Conjugate Schema and Basis Representation of Crossover and Mutation Operators

Sanza T. Kazadi
Department of Computation
and Neural Systems
136-93 Moore Laboratories
Caltech
Pasadena, CA 91125
sanza@cco.caltech.edu

Abstract

In genetic search algorithms and optimization routines, the representation of the mutation and crossover operators are typically defaulted to the canonical basis. We show that this can be influential in the usefulness of the search algorithm. We then pose the question of how to find a basis for which the search algorithm is most useful. The conjugate schema is introduced as a general mathematical construct and is shown to separate a function into smaller dimensional functions whose sum is the original function. It is shown that conjugate schema, when used on a test suite of functions, improves the performance of the search algorithm on 10 out of 12 of these functions. Finally, a rigorous but abbreviated mathematical derivation is given in the appendices.

Keywords

Conjugate schema, representation, genetic algorithm.

1. Introduction

The traditional view of the genetic algorithm (GA) makes use of binary strings in the encoding of optimization problems (Holland, 1975). However, several researchers (Eshelman & Schaffer, 1993) have made use of real-encoded genetic algorithms. A *real-encoded genetic algorithm* (REGA) is a genetic algorithm in which the vectors are represented in the natural real vector notation. In this paradigm, the functions are as initially encoded, and they are acted upon by the standard operators of a genetic algorithm (mutation, crossover, and reproduction), where the actions are on components of the vector, analogously to the binary representation. This use of the genetic algorithm removes several of the limitations of the traditional Holland-style genetic algorithm. These limitations include the difficulty in obtaining precise results, and the inability to dynamically change a GA's representation. Although the use of the real encoding is liberating in some sense, the theoretical underpinning of this type of optimization algorithm is less developed.

In our studies of the real-encoded genetic algorithm, we came to realize that many of the main properties of the original model of the GA remained intact. The concepts of building blocks, schema, the building block hypothesis (Forrest & Mitchell, 1992); (Vose, 1991), and so forth all remain unchanged in REGA models. However, much of the energy spent studying this model has been devoted to creating various operators in the GA paradigm and testing their usefulness on the GA. These new operators perform well on several functions, but much work remains to be done before a true understanding of the genetic algorithm paradigm is built. In this paper, we seek a more complete understanding of the genetic algorithm and what it is doing. We seek to answer the question of why certain encodings, even in the more general framework of the REGA, are more successful in optimizing functions than others. In so doing, we seek to generate a general understanding of the genetic algorithm paradigm that will aid in building more useful simulations, and yield better results when undertaking optimizations. We seek also to answer the question "What is the *best representation* one might hope to obtain and use in implementing a GA, and how can we find such a best representation?"

In Section 2, we give an overview of other work on representational issues in genetic algorithms. In Section 3, we introduce our basic GA paradigm, which will become our test bed for our representational alterations. This paradigm is intended to mimic as closely as possible the binary basis GA (and in fact may indeed be used in a binary vector space). We significantly simplify the paradigm in order to focus attention on representational issues. In Section 4, we provide motivation for considering representational issues. In Section 5, we give an intuitive definition of the concept we call *conjugate schema* and provide motivation for their consideration. In Section 6, two approximation methods for calculating conjugate schema are presented. We show the behavior of both of these approximation methods on several two-dimensional examples. In Section 7, empirical evidence for the usefulness of conjugate schema is presented. In Section 8, a closer look is taken at how the basis variation affects the performance of the GA. Finally, in Section 9, we discuss other possible uses for this formalism and future research directions. A rigorous proof of several facts is given in the appendix. The appendix may be omitted without a loss of understanding of the central concepts.

2. Representation, Mutation, and Crossover

Though a complete understanding still eludes us, a partial understanding of the function of the genetic algorithm has emerged. Among the most important properties of genetic algorithms ranks their implicit parallelism. This property of a GA allows part of a population of a GA to investigate one portion of a vector, while another part of the population investigates another portion of the vector, concatenating the useful parts of these vectors to generate more desirable ones. The key to this ability is the crossover operator, and the crossover operator's behavior under linear transformation (Leipins & Vose, 1990).

The representation of a search space changes the behavior of the search algorithm being used through the operators' altered interaction with the vectors within. By altering the representation of the search space, one can modify the effect of both the mutation and crossover. Rana and Whitley (1997) introduce an exhaustive search of encodings for all possible functions of eight real numbers encoded with three bits per number, using Gray code string representation. They find that a great deal of variability exists in the behavior of extrema, and that by a careful choice of Gray code representation, they can change the number of extrema in the function. This greatly alters the effect of the mutation operator, indicating that the use of differing representations can have a large effect on the operators themselves.

Real-encoded genetic algorithms have the added flexibility of allowing one to change the representation of the operators in a genetic search space, and not simply the neighborhood structure. This allows one to use representations of vectors that may be more conducive to recombination of relevant disparate parts of the vector, without the concomitant damaging effects of crossover. Ono and Kobayashi (1997) implement a crossover operator designed to be independent of the representation of the vectors. This crossover creates children from parents which will be more "like" their parents because they are arranged in space along and nearby the line connecting the parents. This produces offspring that are more likely to be arranged along ridges or valleys produced by the function, a situation impossible for many of their test functions without the use of the invariant crossover operator. In four of five simulations, they report a marked improvement in performance. Moreover, in graphing the elements of the population that occur throughout the search, Ono and Kobayashi find that the new representation-invariant crossover operator (UNDX) is more successful in searching areas of importance than the previous model (BLX-α), and that members of the population in UNDX are more likely than those in BLX-α to be in areas that have high fitness values.

The building block hypothesis is an important notion for GAs. It seems important not only to cross over in a way that preserves "important building blocks," but also to create representations of vectors in which fundamental building blocks may be redundant. Wu and Lindsay (1996) comment on their use of a floating and overlapping representation of building blocks akin to those found in DNA chains. They find that the use of multiple building blocks allows them to have significantly better performance on the royal road function using a floating building block representation than a fixed building block representation. The use of this floating representation allowed building blocks to be duplicated, which reduced the disruptive effects of both crossover and mutation. Moreover, the floating representation allows larger noncoding regions, which seems to allow crossover to fail to disrupt existing building blocks more often, making even single copies more stable under crossover.

Work is still continuing on the use of specialized crossover operators and subspace partitioning. Arabas, Mulawka, and Pokraniewicz (1995) report on the use of adaptive crossover operators that perform better, although in some cases marginally so, on a test suite of functions. Smith and Fogarty (1995) report on the use of gene linking, or connecting different subsets of vector components. They find that gene linking improves the search through a multidimensional space. Tsutsui, Fujimoto, and Ghosh (1997) provide an example of subspace search, in which individual subspaces are searched to alleviate population convergence, providing partial solutions that may be recombined with other partial solutions.

In most of this work, the principal aim is to build an understanding of the subspace structure of a function and how to exploit it (Wright, 1991). As the crossover operator deals with this subspace structure directly, understanding of either one indicates an expanded understanding of the other.

3. The Use of Differing Mutation and Crossover Bases

Our aim in this paper is to understand the effect of the representation of the mutation and crossover bases in the effectiveness of the algorithm. As a first step, we consider a simple two-dimensional example that may be analytically solved.

Let the space be $\Re^2$ and the function be

$$f\left(\overrightarrow{v}\right) = v_1 v_2 \tag{1}$$

Now, the natural representation of this space, and indeed the simplest one, is one in which each of the components is the projection along the basis elements of the basis.

$$B = \{(1,0),(0,1)\} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2}$$

where we take the rows to be the basis elements. Now, let us consider the mutation of a vector $\vec{v} = (v_1, v_2)$ in this basis. Suppose that one element is increased by ϵ, a change equivalent to multiplying the first basis element by ϵ and adding the two vectors. The new function value is

$$f\left(\vec{v'}\right) = (v_1 + \epsilon) v_2 = f\left(\vec{v}\right) + \epsilon v_2 \tag{3}$$

However, if we note that the function may be written as

$$f\left(\vec{v}\right) = v_1 v_2 = \frac{1}{4}\left((v_1 + v_2)^2 - (v_1 - v_2)^2\right) \tag{4}$$

this leads us to consider the basis $\{\frac{1}{\sqrt{v_1^2+v_2^2}}(v_1, v_2), \frac{1}{\sqrt{v_1^2+v_2^2}}(v_1, -v_2)\}$, which becomes $\{\frac{1}{\sqrt{2}}(1,1), \frac{1}{\sqrt{2}}(1,-1)\}$ when $v_1 = v_2$. Now, if we alter the vector by adding a vector aligned with the first basis element, the new function value is

$$f\left(\vec{v''}\right) = \left(v_1 + \frac{\epsilon v_1}{\sqrt{v_1^2 + v_2^2}}\right)\left(v_2 + \frac{\epsilon v_2}{\sqrt{v_1^2 + v_2^2}}\right) = f\left(\vec{v}\right) + \frac{\left(2\epsilon + \epsilon^2\right) v_1 v_2}{\sqrt{v_1^2 + v_2^2}} \tag{5}$$

so that if

$$(2 + \epsilon) v_1 > \sqrt{v_1^2 + v_2^2} \tag{6}$$

the effect of the mutation will be greater than that of an improving step along either a positive or negative step in the direction of the basis element (1, 0) with a similar condition for (0, 1). Using the other basis element gives us a different result.

$$f\left(\vec{v''}\right) = \left(v_1 + \frac{\epsilon v_1}{\sqrt{v_1^2 + v_2^2}}\right)\left(v_2 - \frac{\epsilon v_2}{\sqrt{v_1^2 + v_2^2}}\right) = f\left(\vec{v}\right) - \frac{\epsilon^2 v_1 v_2}{\sqrt{v_1^2 + v_2^2}} \tag{7}$$

making $f(v'') > f(v')$ if $\epsilon < -1$. This would seem to imply that a mutation can be moulded into different forms, with each one providing differing effectivenesses in the optimization, depending on the direction in which the mutation occurs. The most advantageous mutation direction, however, is highly dependent on the position in the space in which the mutation is taking place. Thus, one might expect that there exist more advantageous bases for mutation than the canonical basis at most positions in the search space, though the identification of this basis may be difficult.

A similar analysis can be undertaken for the crossover basis. Let us first consider the effect of a crossover operator. Suppose that we have a basis $B_M = \{\hat{e}_1, \ldots, \hat{e}_N\}$ given by

$$\hat{e}_1 = (1,0,0,\ldots,0), \hat{e}_2 = (0,1,0,\ldots,0), \ldots, \hat{e}_N = (0,0,0,\ldots,1)$$

and two vectors v and u for which we want to carry out a crossover. This is the canonical basis, and may be represented as

$$B = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{8}$$

where we take the rows to represent basis vectors. A standard crossover operation takes one of the projections of the vector along the basis element of the canonical basis and switches it. That is, if $\overrightarrow{w} = (w_1, \ldots, w_k, \ldots, w_N)$ and $\overrightarrow{u} = (u_1, \ldots, u_k, \ldots, u_N)$, then the crossover of the two vectors using the kth basis element will produce the vectors $\overrightarrow{w'} = (w_1, \ldots, u_k, \ldots, w_N)$ and $\overrightarrow{u'} = (u_1, \ldots, w_k, \ldots, u_N)$. Single point and two-point crossover may be built using combinations of simple crossovers built in this way.

