# SEARCH, Computational Processes in Evolution, and Preliminary Development of the Gene Expression Messy Genetic Algorithm

**Hillol Kargupta***

*Department of Computer Science,*
*Washington State University,*
*Pullman, WA 99164-2752*

This paper considers the issue of scalable search with little domain knowledge and explores implications in the context of evolutionary computation. It presents the Search Envisioned As Relation and Class Hierarchizing (SEARCH) framework introduced elsewhere [26, 31] for developing a theoretical understanding of the issue and argues that scalable evolutionary search needs efficient techniques for detecting relations among the members of the evolutionary search space. It offers a perspective of this argument in the context of natural gene expression (representation transformations that construct the protein from the DNA). It also reports on the preliminary development of the gene expression messy genetic algorithm (GEMGA) [27, 28] that exploits the understandings developed here. Theoretical claims are also substantiated by experimental results for a test bed, comprised of different large, multimodal, scaled problems.

## 1. Introduction

Many search problems exist that do not offer adequate domain knowledge for guiding search directions. These problems are common in many practical search, learning, and optimization domains. They are often called blackbox search (BBS) problems. The Search Envisioned As Relation and Class Hierarchizing (SEARCH) framework, introduced in [26, 31], offers an alternate perspective of BBS in terms of relations, classes, and partial ordering. The SEARCH is motivated by the observation that nonenumerative search for an optimal solution in a BBS is essentially an *inductive process* [39] and in the absence of any relation among the members of the search space, induction is no better than enumeration [54]. The SEARCH decomposes BBS into three spaces: relation, class, and sample. The SEARCH identifies the importance of searching for appropriate relations in BBS. No BBS algorithm can efficiently solve a reasonably general class of problems unless it efficiently searches for relations; evolutionary search algorithms are no exception.

*Electronic mail address: hillol@eecs.wsu.edu.

*Complex Systems*, **11** (1997) 233–287; 

This paper considers the issue of relation search in evolutionary computation. It makes use of the SEARCH framework and alludes to an extension of the existing computational perspective of evolutionary computation. It hypothesizes that the process of *gene expression* (construction of protein from the DNA), often neglected in the existing popular models of evolutionary computation (EC) [10, 20, 44], may be responsible for efficient relation search in evolution. The objective of this biological quest is to sketch a possible mechanism from nature that may be responsible for efficient relation search. The work presented in this paper, however, does not make any effort to explicitly model the natural gene expression process. Therefore, at this point, the hypothesis should be considered only in the context of a higher-level functional perspective of EC. This paper also describes the preliminary development of a new BBS algorithm called gene expression messy genetic algorithm (GEMGA), that explicitly processes each of these spaces with careful attention.

Section 2 describes the motivations and the main concepts of the SEARCH framework. The main components of the overall decision making in SEARCH are identified in section 3. Section 4 presents the analysis and some important results. Section 5 discusses information flow in natural evolution from the SEARCH perspective. This section points out a major lacuna in the existing models of EC from the biological perspective. Section 6 further elaborates on the fundamental limitations of evolutionary algorithms in computational terms and offers the motivation behind messy genetic algorithms (GAs). This also presents a brief account of the current status of messy GAs and lists the strengths and main bottlenecks. Section 7 combines the ideas of SEARCH and its biological implications for introducing a new generation of messy GA, the GEMGA, that eliminates many bottlenecks of earlier messy GAs. This is followed by section 8 which presents the test results for large multimodal, scaled, noisy, difficult problems. Section 9 offers additional test results for some popular test functions. Finally, section 10 presents the conclusion of this paper.

## 2. The SEARCH: An informal picture

The SEARCH presents an alternate picture of BBS in terms of relations and classes that can be constructed among the members of the search space. The SEARCH is also a formal framework that helps us quantify different aspects of BBS, such as sample complexity, problem difficulty, and many more. In this section we briefly review the framework and present the main analytical results without presenting the rigorous derivations given elsewhere [26]. Section 2.1 presents the fundamental motivation behind SEARCH. Section 2.2 presents an overview of SEARCH. A more detailed discussion is presented in section 2.3.

### 2.1 Motivation

By definiton BBS is a search for a solution from an unknown space. Probably the simplest way for a BBS to work is to explore the unknown search domain enumeratively until a good enough solution is found. For large problems, enumerative exploration of the search space is computationally expensive. Therefore, a typical nonenumerative BBS uses only samples taken from the search space in order to estimate the location of better solutions. It makes use of a sample set to decide the next set of samples to explore until it finds the desired solution. The success of a BBS algorithm critically depends on the correctness and efficiency of this decision-making process. The SEARCH framework starts its foundation from this issue.

Note that sampling one particular point from the search domain does not necessarily tell us anything about another point. When a BBS algorithm makes a decision to sample another member from the domain, it is performing *induction*—the process of hypothesizing the premise from the consequences [39]. This is because we first observe the objective function values for the members of the sample set and then try to determine whether an unknown point should have a higher or lower objective function value. In other words, it is guessing; it is a fact that induction is impossible when no relation exists among the members [40, 54]. If no prior relation is assumed between them, there is little reason to choose one member over others, and the BBS will be no better than the random search unless the algorithm assumes and exploits some relations among the members of the search domain.

If assuming and exploiting relations among the members of a search space is essential, then it will be wise to isolate this possibility, study it, and see how it can be used to the fullest. The SEARCH framework does that. Recall that SEARCH stands for *Search Envisioned As Relation and Class Hierarchizing*. Searching for better relations and better classes are the primary fronts emphasized in SEARCH. Relations classify the search space into different regions. Some relations classify the search space in such a way that it is relatively easier to detect the class containing the optimal solution. The SEARCH tries to establish such relations among the members of the search space. Instead of directly searching for the best solution from the beginning, the SEARCH tries to find these relations and then uses them to locate the classes containing the optimal solution. The following section presents a brief overview of SEARCH.

