LA-UR- 97-2842

CONF-970740--4

Approved for public release; distribution is unlimited.

Title: Gene Expression: The Missing Link in Evolutionary Computation

Author(s): Hillol Kargupta

Submitted to: Seventh International Conference on Genetic Algorithms, 1977 (July 19-23, 1977)

Los Alamos
NATIONAL LABORATORY

Gene Expression: The Missing Link In Evolutionary Computation

Hillol Kargupta*

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 points out that the traditional perspective of evolutionary computation may not provide the complete picture of evolutionary search. This paper focuses on gene expression—transformations of representation (DNA→RNA→Protein) from a the perspective of relation construction. It decomposes the complex process of gene expression into several steps, namely (1) expression control of DNA base pairs, (2) alphabet transformations during transcription and translation, and (3) folding of the proteins from sequence representation to Euclidean space. Each of these steps is investigated on grounds of relation construction and search efficiency. At the end these pieces of the puzzle are put together to develope a possibly crude and cartoon computational description of gene expression.

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 carrier 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 study the seemingly major steps in gene expression from the perspective of blackbox search or optimization (BBO). We start by revisiting the decomposition of BBO in terms of constructing partial orders in the relation and class spaces, proposed elsewhere (?). Detecting relations that capture patterns of the high fitness regions of the search space require explicit mechanisms for representing the functional definition of patterns and inducing them from samples taken from the search space. Since naive mutation is exponential in time complexity and naive crossover also takes exponential time for learning even similarity based equivalence classes or schemata (let alone more general classes) (?), it is natural to conjecture that may be we are missing another piece of the puzzle of evolutionary computation. In natural evolution proteins define the underlying search space. The shapes of the proteins define their phenotype, in other words their efficacy. Production of a protein is characterized by (1) gene regulatory controls, (2) transformation of sequence reprsentations (DNA→RNA→Amino acid sequence), and (3) folding of the amino acid sequence into Euclidean space. Clearly, gene expression plays a major role in changing and defining the evolutionary representation; also note that representation is a natural way to manipulate and capture relations. Let us summarize the points noted so far to drive the idea home:

1. crossover, mutation, selection alone appear to be computationally inefficient for learning even schemata, let alone more general classes;

2. gene expression constructs representation in natural evolution;

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

3. popular models of evolutionary computation do not explain gene expression well.

This naturally leads us to hypothesize that may be gene expression is the missing piece of puzzle in evolutionary computation that offer mechanisms to learn relations, critical for designing scalable BBO algorithms.

Section 2 reviews the decomposition of BBO proposed by the SEARCH framework.

2 BBO Decomposition In SEARCH

The SEARCH (Search Envisioned As Relation and Class Hierarchizing) framework developed elsewhere (Kargupta, 1995; Kargupta & Goldberg, 1996) offered a decomposition of BBO with a flavor of probabilistic and approximate approach. In this section, we briefly review the decomposition of BBO proposed by the SEARCH framework into relation, class, and sample spaces.

The foundation of SEARCH is based on the fact that induction is an essential part of black box optimization, 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 the scope of the inductive search algorithm to a finite set of relations[1] 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 take an example to illustrate the idea. Say, we have a few 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 dress, shoes 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 dress", such a relation may divide the set of people into different classes, such as cheaply dressed people, very expensively dressed people, 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 set of features that we restrict our attention to may be pre-determined or dynamically constructed during the course of induction. The decomposition of BBO in SEARCH in terms of relation, class, and sample spaces essentially captures this idea. Note that, the search for relations is essential, since some relations are inherently good and some are not. For example, "quality of dress" may be a good one; however, "color of the hair" may not be a good relation for the above mentioned problem. In SEARCH, relations that are inherently good for decision making are said to *properly delineate* the search space. If we construct a partial ordering among the classes, defined by a relation of order k (logarithm of the number of classes defined by the relation), select the "top" 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 as a 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 from top 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.

The traditional perspective toward BBO is often polarized by the desire to find optimal solutions, asymptotic convergence, ways to get out of local optima-s. While they are certainly important, the big picture often gets lost amidst all these issues. SEARCH points out that, since induction is an essential part of BBO, search for appropriate relations is the critical step in BBO. In stead of looking for better solutions from beginning, SEARCH advocates a BBO algorithm to first detect the structure of the search space, induce relations to capture that, and then identify desired quality solutions. Following the SEARCH analysis, we note that two main steps of any BBO algorithm should be,

1. construction/selection of relations that properly delineates the search space;
2. detection of better classes

[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.

Capturing the symmetries and assymetries of a search space in terms of relations is a challenging task and this is the primary topic of this paper.

3 Relations Among What?

Previous sections used the term relation in an abstract set theoretic sense. Although we talked about relations among the search space members in general, recall that our fundamental objective is to identify order-k delineable relations, i.e. relations that identify members of the search space that may contain the desired quality solution. Unfortunately in BBO, we do not know the desired solution a priori. Therefore detection of order-k delineable relations involves two steps: (1) identifying relations that capture better regions of the search space as classes and (2) use these relations to direct the future directions of search. The former step constructs or selects a relation and the latter step evaluates the efficacy of the relation in guiding the search into desired quality solutions.

