Complex Systems **6** (1992) 251–285

Diploidy and Dominance in Artificial Genetic Search

Robert E. Smith*
Department of Engineering Mechanics, University of Alabama, Tuscaloosa, Alabama 35487, USA

David E. Goldberg†
Department of General Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA

Abstract. Genetic algorithms (GAs) continue to receive increased attention as general-purpose methods for search and optimization. GAs are attractive for search in complex spaces where the functional relationships between parameters and objective function values are of unknown, arbitrary mathematical character. Despite their generally robust character, typical GAs are known to fare poorly on functions that vary with time, where the goal is to track nonstationary optima. It has been theorized that in natural genetics, *diploidy* and *dominance* increase the survivability of species in environments that vary with time. This paper examines the effects of diploid representations and dominance operators in genetic algorithms applied to nonstationary search problems. Experimental results indicate that these additions greatly increase the efficacy of GAs in time-varying environments. This increased performance is made possible by *abeyant recessive alleles*. Analytical arguments show that these recessive alleles increase population diversity without the disruptive effects of high mutation rates, thus allowing the GA to renew its search process as the problem varies with time. Analysis also reveals that abeyant recessives are sensitive to past environmental conditions, and can therefore act as a form of *distributed, probabilistic memory* of environmental conditions that occur periodically. Final sections discuss extensions and implications of this work, including multi-locus dominance under genic control, intrachromosomal dominance, and how diploidy may affect the inherently noisy GA search process, if this noise is viewed as a nonstationary aspect of the objective function.

*Electronic mail address: `@ua1ix.ua.edu:rob@galab2.mh.ua.edu`
†Electronic mail address: `goldberg@vmd.cso.uiuc.edu`

1. Introduction

Genetic algorithms (GAs) are robust—broad and efficacious—search procedures that have been employed in a wide variety of optimization and machine-learning applications. GAs are based on an analogy to natural genetics, where a population samples a complex search domain, and selection and recombination are used to construct new populations. Because GAs are based on the idea of biasing samples, they avoid the various mathematical assumptions (e.g., of continuity, "smoothness," modality, etc.) that are inherently involved in most search procedures. This explains the GA's broad applicability. However, despite their robustness, typical GAs are known to perform poorly on nonstationary search problems, where the goal is to track time-varying optima [19]. This is disconcerting, since one of the original motivations for GAs was their use in problems that remain "perpetually novel" [15]. However, research has shown that adding *diploid* representations and *dominance* relationships improves a GA's performance on nonstationary search problems. This paper summarizes the authors' previous research on diploid GAs applied to nonstationary search problems. It also presents more detailed analyses and experimental results that were previously unavailable in the open literature. The paper concludes with a discussion of the implications of this work for search in stationary search problems, where inherent noise in the genetic search process may manifest itself as a recurring misfortune.

To motivate the use of diploid GAs in nonstationary search problems, it is first necessary to motivate GAs themselves, a task taken up in the following section.

2. An introduction to GAs

To motivate the use of a genetic metaphor for a computerized search procedure, consider the following search problem. You are given the task of finding improved designs for solutions to a given problem. Each design can be specified by a list of features. To provide feedback for the search process, you have a "black box" that, given a set of features that fully specifies a design, returns a measure of positive utility called *fitness*. The space of possible designs is far too large to enumerate, and you have *no knowledge* about the mathematical relationship between features and fitness values. You can proceed only by taking sample designs, feeding them to the black box, and using the resulting fitness values to bias subsequent samples.

Clearly, these are very limiting assumptions. In most search problems, more information will be available than is assumed here; however, this worst-case search scenario is useful for motivating a robust search procedure.

An intuitive strategy for this search problem is to feed a sample of designs to the black box, observe apparent correlations between fitness values and features, and bias samples on the basis of these observations. Such correlations are implied by similarities between various designs. For instance, consider designs whose features are encoded as binary strings of length ℓ.

Note that such a space contains 2^ℓ possible designs. If, after feeding a sample of designs to the black box, you observe that all strings that have bits four and seven both set to one have above-average fitness, it would be logical to conclude that setting these bits to one should be tried in subsequent samples. Thus, a similarity in highly fit sample designs implies a correlation between the similarity and high fitness.

For designs represented by binary encodings of length ℓ, the number of such similarities can be quantified by *schemata* (*schema* singular) that may be represented by strings of length ℓ taken from the alphabet {1, 0, *}. For instance, the schema

```
* * * 1 * * 1 * *
```

represents that set of all binary strings of length nine that have bits four and seven similarly set to one. For a space with 2^ℓ possible designs, there are 3^ℓ possible similarities. Clearly, if one is unable to enumerate all possible designs, one is even less able to examine all possible similarities. However, given the previous assumptions, it seems that examining similarities is one of the few logical ways to proceed. Therefore, one must determine a search procedure that examines some similarities and exploits the information obtained in a logical fashion. This is the motivation for GAs.

In its simplest form, a GA proceeds as follows:

Evaluation Fitness values for the members of the current population (sample) are determined.

Selection Individuals (designs) are assigned a number of copies in a *mating pool* that is used to construct the new population. The more fit an individual is, the more copies it receives. A common method is to assign copies by repeated random selection without replacement from the probability distribution

$$p_i = \frac{f_i}{\sum f_j},$$

where f_i is the fitness of individual i, the sum is taken over all population members, and p_i is the probability of individual i receiving an additional copy. Typically, the selection process is continued until the size of the mating pool equals the size of the population. Note that selection is the *emphasis* phase of the GA.

Recombination Individuals from the mating pool are recombined to form new individuals. A common recombination method is *single-point crossover*, where the encodings of two randomly selected individuals from the mating pool are split at a randomly selected *crossover point*, and halves are swapped to form two new individuals (children). As an illustration, consider the following two parents:

```
1 1 0|1 0 1 1
0 1 1|0 1 0 1
```

where a crossover point at position three is marked. Swapping halves on either side of the crossover point yields the following two children:

```
0 1 1 1 0 1 1
1 1 0 0 1 0 1
```

Crossover occurs with probability p_c, which is typically near one. Note that crossover is the *exploration* phase of the GA.

Mutation Mutation is applied to the individuals that result from selection and recombination. Typically, each bit of an individual can be flipped with some small probability p_m. Note that mutation is a mechanism for maintaining *diversity* in the GA population.

The individuals that result from these steps form a new population, and the process is iterated.

One may ask what relationship exists between the simple GA procedure outlined above and the examination of similarities. To answer this question, consider the expected number of individuals containing similarity H in a population at time $t+1$ (denoted by $m(H, t+1)$), given $m(H, t)$. Two properties of schemata are useful in deriving this expected value.

Schema order, $o(H)$, which is defined as the number of non-$*$ bits in schema H.

Schema defining length, $\delta(H)$, which is defined as the maximum number of cross points between non-$*$ bits of H.

Given these quantities, one can derive the *fundamental theorem of genetic algorithms* [14],

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

where $\bar{f}$ is the average fitness of the population are time t, and $f(H)$ is the average fitness of individuals that share the similarity dictated by schema H.

The fundamental theorem indicates that short, low-order schemata that demonstrate above-average fitness will receive exponentially increasing numbers of copies. These schemata are called *building blocks*, and they embody the central assumption in GAs, the so-called *building-block hypothesis*. Simply stated, the hypothesis says that combining short, low-order building blocks should yield higher-order schemata that also demonstrate above-average fitness. This, in essence, is the GA's strategy for examining similarities as a basis for search.

Clearly, the fundamental theorem and the building-block hypothesis are limited in two ways:

- Except in very simple problems, not all building blocks will conform to the building-block hypothesis. In other words, some short, low-order schemata that demonstrate above-average fitness will recombine to yield longer, higher-order, below-average schemata.

- The average fitness $f(H)$ obtained from a given population is only a noisy estimate of $f(H)$ over all possible individuals that share the similarity H.

Two other fundamental theoretical developments address these limitations:

Implicit parallelism The number of building blocks processed by a GA with population size n has been estimated to be $O(n^3)$ [7, 14]. In processing this large number of schemata implicitly and in parallel, the GA often finds sufficient schemata that recombine according to the building-block hypothesis and lead to highly fit individuals.

The *k*-armed bandit argument It can be shown that in a probabilistic decision problem that involves refining estimates through exploration while improving performance through exploitation, a near-optimal strategy is to allocate exponential trials to the observed best. This justifies the GA approach to estimating schema fitness values while allocating an exponentially increasing number of copies [11, 14, 20].

The efficacy of the strategy embodied in these theoretical developments is borne out in numerous GA applications in a wide range of disciplines [7].

3. GAs and nonstationary search

Clearly, the $O(n^3)$ estimate is based on a *diverse* population, where many schemata are represented. However, as exponential allocation of observed-best schemata accrues, one can expect that the number of building blocks processed will decrease. This is an inevitable consequence of convergence in the GA outlined above. After convergence, the GA population will be composed primarily of copies of one individual. The only diversity maintained in the population after convergence is a result of mutation. Note that mutation is a completely random operator that is unguided by the algorithm's observations of fitness values over time.

In traditional optimization applications, convergence of the GA is desirable. However, the lack of population diversity after convergence causes the typical GA described above to fare poorly on problems where the goal of the search is to track a time-varying function. If the character of the objective function changes after the GA population converges, the population does not possess sufficient diversity to allow the search to begin anew.

One solution to this dilemma would simply be to increase the mutation rate such that diversity is maintained. However, this has the effect of disrupting the genetic search process and preventing convergence. Moreover, the mutation provides no mechanism for the GA to "learn" any temporal regularities that exist in the function's variations. Specifically, if the average fitness values of certain building blocks vary with time while the average fitness values of other building blocks remain relatively constant, it would be desirable for the GA to maintain diversity in the former while converging in

the later. This would constitute a form of *temporal memory* of fitness variation. Mutation provides no mechanism for this behavior since it is insensitive to fitness.

Biological theory suggests that diploid chromosomes and dominance mechanisms may improve the survivability of species in time-varying environments [1]. Therefore, as was first realized by Holland [14], it seems logical to examine analogous representations in GAs. The following section introduces diploidy and dominance in GAs.

4. Diploid GAs

The GA described in section 2 assumed a *haploid* representation of potential problem solutions. That is, each population member was a bit string that contained sufficient information to specify a complete solution design. The analogous haploid chromosomes in natural genetics are found primarily in very simple organisms. More complex organisms often have diploid chromosomes, which contain *twice* the information necessary for specifying the organism's structure. Conflicts that occur between the two halves of a diploid chromosome are resolved by a *dominance relationship*, which (in its simplest form) decides on one of the conflicting genes that is eventually expressed in the organism itself. In a GA, a diploid individual has two bit strings, each of which is sufficient to specify a complete solution design. A dominance relationship specifies how these two strings are decoded into a single *expressed* string whose fitness is evaluated. For instance, if we consider a dominance relationship where 1 always dominates 0 (1 is called the *dominant allele* and 0 is called the *recessive allele*), the following diploid individual,

```
1 0 1 1 0 1 0 1 1
0 0 1 0 1 0 1 1 0
```

decodes to the following expressed string:

```
1 0 1 1 1 1 1 1 1
```

How do dominance and diploidy improve GA performance on nonstationary search problems? The simplest explanation is that recessive alleles in a diploid GA preserve population diversity after convergence. When recessive alleles are paired with dominant alleles, they are held in *abeyance* and are effectively shielded from adverse selective pressure. Thus, after convergence, some diversity is retained in the form of abeyant recessives. Holland [14] suggested the value of diploidy as a diversity-preserving mechanism by showing that less mutation is needed to maintain a given level of diversity in a diploid GA than a haploid GA. Consider the expected population proportion of a given allele (1 or 0) at a given bit position at time t, which will be denoted by P^t:

$$P^{t+1} = (1 - \epsilon)P^t + p_m(1 - P^t) - p_m P^t,$$

where $\epsilon > 0$ is the expected change in this proportion due to fitness-proportionate selection for this allele. Note that for a given fitness function, ϵ is a function of allele proportions only. Solving for a steady state ($P^{t+1} = P^t = P_{ss}$) and rearranging yields

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

If one assumes steady-state proportions, ϵ^{ss} can be assumed to be constant. If P_{ss} is a small, desired steady-state proportion of the allele that is less favored by selection ($\epsilon^{ss} < 1$), this equation indicates that the p_m necessary to maintain P_{ss} after convergence is proportional to P_{ss}.

