Illinois Math and Science Academy

From the Selected Works of Sanza Kazadi

July 19, 1997

Conjugate Schema in Genetic Search

Sanza Kazadi

Available at: https://works.bepress.com/sanza-kazadi/20/
Conjugate Schema in Genetic Search

Sanza T. Kazadi
Department of Computation and Neural Systems
139-74 Beckman Institute
California Institute of Technology
Pasadena, Ca 91125
sanza@jisan.caltech.edu

Abstract

Functional optimization is profoundly affected by the use of specific encodings. In one encoding, a particular problem may be simple to undertake, while in another encoding, the program may be intractable. Genetic algorithms solve optimization problems by making use of schema. By locating schema in a solution vector, the paradigm can settle on a solution that makes use of several schema and combines them via crossover.

We propose a generalization of this idea, conjugate schema. Conjugate schema are disjoint subsets of the basis over which the fitness function can be written as a sum of smaller dimensional functions. We find that if conjugate schema exist, this lowering of dimensionality is a direct consequence, yielding the motivation for finding bases containing a maximum number of conjugate schema. We find that in continuous problems, the conjugate schema are the eigenvectors of the absolute Hessian operator,

\[
\begin{bmatrix}
\frac{\partial^2 f}{\partial i \partial j}
\end{bmatrix}_{ij}
\]

We give several examples, and discuss the possible implications.

1 Introduction

In their 1993 paper, Michael Vose and David Battle[1] proved that a genetic algorithm operating on a function transformed by a linear transformation obeyed the same rules and had the same (albeit transformed) schema as the original genetic algorithm operating on the original function. The transformation need not have been unitary for this to occur, yielding a surprising latitude in the creation of new but equivalent genetic algorithms.

This observation having been made, one might ask what 'good' transformations in the space of a genetic algorithm are, and how to find them. It is assumed that if the dimensionality of the problem can be reduced, then its solution may be more readily found. The trick, then, is to lower the dimensionality of the problem, and thus reduce the complexity. Once this is accomplished, standard methods will be able to find the solution with more efficacy.

In this paper, we seek to find some insight that will help us to reduce the dimensionality of a function adaptively while undertaking the optimization. Clearly stated, we wish to discover which transformations of the genetic algorithm's encoding will present the genetic algorithm with a lower-dimensional function. We present the idea of conjugate schema, which are "building blocks", as it were. We show that it is conjugate schema that are sought in the optimization process, and that, once found, the dimensionality of the problem is reduced. We discuss the finding of conjugate schema, and in the end present some of our results using this methodology on several problems.

2 Definition of Conjugate Schema

This section will introduce the concept of a conjugate schema. After an introductory definition of the concept, several useful corollaries will be derived.

Conjugate schema are the basis for the breaking up of large dimensional difficult problems into smaller dimensional, and hopefully more tractable, problems\(^1\). In what follows, \(\Gamma\) is assumed to be an n-dimensional vector space.

\(^1\)In discrete vector spaces, the only unitary basis is the canonical one. However, if we do not restrict ourselves to
vector space. We also assume that there exists a function \( f : \Gamma \rightarrow \mathbb{R} \) which we consider to be our fitness function.

We begin with a definition of functional independence. Given a vector space \( \Gamma \), if two basis elements \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) of \( \Gamma \) have the property that a small variation of a given vector \( \overrightarrow{x} \) along \( \overrightarrow{b}_i \) produces a variation of \( f \left( \overrightarrow{x} + \epsilon \overrightarrow{b}_i \right) \) that is independent of another variation along \( \overrightarrow{b}_j \), then we call the two functionally independent. This means that the function’s behavior in one direction does not depend on the other direction. If this is the case, then the two basis elements will be part of larger structures for which the function may be rewritten as lower dimensional functions.