We may write this crossover as

$$\overrightarrow{w'} = \overrightarrow{w} - \left(\overrightarrow{w} \cdot \hat{e}_k\right)\hat{e}_k + \left(\overrightarrow{u} \cdot \hat{e}_k\right)\hat{e}_k = (w_1, \ldots, u_k, \ldots, w_N) \tag{9}$$

$$\overrightarrow{u'} = \overrightarrow{u} - \left(\overrightarrow{u} \cdot \hat{e}_k\right)\hat{e}_k + \left(\overrightarrow{w} \cdot \hat{e}_k\right)\hat{e}_k = (u_1, \ldots, w_k, \ldots, u_N) \tag{10}$$

In this case, changing the basis will effectively change the crossover, and it is perfectly clear what is meant by altering the basis. Now, consider the problem that we mentioned above. Consider the crossover of two vectors done in the canonical basis. Take, for instance $(5, 6)$ and $(4, 3)$, which have the function values 30 and 12, respectively. Then, the new vectors are $(5, 3)$ and $(4, 6)$ with function values 15 and 24, respectively. What has happened is that the crossover operator has disrupted the connection between the vector components during recombination.

Suppose that we use the basis

$$B_C = \left\{ \frac{1}{\sqrt{v_1^2 + v_2^2}}(v_1, v_2), \frac{1}{\sqrt{v_1^2 + v_2^2}}(v_1, -v_2) \right\} \tag{11}$$

which is a basis as long as neither v_1 nor v_2 is zero. We take the center of our population to be the mean of the coordinates, $(4.5, 4.5)$ in this case. At this point, our new basis is

$$B_C = \left\{ \frac{1}{\sqrt{2}}(1, 1), \frac{1}{\sqrt{2}}(1, -1) \right\} \tag{12}$$

Then, if the crossover is undertaken in the first basis element, we have new vectors given by

$$\begin{aligned}
\overrightarrow{w'} &= \overrightarrow{w} + \left[\frac{(v_1, v_2)}{\sqrt{v_1^2 + v_2^2}} \cdot \left(\overrightarrow{u} - \overrightarrow{w}\right) \right] (v_1, v_2) \\
&= (5, 6) - \frac{11}{\sqrt{2}}(1, 1) + \frac{7}{\sqrt{2}}(1, 1) \\
&= (2.1716, 3.1716)
\end{aligned} \tag{13}$$

$$\begin{aligned}
\overrightarrow{u'} &= \overrightarrow{u} + \left[\frac{(v_1, v_2)}{\sqrt{v_1^2 + v_2^2}} \cdot \left(\overrightarrow{w} - \overrightarrow{u}\right) \right] (v_1, v_2) \\
&= (5, 6) - \frac{7}{\sqrt{2}}(1, 1) + \frac{11}{\sqrt{2}}(1, 1) \\
&= (6.8284, 5.8284)
\end{aligned} \tag{14}$$

The function values are 6.8873 and 39.799, respectively. This computation shows that, instead of radically disturbing the link between vector components, this basis combines vector components in a significantly less damaging way than the previous method. With

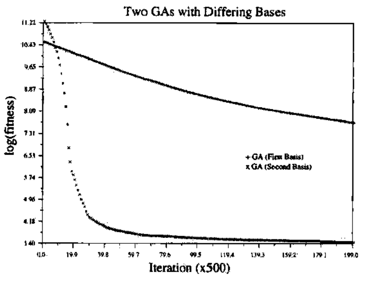


Figure 1. This figure shows the differing performance of two genetic algorithm paradigms. The differences between paradigms are their representations. The striking performance variation manifests itself in the convergence rate and final value.

the other basis element, we obtain the vectors (6.4142, 7.4142) and (2.5858, 1.5858) with function values of 47.556 and 4.1006, respectively.

Let us consider for a moment why we might want to change to this basis as opposed to using the canonical basis. In two out of four crossover products, a function value is produced that is significantly higher than the previous, while the other two are significantly lower. This means that, if we use a proportional reproduction, the new vectors are easily sorted according to their fitness, leaving only the higher scoring vectors and removing the lower scoring vectors. Moreover, the new values are significantly better than the previous ones, indicating a successful crossover operation.

One notable fact appears in this analysis. The effects on the operators are quite different. The effect on the mutation operator is bounded by the use of the step size. Thus, if ε is small enough, the effect on the mutation operator is negligible. On the other hand, the effect on the crossover operator is independent of such parameters and exhibits its effect only in the combinatorial construction of new vectors from existing vectors. We can, therefore, expect the greatest effect of the representation to occur when we are making use of crossover in combinatorially building up vectors. Moreover, we expect the effect on mutation only to come in when precise results are required and the step size is practically limited to be above a particular value.

As an illustration, we include here the effect of three modified representations on "retarded" (more formally defined in Section 5) GA simulations. We test two representations on the function taken from Michalewicz and Schoenauer (1996):

$$\begin{aligned} G7\left(\overrightarrow{v}\right) &= v_1^2 + v_2^2 + v_1 v_2 - 14v_1 - 16v_2 + (v_3 - 10)^2 \\ &\quad + 4(v_4 - 5)^2 + (v_5 - 3)^2 + 2(v_6 - 1)^2 + 5v_7^2 \\ &\quad + 7(v_8 - 11)^2 + 2(v_9 - 10)^2 + (v_{10} - 7)^2 + 45 \end{aligned} \tag{15}$$

This function is borrowed from a bounded optimization problem. We have not included the bounding relations here, as they may be obtained from Michalewicz and Schoenauer (1996). The key idea in Figure 1 is that the mutations and crossovers in one basis may be more or less effective than in another basis. Thus, it is important to understand what a 'good' basis is for each of these operations, and how it can be found. These bases may not be overlapping, however. There is no a priori reason to believe that a (better) basis for crossover will be better when used for mutations, and, in general, we have found this reasoning confirmed. We have found that multiple mutation bases are more often helpful, but specific crossover bases are more advantageous. This difference would seem to stem from the observation that building blocks should be representation invariant at a particular point in space (Battle & Vose, 1993). However, mutations often require weighted alterations along differing directions in the basis being employed to make advantageous steps. Thus, the way in which these concerted alterations are weighted will greatly affect performance, and it is generally reasonable to expect that multiple bases will be able to take advantage of many differing groupings.

4. The Conjugate Schema

Our primary motivation in searching for more effective bases is to remove the damaging effects of crossover. Typically in crossover, there is a tendency to pick groups of vector components without considering the effect of that particular group on the vector as a whole. There is no way that one will know ahead of time that one group of vector components is indeed the set that is providing good function values, and so one typically opts to choose crossover components at random, using a variety of methods.

Our goal, then, is to find bases that partially or wholly remove this uncertainty, so that one might carry out combinatorial trials in the most effective way. For instance, if we knew ahead of time that three basis elements out of N were functionally dependent, but that they were independent of all other basis elements, this would put strong constraints on the form of the function, and indeed its solution. It would indicate that we could carry out a separate optimization on this set of basis elements and simply add that to the rest of the components. In the author's opinion, this is the spirit behind the Schema Theorem. Thus, our goal is to find bases in which there are a maximal number of such subgroups, and to identify those groups of basis elements.

Before proceeding more formally, we consider another point. We have noted in Section 3 that we would like to vary the mutation basis during the optimization. The reason is that the use of varying mutation bases can lead search in different directions, connecting mutations along differing basis elements differently than in previous steps. This type of connection may prove to be quite important to us. Using the conjugate schema basis is one way to have this variation, at no extra computational cost. Since the conjugate schema basis is typically changing, we may simply use it to carry out mutations. This will tend to restrict mutations to occur within single subsets, restricting the mutation to a smaller-dimensional subspace. Such a restriction effectively reduces the dimension of the search by removing mutations across larger-dimensional and functionally independent blocks. Thus, the probability of achieving a good step would tend to be higher than if the whole vector were employed. This would tend to be an important consideration when using functions for which *concerted mutations* of different vector components was important (Page & Hong, 1998).

Let us restate our goal with more rigor. First, we assume that f is a function that maps a vector space to the real numbers. If $\vec{v} = (v_1, \ldots, v_n)$ and we can write f as

$$f = g + h \tag{16}$$

where

$$g(v_1, \ldots, v_n) = g(v_1, \ldots, v_k, *, \ldots, *)$$

and

$$h(v_1, \ldots, v_n) = h(*, \ldots, *, v_{k+1}, \ldots, v_n)$$

(where $*$ means that the element is irrelevant), then the function is separable, and the dimension of the problem has been reduced (provided that k is not zero). If this is the case, then we may optimize h and g, and the building block hypothesis tells us that the solutions of these optimizations may be combined for a single solution of the larger problem.

Let us now define some terms. Given a vector space Γ, if two basis elements $\overrightarrow{b_i}$ and $\overrightarrow{b_j}$ have the property that a small variation of a given vector $\overrightarrow{v}$ along $\overrightarrow{b_i}$ produces a variation of $f(\overrightarrow{v}+\delta\overrightarrow{b_i})$ that is independent of another variation along $\overrightarrow{b_j}$, then we call the two *functionally independent*. If there is a small dependence within a hypersphere of radius δ, we call them *δ-functionally independent*. That is, if

$$\left|f\left(\overrightarrow{v}\right)+f\left(\overrightarrow{v}+\delta\left(\overrightarrow{b_i}+\overrightarrow{b_j}\right)\right)-f\left(\overrightarrow{v}+\delta\overrightarrow{b_i}\right)-f\left(\overrightarrow{v}+\delta\overrightarrow{b_j}\right)\right|=0$$

these definitions hold, with similar conditions for approximations to functional independence.

We may restate our goal in the following way. *We seek bases in which the functional dependence of the separate basis elements is minimal.* That is, we wish to find bases in which the elements may be partitioned into separate groups of elements that have the property that each basis element in one group is functionally independent of any basis element in another group. This allows us to rewrite the function as a sum of smaller-dimensional functions, each acting only on the subspace of the larger-dimensional space spanned by the basis elements in a given group.