Consider a similar formulation for a diploid GA, where P^t is the proportion of the recessive allele at time t. In this situation,

$$P^{t+1} = (1 - 2\epsilon P^t)P^t + 2p_m(1 - 2P^t).$$

Solving for a steady state and rearranging as before gives

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

If P_{ss} is a small, desired steady-state proportion of the allele that is less favored by selection, this equation indicates that the p_m necessary to maintain P_{ss} after convergence is proportional to P_{ss}^2, a much lower value than in the haploid GA.

These steady-state arguments show how diploidy requires a lower level of mutation to obtain a desired level of steady-state diversity. However, examination of the transient behavior reveals more about the utility of diploid GAs in nonstationary problems [13].

Consider a problem in which a given allele at a given position is advantageous for some period of time. After this period, the allele is no longer advantageous; rather, its compliment is favored by selection. In a nonstationary problem, it may be desirable to preserve the originally advantageous allele for some period of time since the conditions under which it was favored may return.

One can examine proportion equations to show how haploid and diploid GAs preserve such an allele. Assume that the originally advantageous allele is recessive in a diploid GA. Recall that this allele appears in both expressed and unexpressed form, depending on whether it is paired with another recessive allele or a dominant allele, respectively. The average fitness of a recessive allele (which will be called f_{r_e}), regardless of whether it is paired with a dominant or a recessive allele, is given by

$$f_{r_e} = f_r(P_r^t) + f_d(1 - P_r^t),$$

where f_r is the fitness of the recessive allele in its expressed form, f_d is the fitness when the dominant allele is expressed, and P_r^t is the proportion of recessive alleles at time t. The average fitness of the population is given by

$$\bar{f} = f_r(P_r^t)^2 + f_d(1 - (P_r^t)^2).$$

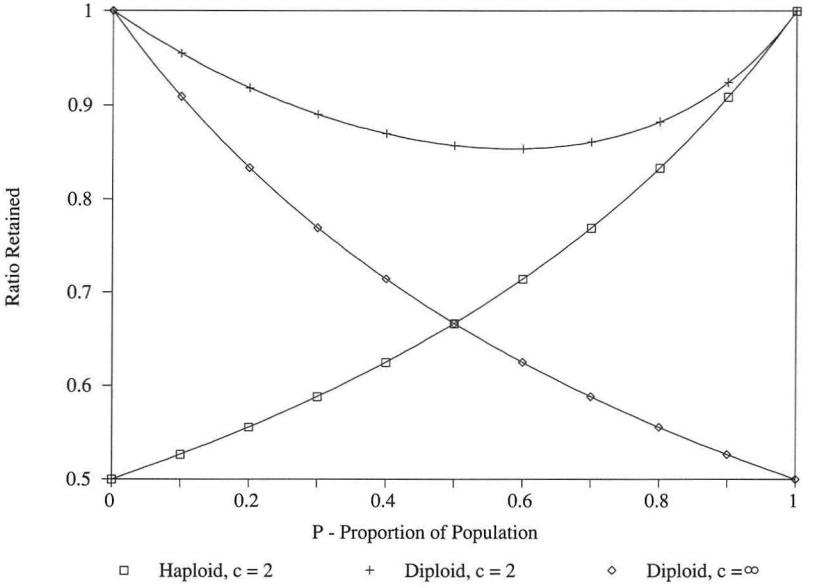

Figure 1: Expected ratio of the proportion of recessive alleles, P_r^{t+1}/P_r^t, versus P_r^t for haploid ($c = 2$) and diploid ($c = 2$ and $c = \infty$) cases [13].

This gives the expected change in the proportion of the recessive allele as

$$P_r^{t+1} = P_r^t K \left[\frac{P_r^t + c(1 - P_r^t)}{(P_r^t)^2 + c\left[1 - (P_r^t)^2\right]} \right],$$

where $c = f_d/f_r$ and K is a constant that expresses the loss due to mutation. A similar expression can be derived for the haploid GA:

$$P_r^{t+1} = \frac{K P_r^t}{P_r^t + c\,(1 - P_r^t)}.$$

Given these equations, one can use the relative rate of change of P_r, namely P_r^{t+1}/P_r^t, as a measure of how slowly recessive alleles are deleted by selection. Figure 1 shows plots of this quantity versus P_r for haploid and diploid GAs with $c = 2$ (where the dominant is twice as advantageous as the recessive), and for the diploid GA with $c = \infty$ (where the dominant is infinitely better then the recessive). The plots show that the rate of change of the proportion of recessives is much slower for the diploid GA than for the haploid GA when $c = 2$. Even for $c = \infty$, the proportion of recessive alleles changes more slowly in the diploid GA than in the haploid GA with $c = 2$ for $P_r > .5$. These results indicate that recessive alleles that have been emphasized during one period of the function's evolution will be retained at

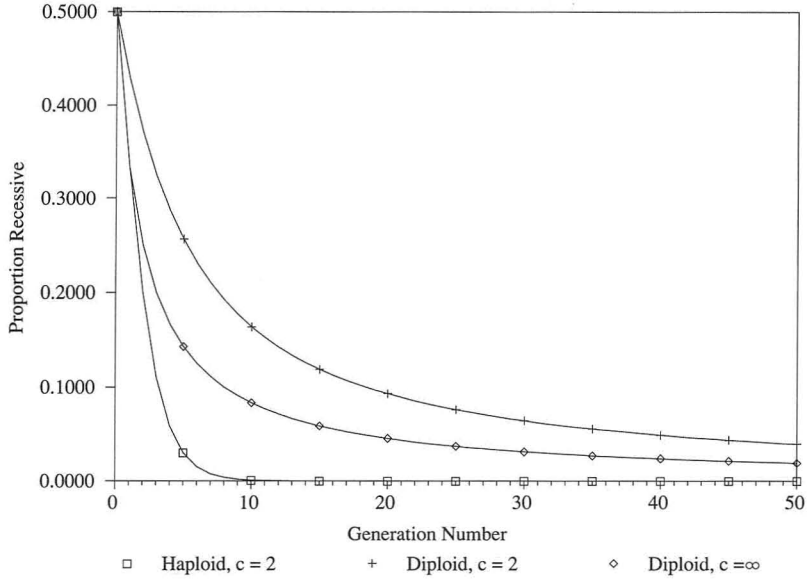

Figure 2: Proportion of recessive alleles P_r^t versus generation number for haploid ($c = 2$) and diploid ($c = 2$ and $c = \infty$) cases [13].

proportions higher than steady state for an extended period of time. Figure 2 illustrates this point by iterating the proportion equations used above.

These results show that diploidy does more than simply maintain diversity after convergence. It allows the GA to maintain extra diversity at positions that have had alternative alleles emphasized in the function's recent past. Unlike mutation, this is a maintenance of diversity that is *sensitive* to the system's fitness history. In effect, the recessive alleles are a form of *memory* of past function conditions. These analytical observations are illustrated in experiments in the following section.

5. Fixed dominance and time-varying functions

As a simple example, consider the *0-1 knapsack problem* [23]. Knapsack problems are a class of common but difficult (NP-complete) problems in operations research. A variety of industrial problems can be reduced to knapsack problems, including cargo loading, stock cutting, project selection, and budget control. Knapsack problems also arise in the consideration of a variety of linear programming problems.

The 0-1 knapsack problem can be defined in terms of the following simple analogy. Consider a scavenger with a bag that holds a maximum weight W. The scavenger has available n objects, each with weight w_i and value v_i. The scavenger's objective is to maximize the value that can be carried in the bag.

Number i	Value v_i	Weight w_i	$W = 60$ Optimal x_i	$W = 104$ Optimal x_i
1	2	12	0	0
2	3	5	1	1
3	9	20	0	1
4	2	1	1	1
5	4	5	1	1
6	4	3	1	1
7	2	10	0	0
8	7	6	1	1
9	8	8	1	1
10	10	7	1	1
11	3	4	1	1
12	6	12	1	1
13	5	3	1	1
14	5	3	1	1
15	7	20	0	1
16	8	1	1	1
17	6	2	1	1
Total:	91	122	13	15
			$\sum_{i=1}^{17} x_i v_i = 71$	$\sum_{i=1}^{17} x_i v_i = 87$
			$\sum_{i=1}^{17} x_i w_i = 60$	$\sum_{i=1}^{17} x_i w_i = 100$

Table 1: The 17-object, 0-1 knapsack problem parameters used here with optimal solutions.

Mathematically, the problem is to find

$$\max \sum_{i=1}^{n} v_i x_i$$

subject to the weight constraint

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

where the x_i's are variables that can be set to either 0 or 1; W, the v_i's, and the w_i's are given problem parameters; and n is the problem size. A 17-object, 0-1 knapsack problem is the basis for the test problems used here [6]. Object weights and values for this problem are shown in table 1. Although this problem has a moderately large search space (2^{17} possible solutions), previous experience [12] indicates that a haploid GA with a moderate population size (150) converges to the problem's optimal solution in approximately 15 generations. This relatively short convergence time made the problem convenient for adaptation to the time-varying optimization experiments.

In the experiments examined in this section, the knapsack problem is made nonstationary through variation of the constraint W. Specifically, the weight constraint is switched every 15 generations between 104 and 60. The optimal solutions associated with these constraint values were determined using standard methods [23]. They are $\sum_{i=1}^{17} x_i v_i = 87$ for the case where $W = 104$, and $\sum_{i=1}^{17} x_i v_i = 71$ for the case where $W = 60$. These optimal knapsacks are indicated in table 1. Note that they differ by two bits.

A version of the haploid GA (described in section 2) is applied to this problem, with the following parameters:

- crossover probability $p_c = 0.750$,
- mutation probability $p_m = 0.001$, and
- population size = 150.

Solutions are coded as strings of length 17, where the ith position represents the x_i variable for the ith object in table 1. The weight constraint is enforced by a penalty function on fitness. Specifically, if the solution dictated by a population member is overweight by ΔW, then its fitness is the solution's value minus $C(\Delta W)^2$, where the penalty coefficient C is 20. Negative fitness values are set to zero. Linear fitness scaling [7] and stochastic remainder selection [2] are used in all experiments. The initial population is generated at random.