### 2.2 Overview

The foundation of SEARCH is laid on a decomposition of the BBS problem into relation, class, and sample spaces. A relation is a set of ordered pairs. For example, in a set of cubes, some white and some black, the color of the cubes defines a relation that divides the set of

cubes into two subsets, the set of white cubes and the set of black cubes. Consider a four-bit binary sequence. There are $2^4$ such binary sequences. This set can be divided into two classes using the equivalence relation (i.e., relation that is reflexive, symmetric, and transitive) $f$###, where $f$ denotes position of equivalence and the # character matches with any binary value. This equivalence relation divides the complete set into two equivalence classes, 1### and 0###. The class 1### contains all the sequences with 1 in the leftmost position and 0### contains those with a 0 in that position. In a BBS problem, relations among the search space members are often introduced through different means, such as representation, operators, heuristics, and others. This example of relations in binary sequences can be viewed as an example of relation in the sequence representation. In a sequence space of length $\ell$, there are $2^\ell$ different equivalence relations. The search operators also define a set of relations by introducing a notion of neighborhood. For a given member in the search space, the search operator defines a set of members that can be reached by one or several applications of the operators. This introduces relations among the members. Heuristics identifies a subset of the search space as more promising than others, often based on some domain-specific knowledge. Clearly this can be a source of relations. Relations can sometimes be introduced in a more direct manner. For example, a bayesian search algorithm is proposed in [42] that divides the search space into Delaunay triangles. This classification directly imposes a certain relation among the members of the search space. The same goes for interval search [43], where the domain is divided into many intervals and knowledge about the problem is used to compute the likelihood of success in those intervals. As we see, relations are introduced in nonenumerative BBS either implicitly or explicitly.

The role of relations in BBS is also very fundamental and important. Although, in a BBS, many relations can be introduced in different ways not all of the relations are appropriate from the search perspective. The objective of sampling-based BBS is to detect regions of the domain that are most likely to contain the optimal solutions. In other words, a BBS algorithm tries to detect those classes defined by a relation which appear more promising. If a relation divides the search space in such a way that such detection is easier, then the relation is appropriate for that problem. We formally define such relations later as those which *properly delineate* the domain. Determining which relation is better requires first constructing a partial ordering among the classes defined by each of the relations. In a sampling-based BBS all this decision making is done by taking a finite number of samples from the domain. Clearly, all the BBS algorithms often implicitly deal with the three distinct spaces: relation, class, and sample. The SEARCH considers all of these explicitly in an effort to understand them rigorously. Figure 1 shows this fundamental decomposition in SEARCH. The major components of SEARCH can be

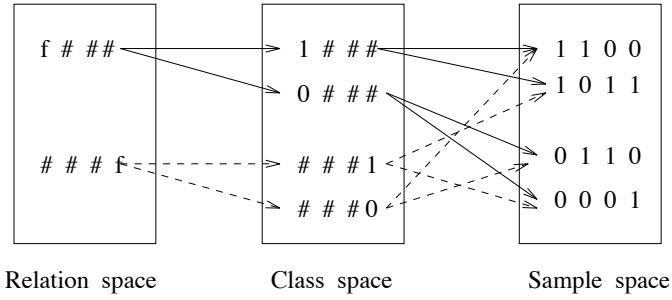


**Figure 1**. Decomposition of BBS in SEARCH.

listed as follows.

1. Classification of the search space using a relation.
2. Sampling.
3. Evaluation, ordering, and selection of better classes.
4. Evaluation, ordering, and selection of better relations.
5. Resolution.

Each of these components is discussed in the following. First we define some notations that are used throughout the paper. A relation is denoted by $r_i$, where $i$ is the index of the set of all relations $\Psi_r$ under consideration of the algorithm. Let $C_i$ be the collection of subsets, created by relation $r_i$. Denote the members of $C_i$ by $C_{1,i}, C_{2,i} \ldots C_{N_i,i}$, where the cardinality of the class $C_i$ is $\|C_i\| = N_i$. Therefore, $C_i$ is a collection of classes. A BBS algorithm may not always have to consider all the relations in $\Psi_r$ for solving a problem. Let $S_r$ be the set of relations actually used by the algorithm in order to solve a given BBS problem.

Once a relation $r_i$ is used to construct $C_i$, the next step is to evaluate the classes in $C_i$. To do that samples from the search domain are needed. A perturbation operator $\mathcal{P}$ is defined as an operator that generates new samples. This operator can either be a random sample generator or a smarter one that exploits information from the relation, class, and sample memory.

The next step is to construct an ordering among the classes in $C_i$. To do that a way is needed to compare any pair of classes. A statistic $\mathcal{T}$ can be computed for each of the classes, and they may be compared on this basis. This statistic is called a *class comparison statistic*. A class comparison statistic can be used for computing a tentative ranking among the classes in $C_i$. For certain choices of $\mathcal{T}$, some classes may not be compared with other classes. This means that sometimes a total order may not be constructed. Therefore, in general, a statistic

$\mathcal{T}$ can be used to construct a partial order on $C_i$. We denote this partially ordered collection by $C_{i[\,]}$. Once the ordering is constructed, the next goal is to select some $1 \leq M_i \leq \|C_i\|$ top ranked classes from $C_{i[\,]}$. $M_i$ represents the total number of top ranked classes that will be selected for future consideration. The exact choice of $M_i$ depends on the decision error probability in choosing an appropriate relation and ordering construction among the classes. For example, if sampling is insufficient, the ordering of classes cannot be relied upon with high confidence, and drastic elimination of classes may not be appropriate. Therefore, a relatively large value of $M_i$ may be used. These $M_i$ classes constitute the updated version of the class search space. In this paper we commonly refer to $M_i$ by "memory size," since the number of such classes that can be kept within the scope of the search often depends on the memory of the algorithm that contributes to the overall cost of computation of the algorithm.

Next, this ordering among the classes is used to evaluate the relation $r_i$ itself. Different kinds of statistics can be used to compare relations with one another. We denote this statistic by $\mathcal{T}_r$ and call it a *relation comparison statistic*. This statistic for relation $r_i$ is now computed. The set of all relations currently under consideration is ordered based on this statistic. Note that, again, this ordering does not have to be a total ordering. The top $M_r$ relations are kept for future consideration and the rest are discarded, in a manner very similar to what was done for the classes.

