NONSTATIONARY FUNCTION OPTIMIZATION USING GENETIC ALGORITHMS WITH DOMINANCE AND DIPLOIDY

David E. Goldberg and Robert E. Smith

Department of Engineering Mechanics
The University of Alabama
Tuscaloosa, AL 35487

ABSTRACT

This paper investigates the use of diploid representations and dominance operators in genetic algorithms (GAs) to improve performance in environments that vary with time. The mechanics of diploidy and dominance in natural genetics are briefly discussed, and the usage of these structures and operators in other GA investigations is reviewed. An extension of the schema theorem is developed which illustrates the ability of diploid GAs with dominance to hold alternative alleles in abeyance. Both haploid and diploid GAs are applied to a simple time varying problem: an oscillating, blind knapsack problem. Simulation results show that a diploid GA with an evolving dominance map adapts more quickly to the sudden changes in this problem environment than either a haploid GA or a diploid GA with a fixed dominance map. These proof-of-principle results indicate that diploidy and dominance can be used to induce a form of long term distributed memory within a population of structures.

INTRODUCTION

Real world problems are seldom independent of time. If you don't like the weather, wait five minutes and it will change. If this week gasoline costs $1.30 a gallon, next week it may cost $0.89 a gallon or perhaps $2.53 a gallon. In these and many more complex ways, real world environments are both nonstationary and noisy. Searching for good solutions or good behavior under such conditions is a difficult task; yet, despite the perpetual change and uncertainty, all is not lost. History does repeat itself, and what goes around does come around. The horrors of Malthusian extrapolation rarely come to pass, and solutions that worked well yesterday are at least somewhat likely to be useful when circumstances are somewhat similar tomorrow or the day after. The temporal regularity implied in these observations places a premium on search augmented by selective memory. In other words, a system which does not learn the lessons of its history is doomed to repeat its mistakes.

In this paper, we investigate the behavior of a genetic algorithm augmented by structures and operators capable of exploiting the regularity and repeatability of many nonstationary environments. Specifically, we apply genetic algorithms that include diploid genotypes and dominance operators to a simple nonstationary problem in function optimization: an oscillating, blind knapsack problem. In doing this, we find that diploidy and dominance induce a form of long term distributed memory that stores and occasionally remembers good partial solutions that were once desirable. This memory permits faster adaptation to drastic environmental shifts than is possible without the added structures and operators.

In the remainder of this paper, we explore the mechanism, theory, and implementation of dominance and diploidy in artificial genetic search. We start by examining the role of diploidy and dominance in natural genetics, and we briefly review examples of their usage in genetic algorithm circles. We extend the schema theorem to analyze the effect of these structures and mechanisms. We present results from computational experiments on a 17-object, oscillating, blind 0-1 knapsack problem. Simulations with adaptive dominance maps and diploidy are able to adapt more quickly to sudden environmental shifts than either a haploid genetic algorithm or a diploid genetic algorithm with fixed dominance map. These results are encouraging and suggest the investigation of dominance and diploidy in other GA applications in search and machine learning.

THE MECHANICS OF NATURAL DOMINANCE AND DIPLOIDY

It is surprising to some genetic algorithm newcomers that the most commonly used GA is modeled after the mechanics of haploid genetics. After all, don't most elementary genetics textbooks start off with a discussion of Mendel's pea plants and some mention of diploidy and dominance? The reason for this disparity between genetic algorithm practice and genetics textbook coverage is due to the success achieved by early GA investigators (Holland, 1975; De Jong, 1975) using haploid chromosome models on stationary problems. It was found that surprising efficacy and efficiency could be obtained using single stranded (haploid) chromosomes under the action of reproduction and crossover. As a result, later investigators of artificial genetic search have tended to ignore diploidy and dominance. In this section we examine the mechanics of diploidy and dominance to understand their roles in shielding alternate...
Most studies of genetic algorithms to date have considered only the simplest genotype found in nature, the haploid or single-stranded chromosome. In this simple model, a single-stranded string contains all the information relevant to the problem we are considering. While nature contains many haploid organisms, most of these tend to be relatively uncomplicated life forms. It seems that when nature wanted to build more complex plant and animal life it had to rely on a more complex underlying chromosomal structure, the diploid or double-stranded chromosome. In the diploid form, a genotype carries a pair of chromosomes (called homologous chromosomes), each containing information for the same functions. At first, this redundancy seems unnecessary and confusing. After all, why keep around pairs of genes which decode to the same function? Furthermore, when the pair of genes decode to different function values, how does nature decide which allele to pay attention to? To answer these questions, let’s consider a diploid chromosomal structure where we use different letters to represent different alleles (different gene function values):

\[
\begin{align*}
&\text{AbCDe} \\
&\text{aBcde}
\end{align*}
\]