Figures 3 and 4 show typical best-of-generation and generation-average results, respectively, from the haploid GA. The GA converges to the optimum in one of the two switching conditions, and fails to have sufficient population diversity to continue searching when the weight constraint changes. After the change, all the solutions suggested by the population are severely overweight and have zero fitness.

Now consider the application of a diploid GA with the fixed 1-dominates-0 dominance map discussed previously. The parameters and operation are identical to those of the haploid GA in the previous experiment, with the exception of alterations in the crossover procedure to account for diploidy. In the diploid GA, crossover first occurs between the strings of a haploid individual, and then these halves are swapped between parents. This is a simulation of *gametogenesis*, and yields the same recombination effects as crossover in the haploid GA.

Figures 5 and 6 show typical best-of-generation and generation-average results (respectively) for the nonstationary knapsack problem with the fixed-map diploid GA. In this case, search continues as the function varies with time. Note that the lower optimum is consistently rediscovered when the function switches. This rediscovery takes place without renewed search. This illustrates that the diploid GA is doing more than simply maintaining diversity across all bit positions (as would a high mutation rate); diploidy is preserving the specific alternative alleles necessary to reconstruct this optimal solution immediately when the function switches.

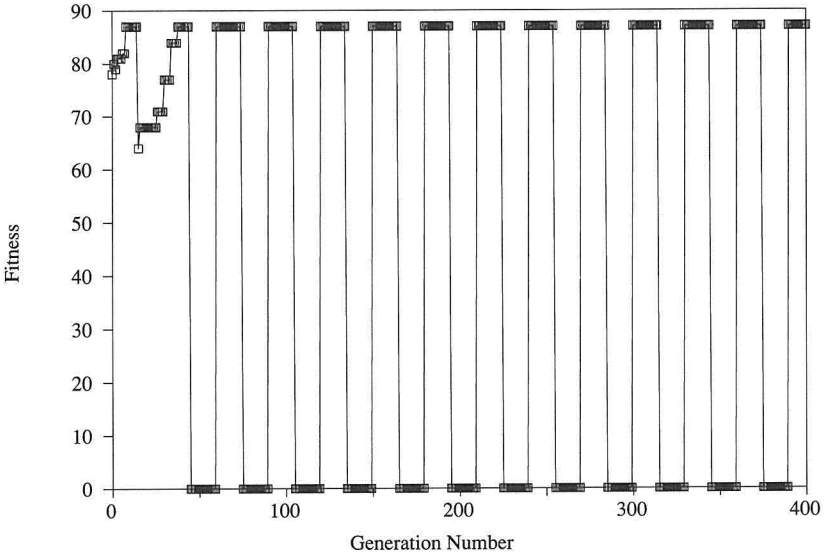

Figure 3: Haploid GA best-of-generation results.

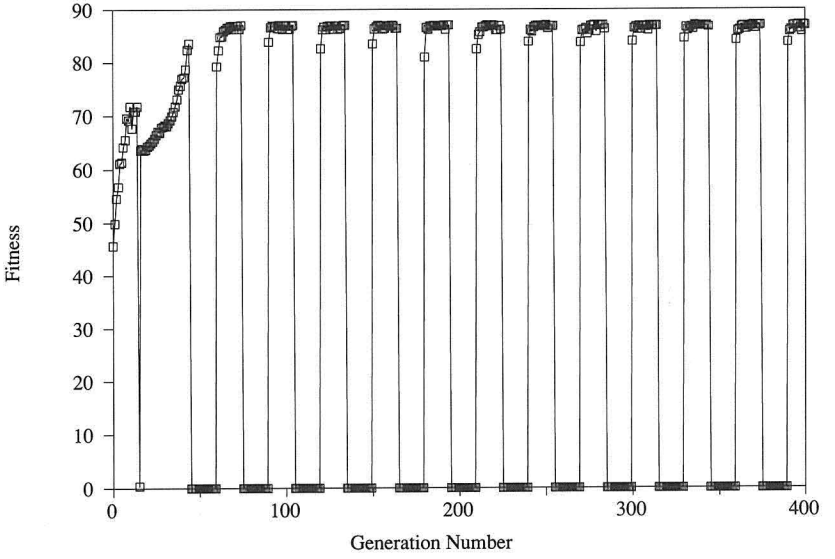

Figure 4: Haploid GA generation average results.

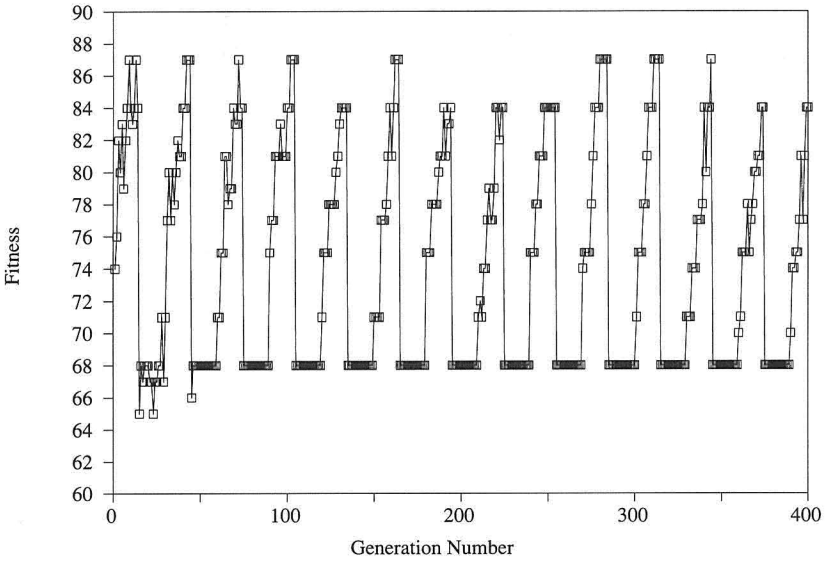

Figure 5: Diploid GA with fixed dominance map (1-dominates-0) best-of-generation results.

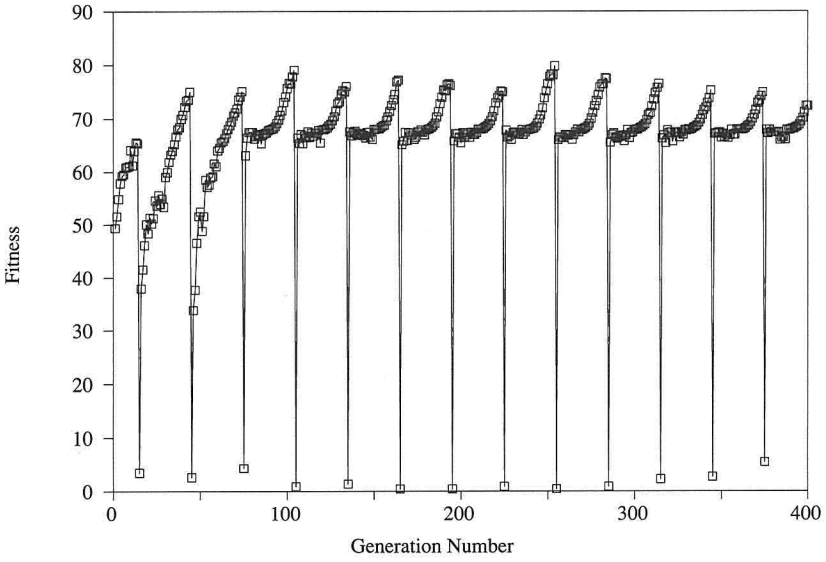

Figure 6: Diploid GA with fixed dominance map (1-dominates-0) generation average results.

Although these results are interesting, the dominance mechanism employed is clearly flawed. Although the higher optimum is rediscovered in some cycles, it is only after renewed search, which often fails to find the true optimum. This is because the higher function value requires more 1-valued alleles in the string, and these alleles cannot be "memorized" as recessives since they are always dominant. To deal with arbitrary nonstationary functions, the diploid GA requires a dominance map that can evolve to have either 0-valued or 1-valued alleles dominate at any position. The following section introduces a scheme suggested by Hollstien [17] and Holland [14] that allows for an adaptive dominance map.

6. The triallelic adaptive dominance scheme

Realizing the need for an adaptive dominance map, Hollstien [17] and later Holland [14] suggested a scheme with three possible alleles at each position, $\{0, 1, 1_0\}$. Under this scheme, both 1 and 1_0 are expressed as 1, but 1 dominates 0 and 0 dominates 1_0. Thus, a 1-valued allele can act either dominant or recessive.

Operation of the diploid GA under this scheme is identical to that of the fixed-map scheme, except for a slight alteration in the mutation operator. In this case, mutation converts each allele type to one of the other two types with equal probability. Thus, mutation accounts for changes in *value* (i.e., from a 1-valued allele to a 0) and changes in *dominance* (i.e., from a dominant 1 to a recessive 1_0). In the following experiments, the population is randomly initialized so that the number of 1-valued and 0-valued alleles is expected to be equal.

Figures 7 and 8 show best-of-generation and generation average results obtained on the nonstationary knapsack problem with a diploid GA and the triallelic dominance scheme. The results indicate that the triallelic GA is superior to both the haploid GA and the fixed-map diploid GA in its ability to track the nonstationary problem. The triallelic GA is frequently able to recall each optimum quickly. More detailed observations reveal how the triallelic GA stores and retrieves alternate information from abeyance.

At generation 135 the population has completely converged to the higher-valued solution, and the weight constraint switching to 60 causes a "crash" in population fitness. In the next generation the alleles necessary to reconstruct the lower-valued optimal solution are brought out of abeyance, and the algorithm recovers the optimal solution for the $W = 60$ case.

Another illustration of the triallelic GA's ability to recall abeyant information is near the end of the run. In the third from last cycle of the run, the algorithm has converged completely to the lower-valued optimum. The higher-valued optimum is not discovered at all for one full cycle, and in the following cycle it is only found in the last generation before the weight constraint switches. After the higher-valued optimum is rediscovered in generation 375, it is quickly recalled from abeyance in the following cycle.

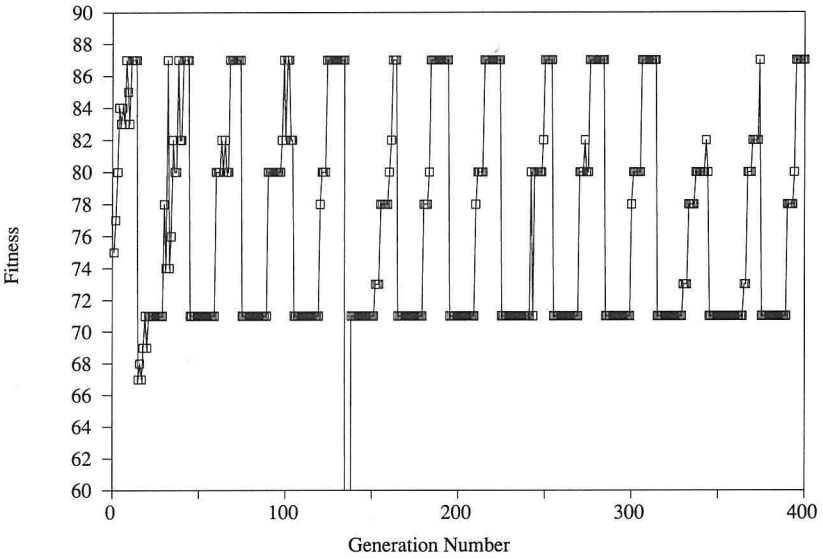

Figure 7: Diploid GA with evolving dominance map (triallelic scheme) best-of-generation results.