Relations that we use for capturing better regions of the search space, need to be either selected from a predefined repository of relations or constructed on the fly from the search space members. In many existing search algorithms, relation space is either implicitly or explicitly defined a priori; Representation, search operators, heuristics often contribute to defining the relation space. If sufficient domain knowledge is available, then highly effective and specialized relation space can be crafted for that domain specific application. Unfortunately, in BBO such luxury is often unaffordable. This lead us to the following dielema. If our BBO algorithm uses a very restricted relation space then the scope of the algorithm is accordingly restricted; on the other hand, as we make the relation space richer and larger, we pay a price in terms of the cost for searching appropriate relations in the relation space. For example, in perceptron the representation was quite restricted which made it capable of learning only linear classifiers (?). On the other hand, the repersentation of a general three layered neural network (3-CNN) is very reach and it is backed by the Kolmogorov mapping theorem (); unfortunately, 3-CNN pays a price for such richness in representation and it has already been shown elsewhere () that 3-CNN is not polynomial time learnable. An example of inadequecy of the representation can be given using the sequence representation used in simple genetic algorithm. Design of simple GA is often based on the processing of similarity based equivalence classes or schemata, defined by the partitions introduced by defining equivalence over variable positions in the sequence representation. Consider Figures 1 and 2 which show two popular

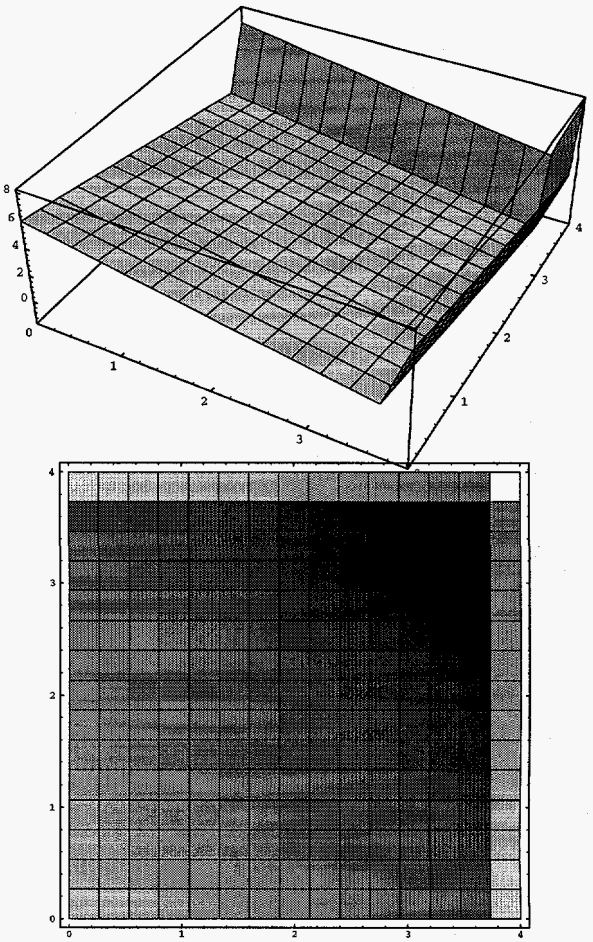

Figure 1: This function is an additive combination of two order-4 deceptive trap functions.

BBO maximization problems. Note that the regions of the search space with higher objective function value can be captured by hyperplanes that are either orthogonal or parallel to coordinate axes. In other words, equivalence classes defined by the partitions over the sequence representation can easily capture better regions of the search space that actually contain the globally optimal solution; for example, in Figure 1 the class 4# defined by the relation $f\#$ contains the globally optimal solution. Similarly in Figure 2 the class 1.5# contains the best solution. Now consider Figures 3 and 4 which two other typical test functions (minimization problems) often used in BBO literature. It is not obvious how such similarity based partitions defined over the sequence representation can be used for effectively capturing better regions of the search space. Nevertheless, the important thing to note that the local optimas or the undulations of the terrains do offer some symmetries or assymetries (to be precisely defined later). In other words, if we can some-

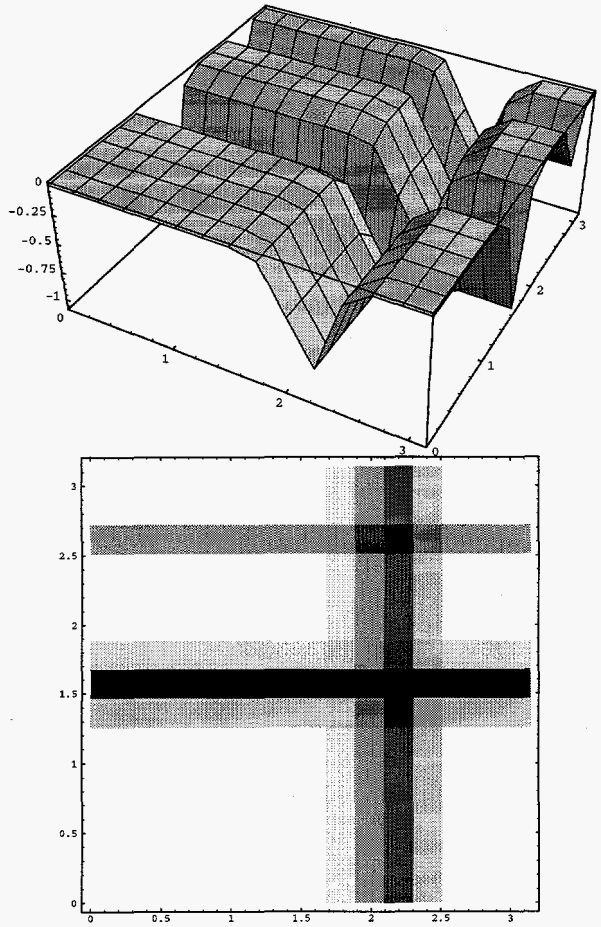

Figure 2: Michalewitz's function.