At each position (locus) we have used the capital form or the lower case form of a particular letter to represent alternative alleles at that position. In nature, each allele might represent a different phenotypic characteristic (or have some nonlinear or epistatic effect on one or more phenotypic characteristics). For example, the B allele might be the brown-eyed gene and the b allele might be the blue-eyed gene. Although this scheme of thinking is not much different from the haploid (single-stranded) case, one difference is clear. Because we now have a pair of genes describing each function, something must decide which of the two values to choose because, for example, the phenotype cannot have both brown and blue eyes at the same time (unless we consider, as nature sometimes does, the possibility of intermediate forms, but we shall not concern ourselves with that possibility here).

The primary mechanism for eliminating this conflict of redundancy is through an operator which geneticists have called dominance. At a particular locus, it has been observed that one allele (the dominant allele) takes precedence (dominates) over the other alternative alleles (the recessives) at that locus. More specifically, an allele is dominant if it is expressed (it shows up in the phenotype) when paired with some other allele. In our example above, if we assume that all capital letters are dominant alleles and all lower case letters are recessive, the phenotype expressed by the example chromosome pair may be written:

\[
\begin{align*}
&\text{AbCDe} \rightarrow \text{ABCD} \\
&\text{aBcde}
\end{align*}
\]
century, black forms were caught in the neighborhood of industrial towns. Careful experiments by Kettlewell (Berry, 1972) showed that the speckled version was advantageous in the pristine setting, while the melanic (dark) form was advantageous in the industrial environment where pollution had killed off the lichen covering the tree trunks. It turned out that the melanic forms were controlled by a single dominant gene, implying that a shift in dominance occurred. When the industrial revolution shifted the balance of power toward the darkened form, the darkened form became dominant and the speckled form was held in abeyance. Note that the melanic form was not a new invention; this was no case of fortuitous mutation magically concocting the needed form. Instead, the black form had been invented earlier, perhaps in response to forests where lichen was naturally suppressed. When the by-products of industry caused the lichen to disappear, the melanic form was sampled more frequently and then evolved to the dominant form. With this alternate solution held in the background, the peppered moth was easily able to adapt rapidly to the selective pressures of its changing environment.

In this example we see how diploidy and dominance permit alternate solutions to be held in abeyance—shielded against over selection. We also see how dominance is no absolute state of affairs. Biologists have hypothesized and proven that dominance itself evolves. In other words, the dominance or lack of dominance of a particular allele is itself under genic control. Fisher (1930) theorized that dominance at a particular position (locus) along a chromosome is actually determined by another modifier gene at another locus. This implies that dominance is an evolving feature of the organism, subject to the same search procedures as any other feature. If a particular allele is favored by selection, it will spread more rapidly if it is dominant. A modifier gene therefore enhances the spread of the gene being modified. This in turn enhances the spread of the modifier. If the two genes are closely linked, this positive feedback quickly propagates both the favored allele and the modifier in the population.

But what sets the dominance of the modifier gene? In order to avoid infinite regress, we recognize that a gene can have more than one effect on the phenotype. In fact, an allele can have several major effects on the phenotype while it affects the dominance at one or more other loci. The presence of such multiple effects is known as pleiotropy. We will use a simple form of pleiotropic modifiers (one with the modifier always attached to the gene it modifies) to introduce some of the methods and ideas developed in Bagley's dissertation (1967) and in his papers on diploid GA application.

In the next section, we examine the diploidy-dominance schemes used in artificial genetic search to see how they incorporate diploid structure, dominance, and the evolution of dominance.