**Definition 1** Let \( \theta \) be a basis in \( \Gamma \). Suppose also that \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) are elements of \( \theta \). Also, let \( \delta > 0 \) and \( \epsilon > 0 \). Suppose that

\[
\left| f \left( \overrightarrow{x} + \epsilon \left( \kappa_1 \overrightarrow{b}_i + \kappa_2 \overrightarrow{b}_j \right) \right) - f \left( \overrightarrow{x}+ \epsilon \left( \kappa_1 \overrightarrow{b}_i \right) \right) - f \left( \overrightarrow{x} + \epsilon \left( \kappa_2 \overrightarrow{b}_j \right) \right) \right| < \delta,
\]

for any \((\kappa_1)^2 + (\kappa_2)^2, 2 \leq 1\). Then we say that \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) are locally \( \epsilon \)-functionally independent to order \( \delta \). If \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) are locally \( \epsilon \)-functionally independent to order \( \delta \) \( \forall \epsilon > 0 \), we say that \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) are locally functionally independent to order \( \delta \). If, given \( \epsilon > 0 \), \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) are locally functionally independent to order \( \delta \) \( \forall \delta > 0 \), then we say that \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) are locally \( \epsilon \)-functionally independent. Finally, if \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) are both locally functionally independent to order \( \delta \) \( \forall \delta > 0 \), and locally \( \epsilon \)-functionally independent \( \forall \epsilon > 0 \), then we say that \( \overrightarrow{b}_i \) and \( \overrightarrow{b}_j \) are simply locally functionally independent.

Let us examine what these definitions mean. First, we investigate continuous maps. We see that continuity guarantees local functional independence to order \( \delta \). This means that for a genetic algorithm, any basis is equally good, as any basis can become functionally independent in a small enough hypersphere.

**Proposition 1** If a real function \( f : \Gamma \rightarrow \mathbb{R} \) is continuous and \( \theta = \left\{ \overrightarrow{b}_1, \ldots, \overrightarrow{b}_n \right\} \) is a unitary basis, then

\[
\text{given } \delta > 0, \exists \ \epsilon' > 0 \text{ such that } \overrightarrow{b}_i \text{ and } \overrightarrow{b}_j \text{ are locally } \epsilon' \text{-functionally independent to order } \delta. \]

**Proof:**

Suppose first that \( f \) is continuous. Now choose \( \delta > 0 \). Given this \( \delta \), \( \exists \ \epsilon > 0 \) such that whenever

\[
\left| \overrightarrow{x}' - \overrightarrow{x} \right| < \epsilon,
\]

we have that

\[
\left| f \left( \overrightarrow{x}' \right) - f \left( \overrightarrow{x} \right) \right| < \epsilon.
\]

Now choose \( \epsilon' < \frac{\epsilon}{\sqrt{2}}. \) Then

\[
\left| \epsilon' \left( \kappa_1 \overrightarrow{b}_i + \kappa_2 \overrightarrow{b}_j \right) \right| < \epsilon
\]

whenever \( \sum_{j=1}^{n} \left( \kappa_j \right)^2 = 1 \), so that

\[
\left| f \left( \overrightarrow{x} + \epsilon \kappa_1 \overrightarrow{b}_i \right) - f \left( \overrightarrow{x} \right) \right| = \Delta_1 < \delta',
\]

\[
\left| f \left( \overrightarrow{x} + \epsilon \kappa_2 \overrightarrow{b}_j \right) - f \left( \overrightarrow{x} \right) \right| = \Delta_2 < \delta',
\]

\[
\Rightarrow \left| f \left( \overrightarrow{x} + \epsilon \left( \kappa_1 \overrightarrow{b}_i + \kappa_2 \overrightarrow{b}_j \right) \right) - f \left( \overrightarrow{x} \right) \right| < \frac{\delta'}{4}
\]

Thus we have shown that any continuous function is locally \( \epsilon \)-functionally independent to order \( \delta \) for any \( \delta > 0 \). □

From this proposition, we see that continuity guarantees local functional independence to order \( \delta \). This means that for a genetic algorithm, any basis is equally good for local search, as any basis can become functionally independent in a small enough hypersphere.