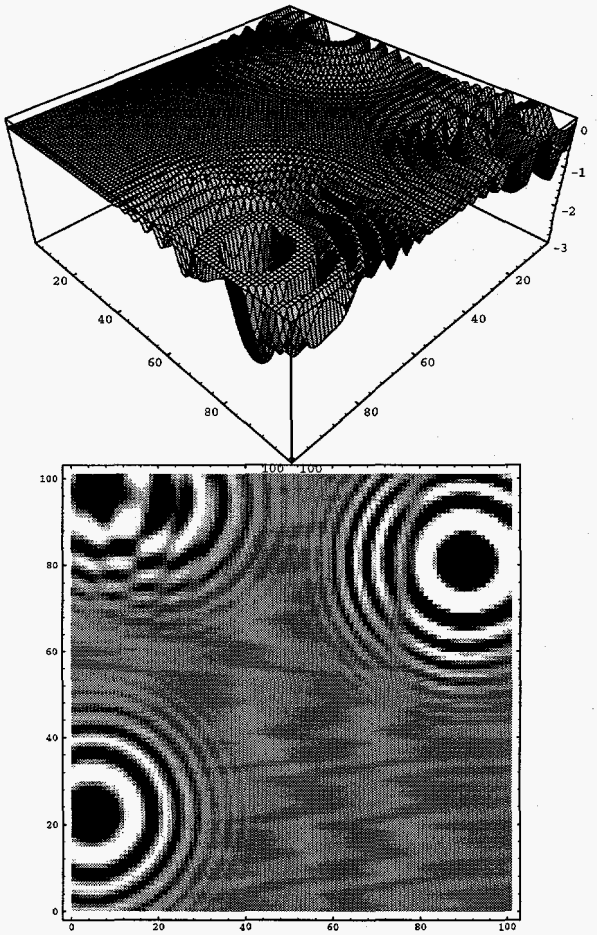

Figure 3: Langerman function

how capture the relations among local optima-s, these relations lead us to the region containing the desired optimal solution. For example in Figure 3 the local minima-s form a pattern similar to concentric circles about different points in the search space. If we can capture the symmetry of concentric circle as a relation and instruct the search algorithm to move along the radius and explore regions along the circles defined by this detected relation, it will soon find the optimal solution. Similarly, in Figure 4 if we connect the local minima-s by a piece-wise straight lines, following these lines may lead us the optimal solution far more efficiently than a pure random search. Simple partitions defined over the variables of a sequence representation are clearly not capable of capturing the complex relations, such as the ones shown in Figures 3 and 4. Clearly, we can always find problems for which a pre-defined relation space in insufficient to capture the patterns of the search space. Therefore, the strategy of constructing relations is more appealing compared to the strategy of selecting appropriate relations from a pre-defined relation space.

4 Learning Relations And Classes In Evolutionary Algorithms

Like any other BBO algorithm, performance of an evolutionary search algorithm depends on how well it captures the regions of the search space with high fitness using relations and classes. A common characteristic among most of the popular evolutionary algorithms is that they all process relations and classes without paying explicit attention and by making subtle but important assumptions that often restrict their efficacy in detecting appropriate relations and classes.

For example, simple GA (sGA) (**?**) assumes relation space is defined by the similarity based partitions of the given representation. These partitions define similarity based equivalence classes, or schemata. Bottom line is that if sGA will work only if the good re-

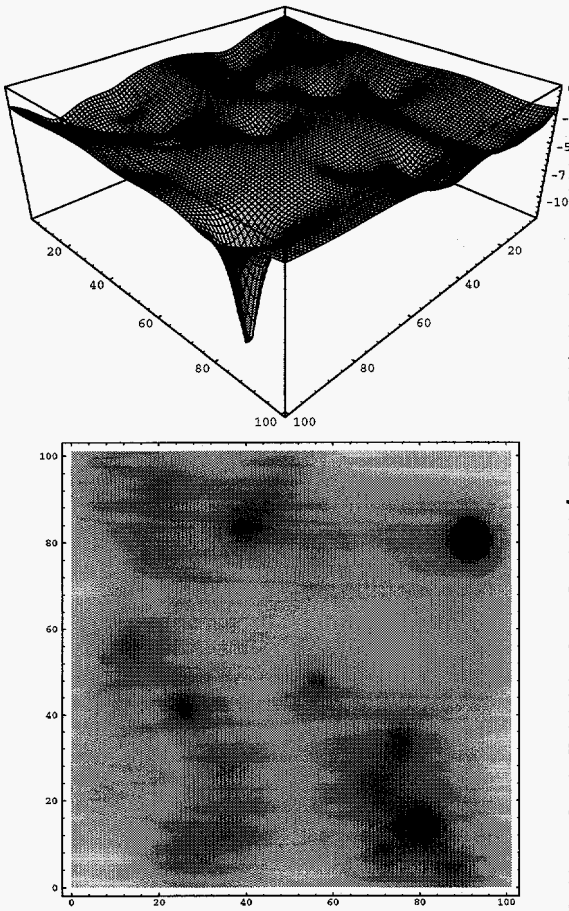

Figure 4: Shekel's foxhole function.

gions of the search space can be captured by low order schemata, defined by adajacently located variables in the chosen representation.