Let us define *conjugate schema* as disjoint subsets of a basis in which any two basis elements chosen from different subsets are functionally independent. Clearly, conjugate schema are not generally global structures. In general, conjugate schema are position-dependent structures and vary with the position in the vector space. Moreover, conjugate schema may be defined differently depending on the degree of locality used. That is, one may compute conjugate schema by using infinitesimally small steps, or one may compute conjugate schema by using extended averages over space. We shall see that this will provide differing degrees of structure.

Conjugate schema provide one with information about the separability (if locally) of a function. Occasionally, such information may be global. This information may be exploited in two ways. The most important way is by crossover. Once conjugate schema have been identified, these portions of vectors may be crossed over with a minimal disruption in the information upon recombination. Making use of the conjugate schema basis minimizes the size of these groups of vector components, making the effective dimensionality of search in these subspaces minimal. The second way is by mutation. Mutations can be restricted to occur within conjugate schema subspaces, again reducing the dimensionality of the problem. Empirically, we have found that the crossover is much more important in this respect than the mutation, though such a restriction of mutation operators does have a large effect in some cases, as in protein design and folding in the HP heteropolymer model.

One possibility for the enhancement of conjugate schema-based methods is to include the gradient vector in the basis used. Such an inclusion would make extremely favorable mutations available via this mutation basis element. However, this is not the goal of the conjugate schema, nor schema in general. The Schema Theorem refers to combinatorial reduction via the use of schema that partition the search space into separate parts which may be independently optimized. These parts are more robustly recombined, and mutations can be restricted to occur within the conjugate schema, effectively reducing dimensionality.

Because of these considerations, the gradient vector is not a natural consequence of the search for conjugate schema. We do not exclude the utility of using the gradient vector for carrying out some mutations. We note only that it is not generally true that a gradient vector will lend itself to functional separability.

Let us consider again our trivial example function $f(\vec{v}) = v_1 v_2$. We already noted that the function may be rewritten as

$$f\left(\vec{v}\right) = v_1 v_2 = \frac{1}{4}\left((v_1 + v_2)^2 - (v_1 - v_2)^2\right) \tag{17}$$

We saw in Section 3 that the transition to another basis yielded a more effective crossover. Now, if our basis takes on the form

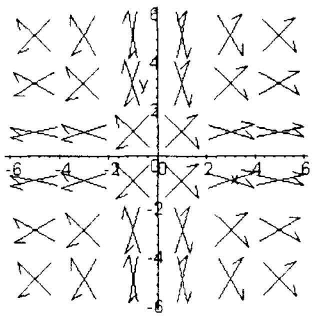
$$B_G = \frac{1}{\sqrt{v_1^2 + v_2^2}} \begin{pmatrix} v_1 & v_2 \\ v_1 & -v_2 \end{pmatrix} \tag{18}$$

and we let $u = \frac{(v_1+v_2)}{\sqrt{v_1^2+v_2^2}}$, $w = \frac{(v_1-v_2)}{\sqrt{v_1^2+v_2^2}}$, then $f(\vec{v}) = g(u, w) = h(u) + k(w)$, where h and k are appropriately defined. This is precisely the result we were after when we defined the conjugate schema. We identify the sets $\{u\}$ and $\{w\}$ as conjugate schema and note that these are global. The use of these conjugate schema in the mutation basis allows us to optimize one function separately from the other. Indeed, k is always nonpositive, and this forces us to choose its optimum as zero, a value it reaches along the diagonal only. This type of insight can be quite useful when considering bounded optimization problems, as we do in Section 7.

If we graph the conjugate schema over the surface, we find that they are indeed variable, but do not coincide with the gradient, as expected. This is not surprising, as conjugate schema give us information about the local separability of the space, and not the optimal mutation direction. There are instances when one of the conjugate schema basis elements will overlap with the gradient, but this simply states that the direction of maximal improvement lies entirely within one conjugate schema, not a surprising or unexpected occurrence. One thing that the conjugate schema do provide, however, is information about how to obtain the maximal vector that is less susceptible to boundaries. If one were to imagine a circular boundary superimposed over the graphs in Figure 2, we would see that the use of the gradient at the boundary would not help us to overcome the boundary, but the use of one or both conjugate schema basis elements would.

As indicated above, the mutation basis is often most effective when one does not make use of a single mutation basis. The conjugate schema basis found by solving for the eigenvectors of the absolute Hessian, defined in the appendices, provide a convenient way of continually modifying the mutation basis, providing a basis that will readily change throughout the search. The crossover basis, B_C, can be found computationally in two steps: (1) Find the eigenvectors of the absolute Hessian matrix, or the matrix formed by computing the Hessian matrix and take the absolute value of each of its components. (2) Multiply each row of the eigenvector matrix by the current position of the population in the space and normalize. We show in the appendices that this is precisely the matrix that yields the conjugate schema for continuous problems, but that these conjugate schema are local. That is, they may change as the position of the population, or the algebraic mean of the vectors in the population, in our solution space changes. However, an analytic expression for these conjugate schema allows us to, at every step of the simulation, have an accurate 'guide' through the fitness landscape.

We may begin to understand how to characterize the various GA operators. If crossovers are limited to crossovers in conjugate schema, then we will be *combining building blocks*



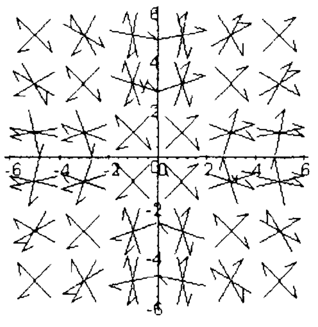


Figure 2. In these graphs, we have the conjugate schema (left) and the conjugate schema with the superimposed gradient vector (right). The fact that there are three unique vectors at most points, and that each vector is different at different points in the space, illustrates the locality of the conjugate schema structure and also that this structure does not, in general, include the gradient, although, in some cases, the gradient is coincidentally identical to one of the conjugate schema basis elements.

during crossover. It can be shown (Kazadi, 1997) (and we do so in the appendix) that the existence of conjugate schema implies separability, as given above, and so any crossover operator built in this way will be 'sorting out' the schema. The reproduction operator will remove those elements from the population that have low function values. Together with the crossover operator, this produces a sort-purge algorithm. Thus, by paying attention to how these crossovers are accomplished, we will develop an efficient sorting algorithm, particularly in, but not limited to, large populations. We return to this characterization of operators in Section 6.

5. Approximations to the Conjugate Schema

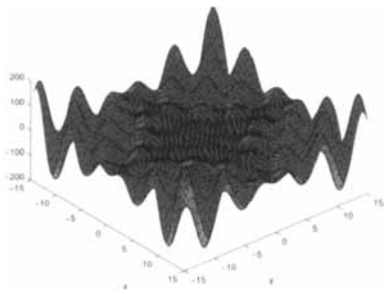
The conjugate schema is a mathematical construct formed by the use of differential relations, and is thus, by definition, local and variable. In many cases, the conjugate schema cannot be generalized to more than neighborhoods. Other times, an analytical expression for the absolute Hessian and its eigenvector matrix is not available. If this is the case, then one must create methods for estimating the conjugate schema numerically. Once this is done, the search may resume, and should be more effective than its predecessor. Suppose that the function that we have chosen is indeed analytical. For instance, suppose that we have a function

$$f(x,y) = xy\sin(x)\sin(y) \tag{19}$$

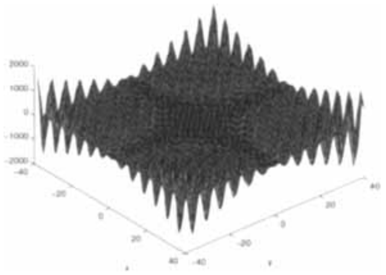
whose graph is shown in Graph 1. This function has the Hessian

$$H = \begin{pmatrix} y\sin(y)(2\cos(x) - x\sin(x)) & (\sin(y) + y\cos(y))(\sin(x) + x\cos(x)) \\ (\sin(y) + y\cos(y))(\sin(x) + x\cos(x)) & x\sin(x)(2\cos(y) - y\sin(y)) \end{pmatrix} \tag{20}$$

whose eigenvectors are wildly varying and are analytically difficult to write out. In order to tackle this problem, one might try to solve the general eigenvector problem analytically, yielding fast computational solutions. However, this is impossible to do in large-dimensional cases, as the algebra quickly becomes intractable, even with the aid of symbolic computer programs.



Graph 1. Graph of function $f(x,y) = xy \sin(x) \sin(y)$.



Graph 2. Larger-scale graph of function $f(x,y) = xy \sin(x) \sin(y)$.

Consider two graphs (Graphs 1 and 2) of equation (19), in which Graph 2 shows more global structure than Graph 1. This second plot has global behavior that might be exploited, but will be obscured by considering only local conjugate schema. In this section, we discuss two methods by which one might build suitable bases that have subbases that approximate conjugate schema. In our study, we implement the latter of these two methods.

5.1 Numerical Hessian

The first method is a straightforward extension of the theory of conjugate schema. In the theory of conjugate schema, one seeks to minimize the functional

$$D(f,B) = \sum_{i \neq j} \left| f\left(\vec{x_0}\right) + f\left(\vec{x_0} + \epsilon_i \vec{b_i} + \epsilon_j \vec{b_j}\right) - f\left(\vec{x_0} + \epsilon_j \vec{b_j}\right) - f\left(\vec{x_0} + \epsilon_i \vec{b_i}\right) \right| \quad (21)$$

under changes of the basis. We have noted, and shall prove in the appendix, that the absolute Hessian matrix achieves this desired goal in a small hypersphere around a point in the n-dimensional space over which the function is defined. Each differential operator of the absolute Hessian may be sampled numerically, yielding an approximate value for the absolute Hessian. Once this has been done, one may solve for the eigenvectors of the matrix.

This method is quite straightforward, but still has some problems. Discretization introduces problems with finding the 'true' value of the absolute Hessian. One must juggle the questions of how small the step must be made when calculating the differentials. The step size determines the accuracy with which the Hessian is calculated. Because of this relationship, and the variation of the function in space, the decision on a step size might have great repercussions on the conjugate schema eventually found.

In our simulations, we define a step size and a "smudge" parameter σ. The value of any particular point is estimated as the average over a hypersphere of radius σ. By increasing σ, we can choose the level of locality our Hessian deals with. We estimate our Hessians using 50 points. This is a large number of estimations, but the estimate may be done with less accuracy with as few as five. These estimations may be done by simply using new vectors created in the population, thus obviating the need for any new fitness calculations.