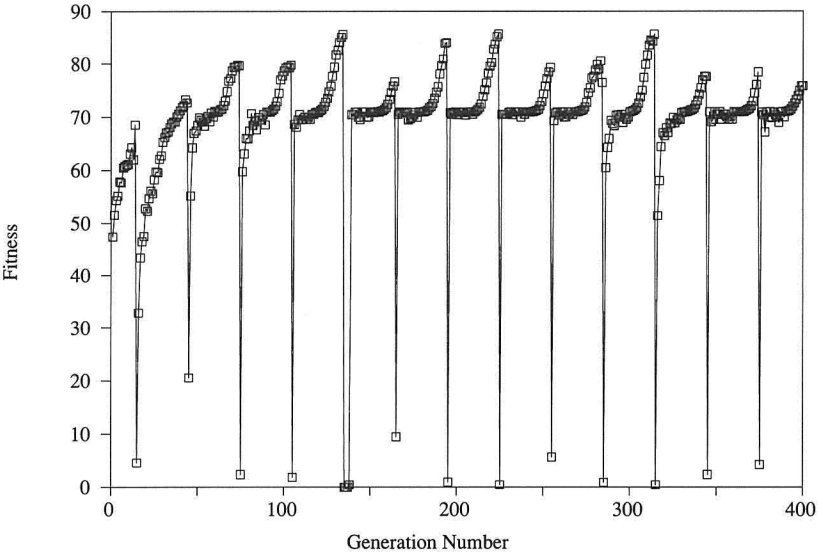

Figure 8: Diploid GA with evolving dominance map (triallelic scheme) generation average results.

This experiment illustrates that diploidy and adaptive dominance greatly improve the GA's ability to track a nonstationary function. However, the exact dynamics of a diploid GA with the triallelic dominance scheme are not entirely apparent. The following section presents some additional analysis of the triallelic scheme.

7. Further analysis of the triallelic dominance scheme

Proportion equations like those used in section 4 are useful in clarifying the behavior of the triallelic dominance scheme. To derive these equations, one must first derive the average fitness values of the three alleles taken over all possible allele combinations in which they appear. These values are given by

$$f(1_{0_e}) = \frac{f_1 P_{1_0}^2 + f_0 P_{1_0} P_0 + f_1 P_{1_0} P_1}{P_{1_0}} = f_1 + P_0(f_0 - f_1),$$

$$f(0_e) = \frac{f_0 P_0^2 + f_1 P_0 P_1 + f_0 P_0 P_{1_0}}{P_0} = f_0 + P_1(f_1 - f_0),$$

$$f(1_e) = f_1,$$

where f_1 is the fitness of any combination of alleles that is expressed as a 1, f_0 is the fitness of any combination that is expressed as a 0, and P_0, P_1, and P_{1_0} are the proportions of 0, 1, and 1_0 alleles, respectively. The average fitness of a triallelic population is given by

$$\bar{f} = f_1\left(1 - 2P_0P_{1_0} - P_0^2\right) + f_0\left(2P_0P_{1_0} + P_0^2\right).$$

For notational convenience, each allele is assigned a reproductive proportion (R) based on its expected average fitness:

$$R_{1_0} = \frac{1}{\bar{f}} P_{1_0}\left[r + P_0(1-c)\right],$$

$$R_0 = \frac{1}{\bar{f}} P_0\left[1 + P_1(c-1)\right],$$

$$R_1 = \frac{1}{\bar{f}} P_1 c,$$

where $c = f_1/f_0$.

Before proceeding to draw the proportion equations, it is useful to split the mutation operator into two separate operators: a mutation of value operator and a *dominance shift* operator. The associated rates of application of these operators will be p_m and p_s, respectively. In the following analysis, dominance shift is defined as a change in the dominance associated with a 1 or a 1_0 allele, and mutation is defined as a change in the value associated with an allele, with dominance determined randomly when the mutation is from a 0-valued allele to a 1-valued allele. These operator definitions are shown in table 2. To avoid the effects of both operators, it is assumed that either mutation *or* dominance shift occurs, but never both. In simulations,

Operator	Allele		
	1	0	1_0
Dominance Shift	changed to 1_0	not affected	changed to 1
Mutation	changed to 0	changed to 1 or 1_0 (with equal probability)	changed to 0

Table 2: The effects of separate dominance shift and mutation operators in the triallelic GA.

the operator that is given the possibility to act is selected by the flip of a fair coin. Note that the common factor of 0.5 introduced into the mutation and dominance shift rates by this procedure is not shown, but is assumed in the following analysis.

With definitions of separate dominance shift and mutation, the proportion equations for the three possible alleles can be stated as:

$$\begin{aligned} P_{1_0}^{t+1} &\geq R_{1_0}^t(1 - p_m - p_s), \\ P_0^{t+1} &\geq R_0^t(1 - p_m), \\ P_1^{t+1} &\geq R_1^t(1 - p_m - p_s). \end{aligned}$$

The addition of mutation and dominance shift source terms in a manner similar to Bridges and Goldberg [3] completes the equations:

$$\begin{aligned} P_{1_0}^{t+1} &= [R_{1_0}^t(1 - p_m - p_s) + R_1^t p_s + \tfrac{1}{2} R_0^t p_m], \\ P_0^{t+1} &= [R_0^t(1 - p_m) + p_m(R_1^t + R_{1_0}^t)], \\ P_1^{t+1} &= [R_1^t(1 - p_m - p_s) + R_{1_0}^t p_s + \tfrac{1}{2} R_0^2 p_m]. \end{aligned}$$

These equations constitute a complete description of the expected proportions of 1s, 0s, and 1_0s for a single locus in the triallelic GA with mutation, dominance shift, and fitness-proportionate selection operators.

As in the analysis of fixed-map diploid schemes, it is useful to examine the steady-state proportion of recessive alleles. Unfortunately, deriving an expression for the steady-state proportions is difficult. Nevertheless, steady states can be found numerically.[1] Although this numerical-solution technique does not *insure* that other steady states do not exist, or that a single trajectory is followed as p_m and p_s are changed, one can make interesting qualitative observations from the results. These qualitative results are confirmed through further experimentation presented later in this paper and elsewhere [21].

Figures 9, 10, and 11 show the spline curves through numerically-determined steady-state values for various settings of p_m and r versus p_s.[2] In these graphs, a 0 is considered recessive if $c > 1$, and 1_0 is considered recessive if $c < 1$. As one might expect, these graphs indicate that the steady-state

[1]The Eureka package from Borland was employed to determine numerical steady states.

[2]Note that the results for $c < 1$ were obtained using a version of the proportion equations with f_0/f_1 as a parameter, rather than f_1/f_0.

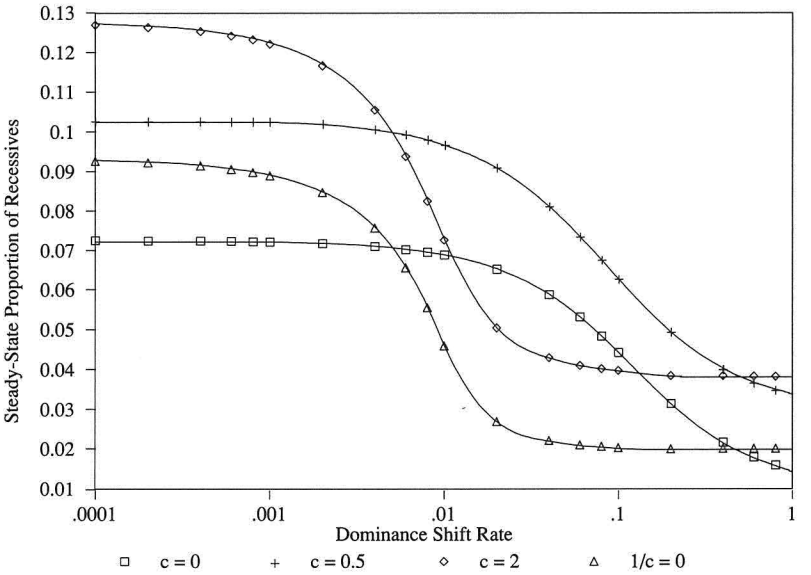

Figure 9: Numerically determined steady-state proportions of recessive alleles versus p_s (log scale) for $p_m = 0.01$. Spline curves through discrete data.

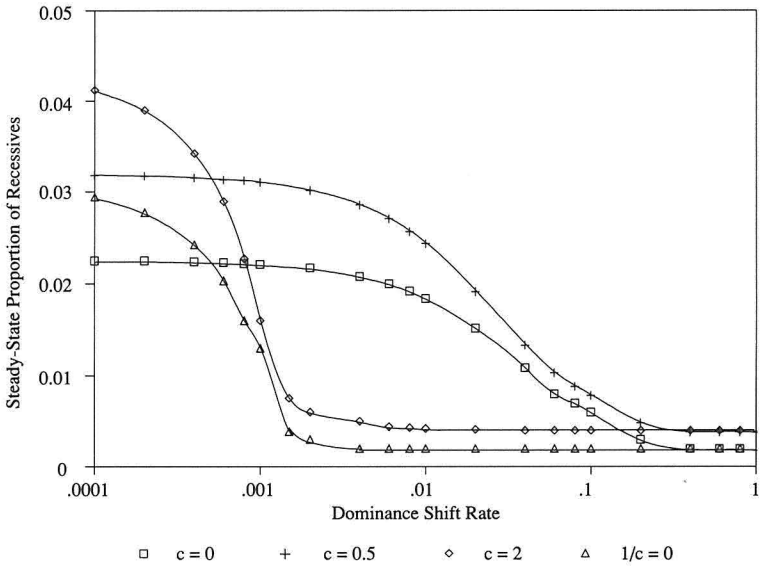

Figure 10: Numerically determined steady-state proportions of recessive alleles versus p_s (log scale) for $p_m = 0.001$. Spline curves through discrete data.

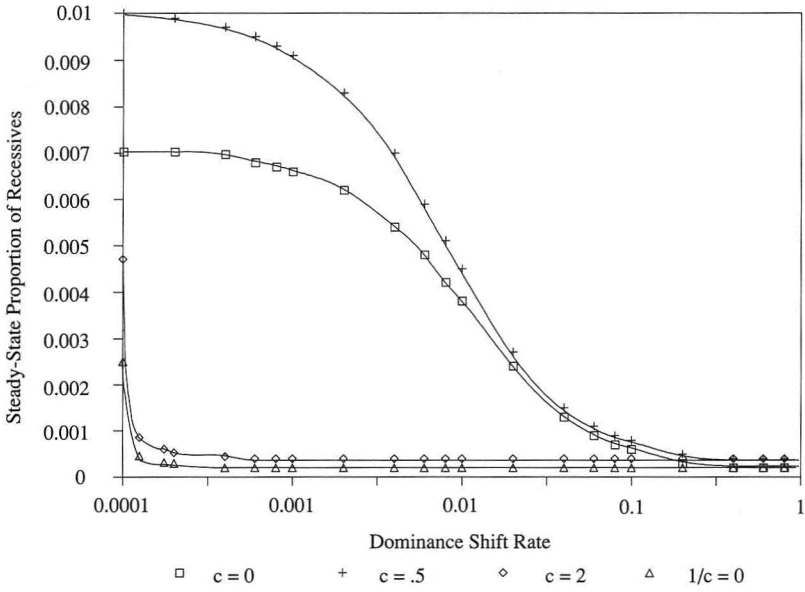

Figure 11: Numerically determined steady-state proportions of recessive alleles versus p_s (log scale) for $p_m = 0.0001$. Spline curves through discrete data.

proportion of recessive alleles falls as p_s is increased. Note that for high p_s the proportion of recessives is on the order of p_m. This is reasonable since for high p_s recessives are expressed often, and are therefore driven out of abeyance in the long run. For low p_s, the steady-state proportion of recessives is on the order of $p_m^{0.5}$, as predicted by the analysis presented previously by Holland [14]. In this case the alternative alleles are seldom expressed (on the order of P_{ss}^2), and are therefore held in abeyance in greater proportion.