Not all the classes defined by a relation need to be considered. As more and more relations are evaluated, the information gathered may be used to prune out different classes before evaluating a new relation. Let $r_0$ be a relation that is logically equivalent to $r_1 \wedge r_2$, where $r_1$ and $r_2$ are two different relations; the sign $\wedge$ denotes the logical AND operation. If either $r_1$ or $r_2$ was earlier found to properly delineate the search space with a certain value of $M_i$, then the information about the classes that were earlier found to be bad can be used to eliminate some classes in $r_0$ from further consideration. BBS algorithms often implement a resolution-like process to take advantage of any such possible decomposibility. If the chosen relation $r_i$ can be decomposed into a collection of different relations, denoted by $\cup_k r_k$, then resolution can eliminate bad classes using the information collected from possible earlier evaluations of some relations in $\cup_k r_k$.

Repeated iterations of the above steps result in gradual focusing into those regions of the search space that appear promising through the window of the chosen class and relation comparison statistics. The set of all these relations $r_i, r_{i+1}, \ldots$ used to solve the problem is denoted by $S_r$. Whether or not the algorithm approaches the globally optimal solution depends on its success in finding proper relations, better classes, and sufficient sampling.

### 2.3 SEARCH: The detailed picture

The objective of this section is to present a more quantitative picture of SEARCH and to formalize the earlier descriptions. The definition of a better relation requires defining what is meant by *better classes*. Therefore, the decision making in the class space is considered first in section 2.3.1. Section 2.3.2 considers the class selection process. Section 2.3.3 discusses the relation search. Finally, section 2.3.4 presents the resolution process of SEARCH.

#### 2.3.1 Classification and ordering of classes

This section considers the decision making process among the classes. Classification of the search space requires defining relations. A relation can be defined using different sources, such as operators and representation. In this section we assume no specific source of relations and simply consider a set of relations $\Psi_r$ as an abstract entity provided to the search algorithm. However, we continue to give illustrative examples whenever required, using relations defined by sequence representation.

Once a relation $r_i$ is used to define the collection of classes $C_i$, each of its members needs to be evaluated. Since we are interested in the relative "goodness" of the classes with an ultimate goal of picking up some and rejecting the rest, a statistic that compares any two classes can serve our purpose. If $\mathcal{T}$ is the class comparison statistic used to compare any two subsets $C_{j,i}$ and $C_{k,i}$, then given any two subsets, there must exist an algorithm that returns the resulting order among the subsets when compared on the basis of $\mathcal{T}$. It may also be possible that the two classes cannot be compared based on $\mathcal{T}$. The relation constructed among two ordered subsets of $C_i$ is represented by $\preceq_{\mathcal{T}}$. In other words, when $C_{j,i}$ and $C_{k,i}$ are compared to each other, then either $C_{j,i} \preceq_{\mathcal{T}} C_{k,i}$ or $C_{k,i} \preceq_{\mathcal{T}} C_{j,i}$, or they cannot be compared. When $C_i$ is partially ordered on the basis of $\preceq_{\mathcal{T}}$, it can be represented by a Hasse diagram. Figure 2 (left) illustrates this representation. In a Hasse diagram, the vertices are the members of a poset (partially ordered set); $C_{1,i}$ is drawn above $C_{2,i}$ if and only if $C_{1,i}, C_{2,i} \in C_i$, and $C_{2,i} \preceq_{\mathcal{T}} C_{1,i}$. We can say that $C_{1,i}$ covers $C_{2,i}$ if $C_{1,i}, C_{2,i} \in C_i$, $C_{2,i} \preceq_T C_{1,i}$, and no element $C_{3,i} \in C_i$ satisfies $C_{2,i} \preceq_{\mathcal{T}} C_{3,i} \preceq_{\mathcal{T}} C_{1,i}$. The depth of a node $C_{j,i}$ in a Hasse diagram is the minimum number of links that need to be traversed to reach $C_{j,i}$ from any node at the highest level. Note that this ordering depends on the chosen class comparison statistic.

In a sampling-based search, the partial-order construction process is based on a finite set of samples taken from each of the subsets, $C_{1,i}, C_{2,i}, \ldots C_{N_i,i}$. We denote the approximate descriptions of these classes using the sample sets by $C_{1,i}, C_{2,i}, \ldots C_{N_i,i}$ by $\hat{C}_{1,i}, \hat{C}_{2,i}, \ldots \hat{C}_{N_i,i}$. Let $C_{i[\,]}$ be the ordering of classes from relation $i$. Denote the class at rank $b$ from the bottom of this ordering by $C_{[b],i}$. This means the top

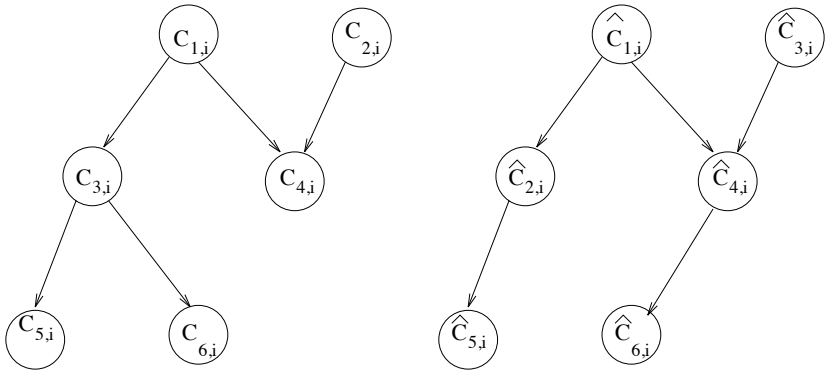


**Figure 2.** Hasse diagram representation of $C_i$ (left) and $\hat{C}_i$ (right).

ranked class in this ordering is denoted by $C_{[N_i],i}$. The partial ordering constructed using the sample estimates may be different from the actual ordering. Figure 2 (right) shows that the partial ordering constructed from sample estimates may differ from the actual ordering of the complete classes with respect to the given $\mathcal{T}$.