Consider again the function

$$f(x, y) = xy \sin(x) \sin(y)$$

with the Hessian as given above. If we take $(x, y) = (1, 1)$, the analytic Hessian becomes

$$H = \begin{pmatrix} 0.20122 & 1.9093 \\ 1.9093 & 0.20122 \end{pmatrix} \tag{22}$$

Numerically, we estimate the Hessian as

$$H = \begin{pmatrix} 0.303044 & 2.043232 \\ 1.863924 & 0.134599 \end{pmatrix} \tag{23}$$

using a step size of 0.1 and a "smudge" parameter of 0.01. The eigenvectors of the analytic basis are

$$B = \left(\begin{pmatrix} 0.70711 \\ -0.70711 \end{pmatrix}, \begin{pmatrix} 0.70711 \\ 0.70711 \end{pmatrix} \right) \tag{24}$$

whereas those of the estimated are

$$B = \left(\begin{pmatrix} 0.70883 \\ -0.70687 \end{pmatrix}, \begin{pmatrix} 0.73785 \\ 0.67497 \end{pmatrix} \right) \tag{25}$$

These agree very closely, considering our purposes for them. If the step size becomes 5.0 with a "smudge" parameter of 1.0, the absolute Hessian becomes

$$H = \begin{pmatrix} 0.205042 & 0.273825 \\ 0.254801 & 0.217083 \end{pmatrix} \tag{26}$$

with eigenvector basis

$$B = \left(\begin{pmatrix} 0.71222 \\ -0.72757 \end{pmatrix}, \begin{pmatrix} 0.70288 \\ 0.68603 \end{pmatrix} \right) \tag{27}$$

Note that the eigenvector basis seems to be quite resilient to changes in the absolute Hessian estimation. These bases mimic that for the function $f(x, y) = xy$, thus apparently overcoming the difficulty introduced by including the sine functions. Although local estimations involving small step sizes and "smudge" parameters seem to estimate the Hessian well, larger values do not, in general. For instance at (2, 1), the eigenvector matrix is

$$B = \left(\begin{pmatrix} -0.99826 \\ -5.8943 \times 10^{-2} \end{pmatrix}, \begin{pmatrix} -5.8943 \times 10^{-2} \\ -0.99826 \end{pmatrix} \right) \tag{28}$$

while estimations using a step size of 0.1 and "smudge" parameter of 0.01 give

$$B = \left(\begin{pmatrix} 0.99359 \\ 0.11301 \end{pmatrix}, \begin{pmatrix} -8.2178 \times 10^{-2} \\ 0.9971 \end{pmatrix} \right) \tag{29}$$

which yields a close match. However, using a step size of 5.0 and "smudge" parameter of 1.0 give

$$B = \left(\begin{pmatrix} 0.84015 \\ -0.54236 \end{pmatrix}, \begin{pmatrix} -0.54186 \\ -0.84047 \end{pmatrix} \right) \tag{30}$$

which is likely a result of the differing global and local behavior, as these seem to more closely agree with the Hessian of $f(x,y) = xy$ at this point.

This estimation technique is a rather straightforward and reasonably robust method that requires minimal (quadratic) computation at sparse intervals. However, the computation of eigenvectors is problematic, and dense for large-dimensional problems, though simplifications exist for symmetric matrices, as the absolute Hessian is. This technique is not implemented here, though it may be in future studies. It would appear to be a useful technique that is capable of taking advantage of the information continually made available through mutation and crossover in the population.

5.2 Basis Rotation

An alternative method that is somewhat more computationally dense is called *basis rotation*, and the resulting algorithm is called a *rotated-basis genetic algorithm* (RBGA). In this scheme, one creates an initial basis and 'rotates' it iteratively to generate a new basis that has a smaller functional dependence between basis elements. Suppose that we begin with a basis

$$B = \{\hat{e}_1, \ldots, \hat{e}_n\} \tag{31}$$

First, the two basis elements $\hat{e}_1$ and $\hat{e}_2$ are chosen. Two new basis elements

$$\begin{aligned} \widehat{e'}_1 &= \hat{e}_1 \cos(\theta) + \hat{e}_2 \sin(\theta) \\ \widehat{e'}_2 &= \hat{e}_2 \cos(\theta) - \hat{e}_1 \sin(\theta) \end{aligned} \tag{32}$$

are generated such that they minimize the functional D, given in Equation (21). The next two elements, $\widehat{e'}_2$ and $\hat{e}_3$ are then rotated, generating new elements $\widehat{e''}_2$ and $\widehat{e'}_3$. The process is continued until the last element is reached, and repeated until the basis converges.

That this method will converge can be seen by the following considerations. At every step, the value of D is decreased, yet it is bounded below by 0. Thus, the values of D form a Cauchy sequence and must converge. That it converges to a unique basis may only be argued if all pairs of basis elements are included in the basis rotation, a process that is more computationally dense than that given. Since this point is more mathematics than method, we mention it only in passing and do not attempt to prove it here. It is not known if such a calculation will lead to a minimum basis, but it is clearly known that it will not, in general, lead to the conjugate schema basis, as that basis is not necessarily orthonormal.

In cases where the eigenvectors are mutually perpendicular, such a rotation algorithm may yield the conjugate schema matrix. In fact, this type of basis construction is capable of finding, in every case, the mutation bases of the function $f = xy$, which may then be modified to form the crossover basis. The mutation bases and the crossover bases are not, in general, identical.

This method also has two main drawbacks. One constraint of the method is that it requires the basis to be orthonormal. Since the conjugate schema basis may not be orthogonal, this method will not be able to find the conjugate schema basis in general, except when it is orthogonal. The second drawback is the discretization problem, as previously mentioned. As in the previous method, differing step sizes may yield different bases, yet we have found this to be a minimal problem. The apparent resilience under changing step sizes of the conjugate schema basis calculation using the Numerical Hessian technique seems empirically to extend to the basis rotation technique, indicating that there is more than an accidental reason for this resilience.

Note however, that this method has the advantage of being derived directly from the conjugate schema formalism. At every step, we reduce the functional that defines the conjugate schema. We will see that this adaptation is curiously successful on our test functions during our simulations. We postulate that this is due to the overwhelming effect of the adapting mutation basis, as this effect largely disappears when mutation is not dominant.

6. A Basic GA Test Bed

Throughout this study, a single, extremely simple model of the GA is used. Although many other authors have discussed more ambitious models of the GA in which specific operators are proposed (Davis, 1991; Eshelman & Schaffer, 1993), we believe that more will be understood by examining the GA through a simple model. The main advantage of this approach stems from the fact that we need not concern ourselves with special purpose operators or carefully crafted constructs. Instead, we deal only with properties of the GA that are shared by all users of this paradigm.[1] In order to more carefully examine the behavior of the genetic algorithm on our optimization problems, we have further "retarded" the GA by limiting its maximum step size so as to slow its convergence. We define a *step size* to be the maximum distance from a current vector that a new vector may be when created by mutation. Clearly, not all algorithms have a well-defined step size, but we employ such a construct here. We further define the *retarded genetic algorithm* to be a genetic algorithm that employs a step size so small that it significantly adversely affects the speed of convergence as compared to larger step sizes. In our simulations, the "unretarded" version is observed to converge in under 10,000 iterations in all cases, and in under 3,000 iterations in many cases. (24.879879 was found in 2,999 iterations using the canonical basis for test function G7, while 24.653364 was found in 2,999 iterations using the rotated basis for the same function.) Since this is not our primary concern, we will not return to this point.

We first give a few comments about each GA operator, and then describe our model. We assume that fitness is a well-defined real-valued quantity, which may be different for each vector in our vector space, and we wish to optimize it.

6.1 The GA Operators and Search Space

In the basic GA paradigm, there are three main operators. The first is a mutation operator. This operator is responsible for introducing new vector components to the population by making an alteration in one or more elements of the GA. The second, the crossover operator, is responsible for the sharing of components between vectors. By swapping vector components between vectors, the operator carries out the combinatorial rearrangement of vector components. Finally, the reproduction operator is responsible for removing lower-scoring individuals from the population, while enhancing the representation of higher-scoring individuals in the population. By removing vectors that have lower fitness from the population, those vector components that cause a vector to score badly are removed, leaving a population with a more desirable set of vector components. In this paper, we take the reproduction operator to be fixed, as representational issues are not important for such operators.

1 Indeed, many of these properties are also shared by other stochastic search paradigms. The upcoming mathematical constructs may be applied to those search paradigms as readily as to genetic algorithms, though their effects are often quite different.

In our study, we take the space of all possible vectors to be a subset of the n-dimensional space $\Re^n$. For this, we consider our population to be a set of n-dimensional vectors, whose representations in any chosen basis can be written in the form

$$\vec{v} = (v_1, \ldots, v_n)$$

In this representation, the mutation operator (M) may be defined as an operator that takes a vector and modifies it by modifying one or more of the components. It is implicitly assumed that the modification of each component is derived from an independent random distribution with mean zero. Many researchers have shown the effectiveness of many different mutation methods, but we choose to simply use a mutation scheme that randomly alters a given vector component by adding to it a number dv_i selected from a bounded, symmetric, uniform distribution around zero. The bound is called the *step size*, which we adapt throughout our simulations. We have commented that the step size doesn't strongly affect the computation of conjugate schema bases. Thus, we can choose any step size and this choice will not greatly affect the use of the conjugate schema. We choose a particularly small step size in each simulation, typically reducing it after 20,000 iterations once convergence has begun, to allow our simulations to evolve slowly and for their relative behaviors to be measurable. We carry out several bounded mutations per iteration, with probability p_m^n of carrying out n mutations per iteration. The constant p_m is dubbed the *mutation probability*, which in this case refers to the probability of a single mutation of a single vector component. The vector components are randomly chosen from the vector, and so this is an example of a uniform mutation operator.

The crossover operator (Cr) is equally straightforward in this representation. The crossover operator takes one or more elements from one vector and swaps it with the same element(s) from another vector. For a given *crossover probability* p_c, we have the probability of k crossovers given by p_c^k. This is a form of a uniform crossover that does not require a specific number of crossover events to occur at each iteration, allowing us to have single crossover events and to have multiple crossover events. This is desirable, as it is more amenable to changing crossover requirements.

6.2 Population Initialization and Parameters

In all our simulations, the population initially created is uniform; all elements of the population are initialized with the same vector. This is quite different from the standard technique found in the literature, in which a random population is initially used. This paradigm allows us to examine the initial behavior of the algorithm as well as the long-term behavior. Initial behavior is typically dominated by mutation, whereas longer-term behavior is increasingly dependent on crossover. Although we found no discernible difference in the final performance of the algorithm due to this modification in comparison to the random initial population model, the initial dynamics are quite different.

Other standard parameters that we made use of are the population size, the mutation probability, and the crossover rate. These are set to population size 30, mutation rate 0.01, and crossover rate 0.4. While the crossover rate would seem to be low, we find no significant improvements by increasing the crossover rate, and chose not to waste the computational resources.

As we will show, this model of the GA is highly dependent on the use of specific bases. The representation of the vectors in our model will change if the basis is changed. This will also affect the sharing of "good" vector components due to crossover and the discovery of new advantageous vector components introduced when undertaking a mutation.