5 From DNA To Protein

DNA molecules consist 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
- protein folding

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 a 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 amino acid sequences fundamentally define proteins, formation of the three dimensional structure of proteins involves 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.

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.

A 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. Table 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 map to Alanine. In other words, this set of four codons offers

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.

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 $\phi(\mathbf{x})$ is 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 keep $\phi(\mathbf{x})$ invariant, they *do 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 (?). 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.

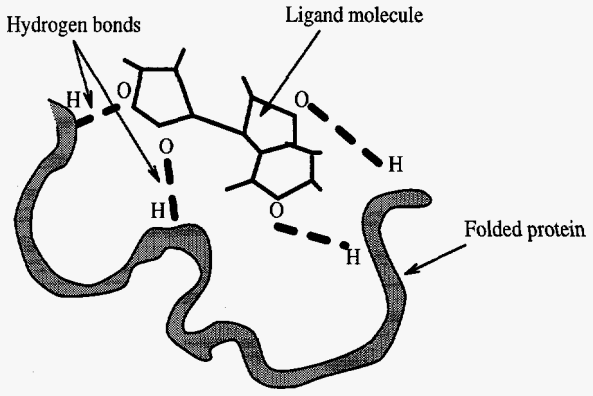

Figure 5: A typical ligand binding site of a protein, like catabolite gene activator protein (CAP).

6 Computational Decomposition Of Gene Expression

6.1 Protein Folding

The shape of the protein decides the efficacy or its fitness. In other words they define the underlying evolutionary search space. The 3-D shape of a protein can be viewed as a structure generated by a set of points in Euclidean space. It is interesting to note that although the DNA and RNA are sequences of neuclic acids, proteins fold into a high dimensional Euclidean space from the sequence of amino acids. It is even more interesting that the actual performance of a protein is often decided by a small region of the surface that comes in direct contact with ligand molecule. Figure 5 shows a typical binding site of a protein. The hydrogen bonds between the protein surface and the ligand molecule determine how well the protein fits with the latter, which in turn determines the fitness of the protein. Although the purpose of the protein surface not in contact with the ligand is not clear yet, biologists believe that it may be needed for making the protein structurally stable. These observations can be summarized in the following manner:

1. high dimensional evolutionary search space is evaluated through a mapping to the Euclidean space

2. only a small fraction of the dimensions in the Euclidean space play critical role in determining the performance of the proteins that essentially defines life.

Let us again recapitulate these observations in a broader context of BBO. Imagine you give the nature a BBO problem. If our conclusions are right, nature is

likely to first project the underlying high dimensional search space to a low dimensional Euclidean space. This would only make sense if such projection does not change the search problem significantly. In other words, we would like the patterns of the search landscape to remain nearly invariant—meaning, regions of high fitness values map to high fitness regions in the projected space and likewise for the regions with relatively low fitness values. If this constraint is satisfied, then such a mapping may be quite useful since reduction of dimensions by a large fraction will reduce the search space significantly; as a result the orginal problem may become computationaly ameneable. Now the question is whether we can say something concrete about such possibility.

Before we answer this question, let us discuss the following. For most of the interesting BBO problems enumeration cannot be afforded. In other words, for all practical purposes, we would like to stop our favorite algorithm to stop after sampling some finite number of points from the search space. Let us say, we sample some n points from a n dimensional space, X^n. If $|\Lambda|$ be the cardinality of each of these n dimensions, then the overall size of the search space is $|\Lambda|^n$. It is quite natural to wonder that, since in BBO our perception of any pattern in the search space is restricted to the information provided by the n samples, whether we need all the n dimensions for describing the pattern. As it turns out, we can design a simple random polynomial time algorithm for capturing such a pattern using only $O(\log^2(n))$ dimensional Euclidean space, that keeps the search space nearly invariant with respect to any arbitrary metric defined over it. In order to make this claim more precise, let us first define the idea of graph isometry. A graph is a collection of nodes and edges. An edge connects two nodes of the graph. Let us view the n samples taken from the search space as the n nodes of a graph and also assume that there exists a distance metric (ρ_x) that defines the distance between any two points from the original search space (X^n). An edge represent the distance between the two nodes it connects. An isometry is a mapping γ from the metric space (X^n, ρ_x) to another metric space (Y^m, ρ_y) such that $\rho_x(x_1, x_2) = \rho_y(\gamma(x_1), \gamma(x_2)$. In other words γ preserves the distance among the nodes. We say the mapping γ to be ϵ-nearly isometric, if and only if $\frac{\rho_x(x_1,x_2))}{\rho_y(\gamma(x_1),\gamma(x_2)} \leq \epsilon$. In this case we may say that the mapping has an ϵ distortion. The following theorem developed elsewhere (**?**) provide the foundation of near isometric mapping of the underlying search space to a Hilbert space.