The inverse implication, however, is not true in general. It is easy to construct a function which is not continuous but is \( \epsilon \)-functionally independent to order \( \delta \).

**Example 1** Let us consider the two-dimensional function \( f(x, y) \) defined by

\[
f(x, y) = \begin{cases} 
9 & x > 0; y > 0 \\
7 & x \leq 0; y > 0 \\
5 & x > 0; y \leq 0 \\
3 & x \leq 0; y \leq 0
\end{cases}
\]
over the basis \{(0,1), (1,0)\}. At (0,0) the function is clearly locally functionally independent, but not continuous.

This is quite important because it indicates that functional independence is more general than continuity. It is this property that the genetic algorithm exploits in its finding of schema.

Now, we can build structures out of functionally independent basis elements. These structures, called conjugate schema, form the basis for the parallel ability of the genetic algorithm.

**Definition 2** Given \( \epsilon, \delta > 0 \). Let \( A \) and \( B \) be two subsets of a basis for \( \Gamma \). Suppose that \( \forall \ b_i \in A, \) and \( \overline{b_j} \in B, \) \( b_i \) and \( \overline{b_j} \) are locally \( \epsilon \)-functionally independent to order \( \delta \). If also \( \forall \ b_i, \overline{b_j} \in A, \) \( b_i \) and \( \overline{b_j} \) are locally \( \epsilon \)-functionally independent to order \( \delta \), and likewise for all elements of \( B, \) then \( A \) and \( B \) are local conjugate schema to order \( \delta \). If \( \forall \ b_i, \overline{b_j} \in A, \) \( b_i \) and \( \overline{b_j} \) are locally \( \epsilon \)-functionally independent to order \( \delta \), and likewise for all elements of \( B, \) then \( A \) and \( B \) are local conjugate schema.

The concept of conjugate schema is perhaps the most important concept when considering how to make ‘useful’ crossovers. This grouping separates vectors into functional units. By mixing and matching these units, the genetic algorithm can combine high-scoring pieces of vectors. It is precisely this property of genetic algorithms which has been touted as the reason for the success of the genetic algorithm. However, it is the functional units, and not the values of the vector components, which define the parallel problem solving ability of the paradigm.

3 Conjugate Schema Create Parallizability

We have said that the genetic algorithm will be successful if it is capable of optimizing sections of vectors in parallel, and mixing and matching them. In this section, we will show that the conjugate schema do precisely this. When conjugate schema exist, the function may be written as a sum of lower-dimensional functions. However, the conjugate schema are dependent on the basis chosen. We present an example of the use of several bases, and illustrate why one is better than another.

Suppose first that \( f : \Gamma \hookrightarrow \mathbb{R} \) is a continuous function. We shall prove the following proposition, which can easily be extended to the more restrictive cases, and will not be done here.

**Proposition 2** Given \( f : \Gamma \hookrightarrow \mathbb{R} \), if \( \theta \) is a basis for \( \Gamma, \) and \( A \rightharpoondown B = \emptyset \) forms a separation of \( \theta, \) then if \( A \) and \( B \) are conjugate schema, then

\[
\begin{align*}
\n f &= g + h \\
\text{with } g : \langle A \rangle \hookrightarrow \mathbb{R} \text{ and } \langle B \rangle \hookrightarrow \mathbb{R} \text{ where } \langle A \rangle \cap \langle B \rangle = \emptyset.
\end{align*}
\]

Proof: Suppose everything is as given above. Then given any \( \epsilon, \delta > 0, b_i \in A, \) and \( \overline{b_j} \in B, \)

\[
\begin{align*}
\left| f\left( x + (\epsilon b_i + \delta \overline{b_j}) \right) - f\left( x + \epsilon b_i \right) \right| = 0
\end{align*}
\]

This means that

\[
\begin{align*}
&f\left( x + (\epsilon b_i + \delta \overline{b_j}) \right) + f\left( x \right) \\
&= f\left( x + \delta \overline{b_j} \right) + f\left( x + \epsilon b_i \right)
\end{align*}
\]