#### 2.3.2 Selection of better classes

Once the classes are partially ordered based on $\preceq_{\mathcal{T}}$, the next immediate objective is to select the $M_i$ "top" subsets. Since $C_{i[\,]}$ is a partial order, the notion of top needs to be properly defined. This is an implementation-specific issue. One possible way to define this may be based on the depth of a subset in the Hasse diagram. For the current purpose, we assume that there exists a subroutine TOP$(C_{i[\,]}, M_i)$ which returns the set of "top" $M_i$ subsets from the collection $C_{i[\,]}$. Denote the particular subset that contains $x^*$—the globally optimal solution—by $C_{*,i}$. If we denote the ordered collection of sample sets $\hat{C}_{1,i}, \hat{C}_{2,i}, \ldots \hat{C}_{N_i,i}$ by $\hat{C}_{i[\,]}$, then we would like $\hat{C}_{*,i}$ to be one among the collection of classes returned by TOP$(\hat{C}_{i[\,]}, M_i)$. If $C_{*,i}$ is not in TOP$(C_{i[\,]}, M_i)$, the class containing the optimal solution is discarded and the search algorithm will fail to find the desired solution. For some relations $C_{*,i} \in$ TOP$(C_{i[\,]}, M_i)$ and for some other relations $C_{*,i} \notin$ TOP$(C_{i[\,]}, M_i)$. The former kind of relations are useful in solving a given problem. The following section formalizes this notion.

#### 2.3.3 Selection of appropriate relations: The delineation property

A relation is not appropriate with respect to the chosen $\mathcal{T}$ and the BBS problem if the class containing the optimal solution is not among the top ranked classes, ordered based on $\mathcal{T}$. If the class $C_{*,i}$ is not among the top $M_i$ classes, the algorithm is not likely to succeed (neglecting any chance that may rank $\hat{C}_{*,i}$ higher than its actual ranking). Let us quantify this

requirement of a relation to be appropriate by a function $\mathrm{DC}(r_i, \mathcal{T}, M_i)$. This function returns a one if $C_{*,i} \in \mathrm{TOP}(C_{i[\,]}, M_i)$; otherwise, it returns a zero. This will be denoted by DC() (short for *delineation constraint*), unless otherwise required. Clearly computation of this function requires prior knowledge about $x^*$. Since that is not available in a BBS, this function cannot be directly evaluated. We use this formalism in order to understand the underlying components of BBS.

**Definition 1 (Proper delineation).** For a given BBS problem, a relation $r_i$, a class comparison statistic $\mathcal{T}$, and a memory size, $M_i$, if $\mathrm{DC}(r_i, \mathcal{T}, M_i) = 1$, we say that $r_i$ properly delineates the search space.

This *delineation constraint* plays an important role in SEARCH. It essentially qualifies or disqualifies a relation for a particular search problem. If a relation does not properly delineate the search space, there is very little chance that the class with the best solution will be detected. Therefore, for a given class comparison statistic, whether or not a relation is appropriate can be directly quantified based on this characteristic function. However, in reality the algorithm does not know this constraint. The algorithm has to decide whether or not a relation properly delineates the search space from the limited number of samples taken from it. Therefore, determining whether or not a relation properly delineates the search space is again essentially a decision making problem.

Given a finite set of samples from the search space, a class comparison statistic $\mathcal{T}$, the memory size $M_i$, and a relation $r_i$, the goal is to determine whether a relation classifies the search space in such a way that $C_{*,i}$ is in $\mathrm{TOP}(C_{i[\,]}, M_i)$. Since the problem is now reduced to a decision making problem instead of the previous binary characteristic function, we can approach it using the same strategy that we took for selecting better classes. In other words, we can start comparing relations, estimate how well a relation would satisfy the delineation requirement compared to another relation, and choose the better relation. This problem is similar to the class selection problem; the only difference is that now we are trying to choose better relations instead of better classes. The first question is: How do we compare two relations? While comparing two classes, we needed a class comparison statistic, $\mathcal{T}$. The same thing can be done for relations. Let us denote a *relation comparison statistic* by $\mathcal{T}_r$. This statistic is used to compute an ordering among the relations. Denote this ordering relation by $\preceq_{\mathcal{T}_r}$. The ordering among the relations in $\Psi_r$ may not remain the same when relations are compared based on a limited number of samples. In other words, if $r_j \preceq_{\mathcal{T}_r} r_i$, then it is not necessarily true that $\hat{r}_j \preceq_{\mathcal{T}_r} \hat{r}_i$; we denote a relation $r_i$ when compared based on limited sampling by $\hat{r}_i$. This process of relation selection involves decision making without complete knowledge and it is therefore susceptible to decision errors.

#### 2.3.4 Resolution of classes

Resolution plays an important role in SEARCH. Resolution takes advantage of possible delineability of relations. Classification of the search space defined by a relation is moderated by the resolution process. If possible, resolution eliminates classes that are not necessary to consider by using the information gathered by previous evaluations of some other relations. Let $r_j$ be a relation that properly delineates the search space with memory size $M_j$. Let $r_i$ be the relation currently being evaluated, and $r_i$ can be logically expressed as $r_j \wedge r_k$, where $r_k$ is a relation. Resolution of $C_i$ with respect to $r_j$ eliminates those classes of $C_i$ that need not be considered using our knowledge about $r_j$. This resolved set of classes in $C_i$ can be formally defined as

$$\bigcup_{b=N_j,\ldots N_i-M_j} \; \bigcup_{a=1,\ldots N_i} C_{a,i} \bigcap C_{[b],j}$$

where the index $b$ varies over all of the $M_j$ top ranked classes of relation $r_j$ and index $a$ denotes the different $N_i$ classes in $C_i$. $C_{[b],j}$ is the rank $b$ member of the ordered collection of classes in $C_j$ and $C_{a,i}$ is a member of the unordered collection of classes $C_i$.

## 3. Decision making in SEARCH

The previous sections presented SEARCH from both informal and formal points of view. They also posed the class and relation selection processes as decision problems without complete knowledge. In this section we analyze these two sources of decision error and combine them to develop an expression for the overall success probability.

