresponds to a

unity width and amplitude, two-pulse sequence (for N=2). f_2 is obviously much easier to "solve". Retaining the f_2 solution through the mechanism of diploid/dominance may be investigated as part of this dissertation. Such an approach can reduce the likelihood of prematurely converging to a local sub-optimum that does not at least include a solution to f_e. This expectation is supported by findings in the *results* section of this proposal for the following problem, which is also relevant to the desired goal of synthesizing networks for low-cost Doppler signal processing. The advantage of diploid/dominance, with or without environment switching will also be investigated if results continue to support its use.

Doppler test problem

The test problem consists of a series of simulated triangular shaped 64-bin spectra whose peaks (modes) are shifted to produce a wave form that has two peaks corresponding to a "normal" common carotid blood velocity wave form. These spectra are then replicated once to produce a series of spectra with peak frequencies (modes) as shown below.

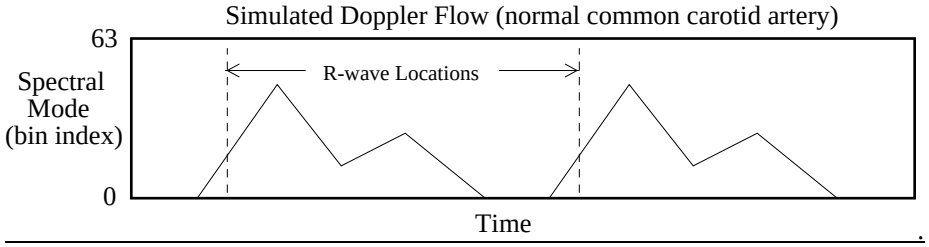


Figure 3.1. Mode frequencies of simulated Doppler signal for R-wave detector problem.

R-wave events are defined to occur at identical x-axis (time) values for the two identical wave forms, as shown: Detection of an R-wave equivalent fiducial point from the Doppler signal has direct application, e.g. in situations where an ECG is difficult to obtain (as mentioned in the introduction). In practice signal noise, probe to vessel movement, and the effects of advanced vessel occlusion can all contribute to make the problem more difficult. To solve these types of problems additional pre-processing can be done, as mentioned in section one. For example, signal energy, energy derivative, and upper 9 dB frequency, in addition to just the Doppler spectra, may be provided as available input data to the previously described "IN" nodes during fitness evaluation.

E. Backcoding

As mentioned in the literature survey, *backcoding* refers to a method to pre-encode chromosomes from a pre-designated network. As mentioned in the introduction, a goal of this proposal is to investigate the possibility of at least partially pre-specifying the network design, to see if the genetic algorithm can then optimize the node parameters or add or delete nodes and connections. A portion of the population is *seeded* in this way with the backcoded chromosome..

Part of this task has been accomplished in that I can specify a list of nodes using node mnemonics and parameter settings and specify a chromosome. For the R-wave test results of the next section, 20% of the population was seeded with a "OS" node connected to a child "MDVI" node, since it is clear that the network needs to produce a unit amplitude pulse and that we would expect the network to utilize a single point estimate of the spectral energy. There is more discussion of backcoding in section 6.

IV. Results

A. 0-1 Knapsack Problem

Test results are provided in Greene (1994) which show the proposed diploid/dominance *sub-fitness* mapping can result in reliable and stable convergence in the following non-stationary, 0-1 knapsack problem. Those results were for a two-environment fitness function defined by two alternating weight constraints. That approach is extended here by defining two additional weight constraints for the knapsack problem. These four weights are alternately applied together with a triploid and tetraploid chromosome structure for each individual (i.e., each individual in the population consists of three or four chromosomes.) The population is comprised of 150 individuals consisting of (polyploid) 17-bit binary strings. The sequence of alternating weight constraints which get repeated (in order) are then: W= 104, 60, 85, 47. The optima (solutions to the knapsack problem) for these four constraints, as determined with an exhaustive search through the 2^{17} bit combinations, are: 113, 129, 120, and 137, respectively. In the experiment below, the four chromosome Genitor GA converges to correct

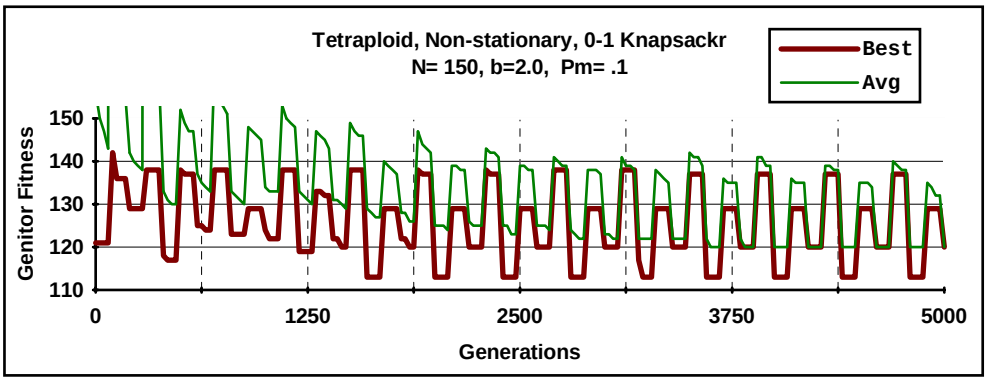


Figure 4.2 Tetraploid (4-homologues) convergence with a 4-state version of the 0-1 knapsack problem.

solutions for the four constraints by t= 3500 generations, after which fitness switching is immediate (within one generation) and stable throughout each fitness environment. As described in the above reference, the population is completely re-evaluated after each fitness change. If a non-overlapping *generational* GA were used instead of Genitor, re-evaluation of the entire population is always carried out every generation, in which case this increased "overhead" disappears. Fitness in these experiments is switched from one optimum to the next every 100 generations.

