LA-UR- 97-454
CONF-970740--3

From DNA To Protein: Transformations And Their Possible Role In Linkage Learning

Hillol Kargupta*and Brian Stafford

Computational Science Methods Group
X Division, Los Alamos National Laboratory
P.O. Box 1663, MS F645
Los Alamos, NM, 87545

Los Alamos National Laboratory Technical Report LAUR-96-XXXX

Abstract

This paper first extends the traditional perspective of *linkage* using the basic concepts developed in the SEARCH framework (Kargupta, 1995; Kargupta & Goldberg, 1996) and identifies the fundamental objectives of linkage learning. It then explores the computational role of gene-expression (DNA→RNA→Protein transformations) in evolutionary linkage learning, using group representation theory. It offers strong evidence to support the hypothesis that the transformations in gene-expression define a group of symmetry transformations that leaves the fitness invariant; however, they change the eigen functions leading to identifying independent subspaces of the search space (a major objective of linkage learning) using irreducible representations of such transformations.

1 Introduction

Intra-cellular expression of genetic information in a living organism plays a critical role in the emergence of different forms of life. Different regions of DNA—the career of genetic information—are *transcribed* in different cells of an organism for producing messenger RNA (mRNA). Messenger RNA sequences are in turn *translated* to produce proteins, which are responsible for almost every activity of a living being. The transformation of the information coded in DNA to the proteins is often called *gene expression*. Little attention has been paid to understand the quantitative role of this intra-cellular flow of genetic information in evolutionary search. Almost all state of the art evolutionary algorithms acknowledge very little computational importance of gene expression.

In this paper we first offer a perspective of linkage learning as identifying delineable relations (Kargupta & Goldberg, 1996) (relations that are inherently appropriate for constructing an ordering among its classes, defined over the search space) in subspaces of the search space. In other words, the task of linkage learning is decomposed in (1) decomposing the search space into different independent subspaces and (2) detecting appropriate relations defined within the subspaces. Next we relate these steps of linkage learning with the series of transformations introduced by the gene expression process and develop a possible perspective of linkage learning in natural evolution using group representation theory.

Section 2 briefly reviews previous work on linkage learning. Section 3 develops the general perspective of linkage using basic concepts of the SEARCH framework. Section 4 first briefly describes the basic steps of gene expression. After reviewing the concepts of group representation theory, the transformations in gene expression are related to the issues of linkage learning.

2 Previous Work

Genetic linkage if often viewed in both genetic algorithm (GA) literature (Goldberg, Korb, & Deb, 1989; Harik & Goldberg, 1996; Holland, 1975) and evolutionary biology (Alberts, Bray, Lewis, Raff, Roberts, & Watson, 1994) as the non-linear epistatic relations among different variables (often called *genes* in the biological context). The scope of linkage learning is often described as identifying the cluster of epistatically related genes. Although the major bulk of the GA literature failed to pay attention to the issue of linkage learning, a significant amount of effort has been

*The author can be reached by email to hillol@lanl.gov

made toward understanding the linkage issue by a relatively closed group of researchers. Goldberg, Korb, and Deb (1989) was first to pay serious attention to linkage issue. They developed the so called *messy genetic algorithms* which targeted learning linkage in problems that are quasi-decomposible. In this effort the subspaces were defined by equivalence classes defined over the representation and they are deterministically enumerated. Equivalence classes were evaluated using a "competiive template" (Deb, 1991). One of the major problem of this approach was the computational cost of explicit enumeration of classes. The *fast messy GA*(fmGA) (Goldberg, Deb, Kargupta, & Harik, 1993; Kargupta, 1995) proposed a probabilistic and approximate way to learn linkage. Although fmGA successfully reduced the computational cost of the messy GAs for some large problems linkage learning in fmGA may be sub-optimal. Kargupta (1995, Kargupta (1996b, Kargupta (1996a) reported the so called *gene expression messy GA* which used a local perturbation based algorithm to determine locally optimal classes. Although the complexity was subquadratic, it is not quite clear how general the technique for determining subspaces is.

Most of the above mentioned techniques of linkage learning were in a way based on heuristics. The present work offers an analytically well grounded approach that follows the emphasis of gene expression developed elsewhere (Kargupta, 1995; Kargupta, 1996b). However, before we do that, let us first understand the scope of linkage learning and its fundamental need in blackbox optimization. The following section does that using the SEARCH framework developed elsewhere (Kargupta, 1995; Kargupta, 1996a).