Two kinds of decision errors may make the selection of better classes erroneous.

1. The relation used to define collection $C_i$ is such that for the chosen $\mathcal{T}$, the subset $C_{*,i}$ is not in $\mathrm{TOP}(C_{i[\,]}, M_i)$. Therefore, despite how well the sampling is done, the selection process will always miss the subset containing $x^*$, unless $\hat{C}_{*,i}$ is ranked higher by sampling error. A search algorithm needs to determine whether or not a relation does this from a finite number of samples. Therefore, this could be a source of error. We call this error the *relation selection error*.

2. Even when $C_{*,i}$ is in $\mathrm{TOP}(C_{i[\,]}, M_i)$, sampling error can produce a different partial order structure for $\hat{C}_{1,i}, \hat{C}_{2,i}, \ldots \hat{C}_{N_i,i}$. As a result, $\hat{C}_{*,i}$ may not be in $\mathrm{TOP}(\hat{C}_{i[\,]}, M_i)$. The sampling error may result in incorrect ordering of the classes and we call this the *class selection error*.

These two dimensions of decision error in BBS determine the success probability. The following sections analyze the success probabilities

associated with each of these dimensions. Finally, they are combined to develop an expression for the overall success probability.

### 3.1 Relation selection success

If an algorithm does not properly delineate the search space, it is not likely to select the class containing the optimal solution. Since, in the absence of knowledge, there is no way to know *a priori* whether a relation satisfies this requirement or not, this can only be estimated based on the sampling information. Relations are ordered based on the measure $\mathcal{T}_r$, and $\|S_r\|$ top relations are selected. Since these top $\|S_r\|$ relations are just the estimated relations that satisfy the delineation constraint, there is the possibility of decision error. If $r_i$ is actually in the top $\|S_r\|$ relations, then the probability that $\hat{r}_i$ will also be within the top $\|S_r\|$ relations depends on correct decision making in comparison with at least $\Psi_r - \|S_r\|$ relations. Denote a relation which actually does not satisfy the delineation constraint by $r_j$. If the minimum probability that $\hat{r}_j \preceq_{\mathcal{T}_r} \hat{r}_i$ over all possible relations is denoted by $\Pr(\hat{r}_j \preceq_{\mathcal{T}_r} \hat{r}_i)_{\min}$, the success probability that $\hat{r}_i$ will be among the top $\|S_r\|$ relations is

$$\Pr(\text{CRS} \mid r_i) \geq \Pr(\hat{r}_j \preceq_{\mathcal{T}_r} \hat{r}_i)_{\min}^{\|\Psi_r\| - \|S_r\|}, \tag{1}$$

where CRS stands for *correct relation selection*.

### 3.2 Class selection success

We now consider the class selection problem. The probability that the best solution is in any of the selected subsets will be denoted by $\Pr(\text{CCS}|r_i)$. CCS stands for *correct class selection* and, conditional to $r_i$, reflects its association with relation $r_i$. Let $\Pr(\hat{C}_{j,i} \preceq_{\mathcal{T}} \hat{C}_{*,i})$ denote the success probability given that $C_{j,i} \preceq_{\mathcal{T}} C_{*,i}$, and let $\Pr(\hat{C}_{j,i} \preceq_{\mathcal{T}} \hat{C}_{*,i})_{\min}$ be the minimum value of $\Pr(\hat{C}_{j,i} \preceq_{\mathcal{T}} \hat{C}_{*,i})$ over every $\hat{C}_{j,i}$ which has a depth greater than that of $\hat{C}_{*,i}$ and there is a link connecting it to $\hat{C}_{*,i}$. Now noting that $M_i$ top classes are selected,

$$\Pr(\text{CCS} \mid r_i) \geq \Pr(\hat{C}_{j,i} \preceq_{\mathcal{T}} \hat{C}_{*,i})_{\min}^{N_i - M_i}$$

This gives the success probability for a particular relation $r_i$.

### 3.3 Overall success

The overall success probability for all the considered relations in $S_r$ then becomes

$$\Pr(\text{CS} \mid \forall r_i \in S_r) = \prod_{\forall r_i \in S_r} \Pr(\text{CRS} \mid r_i) \Pr(\text{CCS} \mid r_i). \tag{2}$$

This equation captures the general idea that will be used in the following sections. As we see, at a higher level, the success of a BBS algorithm depends on the following.

1. The success probability in finding relations that properly delineate the search space.
2. The success probability in detecting the class which actually contains the desired solution.

The following sections specialize the observations of this framework to a specific class comparison statistic and representation. First, we consider ordinal class and relation comparison statistics.

## 4. Ordinal class and relation selection

Different BBS algorithms use different techniques for comparing classes and relations. Since the objective of this section is to understand the cost of ordering classes and relations, we use a distribution-free approach to compare relations and classes.

Construction of a total order and selection of some $M_i$ top subsets from that order have been studied using both parametric and nonparametric approaches [12]. If we are willing to make assumptions about the individual distributions of the members of $C_i$, nice statistics can be formulated to solve this selection problem. However, in the following discussion, we adopt a nonparametric, ordinal approach [6] that allows a distribution-free analysis of the relation and class comparison process. The purpose of this section is to derive bounds on the success probability and sample complexity for quite general ordinal relation and class comparison statistics.

### 4.1 Ordinal class selection

As argued in the section 3, BBS can be viewed as a combined process of search for better relations and better classes defined by each of these relations. We now consider the class comparison process from an ordinal perspective. In order statistics, any two classes can be compared based on their $\alpha$ quantile of the cumulative distribution function (CDF). A quantile of order $\alpha$ can be defined as the number $\Phi_\alpha$, such that $F(\Phi_\alpha) = \alpha$, where $F(\Phi)$ is the CDF of $\Phi$. This definition of quantile is not fully satisfactory when the CDF is discrete and the $\alpha$ quantile may not be unique. In such cases, however, we can define it as any convex combination of points in the closure of the set $\{\Phi : F(\Phi) = \alpha\}$. To convey the main idea without unnecessary cluttering of symbols, we assume that the $\alpha$ quantile is unique. Note that such a quantile-based class comparison will always produce a total order on the collection $C_i$.