DIPLOIDY AND DOMINANCE IN GENETIC ALGORITHMS, AN HISTORICAL PERSPECTIVE

Some of the earliest examples of practical genetic algorithms application contained diploid genotypes and dominance mechanisms. In Bagley's early (1967) dissertation, a diploid chromosome pair mapped to a particular phenotype using a variable dominance map coded as part of the chromosome itself (Bagley, 1967, p. 136):

Each active locus contains, besides the information which identifies the parameter to which it is associated and the particular parameter value, a dominance value. At each locus the algorithm simply selects the allele having the highest dominance value. Unlike the biological case where partial dominance may be permissible (resulting, for example, in speckled eyes), our interpretation demands that only one of the alleles of the homologous loci be chosen. The decision process in the case of ties (equal dominance values) involves position effects and is somewhat complicated so that it will be necessary to outline the process in some detail.

The introduction of a dominance value for each gene allowed this scheme to adapt with succeeding generations. Unfortunately, Bagley found that the dominance values tended to fixate quite early in simulations thereby leaving dominance determination in the hands of his somewhat complicated and arbitrary tie-breaking scheme. To make matters worse, Bagley prohibited his mutation operator from operating on his dominance values, thereby further aggravating this premature convergence of dominance values. Additionally, Bagley did not compare haploid and diploid schemes, and in all of his cases the environment was held stationary. In the end, the convergence of dominance values at all positions led to an arbitrary random choice dominance mechanism and inconclusive results.

Rosenberg's (1967) biologically-oriented study contained a diploid chromosome model; however, since biochemical interactions were modeled in detail, dominance was not considered as a separate effect. Instead, any dominance effect in this study was the result of the presence or absence of a particular enzyme. The presence or absence of an enzyme could inhibit or facilitate a biochemical reaction, thus controlling some phenotypic outcome.

Hollstien's study (1971) included diploidy and an evolving dominance mechanism. In fact Hollstien described two simple, evolving dominance mechanisms and then put the simplest to use in his study of function optimization. In the first scheme, each binary gene was described by two genes, a modifier gene and a functional gene. The
A functional gene took on the normal 0 or 1 values and was decoded to some parameter in the normal manner. The modifier gene took on values of M or m. In this scheme 0 alleles were dominant when there was at least one M allele present at one of the homologous modifier loci. This resulted in a dominance expression map as displayed in Figure 1.

The Hollstien-Holland triallelic scheme is the simplest practical scheme suggested for evolving dominance and diploidy in artificial genetic search. With this scheme, the more effective allele becomes dominant, thereby shielding the recessive. Minimum excess storage is required (half a bit extra per locus) and furthermore, dominance shift can easily be handled as a mutation-like operator, mapping a 2 to a 1 (a 1 to a 10 using Holland’s notation) and vice versa. Despite the clarity of the scheme, Hollstien’s results with this mechanism were mixed. Although his Breed Type III simulations maintained better population diversity (as measured by population variance) than did his haploid simulations, there was no significant overall improvement of either average or ultimate performance. This seems surprising until we recognize that his test bed only contained stationary functions. If the role of dominance-diploidy is shielding or abeyance, we should only expect significant performance differences between haploid and diploid genetic algorithms when the environment changes with time.

Hollstien recognized that this two-locus evolving dominance scheme could be replaced by a simpler one-locus scheme by introducing a third allele at each locus. In this triallelic scheme, Hollstien drew alleles from the 3-alphabet (0, 1, 2). Here the 2 played the role of a dominant "1" and the 1 played the role of recessive "1." The dominance expression map he used is displayed in Figure 2.

The second and third schemes make local dominance decisions based on global population knowledge. The use of global information is questionable since the primary beauty of both natural and artificial genetic search is their global performance through local action. Once global operators are inserted, this attractive feature is destroyed. This is no small matter if we are ultimately concerned with efficient implementation of these methods on parallel computer architectures.

Of the remaining schemes, only the sixth scheme suggested by Brindle uses an adaptive dominance map like those in Hollstien’s (1971) and Bagley’s (1967) earlier work; however, this scheme completely separates the dominance map (the modifying genes) from the normal chromosome (the functional genes) as an added haploid chromosome.
This separation effectively destroys linkage between the dominance map and the functional genes.