Which can be rearranged to read

\[
\begin{align*}
&f\left( x + (\epsilon b_i + \delta \overline{b_j}) \right) - f\left( x + \delta \overline{b_j} \right) \\
&= f\left( x + \epsilon b_i \right) - f\left( x \right)
\end{align*}
\]

and finally,

\[
\begin{align*}
&\frac{f\left( x + (\epsilon b_i + \delta \overline{b_j}) \right) - f\left( x + \delta \overline{b_j} \right)}{\epsilon} \\
&= \frac{f\left( x + \epsilon b_i \right) - f\left( x \right)}{\epsilon}
\end{align*}
\]

so that if we take the limit, we find that

\[
\frac{\partial f}{\partial b_i} | x + \delta \overline{b_j} = \frac{\partial f}{\partial b_i} | x
\]

or that the partial derivative is completely independent of any of the elements in the opposite conjugate schema. \( \Box \)
This proposition gives the motivation for noticing conjugate schema. It indicates that conjugate schema are the benchmark for separable functions, and that finding them allows one to mix and match parts of vectors without worrying about creating entirely new function values when mixing together different parts of a vector. This removes wild variations of the function value under crossover operations.

Let us consider a crossover operator2. Suppose that we have a population \( P \) and have identified conjugate schema \( \{ A_i \}_{i=1}^N \) over the basis \( \theta \). What we have said is that a conjugate schema divides the function into parts as

\[
 f = \sum_{i=1}^N f_i,
\]

where \( f_i : \Gamma_i \mapsto \mathbb{R} \) and \( \Gamma_1 \times \cdots \times \Gamma_N = \Gamma \). Now, let \( \overline{x}_{A_i} \) be the restriction of the vector \( \overline{x} \) to the subset \( A_i \) of \( \theta \), and let \( A_i \cap A_j = \emptyset \) when \( i \neq j \) and \( \bigcup_{i=1}^N A_i = \theta \). The crossover operator behaves in the following way. First,

\[
 \overline{x} = \sum_{A_i} \overline{x}_{A_i},
\]

and

\[
 \overline{y} = \sum_{A_i} \overline{y}_{A_i}.
\]

Then \( Cr(\overline{x}, \overline{y}) = \left( \overline{x}_j, \overline{y}_j \right) \), where \( \overline{x}_j \) and \( \overline{y}_j \) are defined by

\[
 \overline{x} = \sum_{i \neq j} \overline{x}_{A_i} + \overline{y}_{A_j}
\]

and

\[
 \overline{y} = \sum_{i \neq j} \overline{y}_{A_i} + \overline{x}_{A_j}.
\]

Let us make this somewhat more concrete with an example.

**Example 2** Suppose that we have a single-point crossover operator defined in the following way.

Let \( \overline{x} = (a_1, \ldots, a_N) \) and \( \overline{y} = (b_1, \ldots, b_N) \) be vectors found in \( \Gamma \). Let

\[
 Cr(\overline{x}, \overline{y}) = \left( \begin{array}{c}
 (a_1, \ldots, a_{j-1}, b_j, a_{j+1}, \ldots, a_N) \\
 (b_1, \ldots, b_{j-1}, a_j, b_{j+1}, \ldots, b_N)
 \end{array} \right)
\]

where \( j \) is a randomly distributed variable taking on the values \( 1, \ldots, N \) with equal probability. The conjugate schema assumed here are the singleton sets \( \{0, \ldots, 0, 1, 0, \ldots, 0\} = A_j \) and the basis assumed here is \( \theta = \{(1, 0, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 0, 1)\} \), the canonical basis. We call these conjugate schema the canonical conjugate schema.

The crossover operator will only be useful if it does not destroy information by taking part of a vector and replacing it with another part, without regard to which part is being replaced. Only independent (functionally independent) parts should be crossed over (as groups) and smaller or larger parts should be avoided. Any other crossover should be regarded as part of the mutation operator.