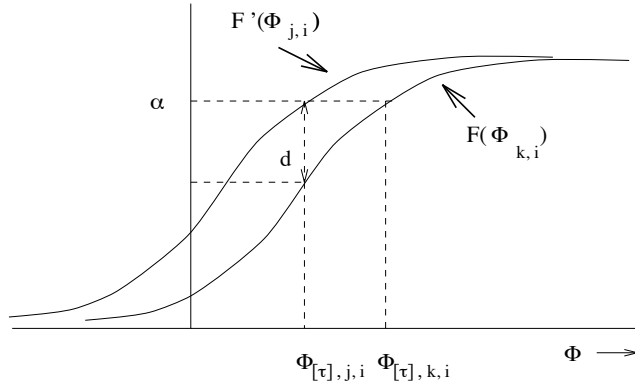


**Figure 3.** Fitness distribution function of two classes $C_{[j],i}$ and $C_{[k],i}$.

Consider the comparison between two classes $C_{j,i}$ and $C_{k,i}$. Assume that we take $n$ samples from each of these classes. We denote $n$ samples from the class $C_{j,i}$ by $\hat{C}_{1,j,i}, \hat{C}_{2,j,i}, \ldots \hat{C}_{n,j,i}$ and the corresponding objective function values by $\Phi_{1,j,i}, \Phi_{2,j,i}, \ldots \Phi_{n,j,i}$. These $n$ samples can be totally ordered on the basis of their objective function values as follows:

$$\hat{C}_{[1],j,i} \preceq_{\Phi} \hat{C}_{[2],j,i} \preceq_{\Phi} \cdots \preceq_{\Phi} \hat{C}_{[n],j,i}$$

where $\hat{C}_{[\omega],j,i} \preceq_{\Phi} \hat{C}_{[\beta],j,i}$ if $\hat{\Phi}_{[\omega],j,i} \leq \hat{\Phi}_{[\beta],j,i}$. We denote the $k$th order statistic by $\hat{\Phi}_{[k],j,i}$. The sample estimate of the $\alpha$ quantile for the class $j$ is denoted by $y_{\alpha,j}$. Define an integer $\tau = \alpha(n+1)$; then, $y_{\alpha,j,i} = \hat{\Phi}_{[\tau],j,i}$. If $\alpha(n+1)$ is not an integer, we can set $\tau$ equal to the largest integer contained in $\alpha(n+1)$ and compute $y_{\alpha,j,i}$ as follows:

$$y_{\alpha,j} = [r + 1 - \alpha(n+1)]\hat{\Phi}_{[\tau],j,i} + [\alpha(n+1) - \tau]\hat{\Phi}_{[\tau+1],j,i}.$$

This basically interpolates between two adjacent order statistics to approximate the point where the CDF is equal to $\alpha$. Again, to keep things simpler, we assume that $\alpha(n+1)$ is an integer.

Figure 3 shows the CDF $F'$ and $F$ of two arbitrary subsets $C_{j,i}$ and $C_{k,i}$, respectively. When these two classes are compared on the basis of the $\alpha$ quantile, then we say $C_{j,i} \preceq_{\alpha} C_{k,i}$, since $\Phi_{[\tau],j,i} \leq \Phi_{[\tau],k,i}$; $\Phi_{[\tau],j,i}$ and $\Phi_{[\tau],k,i}$ are the solutions of $F'(\Phi_{j,i}) = \alpha$ and $F(\Phi_{k,i}) = \alpha$, respectively. We now define

$$d = F(\Phi_{[\tau],k,i}) - F(\Phi_{[\tau],j,i}).$$

We call variable $d$ the *zone of indifference*, which is basically the difference in the percentile value of $\Phi_{[\tau],k,i}$ and that of $\Phi_{[\tau],j,i}$ computed from the same CDF $F$. Figure 3 clearly explains this definition.

It can be easily shown that for $\tau$th order statistics of set $C_{j,i}$ [6], $\Phi_{[\tau],j,i}$,

$$\Pr(\hat{\Phi}_{[\tau],j,i} \leq c') = \sum_{w=\tau}^{n} \binom{n}{w} (F(c'))^{w} (1 - F(c'))^{n-w}. \quad (3)$$

The probability of correct selection among these two classes can be written as [26]

$$\Pr(\hat{\Phi}_{[\tau],j,i} \leq_{\alpha} \hat{\Phi}_{[\tau],k,i}) \geq 1 - 2^{nH(\alpha)} (\alpha - d)^{\alpha n} \quad (4)$$

where $H(\lambda)$ is the binary entropy function, $H(\lambda) = -\lambda \log_2 \lambda - (1 - \lambda) \log_2 (1 - \lambda)$. $H(0) = H(1) = 0$ and $H(\lambda)$ takes the maximum value for $\lambda = 0.5$. For relation $r_i$, if we denote the CDF of the class containing the optimal solution $x^*$ by $F(\Phi_{*,i})$, then define,

$$d'' = \min\{F(\Phi_{[\tau],*,i}) - F(\Phi_{[\tau],j,i}) | \forall j\},$$

the probability that the class $\hat{C}_{*,i}$ will be within the top $M_i$ classes is

$$\Pr(\text{CCS} \mid r_i) \geq [1 - 2^{nH(\alpha)} (\alpha - d'')^{\alpha n}]^{N_i - M_i}. \quad (5)$$

Given relation $r_i$ that properly delineates the search space, equation (5) can be used to compute the probability that $C_{*,i}$ will be within the top $M_i$ classes. Before we proceed toward computing the overall correct selection probability, we need to consider the search in the relation space.