To see if a four-chromosome structure is necessary to solve a four-state problem, the same experiment was run with diploid and triploid populations. As shown in figure 4.21 and 4.22, even with three times as many generations, the diploid and triploid

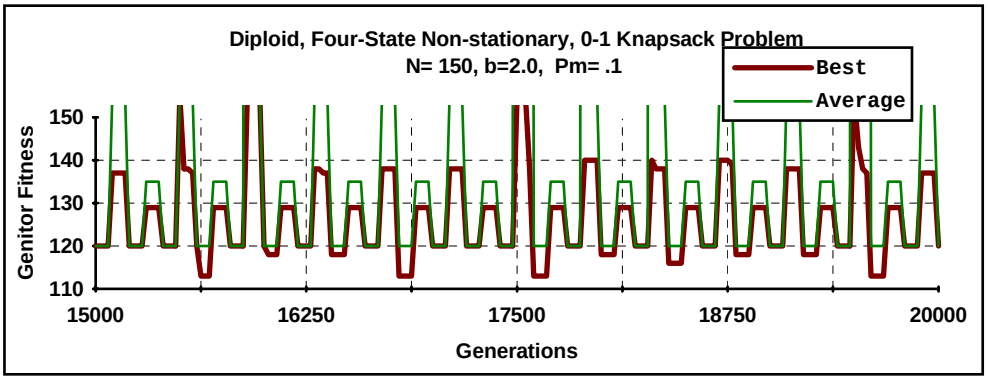


Figure 4.21. Diploid population, still with four-state fitness function.

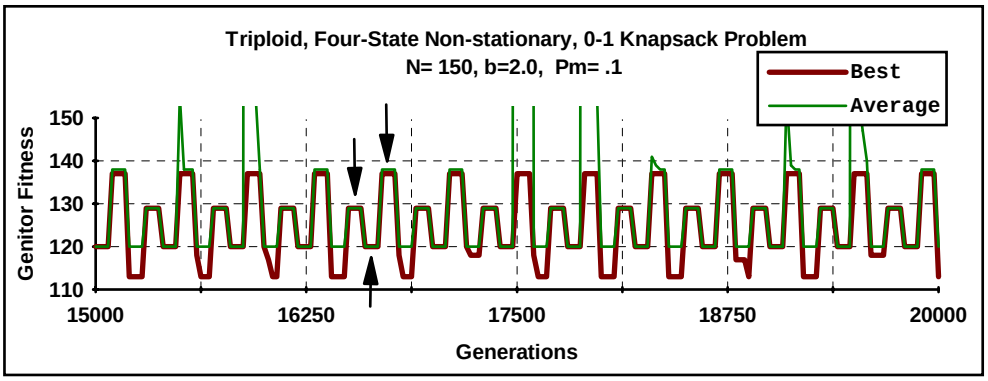


Figure 4.22. Triploid population, still with four-state fitness function. Arrows indicate correct solutions.

populations cannot converge to all four states as can the tetraploid, although the triploid does appear to have done a better job than the diploid in that it gets three of the states correctly, as indicated in fig. 4.22.

These results, and the theory of the previous section, indicate that the proposed "sub-fitness" approach to diploid/dominance improves GA performance in non-stationary fitness environments and that the approach can be extended to a separable fitness function. In addition, the results show that polyploid populations can immediately recall the appropriate solution when a previous fitness criteria is recalled. This suggests the possibility that use can be made of alternating fitness criteria in solving

problems with multiple stages of complexity or sub-functions. In the next experiment, diploid/dominance is used to aid in finding a solution to a simulated Doppler signal processing problem having two such stages of complexity.

B. Two-Heartbeat R-Wave Detector

This series of experiments demonstrates GA development of a network of signal processing operators whose output produces a pulse that coincides with a simulated ECG R-wave trigger, using only the Doppler signal. The Doppler signal is assumed to be available in the frequency domain as a series of 64-bin spectral magnitudes. In typical clinical spectral displays, 128-bin spectra are computed every 10 msec.

As mentioned in the *methods* section, Diploid/dominance may be capable of exploiting a non-stationary fitness environment, or fitness criteria. This idea is extended here, so that one of the (two) fitness definitions is purposefully defined as an incomplete, but relevant, portion of the complete fitness definition. This incomplete function is a term in the (complete) fitness function (eq. 3.20) that monotonically increases with increasing deviation from the correct energy.

The following figures plot Genitor fitness vs. generation for the Doppler R-wave problem as described in the *methods* section. Since individual fitness actually increases as the Genitor "fitness" *decreases* from plus infinity to zero (zero fitness is defined to be a perfect solution) a perfect solution goes off the bottom of the graph as indicated in Fig. 4.11 by the arrow. For each graph, the best and average (Genitor) fitnesses are plotted along with the crossover rate (the "a" term in eqs. 2.42 and 3.2). These tests use a population of N=250 and floating point string length of 30. Fitness is switched every 500 generations producing what appear as oscillations in the graphs. The longer switching times, compared to the knapsack problem, are used to reduce the rate at which the entire population gets re-evaluated. This event takes substantially longer for the more complex R-wave problem.

In fig. 4.1, diploid/dominance with fitness switching is used and the f_e (energy only) fitness function converges to a partially correct solution (energy error is zero) by 1500 generations as shown in Figure 4.1. The network produced at that point has at

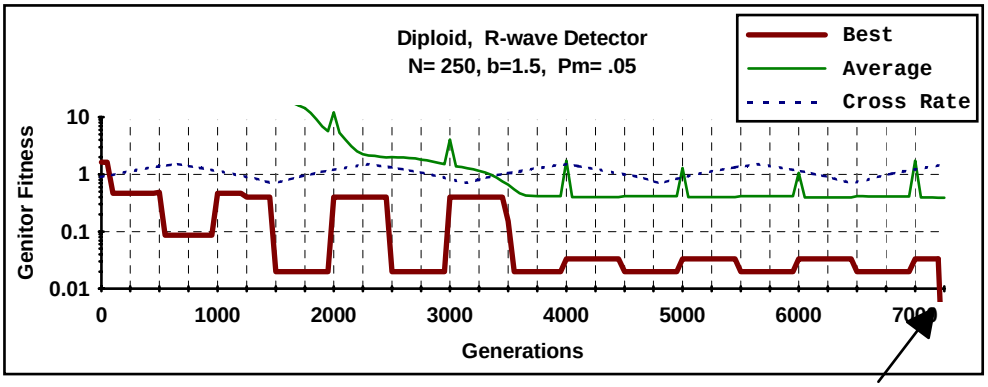


Figure 4.11 Solution Found

its root node a one-shot whose sub-network causes exactly two pulses to be produced, although at the wrong times. Referring to Figure 3.1, the smaller diastolic "hump" of each heartbeat, which is a characteristic feature of common carotid velocity wave forms, has a peak shifted 10 time units (spectra) from the first, larger hump of each heart cycle in the simulation. As the GA run continued, a network emerged at generation 7248 that produces two pulses that are correct both in phase and energy with the (simulated) ECG R-wave events shown previously in Figure 3.1, as shown here:

(Network Output:) <= OS(3.1)<=INV<=NOR <=DLY(10)<=OR <=NULL <=MDVI(o,b) <==Spectral Input <=TAR

Figure 4.115. Synthesized network that solves the Doppler R-wave problem (see appendix for more examples).