Let us now consider another example which serves to illustrate this last point more clearly.

**Example 3** Suppose that the problem is given by

\[
 \Gamma = \mathbb{R}^3,
\]

\[
 f(\overline{x}) = x_1 x_2 + x_3.
\]

Now let \( \overline{v}_1 = (4, 4, 3) \) and \( \overline{v}_2 = (1, 1, 16) \) be two vectors in \( \Gamma \). Note that these have function values that are comparable: \( f(\overline{v}_1) = 19 \) and \( f(\overline{v}_2) = 17 \). Then with single-point crossover, there are three possibilities:

1) \( Cr(\overline{v}_1, \overline{v}_2) = [(1, 4, 3), (4, 1, 16)] \),

2) \( Cr(\overline{v}_1, \overline{v}_2) = [(4, 1, 3), (1, 4, 16)] \), and

3) \( Cr(\overline{v}_1, \overline{v}_2) = [(4, 4, 16), (1, 1, 3)] \).

giving the function values

1) \( f(\overline{v}_1) = 7, f(\overline{v}_2) = 20 \),

2) \( f(\overline{v}_1) = 7, f(\overline{v}_2) = 20 \), and

3) \( f(\overline{v}_1) = 32, f(\overline{v}_2) = 4 \).

While in all cases, a superior vector was created, only the last case produced a particularly high functional value. The conjugate schema assumed here were the canonical conjugate schema. On the other hand, if we had assumed that the sets \( A = \{(1, 0, 0), (0, 1, 0)\} \) and \( B = \{(0, 0, 1)\} \) were the conjugate schema, we would have reproduced the last crossover in one step. This serves to both cement the conjugate schema approach,
and to explain why it is that the single-point crossover operator is sometimes quite useful in the genetic algorithm paradigm. Problems are often written in such a way that such a crossover operator operates by design in the conjugate schema, and in other cases, problems are, by mistake, exploitable by such crossover operators as the functional dependency of different basis elements is quite important.

Although it is clear that the conjugate schema is important, we have not yet established a way of finding conjugate schema. It is not true that every basis chosen will produce a conjugate schema. In fact, in general, it is more likely that there will be no conjugate schema in any basis found. We will illustrate this by extending the last example.

**Example 4** Let us again examine the same function. This time, however, let the basis be given by \( \vartheta = \left\{ \frac{1}{\sqrt{2}} (1, 1, 0), \frac{1}{\sqrt{2}} (1, -1, 0), (0, 0, 1) \right\} \). Then, let
\[
y_1 = \frac{1}{\sqrt{2}} (x_1 + x_2), \quad y_2 = \frac{1}{\sqrt{2}} (x_1 - x_2), \quad \text{and} \quad y_3 = x_3.
\]
Then \( f(y) = \frac{1}{2} (y_1^2 - y_2^2) + y_3 \). In this new basis, the vectors are given by
\[
\overrightarrow{v}_1 = (4\sqrt{2}, 0, 3), \quad \text{and} \quad \overrightarrow{v}_2 = (\sqrt{2}, 0, 16).
\]
Now, then the crossover operator produces three operations:

1) \( Cr(\overrightarrow{v}_1, \overrightarrow{v}_2) = \left( \left[ \sqrt{2}, 0, 3 \right], \left( 4\sqrt{2}, 0, 16 \right) \right) \),

2) \( Cr(\overrightarrow{v}_1, \overrightarrow{v}_2) = \left( \left[ 4\sqrt{2}, 0, 3 \right], \left( \sqrt{2}, 0, 16 \right) \right) \), and

3) \( Cr(\overrightarrow{v}_1, \overrightarrow{v}_2) = \left( \left[ 4\sqrt{2}, 0, 16 \right], \left( \sqrt{2}, 0, 3 \right) \right) \).