### 4.2 Ordinal relation selection

A relation is appropriate if it properly delineates the search space. Determining whether or not a relation satisfies this constraint with absolute certainty is not possible unless we completely enumerate the search space. Therefore, in reality, the characteristic function DC() is replaced by an estimator that measures how likely a relation satisfies the delineation constraint. We now define a measure $\eta : \Psi_r \times 2^C \times 2^X \rightarrow \Re$. $2^C$ denotes the collection of classes and $2^X$ denotes the sample set. For a given relation $r_i$, the corresponding set of classes $C_i$, and a sample set $\mathcal{S}$, the measure $\eta(r_i, C_i, \mathcal{S})$ returns a real value that corresponds to the chances of $r_i$ satisfying the delineation constraint (i.e., $C_{*,i}$ is a member of TOP($C_{i[\,]}, M_i$)). In short, $\eta(r_i, C_i, \mathcal{S})$ will be written as $\eta_i$. This measure will be used to order the equivalence relations $r_i, r_j \in \Psi_r$. We again adopt an ordinal approach to compare different relations, as done for the selection of better classes. For any two relations $r_i$ and $r_j$, the corresponding $\eta_i$ and $\eta_j$ can be treated as random variables. In the class space the random variable was defined to be the objective function value of the samples. Unlike the class space, in the relation space the random variable is the measure $\eta$, which is defined over a collection of

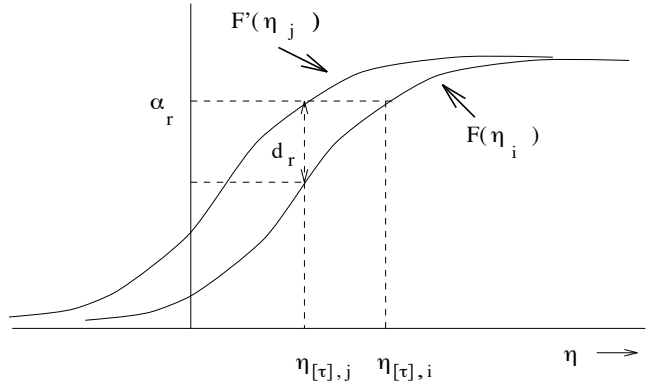


**Figure 4.** CDF of two relations $r_i$ and $r_j$.

classes and a sample set for a given relation. Since, for a given $r_i$, the computation of $\eta_i$ depends on a tuple from $(2^C \times 2^X)$, a collection of $n_r$ such tuples will generate a distribution of different values of $\eta_i$. Figure 4 shows the CDF of two competing relations $r_i$ and $r_j$. Let us say that $r_i$ satisfies the delineation constraint and $r_j$ does not.

If we compare these two relations on the basis of some $\tau_r$th order statistic, the success probability can be computed in exactly the same way that was done for class comparisons. If $\alpha_r$ is the corresponding percentile,

$$\Pr(\hat{r}_{[\tau_r],j} \leq_{\alpha_r} \hat{r}_{[\tau_r],i}) \geq 1 - 2^{n_r H(\alpha_r)}(\alpha_r - d_r')^{\alpha_r n_r} \tag{6}$$

where

$$d_r' = \min\{F(\eta_{[\tau_r],j}) - F(\eta_{[\tau_r],i}) | \forall j, \forall i\}$$

and $F$ is the CDF of the relation comparison statistic of relation $r_i$. $d_r'$ is essentially similar to $d''$, except that this is for relation comparison instead of the previous case of class comparison. In the most general case, a relation needs to be chosen out of all possible relations in $\Psi_r$. However, in reality, it may be true that only one subset of $\Psi_r$ is chosen at a time. In the following analyses we consider the general case, in which all relations in $\Psi_r$ are under consideration. Let us assume that among these $\Psi_r$ relations, the set $\Psi_g \subseteq \Psi_r$ contains all the relations that properly delineate the search space. If relation $r_i \in \Psi_g$, then the probability that $r_i$ will be correctly identified is

$$\Pr(\text{CRS} \mid r_i \in \Psi_g) \geq [1 - 2^{n_r H(\alpha_r)}(\alpha_r - d_r')^{\alpha_r n_r}]^{\|\Psi_r\| - \|\Psi_g\|}.$$

This is the success probability in choosing one good relation. Let $q_r$ be the desired bound on success probability in choosing a relation. If

we need $S_r \subseteq \Psi_r$ relations to solve a problem, we can bound the overall success probability in the relation space as follows [26]:

$$n_r > \frac{\log\left(1 - q^{r^{1/(\|S_r\| \, \|\Psi_r\| - \|\Psi_g\|)}}\right)}{-d_r^*}. \tag{7}$$

Equation (7) can be further rearranged. Define *delineation ratio* as

$$\Omega = \frac{\|\Psi_g\|}{\|\Psi_r\|}. \tag{8}$$

When this ratio is high, searching for appropriate relations is easier, since most of the members of the relation space are appropriate for properly classifying the search space. Using equations (7) and (8) we can write

$$n_r > \frac{\log\left(1 - q^{r^{1/(\|S_r\| \, \|\Psi_r\|(1-\Omega))}}\right)}{-d_r^*}. \tag{9}$$

This bounds the overall computational complexity in the relation space. Equation (9) can be further simplified using the approximation $\log(1 - x) \approx -x$ for $x \ll 1$,

$$n_r > \frac{q^{r^{1/(\|S_r\| \, \|\Psi_r\|(1-\Omega))}}}{d_r^*}. \tag{10}$$

This clearly shows that $n_r$ increases as $q_r$ increases and that $n_r$ increases when $d_r^*$ is reduced. Since $q_r \leq 1$, $n_r$ decreases as $\Omega$ increases. As the number of relations needed to sove the problem $\|S_r\|$ increases, $n_r$ also increases. The collection of relations $\Psi_r$ defines the complete search space for relations. The larger the number of relations in $\Psi_r$, the more computation is required for searching for appropriate relations.