6.3 Bases for Mutation and Crossover

As discussed in Section 3, the use of differing mutation bases may be more advantageous than a single mutation basis. On the other hand, the use of a specific basis, namely the conjugate schema basis, seems appropriate for the crossover operator's representation. As we require a continually changing mutation basis and an adaptive crossover basis, we combine the two throughout the simulation.

In our simulations, we use two differing mutation bases. The first basis is the canonical basis, while the second is the closest available approximation of the conjugate schema basis. We alternate these each iteration, as the conjugate schema-based basis is updated each 500 iterations. The combination of the two seems to outperform either one separately on mutation-dominated improvement.

The crossover basis is either the canonical basis in the canonical GAs, or generated by the conjugate schema, with each one acting separately in a given simulation. In the canonical crossover paradigms, we use only the canonical basis to carry out crossover, whereas in each of the conjugate schema paradigms (rotated basis or analytic) we generate new bases by one of the two methods and use that to carry out crossovers. One of the difficulties also in the use of the conjugate schema basis is that it is continually changing as the population moves in the search space. Thus, a basis that might be appropriate at one point in the search may be inappropriate at a second point in the search. This requires the basis to be recalculated when it has changed significantly. In our simulations, we empirically chose an interval of 500 iterations to carry out this calculation. For small-dimensional problems, this is a simple calculation, but for larger-dimensional problems, the calculation through basis rotation becomes computationally dense. Our simulations were small enough that this wasn't a problem. For bases with an analytical expression, this calculation could be done off line, and the recalculation was very quick.

The design of each of these operators defined the algorithm. Thus, those algorithms that employed only the canonical basis are dubbed the CGA, those in which the conjugate schema bases are analytically found are called the CBGA, and those in which the conjugate schema are found by basis rotation are known as RBGA. Conjugate schema bases that were used for crossover were calculated in the following way. First, the mutation basis was generated by using either the analytic expression for conjugate schema or basis rotation. Once this was done, each component of the basis had its vectors altered by multiplying the vector components by the value of the population's current positions' vector components. This process is called *positional weighting*. The resulting matrix was then the crossover matrix.

6.4 Constraints

We chose a particularly simple method of enforcing the constraints for each of the test functions. The subject of how to introduce constraints is a topic of continuing work, and we do not make any claims as to the generality of these methods. In all of our sample functions, constraints were modeled as

$$f_{c_i}\left(\overline{v}\right) \geq 0 \tag{33}$$

or

$$f_{c_i}\left(\overline{v}\right) = 0 \tag{34}$$

Thus, in each test function we incorporate a penalty function that linearly penalized the amount by which these were not satisfied. Thus, our penalty function was modeled as

$$P_f = A_f g\left(f_{c_i}\right) \tag{35}$$

for the inequalities and

$$P_f = A_f g(-\|f_{c_i}\|) \tag{36}$$

for the equalities, where

$$g(x) = \begin{cases} -x & x < 0 \\ 0 & x \geq 0 \end{cases} \tag{37}$$

and where the constant A was chosen empirically for each function. Again, since our goal here was not to explicitly produce simulations that yielded "best" results, we are not concerned with creating automatic routines for assigning these constants and are quite happy to do this assignment by hand.

7. Test Functions

In this section, we present the results of our simulations. These simulations have all been built using the same genetic algorithms toolbox (a general purpose collection of routines implementing both traditional GA paradigms and methods introduced here) and the same base program. The only difference in all simulations is the basis of mutation and crossover and in the function being optimized.

An important consideration when using these algorithms to optimize these functions is the way that the optimization is carried out, in terms of how the constraints are handled. In our simulations, as discussed in Section 6, constraints are handled by adding linear penalties. In at least one case this technique is not capable of making the algorithm find vectors that bring this constraint to zero. This is because it is often times possible to find vectors that successfully trade off the penalties of the constraint enforcement with improvements in unconstrained variables. How constraints are imposed is an important and ongoing area of study. However, in our application, we are not interested in such questions, as we want only to understand how best to apply the conjugate schema structure of the search space. We present one function in which infeasible vectors minimize the function. The failure here is not in the ability of the search algorithm to find the minimum, but rather in the designer's ability to design rigorous enough constraints to make the search of the infeasible region of space disallowed.

The functions of Michalewicz and Schoenauer (1996) were tested in our study. Because of the nature of these functions, only five were able to be tested using the analytic conjugate schema formalism. A sixth 'home-grown' (HG) function[2] was also tested using the analytic conjugate schema. The remainder were tested exclusively with the basis rotation strategy outlined in Section 5.2 and using the canonical basis. As the functions can be readily found in the literature (Michalewicz & Schoenauer, 1996), and their exhaustive listing is extremely long, we list only several functions, but omit their constraints. One function, G7, has already been given in Section 3, and two more will be given in this section. We list their maxima or minima in Table 1 and those found by our methods.

Table 1 gives the maxima or minima of the test suite of functions, as found in the literature. G4 appears with a star, indicating that the value given is that found during the course of our simulations, and replaces the current literature value. Table 2 gives the average

2 The home-grown function is given by $f(\vec{x}) = \sum_{i=1}^{5} x_{2i-1}x_{2i}$. This function is similar to the simple function discussed in Sections 2 and 3; it should be easy for the conjugate schema GA to optimize it compared to the canonical GA. The constraints in this case are given by $x_i \leq 10$.

Table 1. This table gives the set of maxima for the functions that were used in this study. The starred function is that for which the minimum value is reported for the first time in this paper. Those with plus marks are functions for which the maximum was verified with our model, though often times requiring long (more than 10^5 iterations) runs. We did not find the value quoted in Michalewicz and Schoenauer (1996) for the function G5, as the quoted value was found to be unfeasible.

Functions used		
Function name	Max/Min	Value
G1$^+$	Min	−15
G2	Max	0.803553 (so far)
G3	Max	1
G4*	Min	−31010.437912
G5	Min	5126.4981
G6$^+$	Min	−6961.81381
G7$^+$	Min	24.3062091
G8$^+$	Max	0.1
G9$^+$	Min	680.6300573
G10	Min	7049.330923
G11	Min	0.75
HG$^+$	Max	500

Table 2. This table gives the average performance of 10 runs on the test functions. All simulations were 100,000 generations with the exception of the last one, which was 10,000 generations.

Average performance on test functions			
Function	CGA	CBGA	RBGA
G1	-9.086 ± 0.660	–	-12.1 ± 0.3
G2	0.351 ± 0.036	–	0.329 ± 0.028
G3	0.486 ± 0.0008	–	0.501 ± 0.045
G4	-30994.2 ± 6.4	-30952.5 ± 10.3	-30998.3 ± 4.1
G5	3017.5 ± 0.2	3017.2 ± 0.4	4138.5 ± 59.6
G6	-6938.0 ± 0.1	–	-6961.56
G7	290.0 ± 20.5	142.8 ± 14.8	29.1 ± 0.8
G8	0.002741 ± 0.0008	–	$0.095795 \pm 2e^{-5}$
G9	960.0 ± 0.8	884.6 ± 7.1	909.7 ± 1.3
G10	9932.5 ± 918.2	5175.8 ± 286.5	2369.3 ± 153.2
G11	0.889 ± 0.005	–	0.875 ± 0.005
HG	246.7 ± 0.8	452.3 ± 15.1	198.1 ± 7.1

performance of each scheme over 10 runs, excluding the analytic conjugate schema runs that could not be run. Table 3 gives the best value found during the search and the algorithm that produced it.

During the course of our simulations, the maxima or minima reported in the literature were found in seven of 12 cases, a striking fact, since our intention was not to optimize our routines to find maxima efficiently. In six of those cases, the maxima reported in the literature were confirmed, while in one of those, a new maximum was found. In five of 12

Table 3. This table gives the best performance on these problems obtained in our simulations. All simulations are "retarded," limiting their ability to find the maxima in 100,000 steps. The starred function records the best-known performance on this function, while the double-starred function has a best performance that is due to its smallest constraint levels on this particular run (0.49). Though the other models scored lower final function values, they were much more highly unfeasible.

Best performance		
Function name	Max/Min	Algorithm
G1	−13.739256	RBGA
G2	0.62	RBGA
G3	0.655493	CGA
G4*	−31010.437912	CGA/RBGA
G5**	4362.190060	RBGA
G6	−6961.51	RBGA
G7	26.676231	RBGA
G8	0.095798	RBGA
G9	846.513489	CBGA
G10	2100.143456	RBGA
G11	0.847067	RBGA
HG	497.048187	CBGA

cases, the maxima were not found; the reasons for this result will be discussed in the next section. However, the failure of the algorithm to find the maxima or minima of the function in question was in many cases due to an inefficient way of handling constraints, rather than from a failure of the search algorithm itself. In all but two cases, the CGA scored lower than or equal to one or more of the conjugate schema-based algorithms, CBGA and RBGA. In only one case did CGA score higher, when averaged over all 10 runs. However, even in this case, the RBGA was able to find a higher-scoring peak than CGA. These results indicate that the conjugate schema-based algorithms CBGA and RBGA are capable of taking advantage of local and global information more efficiently than CGA.

8. Discussion

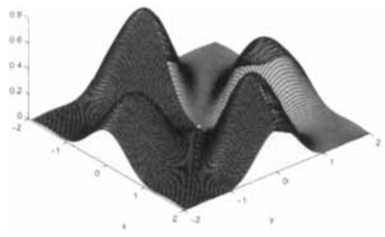
In order to truly understand how these algorithms are behaving, we need to dig somewhat deeper. In this section, we will discuss several of the simulations in detail. We will also discuss the role of mutation basis variation in aiding the generation of lower-level convergences.

8.1 Individual Cases

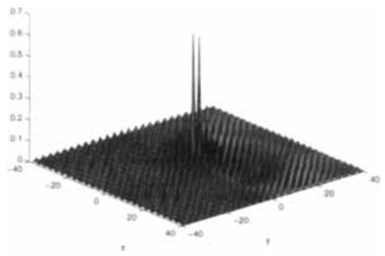
We consider first the case of function G2, given by

$$G2\left(\vec{x}\right) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{38}$$

In two dimensions, this function is as shown in Graph 3, which has a relatively flat global structure, with all near-optimal values located near the origin. This function has a Hessian that is algebraically extremely complicated, and cannot be given here. Because of the nature of the function, the Hessian is rapidly varying, and this means that the eigenvector matrix



Graph 3. Graph of function $G_2(\vec{x})$ etc.



Graph 4. Larger-scale graph of function $G_2(\vec{x})$ etc.

is also rapidly varying. Moreover, there is little global structure to take advantage of, as the function is uninteresting far from the origin. The CBGA was not implemented, since there is no simple analytic form of the conjugate schema matrix. The RBGA was implemented, but as the current best basis for the search changes greatly at each step of the search, it is not expected that the RBGA will have a significant improvement over the canonical GA. The performance is expected to depend, in both models, on the initial position of the population.