In this case,

1) \( f \left( \frac{v'_1}{v_1} \right) = 44, f \left( \frac{v'_2}{v_2} \right) = 7 \),

2) \( f \left( \frac{v'_1}{v_1} \right) = -5, f \left( \frac{v'_2}{v_2} \right) = 18 \), and

3) \( f \left( \frac{v'_1}{v_1} \right) = 7, f \left( \frac{v'_2}{v_2} \right) = 20 \).

All three cases are distinct in their final functional values. Although the last case produced the ‘best’ functional value, it is not clear what caused this change, and the change can be attributed to a mutation which luckily happened to occur in the correct direction. The apparent success can be traced back to the function, which, in our first basis, is monotonically increasing in all three variables. In the example of the completely independent basis, each crossover’s success and failure can be easily predicted and controlled. We must try to create a procedure to find the basis under which crossover is most useful and controlled, and limit our mutations, then to the conjugate schema.

This example has shown that the crossover operator’s success is intimately tied to the basis chosen for the representation of the problem. In most applications of the genetic algorithm paradigm, the basis chosen is the canonical one. This basis may not always be the appropriate basis for search. In the next section, we consider the problem of producing conjugate schema.

### 4 Finding Bases Containing Conjugate Schema

In this section, we will develop theory pertaining to possible methods for finding conjugate schema. We begin with a discussion of the minimization of the functional D introduced in section 2, and finish with a derivation of the eigenvectors of the Hessian matrix for continuous problems as being the basis with the greatest local conjugate schema.

Let us suppose that \( A \) is a linear operator over the linear space \( \Gamma \). Then, also let \( \{ \psi_1, \psi_2, \psi_3, \ldots \} \) be
the set of all eigenvectors of the operator $A$, and \{$k_1, k_2, k_3, \ldots $\}. We assume that $k_i \neq k_j, \ i \neq j$, and $k_i \neq 0 \ \forall i$. Suppose that this linear space contains a zero element $0$ such that $\forall x \in \Gamma$, $x + 0 = x$, and the $Ax = 0$. Now, suppose that there also exists an inner product $\langle \cdot, \cdot \rangle$ defined on the space such that $\langle \cdot, \cdot \rangle: \Gamma \times \Gamma \rightarrow \mathbb{R}$, and obeys all of the standard rules of inner products\footnote{1}. Let

$$f(x) = \langle Ax | x \rangle$$

Then, we have that

$$D = \begin{bmatrix} f(0) + f(\psi_i) \\ -f(\psi_j) - f(\psi_i + \psi_j) \end{bmatrix}$$

where $a$ and $b$ are real constants. More importantly, if $\varphi = a\psi_1 + b\psi_2$ and $\phi = b\psi_1 - a\psi_2$ then

$$f(\varphi) = f(a\psi_1 + b\psi_2) = k_1 a^2 + k_2 b^2$$

$$f(\phi) = f(b\psi_1 - a\psi_2) = k_1 b^2 + k_2 a^2$$

$$f(\varphi + \phi) = f((a + b)\psi_1 + (b - a)\psi_2) = k_1 (a + b)^2 + k_2 (b - a)^2$$

so that

$$D = [2ab(k_1 - k_2)]$$

which will, in general, be zero iff $k_2 = k_1$. Thus, the minimization of $D$, moving to completion, will pick out the eigenvectors of this operator, $A$. We have just proved the following proposition:

**Proposition 3** Given a linear operator $A$ over a linear space $\Gamma$, 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 $A$.

We turn now to the nature of the minimization of $D$ over the function $f: \mathbb{R}^n \rightarrow \mathbb{R}$. We suppose, first, that $f$ is a continuously differentiable function.