(Network Output:) <= OS(3.1)<=OR <=DLY(10) <=MDVI(o,b) <==Spectral Input <=TAR

Figure 4.116. Simplified version of Figure 4.115

The evolved parameter "3.1" for the OS node in Figure 4.115, which is shown with unnecessary nodes removed in Figure 4.116, specifies the threshold at which the one shot triggers. The "DLY" node has an delay of 10, which serves to disable triggering on the second, smaller hump of each heart cycle. The MDVI "o" and "b" parameters are the chromosome supplied offset and bandwidth for the mode function and are equal to 6 and 51, respectively. The solid line from the "TAR" node indicates this node's input target. The spikes visible in the "average" fitness subsequent to each environment change may be due to modifications in the population of *relatively* recessive genes during optimization of the dominant, more highly fit, genes.

The definition of recessive/dominant used here, and as further discussed in the *methods* section, is a "relative" one that can be likened to *incomplete, partial, or co*-dominance in biology [Stansfield, 1983 #98], where each allele can have some effect on fitness. This usage differs from past GA implementation of diploid/dominance (e.g., Smith (1992)), where a gene is *absolutely* recessive or dominant, independent of its homologue. Theory presented in the *literature survey* ("Diploid dominance in biology", above) suggests that such dominance interaction is not uncommon in biology, and results from the natural expression of homologous (diploid) genes and the resulting phenotypic (enzyme) product of each.

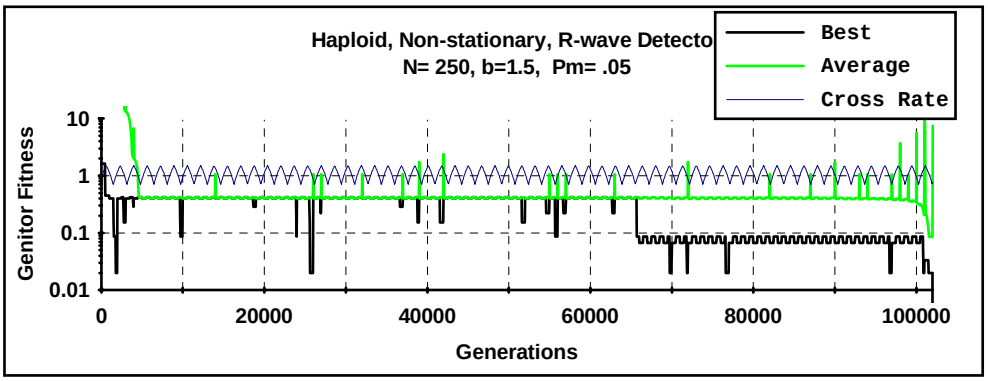


Figure 4.12. Environment switching with a *haploid* population.

The experiment of figure 4.12 solves the above R-wave problem using environment switching only, using a *haploid* population. The significantly faster diploid solution strongly suggests that the Max(f_1,f_2) approach associated with figure 4.11 permits faster convergence to a global optimum.

To show that convergence to a global optimum may be robust, results for a reduced population size and reduced fitness

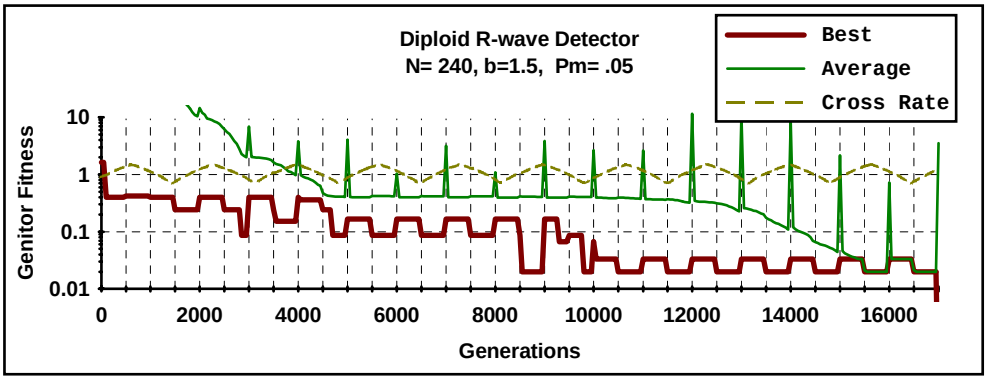


Figure 4.13 Same experiment as Fig. 4.11 with slightly reduced population size.

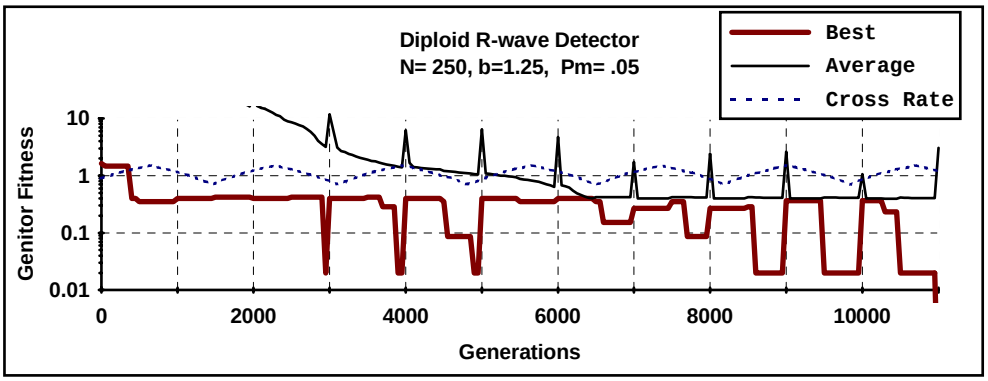


Figure 4.14 Same experiment as Fig. 4.11 with reduced fitness pressure.