Other functions behaved well under our formalism. Noting that the maximum of this function is reported to be 0.803553, we expect that both of the optimization methods being used here will fall short of this. Since the RBGA is an effective way of climbing narrow peaks, we expect that it will be more consistent than the CGA. We also expect that the RBGA will be easily caught by one of the many peaks, trapping it in local maxima. Inspection of Table 2 indicates both of these effects. The variance of the RBGA sampling is smaller than that of the CGA, indicating a more uniform final solution set. Moreover, the homogeneity of the solutions indicates only local search, as expected. A different initial population might have served to alleviate this problem. The best values over 10 runs of the RBGA and the CGA paradigms were 0.62 and 0.53, respectively, from different random number sequences. As shown in Figure 3 below, while the best performance of the RBGA has a higher maxima than the best of the CGA on this problem, the CGA outperforms the RBGA on average. This fact notwithstanding, we find also that the CGA is not as reliable as the RBGA and varies more wildly than the RBGA. Other functions behaved well under our formalism. The most striking of these is the HG function. This function has the form

$$HG\left(\vec{x}\right) = \sum_{i=1}^{5} x_{2i-1}x_{2i} \tag{39}$$

constrained by

$$|x_i| \leq 10 \tag{40}$$

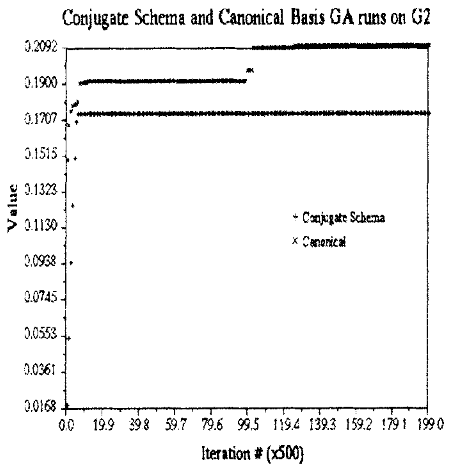


Figure 3. This figure gives the performance of the CGA and the RBGA on G2. In this case, the CGA outperforms the RBGA, but it still fails to find the maximum function value. Moreover, both paradigms are prone to quick convergence and sporadic fluctuations of the population. As we expected, the basis rotation does not help much on this function.

which has the Hessian

$$H = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&1\\ 0&0&1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1&0&0\\ 1&0&0&0&0&0&0&0&0&0 \end{pmatrix} \tag{41}$$

with the eigenvector matrix

$$E = \frac{1}{\sqrt{2}} \begin{pmatrix} 1&0&0&0&0&0&0&0&0&1\\ 0&1&1&0&0&0&0&0&0&0\\ 0&1&-1&0&0&0&0&0&0&0\\ 0&0&0&1&1&0&0&0&0&0\\ 0&0&0&1&-1&0&0&0&0&0\\ 0&0&0&0&0&1&1&0&0&0\\ 0&0&0&0&0&1&-1&0&0&0\\ 0&0&0&0&0&0&0&1&1&0\\ 0&0&0&0&0&0&0&1&-1&0\\ 1&0&0&0&0&0&0&0&0&-1 \end{pmatrix} \tag{42}$$

This basis is a global conjugate schema for this function, and represents one of the simplest functions for which we expect the conjugate schema to do better than any other model. The crossover basis is given by

$$Cr = \begin{pmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} & 0 & 0 & \cdots & 0 \\ \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} & 0 & 0 & \cdots & 0 \\ 0 & 0 & \frac{x_3}{\sqrt{x_3^2+x_4^2}} & \frac{x_3}{\sqrt{x_3^2+x_4^2}} & \cdots & 0 \\ 0 & 0 & \frac{x_4}{\sqrt{x_3^2+x_4^2}} & -\frac{x_4}{\sqrt{x_3^2+x_4^2}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -\frac{x_{10}}{\sqrt{x_9^2+x_{10}^2}} \end{pmatrix} \qquad (43)$$

This basis is clearly not orthogonal and cannot be approximated by a orthogonal basis. Thus, we expect that of the three GA paradigms, the pure conjugate schema paradigm will outperform both other algorithms. Moreover, since each basis element is a substantial departure from the canonical basis, we expect that any basis approximates many of these. On the other hand, our rotation algorithm will alter all vectors significantly. None will approximate, after one or two iterations, the true conjugate schema because of the constraint that the basis must be orthonormal. Thus, a single mutation will affect multiple vector components. Since at most two vector components need be altered simultaneously, we expect that the basis rotation will cause a more problematic algorithm, rather than aiding the algorithm in this case. Because of these concerns, we expect that the conjugate schema will have the best performance, followed by the canonical basis GA, and finally followed by the rotated basis GA.

Dynamically, we expect very similar behavior over an initial population that is uniform and initialized at the origin. This is because all of the vectors of the conjugate schema are nearly identical to a rotated canonical basis, with a small functional dependence. However, the farther the population is from the origin, the better we expect the conjugate schema model to do with respect to the canonical model. The reason is that the mutations will be more likely to lead in advantageous directions, and as one goes on, the effect of correct crossover becomes particularly important as opposed to inefficient crossovers from an inappropriate basis.

The average data over 10 runs is plotted in Figure 3. Initially, all of the algorithms perform similarly. In Figure 4, we plot the average behavior of the various models over 10,000 generations. The conjugate schema model clearly dominates within 500 generations and gives a near-optimal value within 10,000 generations. The next-best performance is, as predicted, the CGA, followed closely by the RBGA. All of these behaviors have been anticipated by consideration of our model.

A more striking example of the difference in performance due to the choice of basis is given in our simulations' performance on the optimization of the function G10 shown in Figure 5. This function, given by

$$G10\left(\overrightarrow{x}\right) = x_1 + x_2 + x_3 \qquad (44)$$

and subject to constraints involving eight variables has a theoretical minimum of 2,100. The constraints make this minimum infeasible, although it is possible to get very close and have all but one constraint satisfied, with the last constraint smaller than 0.4. The performance of the rotated basis model completely outstrips that of the canonical basis model and the conjugate

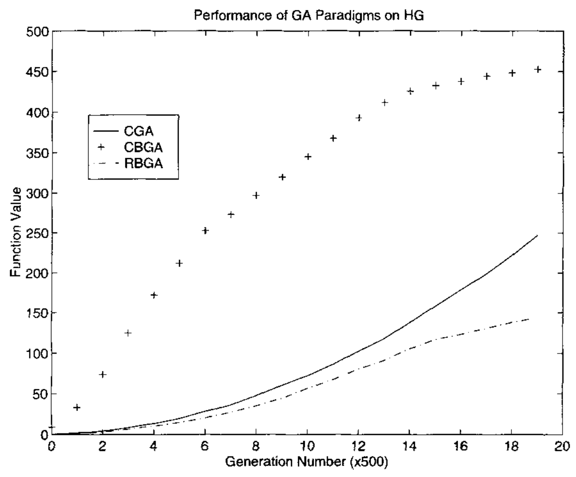


Figure 4. This figure gives the performance of the three GA paradigms on the HG problem. The CBGA model clearly dominates, yielding a near maximal value of 500 in 10,000 generations, while the other two models lag significantly behind.

schema basis model, even though the latter also does significantly better than the first. While the canonical basis model makes almost no progress in 100,000 generations (although the larger step-size models had no trouble optimizing this function), the rotated-basis model finds a minimum value at 2100.143456 in the same number of generations, reaching almost that value in a fraction of that number of generations (<10,000). The conjugate schema model finds a minimum of 4332.699078. All solutions, however, are infeasible. This is viewed as a failure in the inclusion of the constraints, rather than the optimization method, and more successful algorithms must use more aggressive optimization methods. It is somewhat alarming to note that the canonical model seems to have made almost no progress. As stated in Section 5, the simulations have been retarded, making it difficult to optimize the functions, by using small step sizes. For this reason, the canonical model made minimal progress over the entire simulation, though its apparent lack of progress is augmented by the logarithmic scale. However, with the same step sizes, both of the conjugate schema-based algorithms were able to find much lower minima.

That the conjugate schema-based algorithms should be able to do better than the canonical basis is expected from the knowledge that the absolute Hessian matrix is given by

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{45}$$

giving the eigenvector matrix

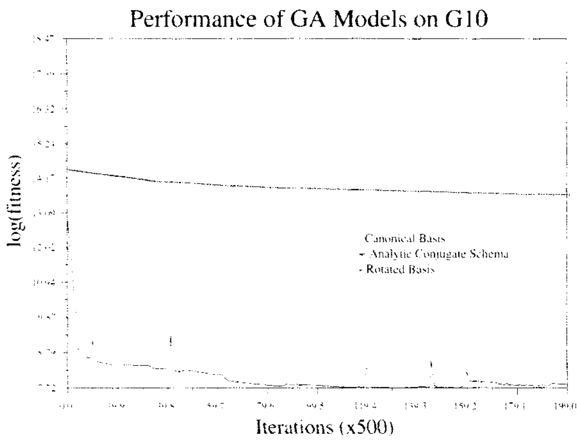


Figure 5. This figure gives the performance of the GA paradigms on G10. The best performance is given by the basis rotation paradigm, followed by the conjugate schema basis, and a distant last by the canonical basis. What seems to have happened here is that the conjugate schema, formed over an approximated functional form, has improved the search, but not quite to the extent that the basis rotation is capable of doing.

$$E = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{2} & -\sqrt{2} \\ 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{46}$$

The fact that this matrix is orthogonal indicates that the RBGA should work very well on this problem, assuming that the basis rotation algorithm can find this basis. That these are global conjugate schema also indicates that the conjugate schema method should work very well. However, it is difficult to know how much local behavior we will run into, and so one expects that the basis rotation will do better, which by its evaluation is more global than the conjugate schema. This behavior is observed.

Through this work, we were able to efficiently find a novel maxima, and to verify the maxima of at least four other optimization problems in the framework of the current simulations (longer simulations, of course, yielded better results). Other simulations were able to verify other maxima published in this test suite more efficiently than with the canonical bases. These data are given in Table 4. More generally, in every simulation except two, the optimum average final state is found by one of either the basis rotation or the conjugate schema. The dramatic improvement of the RBGA in the initial steps of several simulations indicates that the adaptive capability of the RBGA allowed it to take advantage of more useful mutation bases, an advantage not shared in other cases. In most cases, the conjugate schema state did not produce particularly encouraging results. However, notable exceptions to this rule came from G9 and HG. These problems possessed unchanging global conjugate schema and thus could be easily exploited. RBGA's advantage made it a considerably better

Table 4. These functions' maxima are verified during our simulations. In one case (G4), a new minimum is discovered.