**Proposition 4** A minimization of $D$ picks out the eigenvectors of the matrix given by $H = \left[ \frac{\partial^2 f(\varphi)}{\partial x_i \partial x_j} \right]_{ij}$

**Proof.**

$$D_{ij} = \begin{bmatrix} f(\varphi) + f(\phi) & f(\varphi + \phi) \\ -f(\psi_i) - f(\psi_j) & f(\psi_i + \psi_j) \end{bmatrix}$$

Since

$$f(\varphi) - f(\phi) \cong -\varepsilon_i \frac{\partial f(\varphi)}{\partial x_i}$$

for sufficiently small $\varepsilon_i$ and $\varepsilon_j$, so

$$f(\varphi) \cong f(\psi_i) + f(\psi_j) + \varepsilon_i \frac{\partial f(\psi_i)}{\partial x_i}$$

we have

$$D_{ij} \cong \left[ \frac{\partial^2 f(\varphi)}{\partial x_i \partial x_j} \right]_{ij}$$

This shows that for differentiable functions, the matrix whose eigenvectors are being sought is

$$H = \left[ \frac{\partial^2 f(\varphi)}{\partial x_i \partial x_j} \right]_{ij}$$

or the curvature. When such continuous methods are not available, or one wishes to avoid wild variations of the function biasing the conjugate schema, one may employ approximation methods\footnote{5} in which conjugate schema are numerically generated. Such methods are quite successful in improving searches in a variety of problems.

Let us consider a simple example. Suppose that our function is $f(x, y) = xy$. This can be rewritten in the form $f(x, y) = \frac{1}{2} \left[ (x + y)^2 - (x - y)^2 \right]$. From this, it is immediately obvious that the problem can be reduced in dimension if we work in the basis

$$\frac{1}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ x \\ y \\ -y \end{pmatrix}$$

If we use our definition of $H$, we find that

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

and that its eigenvectors are $(1, 1)$ and $(1, -1)$, yielding the vectors $\frac{1}{\sqrt{x^2 + y^2}} (x, y)$ and $\frac{1}{\sqrt{x^2 + y^2}} (x, -y)$, as determined by hand.

Another example is the function $f(w, x, y, z) = wx + yz$, Hessian is

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

and the eigenvectors are $\frac{1}{\sqrt{x^2 + y^2}} (w, x, 0, 0)$, $\frac{1}{\sqrt{y^2 + z^2}} (w, -x, 0, 0)$, $\frac{1}{\sqrt{x^2 + z^2}} (0, 0, y, z)$, and $\frac{1}{\sqrt{y^2 + z^2}} (0, 0, y, z)$.\footnote{2}
Finally, a third example is the function \( f(x, y, z) = xyz \). In this case, the Hessian is
\[
\begin{pmatrix}
0 & z & y \\
z & 0 & x \\
y & x & 0
\end{pmatrix}
\]
and the eigenvectors are \( \frac{1}{\sqrt{2}} (x^2 - y^2 + z^2, yz - xy, yz - xy, z) \). where \( \rho \) is a root of \( (x^2 - y^2 - z^2) \), \( \rho = 2xyz \) and \( N \) is a normalization factor, which may be solved for any given set of values.

At this point, it is important to note that what we have said here does not apply to only genetic algorithms. As much as this is a general fact, it applies to all stochastic and non-stochastic processes. With this in mind, it may behoove us to implement such a calculation along the way on other optimization procedures.

5 Discussion

The importance of this work is straightforward. Since the seminal work of Holland[5], and the creation of the Schema Theorem, researchers have puzzled over the implications of the Schema Theorem. Many have indicated that while the Schema Theorem describes how the schema behave, and the importance of so-called 'building blocks'[2], no robust general methods exist to create building blocks \emph{a priori}. This vacuum is a severe detriment to the genetic algorithm, and more importantly, the evolutionary computation community.

What we have succeeded in doing here is taking one step in the direction of creating building blocks \emph{a priori}. Taking advantage of conjugate schema allows one to use 'a different set of tools' during different parts of search[7]. This set of tools is defined by the search, and when global, may provide researchers with adaptive tools for reducing dimensionality.

Many researchers have lamented that it is currently impossible to add something of the 'flavor' of the problem to the search algorithm, and so it has become impossible to distinguish one search algorithm from another, as their average behavior will be identical. However, the use of the building block approach to optimization does precisely that. We will examine how this is so with a few examples.