Theorem 1 (Bourgain,1985) *Every n-point metric space can be mapped to a $O(\log n)$ Hilbert space with an $O(\log n)$ distortion.*

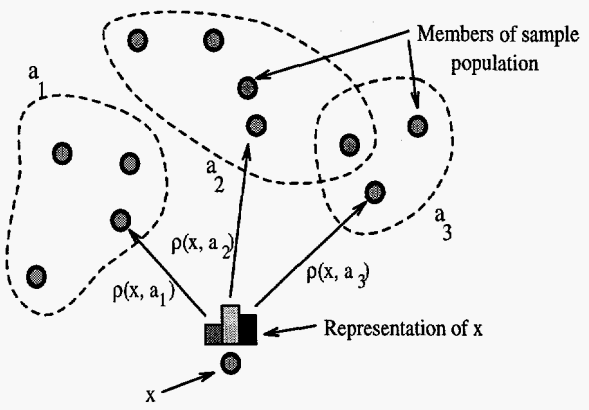

Figure 6: Near isometric representation construction from a finite population of samples.

Proof of the following theorem (**?**) adopts the general scheme of Bourgain's work and it offers a simple random polynomial algorithm to obtain near isometric mapping.

Theorem 2 (Linial et. al., 1994) *In random polynomial time, every n-point metric space can be embedded in $\ell_p^{O(\log^2 n)}$ (for any $p \geq 1$), with distortion $O(\log n)$, where ℓ_p^m is a norm in the Euclidean space $\Re^m$ defined by $||(x_1, x_2, \cdots, x_n)||_p = (\sum |x_i|^p)1/p$.*

Their algorithm works as follows. For each $\kappa < n$, such that κ is a power of 2, randomly pick $O(\log n)$ sets $a_i \subseteq X^n$ of size κ. This will result in selection of $O(\log^2 n)$ sets $a_1, a_2, \cdots a_{\log^2 n}$. Map any point $x \in X^n$ to an $O(\log^2 n)$ dimensional space such each coordinate takes the value $\rho_x(x, a_i) = \min\{\rho_x(x_1, x_2)|x_2 \in a_i\}$. In plain English, it means that we first randomly select $O(\log^2 n)$ subsets of sizes exponentially growing toward n from the sample set. Then we compute the smallest distance between the sample point, x under consideration and each of these selected subsets. We get a vector of these distance values of length $O(\log^2 n)$, which defines the representation of x.

Figure 6 offers a pictorial description of this algorithm, which may be quite plausible in the biological context. In order to judge the utility of this mapping approach, let us look at some experimental results. Consider a binary string space with $\Lambda = \{0, 1\}$ and $n = 32$. We used hamming distance metric. A sample of 30 strings are randomly generated and each of these strings is projected to $\Re^{25}$. Figure 7 shows the original normalized distance matrix for 32 dimensional space. Figure 8 shows the same for the 25 dimensional newly constructed representational space. Figure 9 shows

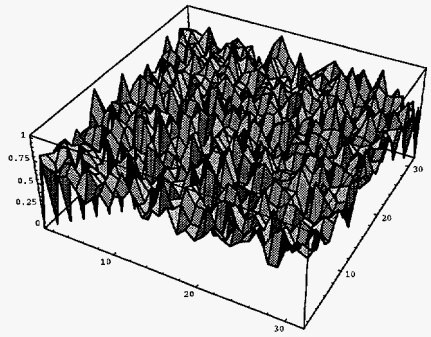

Figure 7: Hamming distance matrix in 32 dimensional space.

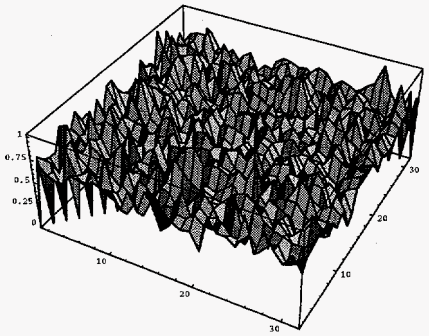

Figure 8: Hamming distance matrix in the projected 25 dimensional space.

the distortion. As we see, the distortion is quite low for most of the members of the sample population.

By construction, the new representation of a sampled point x, is a vector of the distance values between x and a set of its nearby neighbours. Note that for binary search space, with hamming distance metric the maximum value can each of these dimensions can take is bound by the number of dimensions of the original search space, n. Therefore the ratio of the original and constructed search space cardinalities,

$$\begin{aligned} r &= \frac{2^n}{n^{\log^2 n}} \\ &= 2^{n-\log^3 n} \end{aligned}$$

This gives the order of reduction in cardinality in the constructed representation. Apart from the mathematical justifications, this representation also has a simple and intutive appeal. Consider Figure 10, where $x_1 \cdots x_n$ be the set of points used to compute the distance vector, i.e. the new representation of the point x_0. Constructing a relation using these points in order to capture a pattern of high fitness values, is essentially a problem of searching in the function space. Any particular dimension in the projected space can be associated with a function space defined by a pair of points like $(x_0, x_1), (x_0, x_2)$ and so on. For example, we may choose to assign a linear or a quadratic function connecting all such pairings with the point x_0 as shown in Figure 11. Let us recapitulate the main points in order make it more clear. Every coordinate of the new space takes a value defined by a pair of points. In order to define a region or a class using this two points we need to choose a function. Let us denote such a function defined by the points x_0 and x_i by $\xi_i(x_0, x_i)$ (in short ξ_i). Function ξ_i is associated with the i-th dimension of the P_2 space. This essentially defines a local coordinate system that can be used to capture regions of the underlying search space. Figure 12 shows one possible way to capture patterns in the fitness landscape using linear ξ_i functions.