3 Linkage And Optimization

As mentioned earlier, traditionally linkage is viewed from the problem perspective, as the epistatic relation between the problem variables, often represented by one or a collection of genes in genetic algorithms. However, this intuitive definition may not be sufficient to understand the fundamental objectives of linkage learning in the context of optimization.

The SEARCH (Search Envisioned As Relation and Class Hierarchizing) framework developed elsewhere (Kargupta, 1995; Kargupta & Goldberg, 1996) offered an algorithm perspective of linkage in terms of the relations introduced by the representation. In this section, we first briefly review the decomposition of search space proposed by the SEARCH framework into relation, class, and sample spaces. This will be followed by a section, identifying scope of linkage learning.

3.1 Decomposition of blackbox optimization search space in SEARCH

Foundation of SEARCH is based on the fact that induction is an essential part of blackbox optimization (BBO), since in absence of any analytic information about the objective function structure, a BBO algorithm must guess based on the samples it takes from the search space. SEARCH also notes that induction is no better than table look up unless we restrict a finite set of relations among the search space members. If relations are important to consider, then we should pay careful attention to determine which relation is "appropriate" and which is not. Let us say, we have a set of people sitting in a room and we would like to identify the person with highest amount of money in his/her pocket. If we want to do any better than enumeration, i.e. exhaustively picking every person in the room and checking his pocket for the amount of money he or she has, we must make intelligent guesses by observing certain features of the people (e.g. quality of the dress, shoe etc.). If we consider "all possible features" we are again back to enumeration (Watanabe, 1969; Mitchell, 1980). We must consider a certain finite set of features that defines the bias of the process. Features, like quality of dress define relations among the set of people. Depending on what we mean by the "quality of the dress", such relation may divide the set of people into different classes, such as people with cheap dresses, people with very expensive dresses and so on. We consider hypotheses defined by the feature set, use it to divide the search space into different classes, and evaluate hypotheses using samples taken from the search domain. The decomposition of BBO in SEARCH in terms of relation, class, and sample spaces essentially captures this idea. [1] Note that, the search for relations is essential, since some relations are inherently good and some are not. For example, "quality of the dress" may be a good one; however, "color of the hair" may not be a good relation for this problem. In SEARCH, such relations that are inherently good for decision making are said to *properly delineate* the search space. If we construct an ordering among the classes, defined by a relation of order k (the logarithm of this set of classes), in order to select the high ranked classes for further exploration and the class containing the optimal solution is one among those selected classes, then we say that order-k relation properly delineates the search space. The search for appropriate relations and classes can be viewed

[1] A relation is defined as a set ordered tuples. A class is a tuple of elements taken from the domain under consideration. In this paper we will primarily be concerned with tuples taken from space of n-ary Cartesian products of the search domain with itself.

DISCLAIMER

as a stochastic decision making processes in the relation and class spaces respectively. SEARCH offers a general probabilistic and approximate framework to do that. If the relation space provided a priori to the search algorithm contains all the relations needed to solve a problem and the order of all of these suitable relations is bounded by some constant k, then the given problem can be solved in sample complexity (can be loosely defined as the number of samples taken for solving the problem) polynomial in problem size, solution quality, success probability. This class of problem is called the class of order-k delineable problems.

Although, SEARCH has different implications on different grounds, in this paper we need only the following implications of SEARCH:

1. decomposition of BBO in relation, class, and sample spaces
2. assumes that the relation space is defined a priori.
3. once we fix the relation space, an algorithm can only solve the class of order-k delineable problems.

Searching for appropriate relations plays an important role in SEARCH. The following section relates this aspect of SEARCH with linkage learning.

3.2 Scope of linkage learning

The objective of linkage learning is to identify the non-linearly related variables and use them together to define a decision variable. This offers a way to linearly decompose the given set of optimization variables into different subsets, each corresponding to a unique decision variable. This essentially decomposes the global decision making process in the search into different linearly decomposable sub-problems. This idea is rigorously captured by the notion order-k delineability. Non-linearly related genes can be used to define k-dimensional subspaces and relations defined within these subspaces that properly delineate the search space can be used to make correct search decisions. Let us take an example. Consider a sequence representation of four binary variables x_1, x_2, x_3 and x_4. If these variables can be decomposed into two linearly decomposable subsets (x_1, x_2) and (x_3, x_4), then each of them define subspaces which may contain relations that properly delineate the search space. Note that, we may need relations other than simple partitions such as $ff\#\#$, where f denotes the position of equivalence. This is because, the classes defined by such partitions are hyperplanes, either parallel or orthogonal to the axes of reference; for some objective functions such limitations may be be okay. However, in the general case we may not want such restrictions in order to efficiently capture better regions of the search space in classes.