New or verified maxima		
Function	Maxima found	New (Y/N)
G4	−31010.437912	Y
G6	6961.51	N
G7	24.306	N
G8	0.095798	N
HG	500.	N

performer, supporting conjugate schema-based algorithms. Other functions did not contain such conjugate schema, and so the bases were approximations to the conjugate schema except directly after an update (an update occurred each 500 generations). These approximations clearly did not function well.

On the other hand, the basis rotation did yield excellent results, as did the conjugate schema when they were global. Both of these results support the theoretical underpinnings of conjugate schema, and they indicate that conjugate schema are important elements of global search that should be carefully considered during the search.

8.2 The Role of the Basis

In the previous discussions, we have illustrated the use of the basis as a driving force in the success or failure of the GA paradigm. We have seen that in some cases, the conjugate schema basis is very effective. In other cases, however, approximations to the conjugate schema are more effective, as in the case of basis rotation. In still other cases, the canonical basis was as effective a search basis as any. However, this knowledge does not put us much closer to an understanding of why these bases should cause better or worse performance. In this section, we will attempt to ascertain the root of the success or failure of a particular basis.

One characteristic of the crossover basis that seemed to serve as an indicator of the usefulness of the algorithm was its variation over a run. That is, in both the conjugate schema model and the rotated basis model, we use bases that vary as the run progresses. It was noted that the variation of the mutation and crossover bases was commensurate with the success or failure of the GA paradigm.

We define the pseudomeasure between two bases, $B_1 = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_N\}$ and $B_2 = \{\widehat{e'}_1, \widehat{e'}_2, \ldots, \widehat{e'}_N\}$ as

$$d(B_1, B_2) = N - \frac{1}{2}\sum_{i=1}^{N} \max_j \left\{\hat{e}_i \cdot \widehat{e'}_j\right\} - \frac{1}{2}\sum_{i=1}^{N} \max_j \left\{\widehat{e'}_i \cdot \hat{e}_j\right\} \qquad (47)$$

This is not a metric since it does not obey the triangle inequality, but it does serve to illustrate the effect of the variation of a basis on a GA performance. That is, if the basis is varying over iterations, a graph of this pseudomeasure will be varying as well, even though it gives nonspecific information about the relative distance between bases.

Let us reconsider the function HG. On this function, the conjugate schema basis' performance dominated the other two models. We note that the canonical basis' variation over a run is zero, and its performance is not of interest in this section. On the other hand, we would like to know why the rotated basis did not do as well as the *nonrotated basis.*

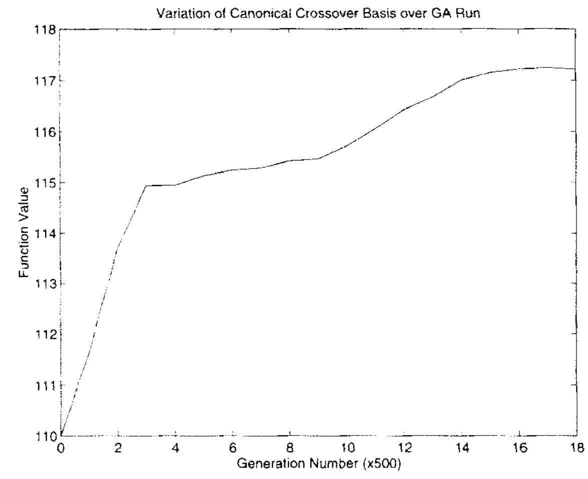


Figure 6a. The variation of the crossover basis over one run of the GA paradigm for the RBGA is rather monotonic. As seen in Figure 3, the method did not succeed in optimizing the function. This may be a useful fingerprint for useful bases.

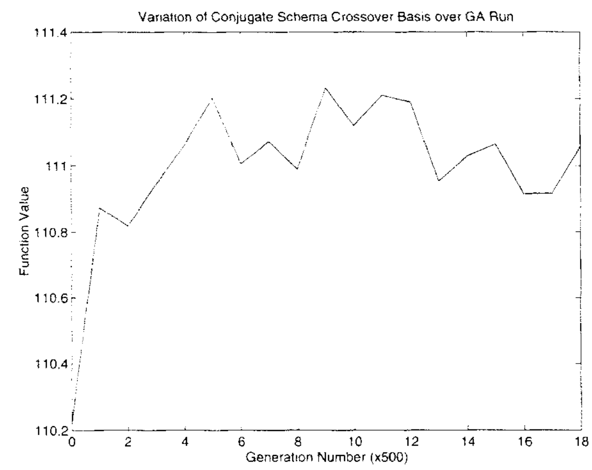


Figure 6b. This figure illustrates the crossover basis variation due to adaptation over the course of one CBGA run. We view the variation of the basis as an indicator of good progress, as we empirically find that small variation accompanies stagnation.

One clue is that the crossover basis seemed to vary monotonically, and over a wide range, during any run. This variation is shown in Figure 6a. In this figure, the variation is more or less monotonic, and takes place over a range of ~7. On the other hand, a nonmonotonic variation was observed in the distance of the basis from its original basis over the simulation of the conjugate schema basis. (See Figure 6b.) The amplitude of this variation had a considerably smaller range than that of the conjugate schema basis, but was less predictable. From this data, we would begin to conclude that the adaptation of the basis is an important tool in the success or failure of the paradigm.

This last point is further supported by a closer look at the performance of the conjugate schema and basis-rotated models of the GA on the function G7. In this function, the variation

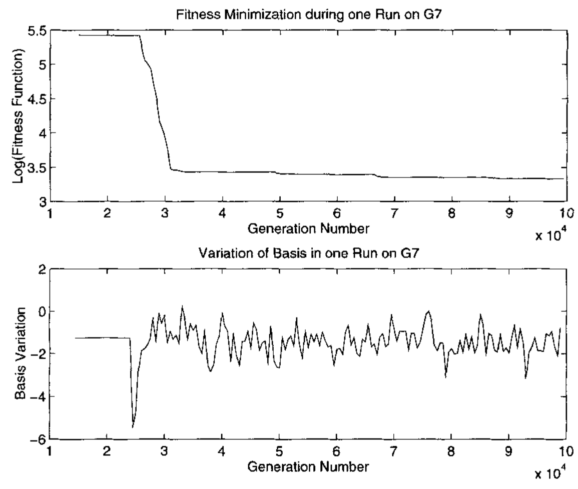


Figure 7. This figure gives the effect of basis variation on the convergence of the GA paradigm in the RBGA. Initially, the basis rotation has been turned off, and the scheme converges to a value that is ~200, much higher than the minimum. Once the basis variation is turned on, as can be seen in the lower graph, the fitness value falls dramatically to nearly the minimum.

of the basis closely follows the optimization of the function. Optimization occurs while the basis is varied, but convergence occurs once the basis variation is ceased. This result can be seen quite strikingly in Figure 7, which gives the effect of basis variation on a simulation that had already converged. As one can see, the convergence is alleviated, and the simulation finds a more minimal fitness value.

Typically, basis variation may be used as a measure of the specificity of the basis being used. If a large variation is found, then the basis is adapting to the search, whereas if a monotonically varying or set basis is being used, the search is independent of the basis.

9. Concluding Remarks

The importance of the correct representation of any problem cannot be overstated. This universal difficulty finds its home in all areas of science from theoretical mathematics to physics. Any improvement in the representation will result in more useful ways of computationally attacking a problem. We have seen that in real-encoded genetic algorithms, representations deriving from conjugate schema provide improved performance both in terms of the convergence speed and maxima located.

Over a space of binary strings, representational issues are also important. It is not difficult to construct an example of a simple function over which modification of single bits is useless, but modification of many bits together will result in improvements in the string's current function value. At least one researcher has found a representational issue for crossover that serves to support the usefulness of conjugate schema. Syswerda (1993) investigated the usefulness of uniform crossover over several standard test problems. His conclusion was that the simulations making use of uniform crossover were more successful in combining useful information than one or two-point crossover. It is straightforward to show that schema define conjugate schema if they do not overlap *in a string. Thus, finding* of conjugate schema, which may be carried out by making use of the basis rotation protocol, may serve to aid in discretely encoded problems.

The preceding fact is quite compelling. If we consider the problem of putting specific elements together, if the elements are not contiguous, then the uniform crossover will be more likely to transfer information between two vectors successfully. This is similar to the use of conjugate schema because it allows arbitrary schema to be addressed with a nonzero probability, while the previous models typically allow it to be addressed with a zero probability. Thus, uniform crossover, like conjugate schema crossover, helps to preserve information. On the other hand, if our prescription for basis rotations is sound, we may bypass uniform crossover, and use conjugate schema subbases. In fact, there are many problems outside of the restricted realm of the genetic algorithm for which this type of analysis can be useful. In molecular dynamics, conjugate schema will produce motion eigenvectors, while in simulated annealing algorithms (Koonin, 1986) the combining of several directions may aid in the creation of useful mutations, yielding better maximal states. If this formalism can be used in discrete bounded vector spaces, then this formalism might be useful for both protein structure determination and Monte Carlo integrations. These possibilities deserve more careful study.

Acknowledgments
I would like to thank Drs. Jerry Solomon and Steve Mayo. This work was completed under their auspices, and both provided the time and location for the work to be completed. Dr. Solomon provided a backboard for the theoretical development of the theory of conjugate schema. It was he who helped me to begin to ask the right questions about the genetic algorithm.

I would like to thank Dr. Scott Page. It was Dr. Page who first introduced me to the genetic algorithm. It was he who introduced me to the building block hypothesis, and who helped me develop earlier versions of this work. His assistance in the steps leading to the present work was invaluable. His critiquing of the final forms of this paper was invaluable.

I would like to thank Dr. Chris Adami for his long discussions with me about this paper. It was his discussions that helped me to finalize the form, and stress important points in the paper. Under his guidance, I was able to extract the details from the theoretical work in a way that made the paper clearer and more incisive.

I would like to thank Ari Hershowitz, who provided useful comments which helped me to finish writing this paper. His careful attention to every detail of the paper was greatly appreciated. I would like to thank the reviewers of this paper for their useful comments. This was invaluable in helping to rearrange and focus the paper.

This work is dedicated to my wife, Jisoo Park.

Appendix A. Proofs of Various Facts

Let us define the functional D. Given a real function f over a linear space Γ and given an orthogonal basis $\{\vec{x_1}, \ldots, \vec{x_N}\}$ is given by

$$D = \sum_{i,j=1}^{N} D_{ij} \tag{A1}$$

where

$$D_{ij} = \left| f\left(\vec{x_0}\right) + f\left(\vec{x_0} + \epsilon_i \vec{x_i} + \epsilon_j \vec{x_j}\right) - f\left(\vec{x_0} + \epsilon_j \vec{x_j}\right) - f\left(\vec{x_0} + \epsilon_i \vec{x_i}\right) \right| \tag{A2}$$

This will form the basis for what is to follow.