6.2 From proteins to RNA sequences

Once the P_2 representation space is constructed for the population members and the coordinates are used to define a local coordinate system that captures good regions of the search landscape, then we need to find a way to use this information to direct future exploration of the search space. Before discussing this, let us review some basic concepts connecting patterns and algebra.

Patterns and algebra have very deep connection. An algebraic system is comprised of a set of objects, a set of operations defined on them, and a collection of properties that they satisfy. A group is a simple algebraic system, defined by a set of elements, a composure operation, such that the elements satisfy associativity property, have unique inverse and an identity element. Readers not familiar with groups may wish to refer to the appendix for a brief exposure to

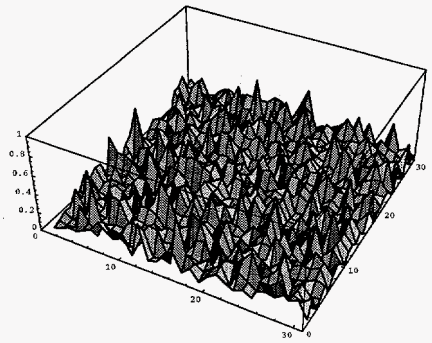

Figure 9: Mapping distortion.

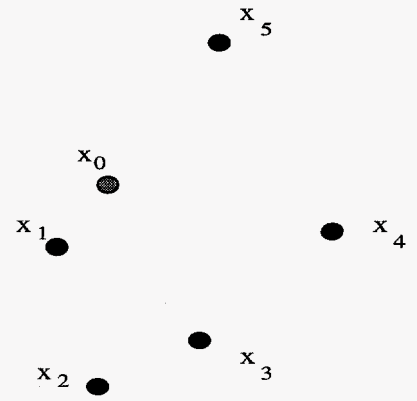

Figure 10: The set of points $x_1 \cdots x_n$, used for constructing the representation of x_0.

elementary group theory. In the following we are primarily going to use the intutive ideas behind groups. As noted earlier, algebraic structures can be used for capturing patterns. Groups can also be used for capturing symmetry, a certain kind of pattern. Consider an equlateral triangular object. Now if our eyes are blind-folded, one way to realize the triangular shape is to rotate the object and measure its surface properties at different locations. This essentially means that we are applying rotational transformations and checking for the invariance of the surface property. The symmetry of such a triangular object can be captured by a group of permutation operations.

Earlier in this paper, we discussed the geometrical perspectives of schemata. Let us now consider a schema in the algebraic context. Consider a space of 5-bit binary strings defined by 5 coordinates $x_1, x_2, \cdots x_5$. Let $ff***$ be the partition that divides the space into four schemata, namely $11***$, $10***$, $01***$ and $00***$. Any similarity based partition like this can be captured by a permutation group. Consider the set of transformations, defined by all possible permutation transformations over the coordinates contributing a wild card ($*$) in the partition. This set of transformations define a permutation group and any schemata defined by this partition is closed under the operation of this group. In the above example, the set of all permutations defined over x_3, x_4, and x_5 captures partition $ff***$. The search for better schemata in genetic algorithms and other similar evolutionary algorithms is essentially equivalent to the search for permutation groups. Obviously, in BBO we do not enumerate all memebers of the schemata; therefore, in BBO we should search for algebraic structures like permutation groups in a probabilistic sense. Now the question is that what kind of structures we should look for. When we interpret this question in our usual domain of schema and partition, the answer is obvious. We need low order schema that capture high fitness region of the search space (precisely speaking order-k delineable schema). This essentially means that we need to look for algebraic structures that preserve fitness invariance in a qualitative and approximate sense. To make this concept more precise, let us define something called (ϵ, δ)-*objective invariance.*

Definition 1 ((ϵ, δ)-Objective invariance) *Given a set of objects, C and an objective function $\phi(\cdot)$, that returns an objective function value for every member of C; let ϕ_m be the mean objective function value. The set C is said to exhibit (ϵ, δ)-objective invariance if and only if no more than δ fraction of the set members has an objective function value, (1) that is $\leq \phi_m - \epsilon$, where maximization is the BBO objective, (2) that is $\geq \phi_m + \epsilon$ for minimization.*

This definition offers just one way to define a measure to represent the quality of the search landscape. Now we can say that the search of "good", low order schemata is essentially same as the search for permutation groups over the coordinates that define (ϵ, δ)-objective invariant equivalence classes.

If permutation groups defined over the coordinates of the search space can capture similarity based partitions and schemata, then we can extend the same concept to our local coordinate system for capturing relations and classes that are not necessarily restricted by similarity basis. The question is how do we do that.