The task of identifying relations of some bounded order-k that properly delineates the search space, can be decomposed into two modules: (1) decomposing the problem into linearly separable subspaces (2) identifying proper relations in these independent subspaces. In this paper we will be primarily concerned with the former objective. The coming sections lays the foundation of a technique for identifying these subspaces and use it to explain the computational role of different transformations of gene expression in living organisms.

4 From DNA To Protein

Information flow in evolution is primarily divided into two kinds:

- **extra-cellular flow**: storage, exploration, and transmission of genetic information from generation to generation;
- **intra-cellular flow**: expression of genetic information within the body of an organism.

The extra-cellular flow involves replication, mutation, recombination, and transmission of DNA (deoxyribonucleic acid) from parents to offspring. On the other hand the intra-cellular flow of information involves transcription and translation of genetic information leading to the computation of the phenotype of the organism. Information flow along these streams depend on each other. Although genetic linkage plays important roles in both extra (e.g. defining the crossing-over sites) and intra-cellular flow, there are strong reasons to believe that development of genetic linkage takes place primarily during the intra-cellular flow. In this section we will primarily be concerned with gene expression—the intra-cellular flow of information.

4.1 DNA, RNA, and Protein

DNA molecule consists of two long complementary chains held together by base pairs. DNA consists of four kinds of bases joined to a sugar-phosphate backbone. The four bases in DNA are *adenine* (A), *guanine* (G), *thymine* (T) and *cytosine* (C). Chromosomes are made of DNA **double helices**. Bases on DNA helices obey the *complementary base pairing rule.* T and G pair with A and C respectively. In other words, if the base at a particular position of a helix is T then the corresponding base in the other helix should be A.

Expression of genetic information coded in DNA into the proteins takes place through several complicated steps. However, the major distinct phases are identified as

- transcription: formation of mRNA (ribonucleic acid) from DNA
- translation: formation of protein from mRNA

Messenger RNA (ribonucleic acid) consists of four types of bases joined to a ribose-sugar-phosphodiester backbone. The four bases are *adenine* (A), *uracil* (U), *guanine* (G), and *cytosine* (C). All the bases defining the RNA are same as those in DNA sequences, except that T is replaced by U. DNA produces mRNA using the RNA Polymerase and the regulatory proteins following the **complementary base-pairing rules** similar to those in DNA.

Messenger RNA acts as the template for protein synthesis. Proteins are sequence of *amino acids*, joined by peptide bonds. Messenger RNA is transported to the cell cytoplasm for producing protein in the ribosome. There exists a unique set of rules that define the correspondence between nucleotide triplets (known as codons) and the amino acids in proteins. This is known as the **genetic code**. Each codon, comprised of three adjacent nucleotides in a DNA chain, produces a unique amino acid. Although sequence of amino acids fundamentally defines proteins, formation of the three dimensional structure of proteins involve a complex non-linear process, which is often called *protein folding*. This process involves interaction between multiple amino acid subsequences. Current understanding of the process can reasonably predict the nature of secondary interaction structure among amino acids. However, the nature of higher order interactions, such as tertiary structure among amino acids is little understood. The following section views different steps of gene expression in the light of linkage learning.

Alanine	GCA GCC GCG GCU
Cysteine	UGC UGU
Aspartic acid	GAC GAU
Glutamic acid	GAA GAG
Phenylalanine	UUC UUU
Glycine	GGA GGC GGG GGU
Histidine	CAC CAU
Isoleucine	AUA AUC AUU
Lysine	AAA AAG
Leucine	UUA UUG CUA CUC CUG CUU
Methionine	AUG
Asparagine	AAC AAU
Proline	CCA CCC CCG CCU
Glutamine	CAA CAG
Arginine	AGA AGG CGA CGC CGG CGU
Serine	AGC AGU UCA UCC UCG UCU
Threonine	ACA ACC ACG ACU
Valine	GUA GUC GUG GUU
Tryptophan	UGG
Tyrosine	UAC UAU
STOP	UAA UAG UGA

Table 1: Genetic code.

4.2 Symmetries in gene expression

Like many other natural processes, steps of gene expression are characterized by different symmetric structures and operations. Let us spend a little time recalling some of these important symmetric properties.