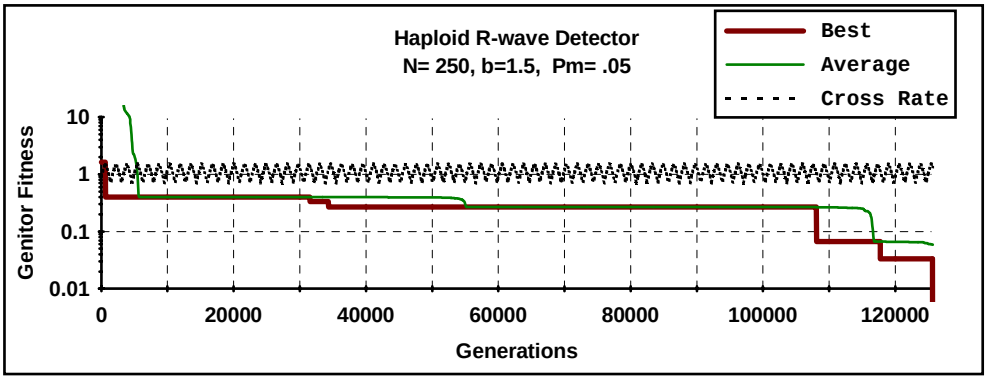


Figure 4.2 Same experiment as Fig. 4.11 using haploid population

pressure are shown in figure 4.13 and 4.14. In comparison to the (non-stationary) diploid population model *with* environment switching, a *haploid* population model, *without* switching (i.e., stationary), required 125,600 generations to converge to a perfect solution as shown in figure 4.2. If a haploid population is run without crossover (same p_m= .05 and b= 1.5), convergence is not obtained even by 200000 generations. This latter result confirms that hill climbing (non-recombinant search) alone is less efficient than either the haploid or diploid GA's used for this problem.

The initial increase in performance for the experiment of Figure 4.11 occurs when a single pulse network is found, followed by another increase when a two-pulse network is found. There are actually a number of networks that can produce two pulses that are incapable of producing pulses at precisely the right points in times. These networks represent *local optima* in the fitness landscape that make the problem relatively difficult to solve using hillclimbing alone. With mutation *only*, a deceptive problem such as this one may require multiple, simultaneously allele mutations to achieve the desired result. Since mutations are done independently, there is a vanishingly small probability of the required mutliple allele modification occurring.

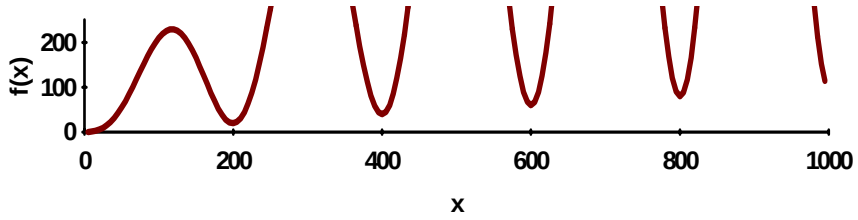
Particularly since parsimony was not addressed in the fitness function, many other networks have been found that completely solve the r-wave problem, of which three are presented in the appendix. A more thorough analysis of the R-wave problem would most likely involve multiple trials using randomly different initial populations. So that others can more readily verify and compare and verify the effects of diploid/dominance, to speed up execution, and to isolate fitness landscape behavior, such an analysis has instead been done with a simpler, and faster to execute, test function which will now be described.

C. Multiple Trial, Multimodal Test Function

A more careful multiple trial analysis to compare haploid vs. diploid was carried out with a multimodal/partially deceptive test function that is many times faster to evaluate and considerably easier to define and replicate than the R-wave problem. Other functions were tried, and rejected, because they were either too easy or difficult, and consequently could not reveal differences in performance between haploid and diploid. The function used is:

$$f(x) = x \bullet [1.1 - \cos(\pi T x)], \quad T = 10^{-2}, \quad 0 \le x < 2^{30} \quad (4.1)$$

Eq. 4.1 has monotonically decreasing minima with decreasing **binary** chromosome value x, and a global minimum at x= 0 as



Firure 4.3. Multimodal fitness function showing decreasing local minima towards global optimum.

shown in Figure 4.3. Other functions considered were Goldberg's concatenated "order-3"and "order-5"deceptive functions Goldberg (1993)(p. 10). While these latter functions may be more tractable per the extent of their "deceptivity", the order-3 was too easy and the order-5 apparently too hard to see any difference between haploid and diploid. That is, both cases either converge in about the same number of function evaluations or didn't converge to a solution in a reasonable time (e.g., 200000 evaluations.) A similar lack of difference was observed (with Eq. 4.1) when a value of T=10 was used. This choice of T reduces any monotonic structure in the function (i.e., the search becomes more like looking for a "needle in a haystack".) Eq. 4.1 is intended to model a well designed, but difficult application, where there are multimodal and deceptive regions in the search space and occasional local optima that are themselves non-deceptive (namely the maxima in the cosine term that map to decreasing local minima, which lead to the global minimum in f(x).)

Results are reported in terms of the number of *fitness function evaluations*, which is nearly proportional to execution time on a single processor machine. This is done because the diploid implementation used here requires two function evaluations per Genitor generation. No explicit environment switching was used because of the added complexity in splitting f(x) into an alternate form, as was done in the R-wave problem. With p_m=.05, b= 1.05, and chromosome length of 30 bits, an evaluation of 10 independent initial populations for each of haploid and diploid resulted in average required evaluations, to zero error[1], as shown in Figure 4.4. If we (generously) give the haploid experiment twice the population size of the diploid

Numerator → ↓ Denominator	Haploid, N=100	Haploid, N=200	Diploid, N=100	Diploid, N= 200
Haploid, N= 200	1.1	-	-	-
Diploid, N=100	4.9	**4.5**	-	-
Diploid, N= 200	5.8	5.3	1.2	-
Haploid, N= 400	5.0	4.5	1.0	0.9

Table 4.1. Ratio of Average Fitness Function Evaluations, 10 Trials

experiment, the ratio between the average number of required function evaluations for haploid and diploid populations is approximately 4.5 to 1, as shown in bold type in Table 4.1. This result is comparable to the ratio between both haploid N=100 *and* haploid N=200 vs. haploid N=400. In other words, Diploid at N=100 gives average efficiency comparable to Haploid at N= 400 for this experiment. In conclusion, the efficiency increase is not due to the increase in diversity due to diploid doubling of string count alone

The reason for comparing with a doubled haploid population size is that the diploid population will still possess the same number of total strings, and hence have equal or less diversity, or available allele values, in the initial population. As mentioned in "Methods", the second homologue of each individual in the *initial* diploid population is generated by making a *direct copy* of each randomly generated first homologue (as opposed to, say, generating the 2nd homologue of each diploid pair randomly.) As a result the diploid N=100 experiment has a generation zero population diversity equal to the haploid *N=100* in these experiments. In subsequent populations mutation will of course generate allele values not initially present.