These graphs indicate another feature of the triallelic dominance scheme. For $f_1 < f_0$ ($c < 1$), there is a relatively smooth growth of P_{ss} as p_s is lowered. However, for $f_1 > f_0$ ($c > 1$), there is a steep jump in the P_{ss} versus p_s curve. The jump occurs around $p_s = p_m$ in all three graphs. Figure 12 explains this phenomenon. This figure shows that for $c > 1$ and $p_s \ll p_m$, the steady-state condition has a proportion of 1s much greater than the proportion of 1_0s. For $p_s \gg p_m$, the system tends toward a steady state in which the proportion of 1-valued alleles is equally distributed between 1s and 1_0s. The latter case allows for much greater expression of 0 alleles; therefore the steady-state proportion of recessive 0s is dramatically lowered as p_s is raised above p_m. A similar effect does not exist for recessive 1s. For some parameter settings, this difference also has the effect of making P_{ss} for 0-valued recessive alleles much lower than that for 1-valued recessive alleles, thus creating a preference for 1 as a recessive allele in the triallelic system. These problems indicate an asymmetry in the triallelic representation caused

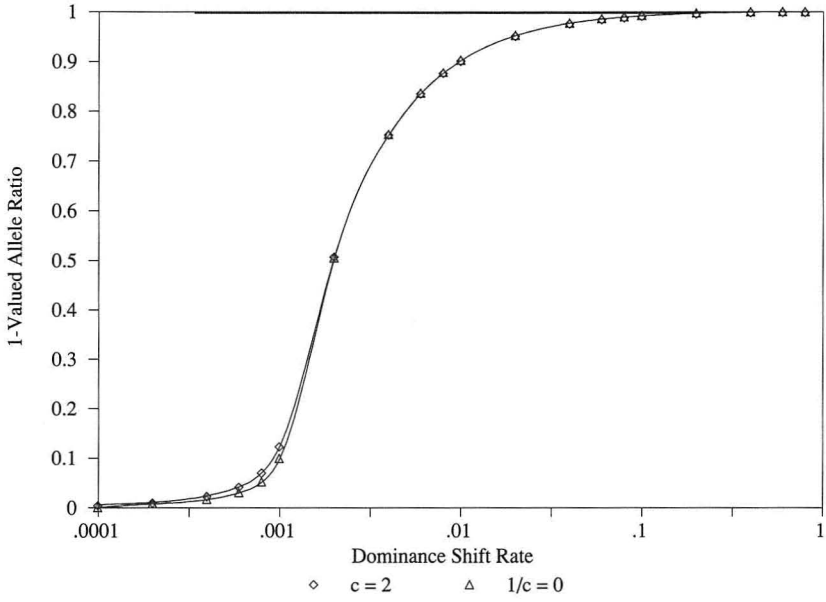

Figure 12: Ratio of numerically determined steady-state proportions of alleles 1 and 1_0 versus p_s (log scale) for $p_m = .01$. Spline curves through discrete data.

by the presence of two 1-valued alleles and only one 0-valued allele. Although it is negligible in many cases, this asymmetry will be evident in other analyses and in experimentation.

As in the analysis in section 4, examination of the transient behavior of the triallelic GA is also revealing. In this case the proportions are solved iteratively (starting from equal expressed proportions of 1s and 0s), and the transient times are estimated as the number of generations clocked until the proportion of recessives changes less than 0.01%. Once again, these results are qualitative, in that there is no proof that steady states and trajectories other than those shown do not exist. However, the qualitative results are confirmed in later experimentation.

Graphs of the time to the steady state versus c and $1/c$ are presented in figures 13 and 14. Note that both c and $1/c$ are plotted on log scales. The plots are spline-interpolated lines through discrete data. Plots of the proportions in the haploid GA and the fixed-map (1-dominates-0) diploid GA proportions are also included.

Figure 13 shows that the haploid and the fixed-map diploid schemes retain alternate alleles at higher-than-steady-state levels for essentially the same short period of time. This result is expected since the 1-dominates-0 scheme cannot hold 1s in abeyance. The haploid and fixed-map diploid schemes quickly discard the 1 allele. By contrast, the triallelic scheme retains the 1

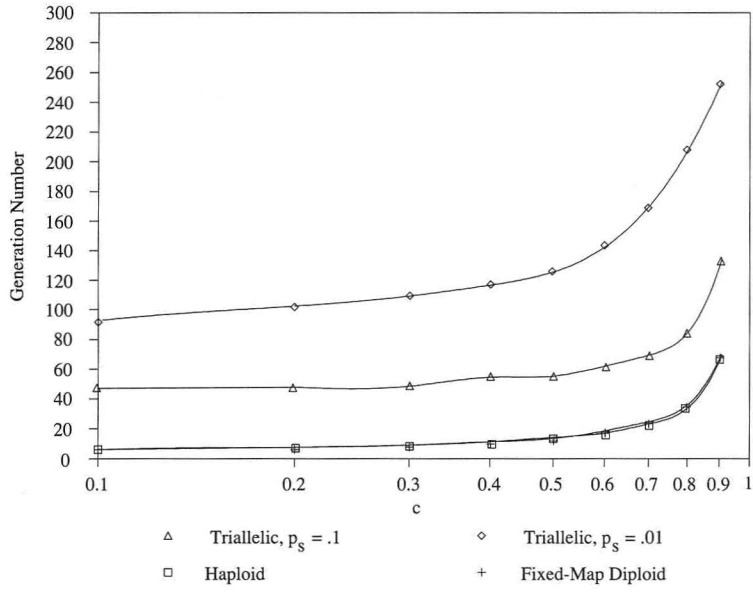

Figure 13: Approximate transient-times to steady-state proportions versus c (log scale) for $p_m = .001$. Spline curves through discrete data.

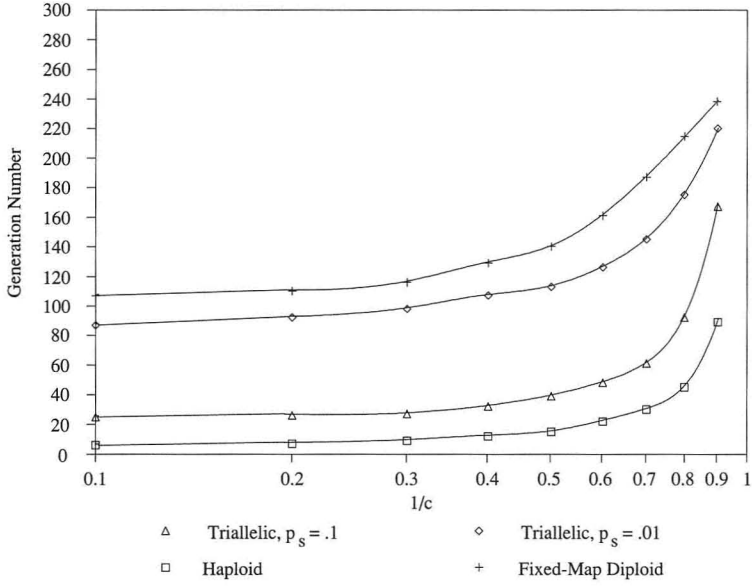

Figure 14: Approximate transient times to steady-state proportions versus $1/c$ (log scale) for $p_m = .001$. Spline curves through discrete data.

allele as an alternate at higher-than-steady-state levels for approximately 65 generations for $p_s = .1$, and approximately 125 generations for $p_s = .01$. The amount of time the triallelic system holds the alternative allele in memory increases rapidly as r approaches 1. Similar results are seen in figure 14 for $c > 1$. However, note that in this case the fixed-map scheme is able to maintain the proportion of recessives at a higher-than-steady-state level for more than 95 generations. This demonstrates how the diploid GA with no shift is able to maintain alternatives as a form of memory for extended numbers of generations. Also note that for large c the triallelic system maintains the 0-valued allele at above-steady-state proportions for a shorter period of time than a 1-valued allele under similar conditions (small c). The differences become less significant in the right-most portions of the graphs, where c is nearer to one. This difference between the two graphs is another indication of the triallelic representation's asymmetry. The 1-valued alternative alleles are held in the diploid GA's population noticeably longer than the 0-valued alternative allele in situations that should be similar.

8. Disruptiveness of maintaining diversity

As noted in previous sections, one of the roles that diploidy plays in improving GA performance on nonstationary problems is to increas the diversity in the population. Whether this diversity is randomly inserted (as in mutation) or is sensitive to the function's history (as in probabilistic memory through diploidy), the diversity introduced moderates selective pressures toward currently observed-best alleles. Therefore, this diversity, albeit necessary in nonstationary problems, is somewhat disruptive to the genetic search process. This section analytically examines the disruptiveness of mutation and dominance shift.

Consider applying only mutation operators (i.e., mutation and dominance shift) to a single-locus population. This locus can be any in a multi-locus population after reproduction and crossover have been applied. If the mutation operators were not applied, one could expect that a certain proportion of each allele value would be expressed. After mutation operators are applied, these expected proportions of expressed alleles are changed. A measure of the degree of population alteration associated with a mutation operator can be defined as the expected proportion of a single-locus population whose expected, expressed values are changed by the application of that operator. This measure will be called the operator's *disruptiveness.*

Consider the disruptiveness of mutation in a haploid GA. Before the application of mutation, the expected proportion of expressed 0-valued alleles is called P_0^t. After the mutation operator is applied, the new proportion (which is called P_0^{t+m}) is given by

$$P_0^{t+m} = P_0^t - P_0^t p_m + P_1^t p_m.$$

The disruptiveness of this operator is given by the absolute value of the

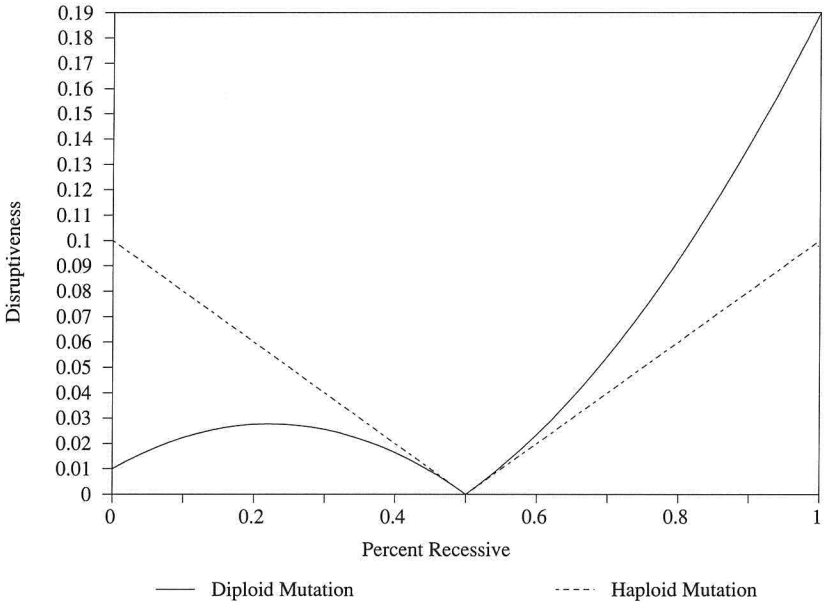

Figure 15: Disruptiveness of mutation versus proportion of recessive alleles for a haploid GA and a diploid GA with a fixed dominance map ($p_m = 0.1$).

difference between these proportions:

$$\Delta_m = |P_0^t - P_0^{t+m}| = |p_m(P_0^t - P_1^t)| = p_m|2P_0^t - 1|.$$

Similar measures can be developed for the dominance shift and mutation operators in the triallelic GA, but it is of some interest to first examine the disruptiveness of mutation in a diploid GA where the dominance map is fixed or has converged.