In addition to the problems with the dominance schemes and test functions, Brindle's work, like studies before, considered only stationary functions. This is a common thread running through all previous genetic algorithm studies of dominance and diploidy. If dominance and diploidy do act to shield currently out-of-favor solutions from excessive selection, use of these operators in static environments is unlikely to show any performance gain when compared to a haploid GA. In the next section, we further buttress the case for dominance and diploidy as abeyance--long term memory--mechanisms through an analysis of schema propagation under these operators.

THEORY OF DOMINANCE AND DIPLOIDY

Before analyzing the specific effects of dominance and diploidy, we briefly review the notion of schemata and the fundamental theorem of genetic algorithms. The cornerstone of all genetic algorithm theory is the realization that GAs process schemata (schema-singular, schemata-plural) or similarity templates. Suppose we have a finite length binary string, and suppose we wish to describe a particular similarity. For example, consider the two strings A and B as follows:

A = 1 0 1 1 1
B = 1 1 1 0 0

We notice that the two strings both have 1's in the first and third position. A natural shorthand to describe such similarities introduces a wild card or don't care symbol, the star *', in all positions where we are disinterested in the particular bit value. For example, the similarity in the first position can be described as follows:

1 * * * *

Likewise, the similarity in the third position may be described with the shorthand notation

* * 1 * *

and the combined similarity may be described with *'s in all positions but the first and third:

1 * 1 * *

Of course, these schemata or similarity templates do not only name the strings A and B. The schema 1**** describes a subset containing $2^4 = 16$ strings, each with a one in the first position. The more specific schema 1*1** describes a subset of $2^3 - 8$ strings, each with ones in both the first and third position.

We notice that not all schemata are created equal. Some are more specific than others. We call the specificity of a schema H (the number of fixed positions) its order, o(H). For example, o(1****) = 1 and o(1*1**) = 2. Some schemata have defining positions spaced farther apart than others. We call the distance between a schema's outermost defining positions its defining length, δ(H). For example, the defining length of any one-bit schema is zero:

$δ(1****) = δ(1*1**) = 0$

On the other hand, the defining length of our order-two schema example may be calculated by subtracting the position indices of the outermost defining positions:

$δ(1*1**) = 3 - 1 = 2$

These properties are useful in the fundamental theorem of genetic algorithms, otherwise known as the schema theorem. Under fitness proportionate reproduction, simple crossover, and mutation, the expected number of copies m of a schema H is bounded by the following expression:

$$m(H, t+1) \geq m(H, t) \cdot f(H) \left[ 1 - P_m \cdot \delta(H) - P_c \cdot o(H) \right]$$

In this expression, the factors $P_m$ and $P_c$ are the mutation and crossover probabilities respectively, and the factor $f(H)$ is the schema average fitness which may be calculated by the following expression:

$$f(H) = \frac{\sum_{s_i} f(s_i)}{m(H, t)}$$

The schema average fitness $f(H)$ is simply the average of the fitness values of all strings s which currently represent the schema H. Overall, the schema theorem says that above average, short, low-order schemata are given exponentially increasing numbers of trials in successive generations. Holland (1975) has shown that this is a near-optimal strategy when the allocation process is viewed as a set of parallel overlapping, multi-armed bandit problems. We will not review this matter in detail here. Instead, we need to look at how dominance and diploidy modify expected schema propagation.

To see the effect of dominance and diploidy on schema propagation, we recognize that the schema theorem still applies: however it is useful to separate the physical schema H from its expression $H_e(H)$. Of course the expression of a schema is a function of the schema, its range of mates, and the dominance map in use. Recognizing all this, we may rewrite the schema theorem in somewhat clearer
form:

\[ m(H_{t+1}) \geq m(H_t) \frac{f(H_{t+1}(H))}{f(H_t)} \left[ 1 - p_e \frac{f(H_{t+1}(H))}{f(H_t)} - o(H) p_e \right] \]

Everything remains the same, except the average fitness of the schema H, \( f(H) \), is replaced by the average fitness of the expressed schema \( H_e(H) \), \( f(H_e(H)) \). In the case of a fully dominant schema \( H \), the average fitness of the physical schema always equals the expected average fitness of the expressed schema \( H_e(H) \):

\[ f(H) = f(H_e(H)) \]

In the case of a dominated schema \( H \), the hope is that the average fitness of the expressed schema is greater than or equal to the average fitness of the physical schema:

\[ f(H_e(H)) \geq f(H) \]

This situation is most likely to occur when the dominance map itself is permitted to evolve. If the average fitness of the allele as expressed is greater than its fitness when homozygous, then the currently deleterious, dominated schema will not be selected out of the population as rapidly as in the corresponding haploid situation. This is how dominance and diploidy shield currently out-of-favor schemata.