A.1 Absolute Hessian

First, we prove a proposition which we will use in our proof of the main result. We assume that Γ is an n-dimensional vector space.

PROPOSITION 1: *Given a linear operator A over a linear space Γ, as defined above, the minimization of D carried out over the inner product function $f(x) = \langle Ax|x\rangle$ picks out the eigenvectors of the operator A.*

PROOF: First, we assume that we have a linear space Γ and that A is a linear operator over Γ with nondegenerate and nonzero eigenvalues $\{k_1, \ldots, k_N\}$. We assume that Γ contains a zero element denoted 0 for which $x + 0 = 0$ and $A0 = 0$. Consider

$$D_{ij} = \left|f(0) + f(\psi_i + \psi_j) - f(\psi_i) - f(\psi_j)\right| \tag{A3}$$

Now, if a and b are real constants, and if $\varphi = a\psi_i + b\psi_j$ and $\phi = b\psi_i - a\psi_j$, then

$$f(\varphi) = f(a\psi_i + b\psi_j) = k_i a^2 + k_j b^2 \tag{A4}$$

$$f(\phi) = f(b\psi_i - a\psi_j) = k_i b^2 + k_j a^2 \tag{A5}$$

$$\begin{aligned} f(\varphi + \phi) &= f\left((a+b)\psi_i + (b-a)\psi_j\right) \\ &= k_i(a+b)^2 + k_j(b-a)^2 \\ &= 2ab\left(k_i - k_j\right) + \left(a^2 + b^2\right)\left(k_i - k_j\right) \end{aligned} \tag{A6}$$

so that

$$D_{ij} = \left|2ab\left(k_i - k_j\right)\right| \tag{A7}$$

which will be zero, in general, iff $k_i = k_j$ or either a or b is zero. Since we already assumed that the eigenvalues are nondegenerate and nonzero, this requires a or b to be zero, meaning that the functional is minimized iff all directions ϕ and φ are along the directions of the basis elements. □

We use this fact in the next proof of our main result. Let us recall first that if two eigenvalues of a symmetric matrix are nondegenerate, the eigenvectors will be orthogonal.

First, we recall that the Hessian matrix is calculated from a continuous function by setting the ijth element to the second order differential $\frac{\partial^2 f}{\partial x_i \partial x_j}$. In other words,

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_N \partial x_N} \end{pmatrix}$$

The *absolute Hessian matrix* is the same as the Hessian matrix with the exception that each element of the matrix is the absolute value of the corresponding element from the Hessian matrix.

$$AH = \begin{pmatrix} \left|\frac{\partial^2 f}{\partial x_1 \partial x_1}\right| & \left|\frac{\partial^2 f}{\partial x_1 \partial x_2}\right| & \cdots & \left|\frac{\partial^2 f}{\partial x_1 \partial x_N}\right| \\ \left|\frac{\partial^2 f}{\partial x_2 \partial x_1}\right| & \left|\frac{\partial^2 f}{\partial x_2 \partial x_2}\right| & \cdots & \left|\frac{\partial^2 f}{\partial x_2 \partial x_N}\right| \\ \vdots & \vdots & \ddots & \vdots \\ \left|\frac{\partial^2 f}{\partial x_N \partial x_1}\right| & \left|\frac{\partial^2 f}{\partial x_N \partial x_2}\right| & \cdots & \left|\frac{\partial^2 f}{\partial x_N \partial x_N}\right| \end{pmatrix}$$

PROPOSITION 2: *A minimization of D picks out the eigenvectors of the matrix given by*

$$AH = \left[\left|\frac{\partial^2 f\left(\vec{x_0}\right)}{\partial x_i \partial x_j}\right|\right]_{ij} \tag{A8}$$

PROOF: Again, let us consider D_{ij}.

$$D_{ij} = \left|f\left(\vec{x_0}\right) + f\left(\vec{x_0} + \epsilon_i \vec{x_i} + \epsilon_j \vec{x_j}\right) - f\left(\vec{x_0} + \epsilon_j \vec{x_j}\right) - f\left(\vec{x_0} + \epsilon_i \vec{x_i}\right)\right|$$

Now, for small ϵ_i, we have

$$f\left(\vec{x_0}\right) - f\left(\vec{x_0} + \epsilon_i \vec{x_i}\right) \simeq -\epsilon_i \frac{\partial f\left(\vec{x_0}\right)}{\partial x_i} \tag{A9}$$

and for sufficiently small ϵ_i and ϵ_j, we have

$$f\left(\vec{x_0} + \epsilon_i \vec{x_i} + \epsilon_j \vec{x_j}\right) - f\left(\vec{x_0} + \epsilon_j \vec{x_j}\right) \simeq -\epsilon_i \frac{\partial f\left(\vec{x_0} + \epsilon_j \vec{x_j}\right)}{\partial x_i} \qquad \text{A10)}$$

so that

$$D_{ij} \simeq \left|\epsilon_i \frac{\partial f\left(\vec{x_0} + \epsilon_j \vec{x_j}\right)}{\partial x_i} - \epsilon_i \frac{\partial f\left(\vec{x_0}\right)}{\partial x_i}\right| \simeq \epsilon_j \left|\frac{\partial^2 f\left(\vec{x_0}\right)}{\partial x_i \partial x_j}\right| \epsilon_i \tag{A11}$$

□

Together with the first proposition, this proposition proves our assertion that the eigenvector matrix of the absolute Hessian matrix will be the minimized form of D.

A.2 Separability

Our next proposition indicates that conjugate schema induce separability in the function. This means that one may separately optimize the parts of the vector, and this will yield the global optimum.

PROPOSITION 3: *Given $f : \Gamma \mapsto \Re$, if ϑ is a basis for Γ, and $A \cap B = \emptyset$ forms a separation of ϑ, then if A and B are conjugate schema, then*

$$f = g + h \tag{A12}$$

with $g : \langle A \rangle \mapsto \Re$ and $h : \langle B \rangle \mapsto \Re$ where $\langle A \rangle \otimes \langle B \rangle = \Gamma$.

PROOF: Suppose everything is as given above. Then given any $\epsilon, \delta > 0$, $b_i \in A$ and $b_j \in B$,

$$\left| f\left(\vec{x}\right) + f\left(\vec{x} + \epsilon \vec{b_i} + \delta \vec{b_j}\right) - f\left(\vec{x} + \epsilon \vec{b_i}\right) - f\left(\vec{x} + \delta \vec{b_j}\right) \right| = 0 \tag{A13}$$

This means that

$$f\left(\vec{x}\right) + f\left(\vec{x} + \epsilon \vec{b_i} + \delta \vec{b_j}\right) = f\left(\vec{x} + \epsilon \vec{b_i}\right) + f\left(\vec{x} + \delta \vec{b_j}\right) \tag{A14}$$

which can be rearranged to read

$$f\left(\vec{x} + \epsilon \vec{b_i} + \delta \vec{b_j}\right) - f\left(\vec{x} + \delta \vec{b_j}\right) = f\left(\vec{x} + \epsilon \vec{b_i}\right) - f\left(\vec{x}\right) \tag{A15}$$

and finally,

$$\frac{f\left(\vec{x} + \epsilon \vec{b_i} + \delta \vec{b_j}\right) - f\left(\vec{x} + \delta \vec{b_j}\right)}{\epsilon} = \frac{f\left(\vec{x} + \epsilon \vec{b_i}\right) - f\left(\vec{x}\right)}{\epsilon} \tag{A16}$$

so that if we take the limit, we find that

$$\left.\frac{\partial f}{\partial b_i}\right|_{\vec{x}+\delta \vec{b_j}} = \left.\frac{\partial f}{\partial b_i}\right|_{\vec{x}} \tag{A17}$$

or that the partial derivative is completely independent of any of the elements in the opposite conjugate schema. □

References

Arabas, J., Mulawka, J., & Pokraniewicz, J. (1995). A new class of the crossover operators for the numerical optimization. In L. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 42–47). San Mateo, CA: Morgan Kaufmann.

Battle, D., & Vose, M. (1993). Isomorphisms of genetic algorithms. *Artificial Intelligence*, *60*, 155–165.

Davis, L. (1991). Hybridization and numerical representation. In L. Davis (Ed.), *The Handbook of Genetic Algorithms* (pp. 61–71). New York, NY: Van Nostrand Reinhold.

Eshelman, L., & Schaffer, D. (1993). Real-coded genetic algorithms and interval schemata. In D. Whitley (Ed.), *Foundations of Genetic Algorithms 2* (pp. 187–202). San Mateo, CA: Morgan Kaufmann.

Forrest, S., & Mitchell, M. (1992). Towards a stronger building-blocks hypothesis: Effects of relative building-block fitness on GA performance, Technical Report SFI-TR-92-06-029, Santa-Fe Institute.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Kazadi, S. (1997). Conjugate schema in genetic search. In T. Bäck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms* (pp. 10–17). San Mateo, CA: Morgan Kaufmann.

Koonin, S. (1986). *Computational physics.* Menlo Park, CA: Benjamin Cummings.

Leipins, G., & Vose, D. (1990). Representational issues in genetic optimization. *Journal of Experimental Theoretical Artificial Intelligence*, *2*, 101–115.

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, *4*(1), 1–32.

Ono, I., & Kobayashi, S. (1997). A real-coded genetic algorithm for function optimization. In T. Bäck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms* (pp. 246–253). San Mateo, CA: Morgan Kaufmann.

Page, S., & Hong, L. (1998). *Diversity and optimality*. Mimeographed article. University of Iowa, Department of Economics.

Rana, S., & Whitley, L. D. (1997). Bit representations with a twist. In T. Bäck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms* (pp. 188–195). San Mateo, CA: Morgan Kaufmann.

Smith, J., & Fogarty, T. C. (1995). An adaptive poly-parental recombination strategy. In T. C. Fogarty (Ed.), *Lecture Notes in Computer Science 993 Evolutionary Computing* (pp. 48–61). New York, NY: Springer-Verlag.

Syswerda, G. (1993). Uniform crossover in genetic algorithms. In D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 2–9). San Mateo, CA: Morgan Kaufmann.

Tsutsui, S., Fujimoto, Y., & Ghosh, A. (1997). Forking genetic algorithms: GAs with search space division schemes. *Evolutionary Computation*, *5*(1), 61–80.

Vose, M. (1991). Generalizing the notion of schema in genetic algorithms. *Artificial Intelligence*, *50*, 385–396.

Wright, A. (1991). Genetic algorithms for real parameter optimization. In G. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp. 205–218). San Mateo, CA: Morgan Kaufmann.

Wu, A., & Lindsay, R. (1996). A comparison of the fixed and floating building block representation in the genetic algorithm. *Evolutionary Computation*, *4*(2), 169–193.