In a fixed-map diploid GA the expected proportion of the population having the recessive allele value expressed is P_r^2. Therefore, the disruptiveness of mutation is given by

$$\Delta_m = |(P_r^t)^2 - (P_r^{t+m})^2| = p_m|2[P_r^t - 2(P_r^t)^2] + p_m[1 + 4P_r^t(P_r^t - 1)]|.$$

Figure 15 shows the disruptiveness of mutation versus the proportion of recessives for both the haploid and diploid GAs. Mutation's disruptiveness on the haploid GA rises to its maximum value of p_m as the population converges to either allele. In the diploid GA, mutation's disruptiveness is at its maximum value of twice the mutation rate when the population converges to the recessive allele. However, if the dominance map has evolved correctly ($f_{r_e} > f_r$) the population converges to mostly dominant alleles. In this case, mutation's disruptiveness falls to p_m^2. This indicates that mutation should

lower population average fitness less in a converged diploid GA than in a converged haploid GA.

The disruptiveness of mutation in the triallelic GA can be found by noting that the expected proportion of expressed 0-valued alleles is given by

$$P^t_{0_e} = (P^t_0)^2 + 2P^t_0 P^t_{1_0}.$$

Therefore,

$$\begin{aligned}\Delta_m &= |(P^t_0)^2 + 2P^t_0 P^t_{1_0} - (P^{t+m}_0)^2 - 2P^{t+m}_0 P^{t+m}_{1_0}| \\ &= p_m|[P^t_0(2(P^t_1 - P^t_{1_0}) - P^t_0) + P^t_{1_0}(P^t_1 + P^t_{1_0})] \\ &\quad + p_m[P^t_1(P^t_1 - P^t_0) + P^t_{1_0}(P^t_0 - P^t_{1_0})]|.\end{aligned}$$

When the triallelic population has converged completely to the 0-valued allele, the disruptiveness of mutation is p_m. However, when the population converges to $P_1 = 1$, the disruptiveness in the triallelic GA is only p^2_m. This difference is another indication of the asymmetry in the triallelic dominance scheme. As indicated previously, it is also possible for the triallelic GA to converge to the following steady-state proportions:

$$\begin{aligned}P_1 &\approx P_{1_0} \approx 0.5, \\ P_0 &\approx 0.\end{aligned}$$

In this case the disruptiveness of the mutation operator is approximately p_m.

The disruptiveness of the dominance shift operator is given by

$$\Delta_s = p_s P^t_0 |P^t_1 - P^t_{1_0}|.$$

This value tends toward zero as the algorithm converges to any of the three steady-state conditions.

Figures 16 through 19 show plots for the iterative solution of the triallelic proportion equations and the accompanying measures of disruptiveness. For these figures $c = 2$ and $c = 0.5$ (respectively), and $p_s = p_m = .01$ for both graphs. These graphs show that as the triallelic GA converges, dominance shift becomes less disruptive than mutation. This indicates that dominance shift should lower population average fitness less than mutation in a converged triallelic GA. One can also conclude from these graphs that mutation and dominance shift should both tend to lower population average fitness less in the triallelic GA than mutation in the haploid GA. One could also conclude that the diploid GA could sustain much higher dominance shift rates than mutation rates for an acceptable limit on disruptiveness. The following section presents experiments that illustrate these observations.

9. Additional experiments with the triallelic GA

To illustrate the qualitative and analytical observations of the previous section, consider another version of the nonstationary knapsack problem used in previous experiments. In this problem, the weight constraint remains constant while the interpretation of certain bits is varied. In particular, the

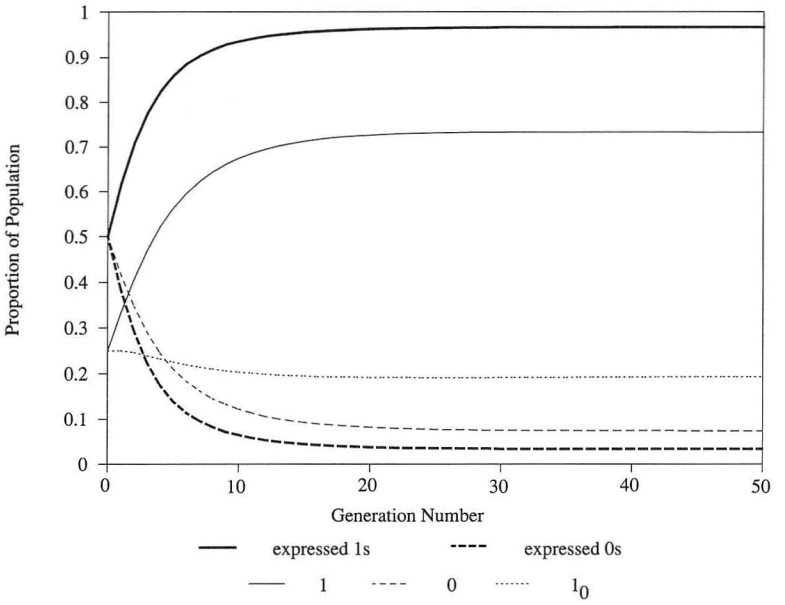

Figure 16: Proportions of alleles versus generation number for single-locus triallelic population with $c = 2$ ($p_m = p_s = 0.01$).

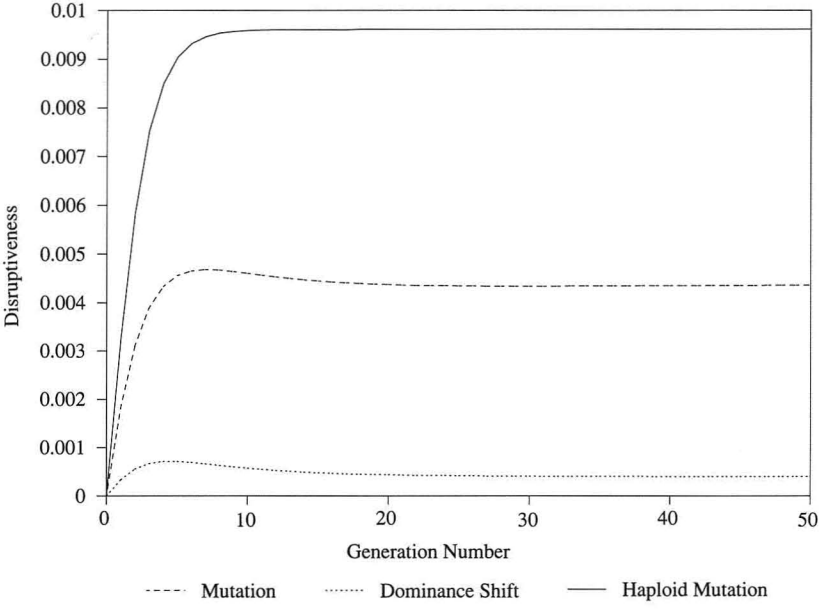

Figure 17: Disruptiveness of mutation and dominance shift associated with the proportions in figure 16.

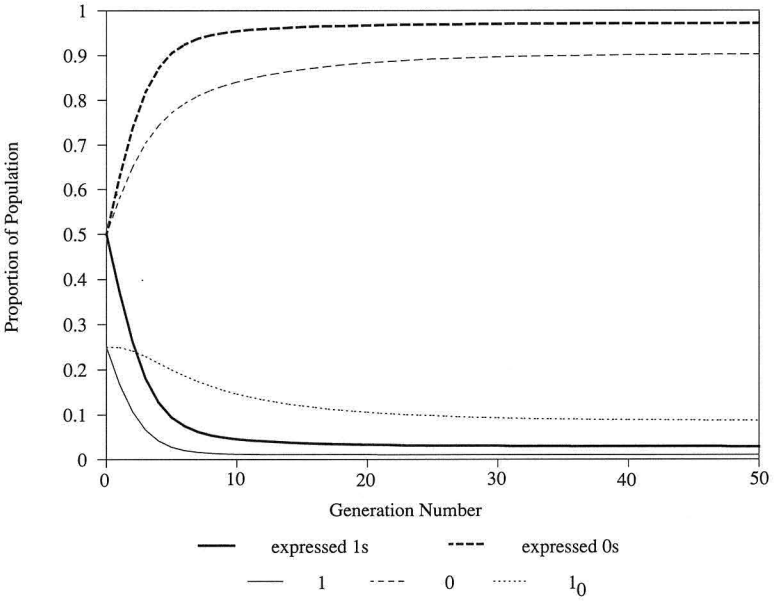

Figure 18: Proportions of alleles versus generation number for single-locus triallelic population with $c = 0.5$ ($p_m = p_s = 0.01$).

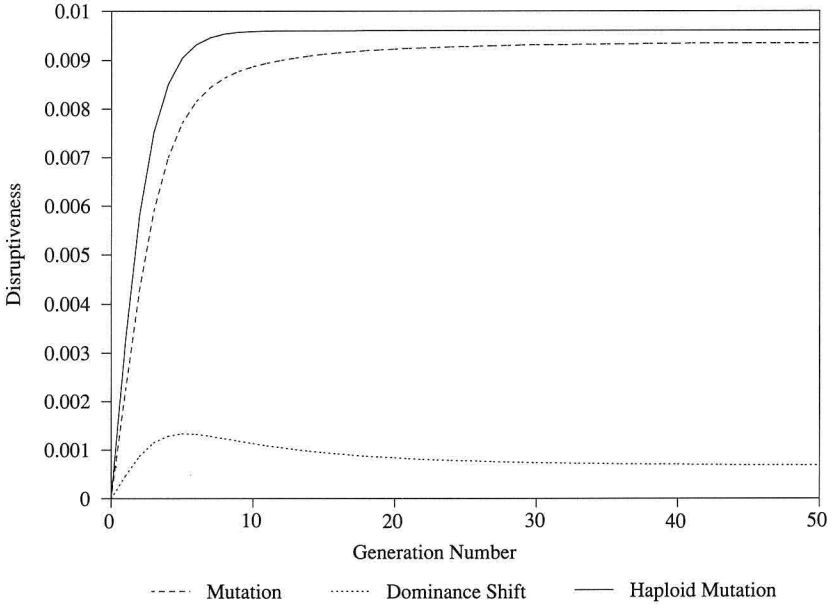

Figure 19: Disruptiveness of mutation and dominance shift associated with the proportions in figure 18.