DNA double helix is comprised of the two complementary chains of nucleic acid bases. The notion of complementary base pairs exists due to the fact that (T→A, A→T) and (C→G, G→C). These pairs define two disjoint cyclic permutations over the set of four nucleic acid bases. Similarly, the DNA→mRNA mapping exhibits cyclic pairs (T, U) and (C, G). The genetic code that maps the mRNA into the amino-acid sequence in protein, also offers interesting symmetry properties. Figure 1 tabulates the nucleic acid codons and their corresponding amino acids. Note that most of the rows of the table have multiple codons listed against one amino acid. For example, the first row shows that GCA, GCC, GCG, GCU—all of them maps to Alanine. In other words, this set of four codons offers an invariant transformation to the mRNA. Since the "fitness" of a living organism depends on its protein structure, which is determined by the amino acid sequence in the protein, the "fitness" remains invariant if any member of the set of four codons is replaced by another member. Such transformations are called *fitness invariant symmetry* transformations. Formally speaking if $\phi(\mathbf{x})$ be an arbitrary function, $\mathbf{x} = T\,\mathbf{X}$, where T is a linear transformation, and $\phi(\mathbf{x}) = \phi(\mathbf{X})$, then we say T is a fitness invariant symmetry transformation. Although such transformations keeps $\phi(\mathbf{x})$ invariant, they *does not* in general keep the eigen functions invariant. Ψ, an eigen function of the operator ϕ is a state function that satisfies $\phi(\Psi) = E\Psi$, where the values of E are the eigen values. In the coming sections these fitness invariant symmetry transformations will play an important role.

Capturing the abundance of symmetries in gene expression is a challenging task. However, group theory offers some interesting tools to deal with symmetry in both physical and abstract systems. Group theory

has been successfully used for exploiting symmetries in quantum mechanics (Heine, 1993). Group theory can also be used to study the computational rationale behind the transformations in gene expression. The following section presents a brief review of the necessary concepts of group theory used in this paper.

4.3 A brief review of group theory

Groups are best explained in terms of their definition. After the following definition, a series of examples will show some groups and lead into their desired application. A group is a set of elements and an operation on the elements with four properties. Say we have the set $\mathcal{E} = \{a, b, c, ...\}$ and an operation $\otimes$. We can write any table for $\otimes$ which has the following properties:

1. $\otimes$ must be closed:
   For any x_1, x_2 in $\mathcal{E}$, $x_1 \otimes x_2 = x_3$ where x_3 must be in $\mathcal{E}$.
   This means that the result of using the operation on $\mathcal{E}$'s elements must also be in $\mathcal{E}$.

2. $\otimes$ must be associative:
   For any x_1, x_2, x_3 in $\mathcal{E}$,
   $(x_1 \otimes x_2) \otimes x_3 = x_1 \otimes (x_2 \otimes x_3)$.
   When the operation is used more than once, it does not matter which operation is performed first.

3. There is a unique identity element (usually written as e):
   For any x_1 in $\mathcal{E}$, $x_1 \otimes e = x_1 = e \otimes x_1$.
   In arithmetic, 0 is the identity for the addition operator and 1 is the identity for the multiplication operator.

4. Every element has a unique inverse:
   For each x_1 in $\mathcal{E}$, there is a single x_2 in $\mathcal{E}$ such that $x_1 \otimes x_2 = e = x_2 \otimes x_1$.
   An element operated with its inverse equals the identity.

As a small example of a group, take a set of two elements $\mathcal{E} = \{e, a\}$. By keeping the properties of $\otimes$ in mind, an operation table can be written. Let e be the identity and note that $e \otimes e = e$ satisfies the third and fourth properties, showing that e is always its own inverse. As e is the identity, $a \otimes e = a = e \otimes a$, by the third property. The only remaining combination is $a \otimes a$. As we haven't found an inverse for a, $a \otimes a$ must be a. As it turns out, there is only one possible way to define $\otimes$ for a set of two elements once the identity has been chosen. The operation table is shown in the following:

$\otimes$	e	a
e	e	a
a	a	e

There is a simple theorem which can explain the most important aspects of groups.

Theorem 1 (Cayley's Theorem) *Every group is isomorphic to a group of permutations.*

Here, *isomorphic* is the notion of 'equal', but with care to remember that sets are not ordered. In general, if a group is defined by a set $\mathcal{E} = \{a_1, a_2, \ldots, a_n\}$ and an operation $\otimes$ on those n elements, then the group can impose its structure on another set $\mathcal{E}' = \{b_1, b_2, \ldots, b_n\}$ or just its original set in a different order. In the case of a new set, each element of $\mathcal{E}'$ is associated with one element of $\mathcal{E}$, not leaving anything un-associated in $\mathcal{E}$. Thus we can define a group on one set and let it 'act' on another set of the same size. A *group of permutations* is just a group defined over a set of *permutations* (functions that reorder elements in an ordered sequence). So the theorem says that any group on a set of n elements can 'act' on a set of n permutations — with the implication that its operator $\otimes$ will be the same as the way permutations are defined to be multiplied.