To make this argument more quantitative, let's consider a simple case where only two alternative, competing schemata may be expressed, one dominant and the other recessive. Physically, we can think of this as representing either two alternate alleles at a particular locus, or two multi-locus schemata that have come to dominate a particular set of loci. In either case, we assume that the dominant alternative is expressed whether heterozygous or homozygous, and we assume that the recessive alternative is expressed only when homozygous. Rearrangement of the schema growth equation permits us to calculate the proportion of recessive alleles, \( p_{t+1} \), in successive generations, t. If we assume that there are only two alternatives and the dominant form has a constant expected fitness value of \( f_d \) and the recessive form when expressed has a constant expected fitness value of \( f_r \), the proportion of recessives expected in the next generation may be calculated as follows:

\[ p_{t+1} = p_t \frac{K \left( \frac{p_t^* + r(1-p_t^*)}{(p_t^*)^2 + r(1-(p_t^*)^2)} \right)}{p_t^* + r(1-p_t^*)} \]

where \( r = f_d/f_r \), and \( K = \) crossover-mutation loss constant.

A similar equation may be derived for the haploid case where the deleterious alternative (we will still call this recessive even though it no longer recesses) is always expressed when present in a haploid structure:

\[ p_{t+1} = p_t \frac{K \left( \frac{p_t^* + r(1-p_t^*)}{(p_t^*)^2 + r(1-(p_t^*)^2)} \right)}{p_t^* + r(1-p_t^*)} \]

Proportion ratio (\( P_{t+1}/P_t \)) and proportion versus time graphs are plotted for haploid and diploid cases in Figures 3 and 4. From Figure 3 we note that for a comparable proportion of alleles, the haploid case always destroys more recessives (always has a smaller proportion ratio) than the corresponding diploid case. Of course, this does not imply that the diploid case has a low on-line performance measure. In fact, the sampling rate remains low (proportional to \( P_t^* \)) for the poor (recessive) alleles in the diploid case.

Figure 3. Expected ratio of recessives proportion \( P_{t+1}/P_t \) versus \( P_t^* \), for haploid (\( r = 2 \)), diploid (\( r = 2 \)), and limiting diploid (\( r = \infty \)).

Figure 4. Proportion of recessive alleles \( p_t^* \) versus generation t for haploid (\( r = 2 \)), diploid (\( r = 2 \)), and limiting diploid (\( r = \infty \)).

The previous analysis clearly demonstrates the long term memory induced by diploidy and dominance. Because of this effect we also expect
that mutation should play even less of a role in the operation of a diploid genetic algorithm. Holland (1975) has presented an analysis of the steady state mutation requirements of diploid structures as compared to haploid structures. We reproduce his arguments to understand the ergodic performance of these mechanisms.

For a haploid structure under selection and mutation it may be shown that the proportion of recessive alleles in the next generation $P_{r}^{t+1}$ is related to the proportion in the current generation, $P_{r}^{t}$ by the following equation:

$$P_{r}^{t+1} = (1-\epsilon)P_{r}^{t} + P_{m}(1-P_{r}^{t}) - P_{m}P_{r}^{t}$$

Here we have the sum of three terms, the proportion due to selection, the source of alleles from mutation and the loss of alleles from mutation. The $\epsilon$ factor is the proportion lost due to selection and other operator losses. At steady state,

$$P_{r}^{t+1} = P_{r}^{t} = P_{ss}$$

Solving for $P_{m}$ we obtain the following equation

$$P_{m} = \frac{\epsilon P_{ss}}{1 - 2P_{ss}}$$

This equation suggests that the final steady state proportion of alleles is directly proportional to the mutation rate (with large $\epsilon$ and small $P_{ss}$).