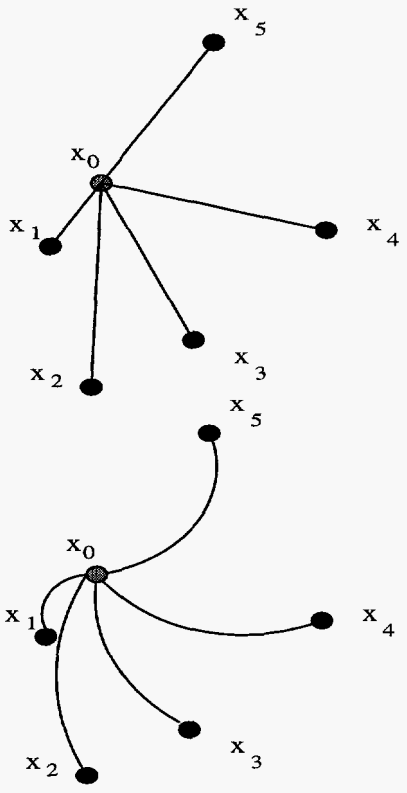

Figure 11: Different possibilities of functions for capturing a pattern.

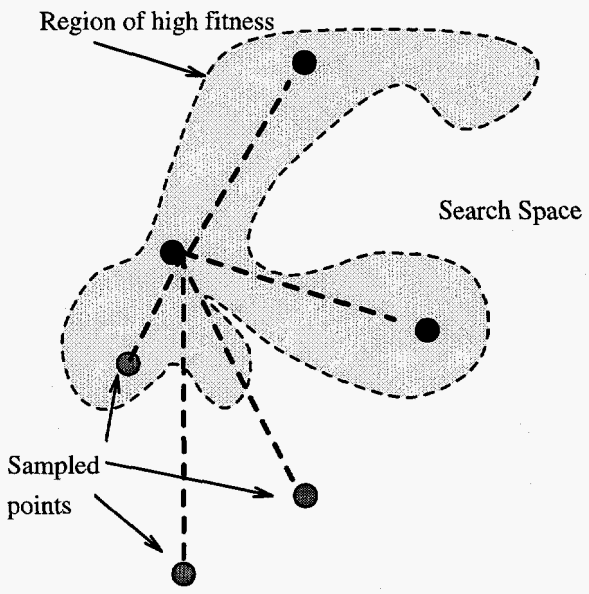

Figure 12: Capturing patterns of the fitness landscape using linear ξ_i functions.

In case of similarity based partitions and schemata, it was quite natural since we focussed only on the given natural coordinates of the search space. The local coordinates in the P_2 space are however functions of the given natural coordinates of the problem. Before we answer this question we need to note that, if permutations are directly applied to the P_2 space, then they may cause large change in the P_2 vector since the cardinality of the coordinates of P_2 can be high (e.g. n for hamming distance metric). This points out the fact a permutation operation directly on the P_2 space will involve changing at least a pair of dimensions, unless it is an identity operation. Now since the P_2 space is a finite subset of the Euclidean space, the cardinality of the alphabet set for each of the dimensions can be as large as determined by the chosen distance metric on the underlying search space. What we need is a way to apply permutations that have a control over a range of granularity level of perturbations. One possible solution to this problem is to map the P_2 to a sequence representation. For example, an n valued dimension of P_2 can be mapped to a sequence space of $\log n$ bits. What we really need is to represent the functions associated with each of the P_2 dimensions in the sequence space. In the following section we point out that the alphabet transformations in translation and transcription may provide one solution to this problem.

6.3 Alphabet transformations in translation

The three dimensional structure of the protein is created from the sequence of amino acids. The amino acid sequence is generated by translation, which is essentially a transformation defined over neuclic acid triplets. The mRNA is grouped into a sequence of codons (triplets) which are translated into a sequence of amino acids. Since the cardinality of the alphabet set of mRNA is 4 (U,C,G,A), there are 64 unique codons. For every codon there is an amino acid (not necessarily unique) as shown in 1. Since the genetic code is defined on alphabet triplets regardless of the coordinate positions, all the coordinate triplets which are associated with codons get mapped to the same amino acid. We can therefore, divide the set of all coordinates triplets into at most 64 different subsets, where members of any such subset contain the same codon. This essntially means that any translation transformation (different possible genetic codes) can only independently change along at most 64 dimensions (since within any subset corresponding to a particular codon, all the coordinate triplets map to the same amino acid).

Such alphabet transformations may provide a solution to our problem of mapping the $\log^2 n$ dimensional P_2 space. Let us consider a sequence space, M^{n_r} of n_r dimensions, with alphabet set size Λ_M. In general, for a tuplet size of k_M, there are $\Lambda_M^{k_M}$ unique dimensions associated with the coordinate tuplet with the same alphabet tuplet value. Now, we can associate these dimensions with the dimensions of the P_2 space. This essentially means that, $\Lambda_M^{k_M} \geq \log^2 n$ i.e., $k_M \geq (\log^2 n)/\log \Lambda_M$. Now that we have a way to construct a sequence representation of the P_2 space, we need to find a way to apply permutation groups defined over dimensions of the newly constructed representation and identify the set of dimensions that offer (ϵ, δ)-objective invariance.

If our above explanation has some truth, then the independent dimensions associated with each of the unique codons should exhibit some degree of objective invariance. This essentially means that any such objective invariance is bound to be reflected by fitness invariance by permutation groups defined over a subset of unique codons. It is quite interesting that, this is a very important characteristic of the genetic code that governs the translation transformation. Note that, all the unique codons of any row of Table 1 map to same amino acid. For example, in the first row, GCA, GCC, GCG, GCU all map to the amino acid Alanine and therefore they all map to the same protein structure; as a result the fitness of the genome remains invariant under the group of permutation transformations defined over these codons. As we see, the genetic code offers a high degree of such objective invariance.