The important points are that a group can act on any set of the same size as the set of its original definition. And that a group can be thought as a set of functions which manipulate the order of an ordered list, with $\otimes$ serving to combine a sequence of such functions. An interesting corollary is that every group is a subgroup of a *symmetric group*. For this paper, the import is that every group behaves (acts) the same as one of the ways a symmetric group acts. A symmetric group captures all the symmetries in the most perfectly symmetric way to arrange a number a points. A graphical example will be developed next.

Groups evolved as a means to express the symmetries in a problem. And especially to use symmetries to simplify problems. The power of behind the idea of groups is that a group can act on a set of functions. By studying functions which preserve the shape of a geometric figure, the symmetries of a figure can be expressed. The boon is that these functions can be used to manipulate a figure without changing its essential nature. As an example consider an equilateral triangle.

There are three points, P_1, P_2, P_3, connected by equal length segments. The identity transformation (function), just leaves the triangle unchanged. The next type of transformation is a rotation by a multiple of 120°. The set of three rotations $\mathcal{E} = \{0° = 360°, 120°, 240°\}$ forms a 'rotation' group with 0° as the identity, and $\otimes$ serving to add rotations.

$\otimes$	0°	120°	240°
0°	0°	120°	240°
120°	120°	240°	0°
240°	240°	0°	120°

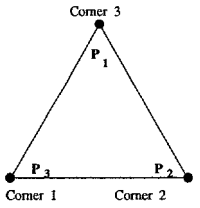

Figure 1: .

The equilateral triangle has the symmetry of a 120° angle between points on the corners. Another symmetry is that any two adjacent corners can be exchanged by flipping the triangle. There are three possible pairs of corners, $(12), (13), (23)$. If we use the notation that (1) means to leave the triangle unchanged, and (12) means to flip the triangle to exchange the lower two points, then $\mathcal{E} = \{(1), (12)\}$ and the following $\otimes$ form a group.

$\otimes$	(1)	(12)
(1)	(1)	(12)
(12)	(12)	(1)

Note that the (12) notation always refers to the bottom corners rather than points P_1, P_2. The combination of the identity, the rotations, and the three flips form a set $\mathcal{E} = \{0° = (1), 120° = (123), 240° = (132), (12), (13), (23)\}$. The *permutation* notation (123) means to move the point at corner 1 to corner 2, the point at corner 2 to corner 3, and wrapping around the two ends of (123) means to move the point at corner 3 to the point at corner 1. Thus (123) can mean the same as rotating 120° about the center. If we use the elements of $\mathcal{E}$ in permutation notation, then the group with $\mathcal{E}$ and $\otimes$, as defined below, is called the symmetric group on three points, or S_3.

$\otimes$	(1)	(123)	(132)	(23)	(13)	(12)
(1)	(1)	(123)	(132)	(23)	(13)	(12)
(123)	(123)	(132)	(1)	(13)	(12)	(23)
(132)	(132)	(1)	(123)	(12)	(23)	(13)
(23)	(23)	(12)	(13)	(1)	(132)	(123)
(13)	(13)	(23)	(12)	(123)	(1)	(132)
(12)	(12)	(13)	(23)	(132)	(123)	(1)

The symmetric group S_4 has all the symmetry transformations of a pyramid. S_n refers to a regular pyramid in $n-1$ dimensions with n corners and a line of unit length between each possible pair of points. While it is not necessary to learn how to do geometry in large numbers of dimensions, there is the implication that S_n gets complicated for large n. In general, S_n has $n!$ elements in its set of transformations, and describes all possible symmetry transformations (shape and scale preserving functions) for a set of n points.

Groups can also be formed over sets of transformations which can vary a non-geometric problem without varying the problem's solution. The advantage of groups will be in showing how a problem space can be broken into independent subsets of dimensions (subspaces). Each subspace will be invariant under a corresponding group of symmetry transformations. Representing groups with matrices is the best way to discuss the issues of independence among subspaces and how subspaces can be invariant under symmetry transformations.

Group Representation Theory

General groups can be represented in matrices by finding a set of matrices to correspond to the group's set $\mathcal{E}$. Where the group operation $\otimes$ can be replaced by ordinary matrix multiplication. For instance, an $n \times n$ matrix can represent a rotation, flip (reflection), or any other symmetry transformation possible in n dimensions. The entire space can be transformed, but in the context of this paper, only a set of discrete points need to be transformed, while an underlying continuous space can remain unchanged.