For a diploid structure under selection and mutation it may be shown that the proportion of recessive alleles in the next generation is related to the number in the current generation by the following equation:

$$P_{r}^{t+1} = (1-2\epsilon P_{r}^{t})P_{r}^{t} + 2P_{m}(1-2P_{r}^{t})$$

At steady state we obtain a relationship between the required mutation rate and the steady proportion of recessive alleles:

$$P_{m} = \frac{\epsilon P_{ss}^{2}}{1 - 2P_{ss}}$$

For small steady state proportions of recessive alleles, $P_{ss} < 1$, this equation suggests that the mutation rate required to keep a certain proportion of recessive alleles around is proportional to the square of the proportion. Of course, the presence of the same proportion of alleles in the diploidy case does not mean that they will be sampled as frequently. Although the same proportion is present, they are sampled as the square of their proportion. This underlines the need for occasional dominance changes so stored alleles may be sampled in the current context.

**EXPERIMENTAL RESULTS**

To investigate the behavior of a genetic algorithm with dominance and diploidy, we compare three schemes: a simple haploid GA, a diploid GA with a fixed dominance map (1 dominates 0) and a diploid GA with an evolving dominance map (Hollstein-Holland triallelic scheme). We apply all three GAs to a 17-object, blind, nonstationary 0-1 knapsack problem where the weight constraint is varied in time as a periodic step function.

The 0-1 knapsack problem is an NP-complete problem of operations research where we maximize the total value of a subset of objects (selected from a set of $N$ possible objects) that we place in a knapsack subject to some maximum load or weight constraint. More mathematically, we associate a value $v_i$ and a weight $w_i$ with the $i$th object. The problem may then be stated as follows:

$$\max_{i=1}^{N} \sum_{i=1}^{N} v_i x_i$$

subject to the weight constraint

$$\sum_{i=1}^{N} w_i x_i \leq W$$

Here the $x_i$ variables take on the values 1 or 0 as the object is in or out of the sack respectively, and $W$ is the maximum permissible weight.

Table 1

<table>
<thead>
<tr>
<th>Object Number $i$</th>
<th>Object Value $v_i$</th>
<th>Object Weight $w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

Total: $\sum v_i = 91$, $\sum w_i = 122$

We have adopted weights and values from the literature (Gillett, 1976) as shown in Table 1, but we have made the problem more difficult along two dimensions. First, as with most GAs, the problem
is presented to the GA blindly. In other words, the algorithm has no knowledge of the problem structure or problem parameters. It is forced to exploit high performance similarities in the problem coding. Second, since we are primarily interested in observing the long term memory induced by dominance and diploidy, we have introduced a nonstationarity in our problem to test this capability. In particular we cause the weight constraint to vary as a step function between two values: 80% and 50% of the total object weight. The weight is shifted between the two values at 15 generation intervals (a total period of 30 generations).

We code the problem as we did in an earlier set of haploid-only experiments (Goldberg & Smith, 1986). The 17 $x_i$ values are concatenated position by position to form a 17-bit string. In the diploid cases a second homologous string is carried along, and in the triallelic case three different alleles (1, 10, 0) are permitted at each locus within each homologous string. The constraint inequality is adjoined to the problem with an external penalty method where weight violations are squared and multiplied by a penalty coefficient, $C_{\text{penalty}} = 20$. Negative fitness values that result from the application of the penalty function are set to zero.

Optimal solutions to the problem at both weight constraints have been calculated using standard methods. In the 80% constraint case, the optimal sack value is 87 and the optimal string is 01111101111111111 (low object number to high object number from left to right). In the 50% constraint case, the optimal sack value is 71 and the optimal string is 01101101111111011.

All three GAs have been coded in Pascal and executed on IBM PCs. The details of the haploid GA are similar to those covered elsewhere. In the diploid GAs, we need to discuss one important implementation detail: the appropriate place to perform crossover. In nature, crossover occurs during the pachytene stage of the first prophase during meiosis. For GAs the important point to keep in mind is that the cross occurs (if it does) between the homologous chromosomes of a mature individual. This detail has been overlooked by some genetic algorithmists; however, it is important, because the cross within a viable individual is a less risky affair than a cross of chromosomes from random individuals (the usual haploid GA cross). There is always the possibility that when stripped of its current context (the new genome's homologous complement) that a few recessive alleles will be expressed with possible negative results; however, the new chromosome (regardless of a cross) did at least make it to adulthood in some context. In a way we may think of this as a safe cross.