In order to compare greater and less haploid population sizes with diploid N= 100, haploid experiments with N= 100 and N= 400 were also done. The ratio of average function evaluations in these cases are 4.9 and 1.0 to 1, respectively.

[1]One haploid N= 100 trial was stopped at 200,000 function evaluations (generations) prior to complete convergence.

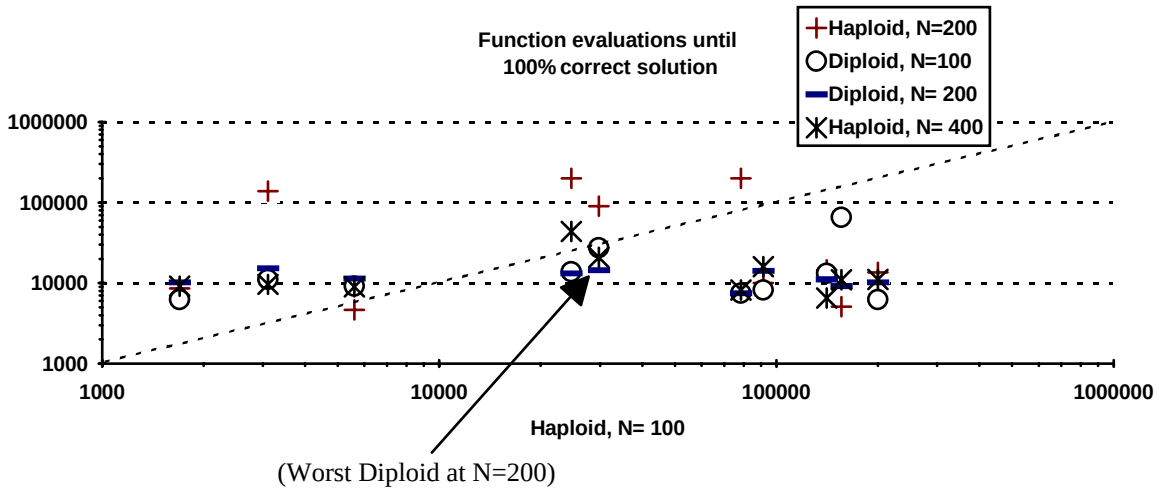


Figure 4.4 . Scatter gram of GA evaluations various haploid and diploid population sizes: Ten trials each.

The longest convergence times occur with haploid N= 100 and N= 200, as shown in Figure 4.4, with 3 trials in each case exceeding 100,000 evaluations. Both the average trend as outlined in the Table 4.1 *and* protection against routine, very large convergence times appear improved with the diploid scheme when compared to the haploid at population sizes N= 100 or higher. Diploid N=200 results in smaller *maximum* convergence when compared to any other experiment (14,400 function evaluations), The scatter of Haploid N=200 compared to the lowest five or six Haploid N=100 values are substantially *worse,* whereas diploid N=100 is only slightly worse than haploid N=100 and then only at efficiencies less than 10000. Diploid N=100 is usually better than haploid N=200, and diploid N=200 always is. This suggests that diploid may provide protection against premature convergence over an identical haploid implementation. Of course, as population size increases, both haploid *and* diploid should improve and perhaps reach a very similar number of average required evaluations. Making the above tests beginning at N= 100 appears to give sufficient but not excessive difficulty to make an effective comparison. In any case the use of diploid/dominance, even without environment switching, has a generally favorable effect on GA efficiency for this fitness function.

V. Conclusions

Results of synthesizing a signal processing network that correctly recognizes fiducial points in a simulated two-heart cycle, spectrally represented, wave form suggests the ability to handle similar applications with real clinical Doppler data. The solution described in the previous section made use of a delay element that matches the heart-cycle period and is otherwise sensible. Search difficulty was increased by including in the function set a number of function/operators not actually needed to solve the problem. This was done purposely to eliminate the necessity of defining a problem dependent function set as may be necessary for medical data.

A multiple trial, multi-modal, partially deceptive test problem provide further evidence that the Max(f_1,f_2) diploid/dominance implementation can provide better than or equal processing efficiency, compared to haploid. This conclusion is supported by a similar, though less thorough, comparison using the R-wave network synthesis problem. The Max(f_1,f_2) approach has been observed to do about the same as haploid with either very simple (e.g., unimodal) or very difficult or poorly formulated problems. Diploid/dominance as implemented here can be utilized *in conjunction* with other improvements (e.g., more refined crossover, inversion, species formation, etc.) to the standard GA. The experiments with alternating fitness environments show that multiploid populations are capable of storing and rapidly recalling as many global optima as there are homologues in each individual chromosome and shows that diploid/dominance retains recessive alleles and schema.

The diploid approach could immediately make use of a two-processor system, since the algorithm used involves two function evaluations per generations.

VI. Future Directions

The focus of my dissertation will be to further develop a methodology to detect Doppler flow states such as flow signal dropout and turbulent flow leading to development of a more robust upper 9dB spectral limit These are more complex problems than the simulated R-wave problem especially due to the associated use of "real" clinical data. An increase in power over the methods previously described can be achieved using improvements that will be discussed. For example,

a*utomatically defined functions*, have been found by Koza and others to encourage propagation and re-use of effective sub-networks or functions. Improvements may also be possible using indexed (global) memory and conditional execution nodes. First, the steps needed to analyze the existing clinical patient data will be described.

- ***Doppler Signal Processing***
  - ***Creation of test data from digitally recorded wave forms***

  Digitized clinical Doppler data can already be FFT analyzed, displayed, and recorded. This ability will be extended to manually mark events, in particular signal "drop-out", and subsequently store "marked" status in replacement data files. The spectral amplitude data from these files will be decimated during early generations as suggested by Fitzpatrick (1988) and as discussed in the literature survey. Decimation can be achieved by periodically deleting entire heart cycles, for example. Results in later generations, when reasonable solutions start to emerge, (typically N > 10000), would use all the data. Energy and spectral parameters, such as raw energy and upper 9 dB frequency, can be made available to a candidate network as *scalar inputs* through pre-defined "IN" node target addresses as mentioned in the *methods* section.

  - *Valid/invalid signal detection*

  First, clinical carotid and fetal data will be marked, manually, to indicate regions where the signal has become "invalid". These states are easy to spot visually, similar to the example given in figure 1.2. Networks will be evolved to produce output with range [0,1] so that greater than .5 indicates a valid signal or heartbeat. The raw energy, mode, and upper 9 dB will be utilized as inputs to the network, using the existing "IN" operator, in addition to the Doppler spectra. The calculation of upper 9 dB can then be modified depending on spectral bandwidth, and the line color or style altered if invalid data are detected. One or two specific data segments can be used to define alternative fitness evaluations for environment switching, if necessary. A superset of these can be used to test for actual fitness. If need be, the "size" of the training/test data base can be artificially enlarged by introducing random Gaussian noise in the time domain or artificial bruit noise, for example, in the frequency domain.