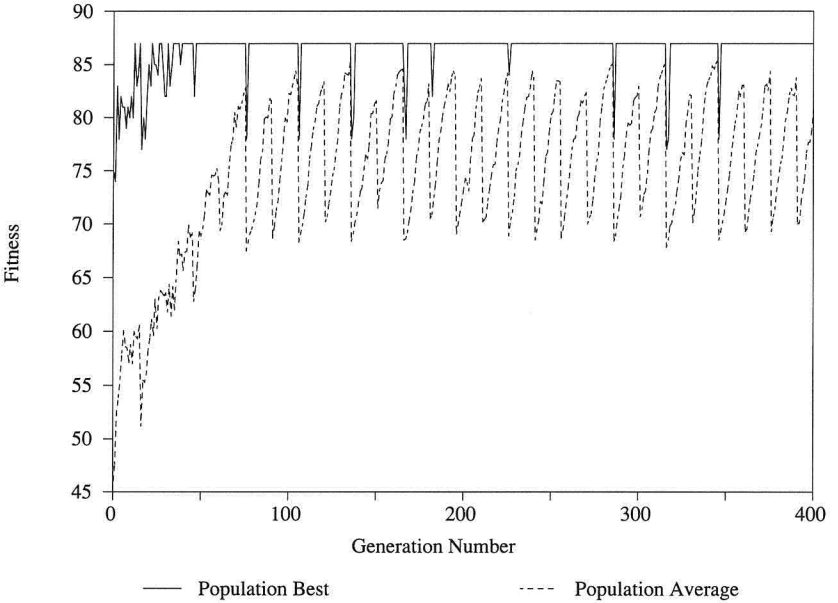

Figure 20: Results from the triallelic GA applied to a nonstationary knapsack problem with $p_s = 0.1$ and $p_m = 0.001$.

interpretation of the ith bit can be varied from meaning x_i to meaning x_i's complement. This form of the nonstationary knapsack problem is desirable since the relative selection pressure for each alternative bit remains constant, and the problem's variations are symmetrical. In the problem used here, the interpretation of bits 3 and 10 are switched every 15 cycles. Experiments were performed for various settings of p_m and p_s.

Figure 20 shows the results of an experiment with $p_s = 0.1$ and $p_m = 0.001$. Of those examined, these parameter settings gave the triallelic GA the greatest ability to track the nonstationary test problem. The high dominance shift rate allows the algorithm to recall the alternate alleles necessary to maintain an optimal solution in almost every generation of the run. The population average fitness also indicates a high degree of convergence in each cycle of the test problem. These results are consistent with the previous section's conclusion that dominance shift is less disruptive than mutation in the triallelic GA. Note that the peak of the average-fitness curve is slightly lower in one of the two function conditions. This is the condition where the nonstationary bits should be set to 1 in the optimal string, and it indicates the asymmetry of the triallelic representation discussed in previous sections. This confirms the results illustrated in figure 10.

It is also interesting to examine changes in the diversity of the population at bits that have stationary and nonstationary interpretations in this experiment. Figure 21 shows the comparative convergence of the expressed

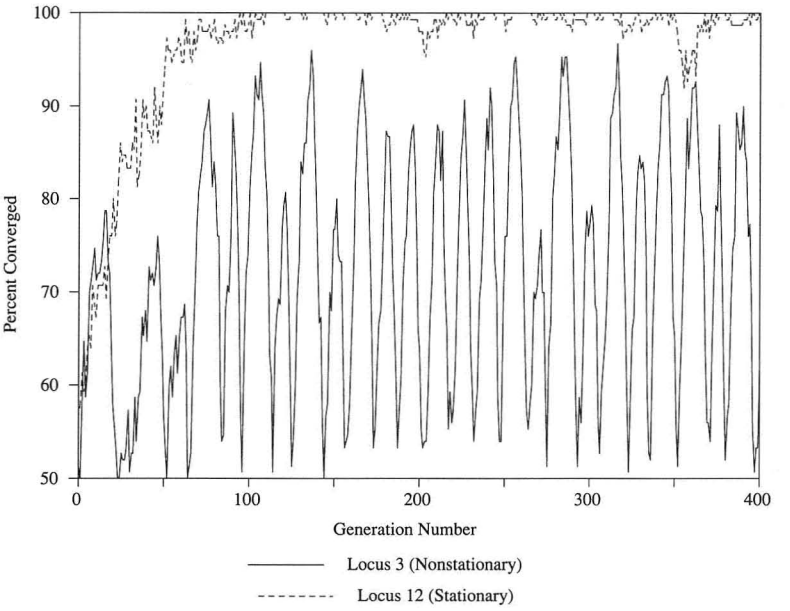

Figure 21: Expressed allele convergence at loci 3 and 12 from the experiment presented in Figure 20.

allele values versus generation number for loci 3 and 12 in this experiment. The plot shows the proportion of the most-expressed allele versus generation number. The population quickly converges with regard to locus 12, which has the same optimal allele value for both coding maps. The population oscillates with regard to the expressed value at locus 3, indicating the algorithm's response to this locus being nonstationary. Figure 22 shows a graph of the actual (rather than expressed) convergence at loci 3 and 12. The value plotted against the vertical axis is the highest of the proportions of the three alternative alleles. Note that locus 12 has converged to approximately equal proportions of 1s and 1_0s. Taken together, the previous three graphs show how the diploid GA maintains diversity at bits that have nonstationary values, which allows for tracking of the problem's variations while maintaining a high population average fitness.

For higher values of mutation and lower values of dominance shift, different behavior is observed. Figure 23 shows results from an experiment with a moderate dominance shift rate ($p_s = 0.001$) and a high mutation rate ($p_m = 0.01$). Although mutation promotes sufficient population diversity for the algorithm to maintain the optimal solution throughout the majority of the run, the random effects of mutation result in substantially less convergence of the algorithm due to lower population average fitness. This result is in agreement with the previous section's conclusion that mutation is more disruptive than dominance shift in the triallelic GA. Additional experiments

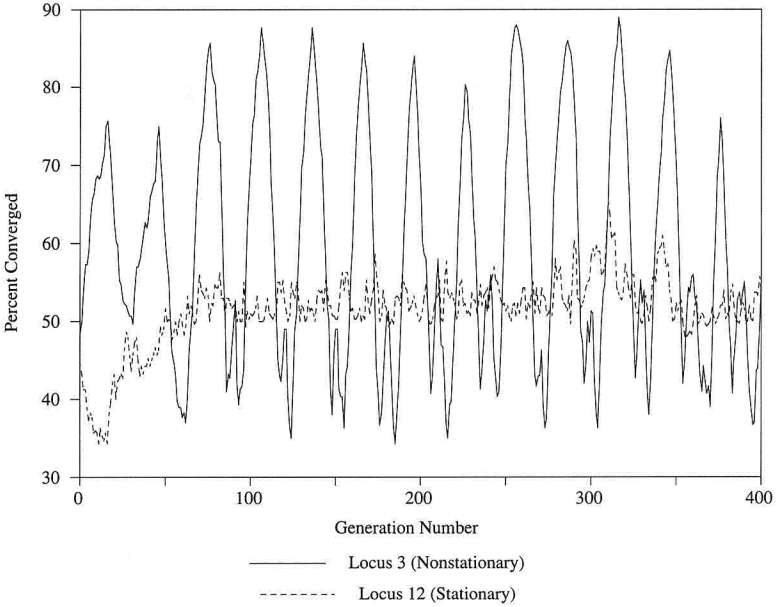

Figure 22: Actual allele convergence at loci 3 and 12 from the experiment presented in figure 20.

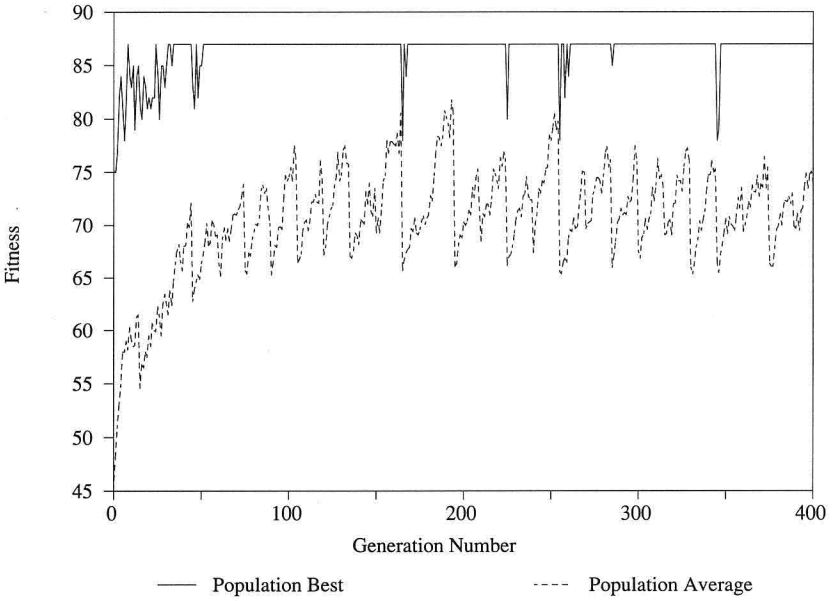

Figure 23: Results from the triallelic GA applied to a nonstationary knapsack problem with $p_s = 0.001$ and $p_m = 0.01$.

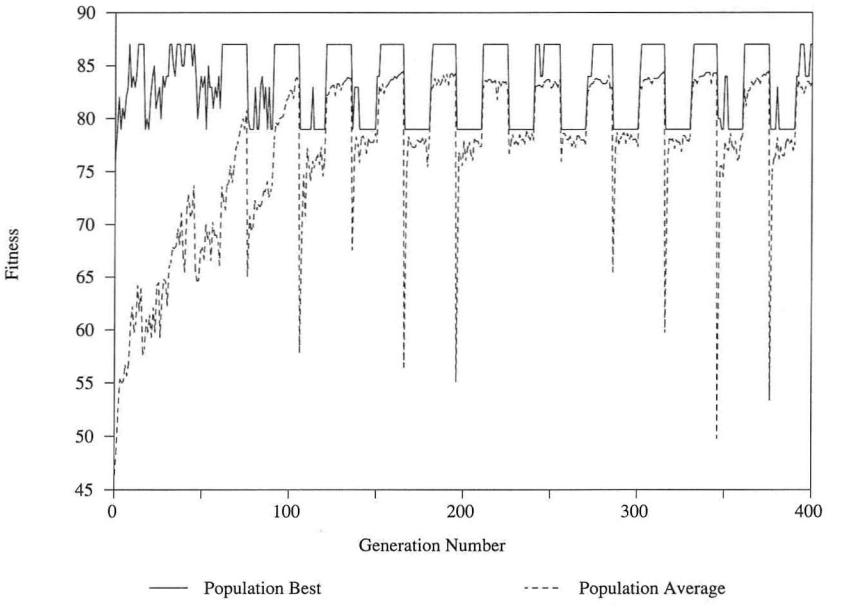

Figure 24: Results from the triallelic GA applied to a nonstationary knapsack problem that requires the recall of recessive building blocks.

are not included for the sake of brevity, but other combinations of p_s and p_m have shown that high dominance shift rates coupled with low mutation rates yield the most reliable tracking of nonstationary functions with various oscillation frequencies and high population average fitness (see Smith [21] for an account of additional experiments).