Matrix representations are easiest to interpret for permutations. For an example consider five points. Consider the first two points as equivalent under the symmetry transformations of S_2, and let the last three points be equivalent under the transformations of S_3. Represent the points by their place in a vector: $(P_1, P_2, P_3, P_4, P_5)$ such that $P_1 = (1,0,0,0,0), \ldots, P_5 = (0,0,0,0,1)$. A transformation which preserves the symmetry of equivalence among the points has the form: $T = \begin{pmatrix} S_2 & 0_{2\times 3} \\ 0_{3\times 2} & S_3 \end{pmatrix}$. Where $0_{n_1 \times n_2}$ represents an n_1 by n_2 matrix of 0's. The two elements of S_2 are representable by $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, and $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. The first serves as the identity, and the second interchanges the first two values in the point vector. The six elements of S_3 are:

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

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

The first serves as the identity, and the next two as the rotations. The last three are flips. The two transformations of S_2 act independently of the six transformations of S_3. This can be seen as S_2 and S_3 do not share any rows or columns when they are combined together. There are twelve combinations of the two groups. As an example of a transformation that swaps the first two points and rotates the last three points:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{pmatrix} = \begin{pmatrix} P_2 \\ P_1 \\ P_5 \\ P_3 \\ P_4 \end{pmatrix}$$

In general, representations may have larger matrices than necessary. There are representations which have minimal size and are referred to as irreducible representations. Representations need only preserve part of the structure of a group, but our attention is upon irreducible representations which capture the full structure of a group. In particular, the irreducible representations we looking for can be represented in a block diagonal form. With one block for each independent group placed along the diagonal of a matrix which represents a combination of groups. Each block/group will act on its own subset of the variables represented in a vector. Just as S_2 acts on the first two elements in the point vector and S_3 acts on the last three elements. In terms of symmetry, points P_1, P_2 can be considered independently of the points P_3, P_4, P_5. This can greatly reduce the size of a problem.

By taking equivalences:

$$P \equiv P_1 \equiv P_2, \quad P' \equiv P_3 \equiv P_4 \equiv P_5$$

An original problem on five points can be considered as a problem in terms of the two points P, P'. The relatively simple examples used in this paper may not make this seem like a great savings. The general case may use variables, each representing a separate dimension, instead of points. Thus the above example could be seen as reducing a five dimensional problem into a two dimensional problem. With the analogous relations:

$$U \equiv x_1 \boxminus x_2, \quad V \boxminus x_3 \boxminus x_4 \equiv x_5$$

Where U and V are considered independent subspaces of some five dimensional problem. In the most abstract sense, a subspace can be expressed a function of its underlying dimensions. Even then, there is a great simplification in that the subspaces do not share underlying dimensions. Each dimension is associated with a single subspace. Each subspace corresponds to one of the blocks along the diagonal of representation. Each block represents a group of symmetry transformations which acts on the corresponding subspace. Each subspace can be said to be invariant under the group of transformations which act on the subspace.

4.4 Transformations in gene expression

Earlier in this section we have described the structure of DNA, mRNA, proteins and the transformations that define the process of gene expression in qualitative terms. Little work has been done that establishes such qualitative descriptions on computational grounds and justify their purpose in natural evolution. In this section we shall hypothesize that these transformations offer a well grounded mathematical technique for identifying linearly decomposable subspaces of the search space. We shall also provide examples to demonstrate that the biological events in gene expression do seem to follow our line of mathematical arguments.

In the previous section we saw that the block diagonal elements of irreducible representations of a group of invariant transformations can be used to produce the linearly independent subspaces of the search space. However, in the context of optimization, transformations that change the objective function are of little use, since the solution of the transformed will not remain invariant in the general case. Therefore we need fitness invariant symmetry transformations (defined earlier) which keeps the objective function invariant but changes the underlying eigen functions. The following theorem is useful for further understanding of groups of fitness invariant symmetry transformations.

Theorem 2 *Fitness invariant symmetry transformations, that are non-singular, always form a group.*

Proof sketch: If Q_1 and Q_2 are two fitness invariant symmetry transformations, then it can be easily shown that $Q_1 Q_2$ is also a fitness invariant symmetry transformation. Trivial to show the associativity property. We can always have the identity transformation. Since the transformations are non-singular, there always exists the inverse, Q_i^{-1}. Please see (Kargupta & Stafford, 1997) for details. Detailed proof is not provided here due to lack of space. □

This theorem clearly says that non-singular, fitness invariant symmetry transformations can be naturally studied as a group. In the following, we shall study the transformations in transcription, translation and discuss their role in the context of fitness invariant transformations.