As a benchmark we examine the performance of a simple haploid GA on the oscillating knapsack problem. We select GA parameters as follows:

crossover probability $p_c = 0.75$,

mutation probability $p_m = 0.001$,

and population size $N = 150$

These parameters are held constant throughout all simulations (both haploid and diploid), and stochastic remainder selection (Booker, 1982) is used throughout.

![Figure 5. Haploid GA best-of-generation results](image)

Figure 5 shows the best-of-generation results with the haploid GA. Starting from a randomly generated population, the haploid GA quickly decides on the solution with the higher fitness value (the 80% weight constraint case), and thereafter each oscillation to the other problem (50% constraint) causes a crash to infeasible, low fitness solutions. This same set of circumstances is reflected in the generation average results as shown in Figure 6. These results are not unexpected as earlier experiments with haploid GAs in nonstationary problems (Pettit & Swigger, 1983) also were unable to track changing environmental conditions.
The need for observations underscores the need for evolving dominance maps.

In the third and final experiment, we execute the triallelic diploid GA. Figures 9 and 10 show best-of-generation and generation average results respectively. The triallelic scheme with an evolving dominance map demonstrates better tracking of time-varying optima than either the haploid or fixed dominance schemes: the algorithm is able to discover and retain both optima. Despite the fact that the higher-valued solution is periodically unacceptable (overweight), the population retains the information necessary to quickly rediscover this solution in the majority of the problem cycles. Other important features of the triallelic diploid GA's performance occur at generation 135, and in the last 3 sets of cycles near the end of the run. At first glance these points seem to show drastic performance failures, but further inspection shows desirable memory-like activity at work. At generation 135, the algorithm has completely converged on the higher-valued (80% constraint) optima, and the population fitness
collapses after the weight constraint shifts; in the following generation, the alleles necessary to construct the lower-valued optima are brought out of abeyance (through expression in their homozygous form), and the algorithm recovers the optimal solution for the 50% weight constraint case. Near the end of the run, the GA does not find the higher-valued optima for two consecutive cycles; after rediscovering this optima in generation 375, however, the solution is quickly recalled in the following cycle. These are important demonstrations of the triallelic diploid GA’s ability to recover solutions from its past.

CONCLUSIONS

We have examined the use of diploidy, dominance, and evolving dominance maps in genetic algorithm search. We have found these mechanisms to be an effective means of inducing a form of long-term distributed memory within a population of structures.

Specifically, we have examined the theory and implementation of these mechanisms. The schema theorem has been extended to distinguish between a physical schema and its expression. The theory suggests that evolving dominance permits currently deleterious alleles to become recessive thereby providing an extra element of shielding against excessive selection.

Computational experiments have shown the superiority of diploidy over haploidy in a nonstationary knapsack problem. A 17-object, 0-1 knapsack problem with an oscillating weight constraint has been optimized using three GAs: haploid, diploid with a fixed dominance map, and diploid with an evolving dominance map. Both diploid schemes were better able to satisfy the switching requirements of the nonstationary environment than was the haploid scheme. The diploid scheme with the evolving dominance map was better able to quickly serve both optima as the environment changed than was the diploid scheme with the fixed map. These proof-of-principle results demonstrate the effectiveness of diploidy, dominance, and evolving dominance maps as a means of achieving faster response to severe environmental nonstationarities.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant MSM-8451610. The second author gratefully acknowledges assistantship support from the Rochester Products Division of General Motors.

REFERENCES


GENETIC ALGORITHMS
AND THEIR APPLICATIONS:
Proceedings of the
Second International Conference on
Genetic Algorithms

July 28-31, 1987
at the
Massachusetts Institute of Technology
Cambridge, MA

Sponsored By
American Association for Artificial Intelligence
Naval Research Laboratory
Bolt Beranek and Newman, Inc.

John J. Grefenstette
Naval Research Laboratory
Editor

LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
1987 Hillsdale, New Jersey Hove and London