  - *Laminar/turbulent flow detection*

  As with "valid/invalid", segments of digitized data will be marked, but for clear cases of either laminar or turbulent instead of valid/invalid. The remainder of the procedure should also be similar to the valid/invalid problem.

- ***Proposed additions to node function/terminal set***
  - *Enhanced MDVI node*

  Returns vector mode with *adaptive* band limiting. That is, band offset and widths would be taken from a sub-tree as opposed to only taking it from the evolved chromosome values. This would adapt the range of spectral interest, for example, when bruit noise is detected or when substantial high frequencies are present as a result of flow turbulence..

  - *Vector filtering operator*

  This operator takes a vector (e.g., FFT magnitudes) input and produces a vector output. To encourage formation of frequency domain filters, e.g., for bruit noise elimination, this node would be decoded in a *context sensitive* fashion, since such a node would require a parent that was expecting a vector input. Alternatively, this function could be built into the existing MDVI node. The advantage of a separate operator is they could be stacked in serial or parallel.

  - *ADF's*

  Automatically define functions introduces a very powerful mechanism to define and recall useful sub-trees Koza (1993). Sub-trees are *encapsulated* by defining a new entry into the function set, after which the ADF can be recalled as a node/operator. These nodes can also be given arity > 0 if they are provided with internal *arguments*.

  - *IT-THEN-ELSE*

  This node operator Selects one of two inputs for output depending on the *sign* of a third input (arity= 3) and gives the network the explicit building blocks to have conditional flow of "program" execution.

  - *READ/WRITE*

  Automatically defines variables in a shared memory space along with *read* and *write* nodes to access them. "Write" stores its input value in a chromosome specified memory location, and simultaneously provides the same value at its output. "Read" acts as terminal node that outputs the value of a chromosome specified memory location. Specification of memory locations can occur when the read or write node is instantiated Teller (1993). The use of read/write nodes has been shown to give genetic programming the power to represent *any algorithm* in terms of its Turing completeness [Teller, 1994 #99]. These nodes have arity= 1.

- ***Extended investigation and use of diploid/dominance***

It appears from the experimental results that diploid/dominance *without* switching may be nearly as effective as with switching. This should be tested using a multimodal fitness function similar to eq. 4.31. One way to create an alternating fitness environment for this simple equation is to alternately fix odd bits in the chromosome to zero (every 100 or so generations.) If switching *is* found to be important, omission of every other heart cycle can provide the basis for an alternate fitness environments in solving the low-cost Doppler problems..

Additional mathematical analysis of the Max(f_1,f_2) approach, following direction provided by Holland and Smith should be investigated. Goldberg has said that this phenomenon may be due to random, repetitive runs of "bad luck" in the evolution process that gives the same effect as deliberate fitness switching [Goldberg, 1994 #100]. One approach to such an analysis was suggested in the *methods* section where absolute dominance is defined, during a given generation, with a fitness threshold based on the current distribution of existing haploid individuals. The suggestion there is that recessive allele retention for the proposed method is similar to that predicted by Smith for triallelic dominance. A logical next step might be to apply the difference equation approach to modeling highly fit allele propagation (please see "***Propagation rates for two GA implementations ...***" in the literature review section.) These latter equations set an arbitrary fitness threshold for allele consideration, similar to what I suggested in the *methods* section. This approach might be usable in a diploid scenario in which case more precise insight can be gained into recessive allele retention using Max(f_1,f_2). Further study of biological population genetics, from which much of the existing theory has come, will likely provide additional insight.

Further careful study should examine the range of the "T" parameter in eq. 4.31 for which diploid/dominance still shows improvement and in addition a study of the effects of varying p_m rates. I would expect results consistent with those already obtained for $T/2 < T < 2T$, or wider, if the observed diploid improvement is due to an overall well behaved fitness function that has intervening deceptive regions. Additional trials might indicate whether haploid N=100 populations can be compared to "self-initialized" diploid N=100 experiments in terms of their population diversity, by looking for matching cases having identical initial populations (due to allele proportion deficits) as mentioned in part "C" of the results. The result of this would help clarify whether one or more of the longer diploid runs observed in the multiple trials experiment were due to a severe lack of diversity in the initial population Investigations can be done to see if triploid and tetraploid dominance provide additional improvement over diploid, and what the effect would be of switching the order of gamete production with recombination, to make the approach more correct biologically.

- ***Crossover***

The current approach to chromosome crossover was a clear improvement over completely random crossover , based on an informal comparison of R-wave problem solution efficiencies. This observation is consistent with identical findings ofKoza and others, indicating that the details of the crossover algorithm can be important. One improvement to the existing crossover method might be to more systematically encourage swapping of intact sub networks One way to accomplish this is to set the second crossover point at the end of the subnet as defined by first crossover point, instead of setting the second point at the beginning of *any* node as is done now.

- *Backcoding*

To back code a more complex network than currently is possible (see *Methods*), the string position(s) of a particular node to be encoded must be determined automatically so that when the string is instantiated, the assumed depth first encoding will produce the desired network structure. This can be accomplished by including node index specifications for each specified node, and then specifying the desired children for each node by referencing the parent through its index. Also, the maximum tree depth (smaller maximum depth causes more branching) must be specified and taken into account. The algorithm would take the desired nodes and their index specifications and match up children with parents, and in the process instantiate a temporary network. The resulting temporary network would then be read out depth first, with or without specified parameter values to create a chromosome with which to seed the population. Parameter values left unspecified would be specified with a uniformly distributed random number generator within the seeded subset of the population.

- *Network Reduction and Simplification*

Simplification of network architecture is possible by deleting nodes that have *NULL* inputs (e.g., output= OR(NULL, subnet) during network instantiation. This could speed up fitness evaluation for larger networks. The simplification can occur during a final instantiation *pass* where redundant or unused nodes detected and spliced out prior to execution. and should be facilitated by the way nodes are connected to each other using C++ object pointers.

Appendix

Genetic Algorithm Processing Efficiency Goldberg (1989a)
(Expected rate of schema processing for a given population size)

Given: Chromosome length l, population size n.
To Find: Effective number of schemata processed each generation, O(?).