Transcription

Earlier in this section we have described the structure of DNA, mRNA, proteins and the transformations that define the process of gene expression in qualitative terms. In this section we shall capture them in a quantitative manner. Denote the DNA double helix as $\mathcal{D} = (\mathcal{D}_a, \mathcal{D}_b)$; where $\mathcal{D}_a = d_{a1}, d_{a2}, \cdots d_{ai} \cdots d_{an}$. Any element d_{ai} $in \Lambda_d$, where Λ_d is the alphabet set for DNA.

Mathematically transcription can be viewed as a process that transforms a DNA sequence into the mRNA sequence. Let us denote the alphabet set of nu-

cleic acids in mRNA by Λ_r. An mRNA sequence can be denoted as, $\mathcal{R} = r_1, r_2, \cdots r_i \cdots r_n$, where $r_i \in \Lambda_r$.

The complementary pairing between the two strand of DNA can be modeled by the group S_2. The identity transformation will transcribe $\mathcal{D}_a$ to mRNA. If $\mathcal{D}_b$ is to be transcribed, it can be determined from $\mathcal{D}_a$ by the complementary relation between strands. To demonstrate the latter, consider each d_{ai} as a vector and the matrix $M_{complement}$ for complementary pairing:

$$T = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, A = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, G = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, C = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

$$M_{complement} = \begin{pmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}$$

Then $M_{complement} \times d_a i$ translates one strand into the other, after which there is a one to one translation from DNA to mRNA with the substitution of U for T. The symmetry of the complementary pairing can be express by S_2, The transformation $M_{complement}$ is not fitness invariant as each strand will lead to a functionally different protein.

Translation

RNA is grouped into a sequence of codons (triplets) which are translated into a sequence of amino acids. In terms of amino acids, a set of codons on the same row of table 1 are equivalent. As from the first row, GCA, GCC, GCG, GCU all result in the same amino acid and can be considered in an equivalence class. By forming a separate equivalence class for each row of table 1, the symmetries of the table can be expressed by a group. The group which contains all the symmetries for m elements is S_m. Thus the symmetry transformations for the first row form the group S_4. Remembering that the irreducible representation of S_m in terms of permutation matrices is an $m \times m$ matrix, the following matrix shows the block diagonal form of the group of all symmetries of table 1. $M_{translate}$ looks like:

$$\begin{pmatrix}
S_4 & & & & & & & & & & & & & & & & & & & & & & & \\
 & S_2 & & & & & & & & & & & & & & & & & & & & & & \\
 & & S_2 & & & & & & & & & & & & & & & & & & & & & \\
 & & & S_2 & & & & & & & & & & & & & & & & & & & & \\
 & & & & S_2 & & & & & & & & & & & & & & & & & & & \\
 & & & & & S_4 & & & & & & & & & & & & & & & & & & \\
 & & & & & & S_2 & & & & & & & & & & & & & & & & & \\
 & & & & & & & S_3 & & & & & & & & & & & & & & & & \\
 & & & & & & & & S_2 & & & & & & & & & & & & & & & \\
 & & & & & & & & & S_6 & & & & & & & & & & & & & & \\
 & & & & & & & & & & S_1 & & & & & & & & & & & & & \\
 & & & & & & & & & & & S_2 & & & & & & & & & & & & \\
 & & & & & & & & & & & & S_4 & & & & & & & & & & & \\
 & & & & & & & & & & & & & S_2 & & & & & & & & & & \\
 & & & & & & & & & & & & & & S_6 & & & & & & & & & \\
 & & & & & & & & & & & & & & & S_6 & & & & & & & & \\
 & & & & & & & & & & & & & & & & S_4 & & & & & & & \\
 & & & & & & & & & & & & & & & & & S_4 & & & & & & \\
 & & & & & & & & & & & & & & & & & & S_1 & & & & & \\
 & & & & & & & & & & & & & & & & & & & S_2 & & & &
\end{pmatrix}$$