Suppose that one is concerned with building a protein folding simulation in which a particular protein is being folded into its most compact form. This folding simulation may be built in one of many different ways, one being the use of continuous parameters defining inter bond distances and positions. In this model, it may be possible that in a given intermediate state of the fully folded protein, one might have a particular step to accomplish which might entail the concerted motion of more than one element. In nature, this is very easy to do, but computationally, this is often quite difficult. However, by finding the basis with the maximal number of conjugate schema, the chances of a simple multi-atom beneficial mutation may increase a great deal.

The immediate criticism of this technique is that it is not applicable to a wide range of problems, since they are not continuous, and so the calculation of the Hessian matrix will have no meaning. While this is certainly a valid concern, our preliminary studies of the use of this technique in protein folding have shown this to be less of a concern than previously thought[4]. Moreover, a suitable embedding of any discrete space into a continuous function might produce conjugate schema that are useful. The embedding would have to be carefully chosen. For instance, a simple linear extrapolation of the points given would produce a surface that might yield the correct (or at least useful) conjugate schema.

While this formalism has been developed with a genetic algorithm in mind, it is clear that this formalism easily extends to other stochastic search algorithms for which representational issues are important. For instance, in the simple function mentioned above
\[
f(x, y) = xy
\]
even the efficiency of a hillclimbing method would be improved by using the vectors indicated by our method. Possible uses for this formalism include Monte Carlo integration using the Metropolis et. al. method[6]. The method requires the creation of a non-correlated data set. The use of the conjugate schema may allow the correlations across conjugate schema to be partially eliminated, yielding 'cleaner' data sets, and improving the efficiency of the method.

For genetic algorithms, this generation of conjugate

---

4A quick calculation might show the difficulty. If three residues must be moved in tandem, each one might be carried out with a probability \( p \), making the probability of the useful mutation \( p^3 \). If \( p = 0.1 \), this lowers the probability of a useful mutation to 1 out of 1000. However, if the basis already moved all three in tandem, then the probability would simply be 10\%, a difference of two orders of magnitude. In real systems, however, it is realistically required to move merely ten or twenty residues at once, due to bonding constraints, etc. This would then cause, if correctly implemented, an alteration in the probability of a useful mutation of the order of 10- or 20-fold.
schema provides another service. We consider this to be a fundamental delineation of the search paradigm. In one fell swoop, the creation of conjugate schema alters the search algorithm to a search-sort algorithm. Let us see how. When all crossovers are done in the conjugate schema, there is no added information in the crossover. Rather, what happens is that one crossover combines different blocks of the vectors together. The reproduction operator will eliminate those vectors that contain 'bad' parts, on average, leaving only the 'good' schema, as the Schema Theorem indicates. In this way, the crossover and reproduction together become a sorting algorithm. Now, if the mutations are done in the conjugate schema, this means that each mutation is carrying out a search in a smaller dimensional space, the dimension of which is defined by the cardinality of the conjugate schema in which this mutation is taking place. Thus, the use of conjugate schema at once reduces the dimensionality of the problem, and reduces the concerted use of crossover and reproduction to a sorting algorithm.

Other uses for this type of analysis are likely to arise. As a means of answering the fundamental question of finding building blocks, this is a promising first step.

6 Acknowledgments

I would like to thank Dr. Jerry Solomon for his encouragement and patience during this venture. Dr. Solomon provided both the location for this work to take place, and a useful backboard for bouncing around interesting ideas. His interest in protein folding as a concrete application helped to solidify the work into a real use. Most importantly, Dr. Solomon gave me time and freedom to explore.

I would like to thank Scott Page for his assistance in this work. It was Dr. Page who introduced me to this topic originally, and many discussions of the subject with him helped me to understand the topic more clearly. Dr. Page also provided me with valuable advice during the writing of this paper, and even aided in the more mundane tasks, which included proofreading the paper.

Finally, I would like to thank my wife, Jisoo Park, for enduring my long hours and providing my inspiration during this endeavor.

References