1. Define a schema of length $l_S < l$. The number of schemata for a given length l_S, or less, at a given spot in a given chromosome, is $2^{(l_s-1)}$. The number of schemata at all spots in a given chromosome is $(l-l_s-1)\cdot 2^{(l_s-1)}$.
3. In a population of n chromosomes, there will then be $n\cdot(l-l_s+1)\cdot 2^{(l_s-1)}$ total schemata, although some of the low order schemata will be duplicates.
4. By choosing n= $2^{l_s/2}$, there will be less than or equal to 1 instance of every possible schemata of order $l_S/2$ or more, since there are insufficient number of possible schemata (i.e., there can be no duplicate schema of order $l_S/2$ or more.).
5. If n= $2^{ls/2}$, and since the number of schemata must be binomially distributed, we have half the schemata of higher order than or equal to $l_S/2$ and half the schemata of order less than $l_S/2$. If we count only the schemata with order $\geq l_S/2$, there will be no duplicates, and a lower bound on the number of schemata in the entire population is then:
   $n_{LB} \geq n(l-l_s+1)\cdot 2^{(l_s-2)}$. Therefore,
   $$n_{LB} \geq n(l-l_s+1)(n^2/4) = (n^3/4)(l-l_s+1) = (const)(n^3) \Rightarrow O(n^3)$$

Examples of Synthesized Networks than Solve the "R-Wave" Problem

****

Example 1:
Total generations to solution: 2000 B/   0 W/1.7E+02 M/ 0.4 A/ 7.1 P/ 1.2 N/2 E/1 L 26 G16
ID No. f0512520-Diploid_W/_SWITCHING,b=1.5/m=.05_p125_DIPLOID
pop= 125, len- 30, ploids= 2, mu= 0.05 fitness/ploid=   0/ 0 re-evaluated fitness/ploid=   0/ 0
****

```
OS( 9.6, 1, 1.0)15-> 0->AM(3)14-> 0->NAND(2)11-> 0->AMW(2)-0.46,0.607-> 0->IR1(0.34,1)1-> 0->MdVI(7,54)0->
                                                                                  1->Dly(1,17)6-> 0->IRB(0.70,0.57,5)5-> 0->OS( 3.1, 1, 1.0)4-> 0->Null3->
                                                                                                                    1->MdVI(8,54)2->
                                                                              1->DFR(0.96,9)10-> 0->DFR(0.96,9)9-> 0->MdVI(0,63)8->
                               1->MdVI(0,63)12->
                              2->MdVI(0,63)13->
```

****

Example 2:
Total generations to solution: 6700 B/   0 W/0.27 M/0.27 A/0.18 P/0.99 N/2 E/1 L 26 G14
ID No. f051052n-DIPLOID
pop= 100, len- 30, ploids= 2, mu= 0.05 fitness/ploid=   0/ 1 re-evaluated fitness/ploid=   0/ 1
****

```
OS( 8.7, 1, 1.0)18-> 0->
 AMW(2)0.75,0.7517-> 0->Dly(0,3)3-> 0->IRB(0.55,0.51,6)2-> 0->INr(1.000,0)1->
                                                             1->TAR(1)0->
                        1->AM(3)16-> 0->OR(2)13-> 0->AM(3)11-> 0->Dly(4,12)5-> 0->MdVI(0,63)4->
                                                                 1->OS( 8.7, 1, 1.0)9-> 0->IRB(0.87,0.87,8)8-> 0->MdVI(0,62)7->
                                                                                                            1->Null6->
                                                                2->MdVI(0,62)10->
                                              1->MdVI(0,62)12->
                                  1->MdVI(0,62)14->
                                     2->MdVI(0,62)15->
```

****

Example 3:
Total generations to solution: 36200 B/   0 W/0.033 M/0.033 A/0.033 P/ 1.2 N/2 E/1 L 26 G14
ID No. f6105227_Diploid_#27
pop= 100, len- 30, ploids= 2, mu= 0.05 fitness/ploid=   0/ 0 re-evaluated fitness/ploid=   0/ 0
****

```
OS( 9.8, 1, 1.0)11-> 0->MUL( 9.6)10-> 0->DFR(0.98,9)9->
    0->AMW(2)0.96,0.968-> 0->IRB(0.38,0.44,2)3-> 0->DFR(0.81,8)2-> 0->MdVI(0,63)1->
                                                  1->TAR(4)0->
                           1->DFR(0.25,4)7-> 0->ADDC(-2.3)6-> 0->OS( 1.2, 1, 1.0)5-> 0->MdVI(8,54)4->
```

Bibliography

Antonisse, J. (1989). A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint. Proceedings of the Third International Conference on Genetic Algorithms, 3, 86-91.

Avriel, M. (1976). Nonlinear Programming. Englewood Cliffs: Prentice-Hall.

Brill, F., Brown, D., Martin, W. (1992). Fast Genetic Selection of Features for Neural Network Classifiers. IEEE Transactions on Neural Networks, 3(2), 324-328.

Caudell, T. P. (1992). Genetic Algorithms as a Tool for the Analysis of Adaptive Resonance Theory Network Training Sets. Combinations of Genetic Algorithms and Neural Networks, 184-200.

Crow, J. F. (1983). Genetics Notes (8 ed.). Minneapolis: Burgess.

DeJong (1980). Adaptive System Design: A Genetic Approach. IEEE Transactions on Systems, Man, and Cybernetics, SMC-10(9), 566-574.

DeJong, K. A. (1975) Analysis of the Behavior of a Class of Geneetic Adaptive Systems. PhD. Dissertation, University of Michigan.

Eshelman, L. J. (1991). The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In G. Rawlins (Eds.), Foundations of Genetic Algorithm (pp. 265-283). San Mateo: Morgan-Kaufmann.

Eshelman, L. J., Schaffer, J.D. (1992). Real-Coded Genetic Algorithms and Interval Schemata. In D. J. Whitley (Eds.), Foundations of Genetic Algorithms II (pp. 187-202). San Mateo: Morgan-Kaufmann.

Fitzpatrick, J. M., and Grefenstette, J.J. (1988). Genetic Algorithms in Noisy Environments. Machine Learning, 3(2/3), 101-120.

Goldberg (1989). Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis". Complex Systems, 3(153-171).

Goldberg, D., Rudnick, M. (1991). Genetic Algorithms and the Variance of Fitness. Complex Systems, 5, 265-278.

Goldberg, D. E. (1985). Optimal Initial Population Size for Binary-Coded Genetic Algorithms (TCGA No. 85001). University of Alabama.