All entries outside of the blocks on the diagonal are 0. If there were room to expand each block, $M_{translate}$ would be seen as a 64×64 matrix. Codons are represented by 64×1 vectors, with exactly one 1 to mark which of the 64 codons is to be represented. The codons must be ordered as in table 1. As in the example of $S_2 \times S_3 = \begin{pmatrix} S_2 & 0_{2\times3} \\ 0_{3\times2} & S_3 \end{pmatrix}$ shown in the group theory section, each block can be any $m \times m$ permutation matrix. The choice of which particular matrix (identity, rotation,...) to use in a block is independent of the choices made in other blocks. Thus the total number of symmetry transformations for the table is $2!^9 \cdot 3! \cdot 4!^5 \cdot 6!^3 = 339,738,624$. The group of the symmetry transformations is a product of symmetry groups, each given as an irreducible representation, so the overall representation is irreducible. Each block corresponds to a subspace of codons, and the codons in that subspace have fitness invariant symmetry transformations represented by the block.

Regarding k-delineable problems, k is bound by the largest number of codons in a row of table 1. This also applies to a sequence of codons $C = c_1, c_2, \ldots, c_n$.The required representation $M_{translate,equence}$ can be constructed by placing n copies of $M_{translate}$ along the diagonal, with the rest of $M_{translate-sequence}$ as 0's:

$$\begin{pmatrix} M_{translate} & & & 0 \\ & M_{translate} & & \\ & & \ddots & \\ 0 & & & M_{translate} \end{pmatrix}$$

The vector for C is just a single vector formed by appending the c_i vectors in sequence order. Thus as long as a coding table has bounded row length, then the problem is k-delineable, even for a sequence of codons.

5 Conclusions

Linkage learning plays an important role in blackbox optimization. The scope of linkage learning may be divided into two groups, namely, (1) identifying the linearly decomposable subspaces and (2) identifying relations within these subspaces that properly delineate the search space. In this paper we proposed a possible strategy for identifying the linearly decomposable subspaces, if they exist for the given problem. Searching for a group of fitness invariant transformations and constructing the irreducible representation of the group are the two main components of this approach. In future work on protein structure, we will use more general unitary matrices for transformations and group representations. This will allow hyperplanes at arbitrary angles as may be required by the

geometry of protein search spaces. We noted that the transformation in transcription (DNA→RNA) does not introduce any fitness invariance. Therefore, RNA representation does not let us identify the linearly decomposable subspaces. However, the RNA→Protein transformation, i.e. the genetic code offers interesting characteristics. It introduces fitness invariant transformations. Moreover, the transformations appears to be in irreducible form. It is interesting to note that many biologists conjecture that RNA came to existence before proteins. However, during the course of evolution, RNA→Protein transformation appeared and proteins took the responsibility of being basic functional units. Our analytical arguments offers a justification behind this natural phenomena.

Acknowledgment

This work was supported by Los Alamos National Laboratory and United States Department of Energy.

References

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., & Watson, J. D. (1994). *Molecular biology of the cell.* New York: Garland Publishing Inc.

Belew, R., & Vose, M. (Eds.) (1996). *Foundations of Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann.

Deb, K. (1991). *Binary and floating-point function optimization using messy genetic algorithms* (IlliGAL Report No. 91004). Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Goldberg, D. E., Deb, K., Kargupta, H., & Harik, G. (1993). Rapid, accurate optimizaiton of difficult problems using fast messy genetic algorithms. In Forrest, S. (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 56–64). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, *3*(5), 493–530. (Also TCGA Report 89003).

Harik, G., & Goldberg, D. E. (1996). Learning linkage. See Belew and Vose (1996). To be published.

Heine, V. (1993). *Group theory in quantum mechanics.* New York: Pergamon Press.

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

Kargupta, H. (1995, October). *SEARCH, Polynomial Complexity, and The Fast Messy Genetic Algorithm.* Doctoral dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. Also available as IlliGAL Report 95008.

Kargupta, H. (1996a). The gene expression messy genetic algorithm. In *Proceedings of the IEEE International Conference on Evolutionary Computation* (pp. 814–819). IEEE Press.

Kargupta, H. (1996b, January). *SEARCH, evolution, and the gene expression messy genetic algorithm.* Los Alamos Unclassified Report LA-UR-96-60.

Kargupta, H., & Goldberg, D. E. (1996). SEARCH, blackbox optimization, and sample complexity. See Belew and Vose (1996). To be published.

Kargupta, H., & Stafford, B. (1997, January). *Transformations in gene expression.* In preperation.

Mitchell, T. M. (1980). *The need for biases in learning generalizations* (Rutgers Computer Science Tech. Rept. CBM-TR-117). Rutgers University.

Watanabe, S. (1969). *Knowing and guessing - A formal and quantitative study.* New York: John Wiley & Sons, Inc.