6.4 Alphabet transformations in transcription

Denote the DNA double helix as $\mathcal{D} = (\mathcal{D}_a, \mathcal{D}_b)$; where $\mathcal{D}_a = d_{a1}, d_{a2}, \cdots d_{ai} \cdots d_{an}$. Each $d_{ai} \in \Lambda_d$, where Λ_d is the alphabet set for DNA. The complementary pairing between the two strands of DNA can be modeled by a permutation group. This group of all possible permutations contain $|\Lambda_d|!$ members. If we consider natural DNA where $\Lambda_d = \{T, A, G, C\}$ and the existing complementary pairing rules of DNA, then the transformation can be written as follows:

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

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

Each d_{ai} is considered as a vector and the matrix $M_{complement}$ shows the complementary pairing transformation.

Mathematically, transcription can be viewed as a process that transforms a DNA sequence into the mRNA sequence. The same permutation group can be used to capture this transformation. The only difference is that in nature the necleic acid Uracil (U) is used in mRNA instead of the corresponding Thiamin (T) of DNA.

Note that transcription introduces transformations that offer very little fitness invariance. For example, GCA maps to alanine, but the complement is CGU which maps to arginine. If we carefully examine Table 1 we shall note that except the row corresponding to serine there is no other amino acid which maps back to complementary codons (e.g. AGC and UCG). Since, in general the transcription does not offer any fitness invariance, it alone is of little use in the context of identifying linearly decomposable subspaces. In order to fulfill this objective nature needs some richer set of transformations that offer fitness invariance.

6.5 Transcriptional regulation

Acknowledgments

This work was supported by Los Alamos National Laboratory, United States Department of Energy. The author acknowledges many useful discussion with Sanghamitra Bandyopadhyay and David E. Goldberg.

References

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., & Goldberg, D. E. (1996). SEARCH, blackbox optimization, and sample complexity. In Belew, R., & Vose, M. (Eds.), *Foundations of Genetic Algorithms* (pp. 291–324). San Mateo, CA: Morgan Kaufmann.

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.

Appendix: Groups And Semigroups

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 $\mathcal{E}$, $x_1 \otimes x_2 = x_3$, 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. A unique identity element (usually written as e):
   For any x_1 in $\mathcal{E}$, $x_1 \otimes e = x_1 = e \otimes x_1$.
   For arithmetic operators, 0 is the identity for addition and 1 is the identity for multiplication.

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 returns 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 e. 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

In preparation for a central theorem of group theory, the terms *isomorphic* and *group of permutations* will be explained. Saying that two groups are isomorphic is the notion of "equal" for groups, 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 even its original set can be used 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 $\mathcal{E}$. The other difference between an isomorphism and equality is that the operators may not be equal, but must behave (act) the same way on the respective sets of two groups. Thus we can define a group on one set and let it 'act' on another set of the same size. *Permutations* are functions that reorder elements in an ordered sequence. A *group of permutations* is just a group defined over a set of *permutations*, with $\otimes$ serving to represent a permutation which is equivalent to a pair of successive permutations.

Theorem 3 (Cayley's Theorem) *Every group is isomorphic to a group of permutations.*

The theorem says that any group on a set of n elements can act on a set of n permutations. 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 of as a set of functions which manipulate the order of an ordered list, with $\otimes$ serving to combine a sequence of such functions.

Alternative statements of Cayley's theorem refer to symmetry in an idealized geometric shape. The shape is that of a regular (equal sided) pyramid, with a triangle on each face. For two or three points the shape is just a line or a triangle. For n points, the shape is an $n-1$ dimension pyramid. All the symmetries of the points at the corners of such a pyramid can be represented by permutations. In fact the set of symmetry transformations is exactly the set of all possible permutations for n points. The group of all permutations of n points is called the *symmetric group* or S_n. If the elements of a group can be evenly divided into subsets, and if the group operator $\otimes$ can form a new group on each of these subsets, then the new group is called a *subgroup* of the original group. Subgroups exhibit a portion of the behavior of the original group.

Theorem 4 (Alternate of Cayley's Theorem) *Every group of n elements is isomorphic to a subgroup of S_n.*

Meaning that every group behaves (acts) the same as one of the ways a symmetric group acts. Or that every group can be interpreted in terms of symmetries. A graphical example will be developed next.

Groups evolved as a means to express the symmetries in a problem. And especially to simplify problems by using symmetries. The power 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

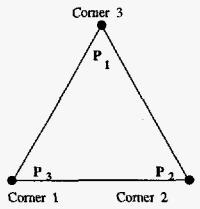

Figure 13: .

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°

This group of rotations is a subgroup of S_3. The next subgroup of S_3 will serve to introduce permutation notation.

Another symmetry of the equilateral triangle 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 points at corners 1 and 2. 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 corners 1 and 2 rather than points P_1, P_2. 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. The combination of the identity, the rotations, and the three flips form a set $\mathcal{E} = \{0° = (1), 120° = (123), 240° = (132), (12), (13), (23)\}$. If we use the elements of $\mathcal{E}$ in permutation notation, then the group with $\mathcal{E}$ and $\otimes$, as defined below, is 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 upper left quarter of the $\otimes$ table shows that the subset $\mathcal{E}' = \{(1), (123), (132)\}$, which corresponds to the rotation group shown earlier, forms a subgroup of S_n.

DISCLAIMER