Goldberg, D. E. (1989a). Genetic Algorithms in Search, Optimization & Machine Learning. Menlo Park: Addison-Wesley.

Goldberg, D. E. (1989b). Sizing Populations for Serial and Parallel Genetic Algorithms. In S. Forest (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, 3 (pp. 70-79). University of Illinois: Morgan Kaufmann.

Goldberg, D. E., and Deb, K. (1991a). A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Eds.), Foundations of Genetic Algorithms (pp. 69-93). San Mateo: Morgan-Kaufmann.

Goldberg, D. E. (1991b). Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking. Complex Systems, 5, 139-167.

Goldberg, D. E., Deb, K., Kargupta, H., Harik, G. (1993). Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. .

Greene, F., Beach,, K., Strandness, D., Fell, G., Phillips, D. (1982). Computer based pattern recognition of carotid arterial disease using pulsed-Doppler ultrasound. Ultrasound in Med. and Biol., 8(2), 181-176.

Greene, F., Phillips, D., Beach, K., Primozich, J., Strandness, D. (1989). Computer aided pattern recognition and 3-D reconstruction of carotid stenosis. In IEEE EMB Society 11 th Ann. Conference, 11 (pp. 112-113).

Greene, F. (1994). A method for utilizing diploid/dominance in genetic search. In World Conference on Computational Intelligence, ICEC-1 (pp. 439-444). Orlando, FL: IEEE.

Grefenstette, J. J. (1986). Optimization of Control Parameters for genetic algorithms. IEEE Trans Systems, Man, and Cybernetics, SMC-16(1), 122-128.

Gruau, F. (1992). Genetic synthesis of boolean neural networks with a cell rewriting developmental process. In Combinations of Genetic Algorithms and Neural Networks, . Baltimore, Maryland: IEEE Computer Society Press.

Gruau, F., Whitley D. (1993). The Cellular Development of Neural Networks: The Interaction of Learning and Evolution (Research Report No. 93-04). Laboratoire de L'Informatique du Parallelisme.

Harp, A., Samad, T., Guha, A. (1989). Towards the Genetic Synthesis of Neural Networks. Proceedings of the Third International Conference on Genetic Algorithms, 3, 360-369fÄÄÄ.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press.

Hollstien, R. B. (1971) Artificial genetic adaptation in computer control systems. Doctoral Dissertation, University of Michigan.

Hwang, H., Choi, J., Oh, S.,, Marks, R. (1990). Query learning based on boundary search and gradient computation of trained multilayer perceptrons. In International Joint Conference on Neural Networks, June . San Diego:

ICGA-93 (1993). Proceedings of the Fifth International Conference on Genetic Algorithms. In Forrest (Ed.), . University of Illinois at Urbana-Champaign: Morgan-Kaufmann.

Keith, M. J., Martin, M.C. (1993). Genetic Programing in C++ No. Allen Bradley Corporation, Heighland Heights, Ohio, 44143, Keithm@odin.icd.ab.com, (216) 646-3464.

Keller, H., Meier, W., and Anliker, M. (1976). Noninvasive measurement of velocity profiles and blood flow in the common carotid artery by pulsed Doppler ultrasound. Stroke, 7, 370-377.

Kitano, H. (1990). Designing Neural Networks Using Genetic Algorithms with Graph Generation System.Complex Systems, 4, 461-476.

Koza, J. (1993). Genetic Programming (3rd Printing ed.). Cambridge: MIT Press.

Kung, S. Y., Whitehouse, H.J., and Kailath, T. (1985). VLSI and Modern Signal Processing. Prentice Hall.

Liepins, G. E., Vose, M.D. (1991). Deceptiveness and Genetic Algorithm Dynamics. In G. J. E. Rawlins (Eds.), Foundations of Genetic Algorithms I (pp. 36-50). Morgan-Kaufmann.

Lindenmayer, A. (1990). The Algorithmic Beauty of Plants. New York: Springer-Verlag.

Martin, R. S., and Knight, J.P. (1993). Genetic Algorithms for Optimization of Integrated Circuits Synthesis. In Fifth International Conference on Genetic Algorithms, (pp. 432-439). Morgan-Kaufmann.

Maulik, D., Saini, V (1982). Doppler evaluation of fetal hemodynamics. Ultrasound in Medicine and Biology, 8, 705.

Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs. Berlin: Springer-Verlag.

Miller, G. F., Todd, P.M., Hegde, S.U. (1989). Designing Neural Networks using Genetic Algorithms. Proceedings of the Third International Conference on Genetic Algorithms, 3, 379-384.

Muller, H. J. (1950). Evidence of the precision of genetic adaptation. In C. C. Thomas (Eds.), The Harvey Lecture Series (pp. 165-229). Springfield, Illinois:

Novikoff, A. B., Hotzman, E. (1970). Cells and Organelles. New York: Holt, Rinehart, and Winston, Inc.

Park, N., and Parker, A. (1985). Synthesis of optimal clocking schemes. In Proceedings of the 22nd Design Automation Conference, ACM IEEE, .

Radcliffe, N. J. (1991). Forma analysis and random respectful recombination. In Proceedings of the Fourth International Conference on Genetic Algorithms, (pp. 222-229). Morgan-Kaufmann.

Roughgarden, J. (1979). Theory of Population Genetics and Evolutionary Ecology: An Introduction. New York: McMillan.

Rudnick, M., Goldberg, D. (1991). Signal, noise, and genetic algorithms (IlliGAL No. 91005). University of Illinois.

Rutherford, W., Kreutzer,H. (1977). The use of velocity waveform analysis in the diagnosis of carotid artery occlusive disease. Surgery, 695-702.

Saini, V. M., D. (1986). A microcomputer-based real-time system for measurement and analysis of umbilical flow velocity waveforms obtained by Doppler ultrasound. American Journal of Perinatology, 3(1), 74-76.

Smith, R. E., Goldberg, D.E. (1992). Diploidy and dominance in artificial genetic search. Complex Systems, 6, 21-285.

Teller, A. (1993). The Evolution of Mental Models. In Fifth International Conference on Genetic Algorithms (Genetic Programming Workshop), (pp. 1-13). Urbana, IL:

Whitley, D., Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connection and connectivity. Parallel Computing, 14, 347-361.

Whitley, D. (1991). Fundamental Principles of Deception in Genetic Search. In G. Rawlins (Eds.), Foundations of Genetic Algorithms (pp. 221-241). Morgan-Kaufmann.

Whitley, D. (1993). A genetic algorithm tutorial. .