10. Extensions

Previous experiments indicate that recessive alleles induce non-disruptive population diversity that is sensitive to past environmental conditions. These recessive alleles can be recovered quickly through dominance shift. However, in some time-varying problems it may be necessary to simultaneously reveal two or more recessive alleles within the same population member. This condition will occur if two or more alleles that are affected by the problem environment's time variation have a bitwise-nonlinear (or *deceptive* [7]) effect. Figure 24 shows the results of the triallelic scheme applied to a nonstationary knapsack problem that incorporates a bitwise nonlinearity.[3] The GA is unsuccessful in tracking the problem's variations because the probability of the two necessary recessive alleles being simultaneously revealed and

[3]This problem is similar to that used in previous experiments, but in this case the interpretation of bits 7 and 8 are switched every 15 cycles. Due to a slight recoding, these bits represent objects 1 and 7 in table 1 [21].

evaluated is relatively low (on the order of p_s^2). An extension that may overcome this limitation is the addition of a dominance shift operator that acts with high probability on multiple, tightly linked recessive alleles (recessive building blocks).

One way to induce multi-locus dominance shift would be to insert extra bits in the problem encoding that directly control dominance at other loci. Note that the triallelic scheme incorporates genic control of dominance at a single loci to some extent, in that the subscript of the 1-valued alleles is a type of gene that controls dominance at these alleles. However, in this case the dominance-controlling gene effects only a single locus, to which it is permanently linked. A more general form of genic dominance control could allow for direct recombination of dominance relationships. Since selection and recombination are the primary elements of the traditional GA search process, this modification could significantly affect the diploid GA's performance. Multi-locus dominance under genic control could also allow for the abeyance and simultaneous revelation of recessive building blocks.

Genic control of dominance is similar to the idea of *intrachromosomal* dominance in haploid encodings. In an intrachromosomal dominance scheme, genes along a haploid chromosome can activate and deactivate the effects of other genes. One example of intrachromosomal dominance is in messy GAs (mGAs) [5, 10, 8, 9], where variable-length chromosomes are used separately to evaluate building blocks and overcome deception. In mGAs, chromosomes can become *overspecified*, such that alternate alleles for the same gene are in conflict. In such cases, left-most genes dominate. This *positional dominance* may have effects that are similar to those of diploid GAs with dominance. The analysis presented in this paper may also have analogues in intrachromosomal dominance schemes. Investigation of these issues is an important extension of this work.

In a study that eludes to extensions of dominance in GAs, Dasgupta and McGregor [4] suggested a GA technique that incorporates genic control of gene expression in overspecified, haploid GA encodings. Experiments demonstrate that this scheme is effective in tracking a nonstationary knapsack problem. However, because of restrictions on genetic operators, the experiments avoid the need to consider intrachromosomal dominance effects necessary for overspecified encodings. These restrictions also avoid the potential for underspecification that exists in this scheme. If overspecification is considered, this type of GA will require an intrachromosomal dominance scheme, perhaps similar to that used in the mGA. Examination of such schemes is an important area for further investigation.

Another important extension to this paper is the re-examination of diploid GAs applied to *stationary* search problems, where the inherent noise of the genetic search process is considered nonstationary. Recall that a GA searches by implicitly selecting building blocks on the basis of population average fitness values, and recombining. This process is clearly noisy since schema average fitness values vary from sample to sample, and thus from population to population. Goldberg and Rudnick [11] use the term *collateral noise* to

refer to this variance in schema average fitness values. How collateral noise can be viewed as nonstationary can be illustrated by example. Consider the competing schemata * * 1 and * * 0 in a population of three-bit individuals, or as three bits of a larger problem. Imagine that at some point in the GA search process, a population is dominated by individuals with schemata 0 1 * and 1 0 *, with few or no individuals with schemata 0 0 * and 1 1 *. Further imagine that, based on this population, schemata average fitness values indicate that selection should strongly favor * * 0 over * * 1, while there is fitness difference between 0 1 * and 0 1 *. After a number of generations, selection causes the population to be dominated by individuals with schema * * 0, while recombination re-biases the population toward a more equal representation of 0 0 *, 0 1 *, 1 0 *, and 1 1 *. Based on this re-biased population, one can imagine that variance in the fitness values could dictate selection strongly favoring * * 1 over * * 0. If the optimal solution is 1 1 1, the GA may have difficulty since early selective pressure eliminated 1-valued alleles in the third position. It is easy to imagine more complex situations in which the variance values of a number of schemata affect the search process in a similar way. Mutation could be used to maintain diversity such that a sufficient number of schemata with high variance are maintained despite early selective pressure. However, previous arguments have indicated that mutation is disruptive to the genetic search process (in fact, mutation can be seen as an additional source of noise). Clearly, the non-disruptive diversity introduced by diploidy and dominance may be useful in these situations. One can also envision situations in which schema variances and repeated biases in the population could yield an advantage for the probabilistic memory introduced by diploidy and dominance.

Inherent temporal variation in the genetic search process may also play a significant role in genetics-based learning systems like the *learning classifier system* [16] or certain genetically-structured neural networks [18]. In these systems, each population member's fitness can depend on the entire composition of the population, as in a natural ecology [22]. In such problems, changes in the population composition imply a nonstationary character. Thus, diploidy and dominance could prove useful.

11. Conclusions

Several conclusions can be drawn directly from the information presented in this paper. As suggested by Holland [14], diploidy allows for a higher level of steady-state diversity in a converged GA population for a given level of mutation. However, analytical and experimental evidence shows that a diploid GA maintains extra diversity at loci where alternative alleles were emphasized in the recent past. In effect, diploidy embodies a form of temporal memory that is distributed across the population. This added diversity is more effective than that induced by increased mutation because of its sensitivity to function history. Also, added diversity (whether due to abeyant recessives or added mutation) is less disruptive to the genetic search process

in a diploid GA than in a haploid GA. Therefore, a diploid GA not only requires a lower mutation rate for a diverse population, but can also sustain a higher rate of mutation without disrupting the search process.

Experiments show that an adaptive dominance map is necessary to effectively exploit the advantages of diploidy. Analysis of and experiments with the triallelic adaptive dominance scheme have shown the importance of a dominance shift operator in such schemes. The dominance shift rate is a control on probabilistic memory in a diploid GA, with recessive alleles retained longer and recalled less quickly for lower dominance shift rates. Experiments also show that the diploid GA can sustain rates of dominance shift that are much higher than those typically used for mutation.

The results presented in this paper have clear-cut implications for nonstationary versions of the ill-conditioned search problems for which haploid GAs would typically be employed. As previously noted, diploidy and dominance could also aid in stationary search problems that include noisy evaluations of schemata average fitness values, and in genetics-based learning systems that have an ecological character. In such problems, internal aspects of the genetic search process induce virtual nonstationarities that may be addressed effectively through diploidy and dominance. Thus, diploidy and dominance are a naturally inspired addition that may improve the performance of GAs in a broad class of search, optimization, and learning tasks.

12. Acknowledgments

The authors gratefully acknowledge support provided by NASA under Grant NGT–50224, and by the NSF under Grants CTS-8451610 and ECS-9022007.

References

[1] R. J. Berry, *Genetics* (London, English University Press, 1965).

[2] L. B. Booker, "Intelligent Behavior as an Adaptation to the Task Environment" (Doctoral dissertation, University of Michigan), *Dissertation Abstracts International*, **43**(2) (1982) 469B (University Microfilms No. 8214966); also Technical Report No. 243, Logic of Computers Group, University of Michigan.

[3] C. L. Bridges and D. E. Goldberg, "An Analysis of Reproduction and Crossover in a Binary-Coded Genetic Algorithm," *Proceedings of the Second International Conference on Genetic Algorithms*, (1987) 9–13.

[4] D. Dasgupta and D. R. McGregor, "A Structured Genetic Algorithm: The Model and First Results," Technical Report IKBS-2-91, Department of Computer Science, University of Strathclyde, Glasgow, Scotland (1991).

[5] K. Deb, "Binary and Floating-point Optimization Using Messy Genetic Algorithms" (Doctoral dissertation), TCGA Report No. 91004, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa (1991).

[6] B. E. Gillett, *Introduction to Operations Research* (New York, McGraw-Hill, 1976).

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Reading, MA, Addison-Wesley, 1989).

[8] D. E. Goldberg, K. Deb, and B. Korb, "An Investigation of Messy Genetic Algorithms," TCGA Report No. 90005, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa (1990).

[9] D. E. Goldberg and T. Kerzic, "mGA1.0: A Common LISP Implementation of a Messy Genetic Algorithm," TCGA Report No. 90004, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa (1990).

[10] D. E. Goldberg, B. Korb, and K. Deb, "Messy Genetic Algorithms: Motivation, Analysis, and First Results," *Complex Systems*, **3** (1989) 493–530.

[11] D. E. Goldberg and M. Rudnick, "Genetic Algorithms and the Variance of Fitness," *Complex Systems*, **5** (1991) 265–278.

[12] D. E. Goldberg and R. E. Smith, "AI meets OR: Blind, Inferential Search with Genetic Algorithms," TCGA Report No. 86002, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa (1986).

[13] D. E. Goldberg and R. E. Smith, "Nonstationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy," *Proceedings of the Second International Conference on Genetic Algorithms*, (1987) 59–68.

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

[15] J. H. Holland, "Genetic Algorithms and Classifier Systems: Foundations and Future Directions," *Proceedings of the Second International Conference on Genetic Algorithms*, (1987) 82–89.

[16] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard, *Induction: Processes of Inference, Learning, and Discovery* (Cambridge, MIT Press, 1986).

[17] R. B. Hollstien, "Artificial Genetic Adaptation in Computer Control Systems" (Doctoral dissertation, University of Michigan), *Dissertation Abstracts International*, **32**(3) (1971) 1510B (University Microfilms No. 71–23,773).

[18] H. Kargupta and R. E. Smith, "System Identification with Evolving Polynomial Networks," *Proceedings of the Fourth International Conference on Genetic Algorithms*, (1991) 370–376.

[19] E. Pettit and K. M. Swigger, "An Analysis of Genetic-based Pattern Tracking and Cognitive-based Component Tracking Models of Adaptation," *Proceedings of the National Conference on Artificial Intelligence*, (1983) 327–332.

[20] T. Sivapalan and D. E. Goldberg,, "The Two-armed Bandit Problem: A Bibliography 1952–Present," TCGA Report No. 87002, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa (1987).

[21] R. E. Smith, "An Investigation of Diploid Genetic Algorithms for Adaptive Search of Nonstationary Functions," TCGA Report No. 88001, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa (1988).

[22] R. E. Smith, S. Forrest, and A. Perelson, "Discovering and Maintaining Diversity in a Genetic Algorithm" (forthcoming).

[23] M. M. Syslo, N. Deo, and J. S. Kowalik, *Discrete Optimization Algorithms with Pascal Programs* (Englewood Cliffs, NJ, Prentice-Hall, 1983).