### 4.3 Overall selection success

We now combine the search for better relations and classes and compute the overall success probability. Define

$$d' = \min\{F(\Phi_{[\tau],*,i}) - F(\Phi_{[\tau],j,i})|\forall j, \forall i\}.$$

$d'$ is basically the minimum possible value of $d$ over all classes (index $j$) which are compared with the class containing the optimal solution and all relations (index $i$) in $S_r$. Now let us consider the overall class selection success probability given by equation (2). Note that the relation $\leq_\alpha$ imposes a total order onto $C_i$. Define $N_{\max}$ as the maximum possible value of $N_i$ over all relations in $S_r$. Let $M_{\min}$ be the minimum value of memory size $M_i$. In formal notation,

$$N_{\max} = \max\{N_i|\forall r_i \in S_r\}$$
$$M_{\min} = \min\{M_i|\forall r_i \in S_r\}.$$

If $d^*$ is a constant such that $d' \geq d^*$, just like the previously defined $d_r^*$, then the overall success probability can be bounded as follows [26]:

$$\left[(1 - 2^{nH(\alpha)}(\alpha - d^*)^{\alpha n})^{(N_{\max} - M_{\min})}\right]^{\|S_r\|} q_r \geq q$$

$$n > \frac{\log\left(1 - \left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}}\right)}{\alpha \log(\alpha - d^*)}. \tag{11}$$

The denominator of equation (11) can be simplified to,

$$n > \frac{\log\left(1 - \left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}}\right)}{-d^*}. \tag{12}$$

This inequality bounds the number of samples needed from each class to achieve an overall success probability of $q$ in the combined relation and class spaces and $q_r$ gives the desired level of success probability in choosing $\|S_r\|$ relations correctly. The cost of increasing the bound $q_r$ can be realized using equation (9).

### 4.4 Sample complexity

We define the sample complexity of a BBS algorithm to be the total number of objective function evaluations required for solving a class of problems for a desired degree of accuracy. The bounds on the overall success probability, derived in the previous section, can be directly used to bound the overall sample complexity with respect to the ordinal class and relation comparison statistics. The following analysis considers the contribution of the relation and class search to the overall sample complexity

$$\mathrm{SC} \leq \frac{N_{\max}\|S_r\| \log\left(1 - \left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}}\right)}{-d^*}. \tag{13}$$

This inequality gives the overall sample complexity when the probability of finding the globally optimal solution is at least $q$. This expression can be further simplified using reasonable approximations to clearly explain its physical significance. Since $(q/q_r)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}} \leq 1$ and $\log(1 - x) \approx -x$ for $x \ll 1$, we can approximate equation (13) as,

$$\mathrm{SC} \leq \frac{N_{\max}\|S_r\|}{d^*}\left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}}. \tag{14}$$

Equation (14) presents a clear picture of the contributions of different parameters of the SEARCH framework into the sample complexity. Recall that $q$ is the bound on the overall success probability in the relation

and class spaces combined. Clearly, sample complexity SC grows polynomially with $q$. On the other hand, $q_r$ is the minimum bound in the success probability in choosing all $\|S_r\|$ relations correctly. The cost of demanding higher success probability in the relation space shows up in equation (9). However, as we increase our success probability in the relation space, the overall success probability in the combined relation and class spaces also increases. The sample complexity should therefore decrease as success probability in the relation space increases. Equation (14) clearly shows that SC decreases with an increase in $q_r$. Note that the ratio $(q/q_r)^{\frac{1}{\|S_r\|(N_{\max}-M_{\min})}}$ approaches 1 in the limit as $\|S_r\|(N_{\max}-M_{\min})$ approaches infinity. Therefore, SC grows at most linearly with the maximum index value $N_{\max}$ and the cardinality of the set $S_r$. Recall that $d^*$ defines the desired region of indifference; in other words, it defines a region in terms of percentile within which any solution will be acceptable. Sample complexity decreases as $d^*$ increases.

This bound on sample complexity establishes an insight introduced earlier in this section. In the beginning of section 2, we argued that BBS can perform no better than random enumeration unless we try to exploit the relations among the members of the search space. Now that we have a closed-form bound on sample complexity, let us investigate the case when no relations are assumed among the members. Saying no relations are assumed essentially means that there exists only one relation in $\Psi_r$ that basically divides the complete search space into a set of singleton classes. For our four-bit problem representation, this could be the relation *ffff*. This relation divides the search space into 16 singleton classes, which is essentially the complete search space. From the definition of global optima, we know that such a relation always properly delineates the search space. Therefore, $S_r = 1$ and $q_r = 1$. The index of this relation is the same as the cardinality of the search space. So, $N_{\max} = \|\mathcal{X}\|$, where $\|\mathcal{X}\|$ denotes the size of the search space $\mathcal{X}$. Substituting these into equation (14) we get

$$\mathrm{SC} \leq \frac{\|\mathcal{X}\| q^{\frac{1}{\|\mathcal{X}\|-M_{\min}}}}{d^*}. \tag{15}$$

This inequality clearly tells us that the overall sample complexity becomes the size of the search space when we completely neglect all relations that put at least two members together in a class. The only advantage that we get comes from our relaxation in the desired solution quality ($d^*$) and the overall success probability ($q$). This confirms that although SEARCH provides one particular perspective of BBS, the importance of relations is fundamental and should be emphasized in all possible models of BBS that aspire to guide designing BBS algorithms which perform better than enumerative search. No BBS algorithm can transcend the limit of enumerative search without inducing relations among the members.

Since SC grows as $N_{\max}$ and $\|S_r\|$ grow, polynomial-complexity BBS requires bounding them by polynomials. We define the order ($o(r_i)$) of a relation $r_i$ to be the logarithm of the number of classes that $r_i$ defines. Clearly if $o(r_i) = O(\ell)$ where $\ell$ is the problem size, then $N_{\max}$ will be exponential in $\ell$. As a result, polynomial-complexity BBS are not possible unless we choose to discard a large fraction of them without any consideration. So let us bound $o(r_i)$ by some constant $k$, that is, $o(r_i) \le k$.

As mentioned earlier, equation (13) considers only the cost of correctly ordering the classes for every relation in $S_r$. Once the good classes are selected then resolution can be used to identify the class that may contain the desired solution. Note that this analysis does not require all the relations to be evaluated at the same time. Relations can be evaluated sequentially if desired and knowledge of better classes identified by previously evaluated relations can be used to resolve subsequent relations. The outcome of resolution among all the relations in $S_r$ is a subset of the domain that is likely to contain the desired solution (at least according to the algorithm). Since all relations in $S_r$ are already used up, the only option left is to enumerate this subset of the domain. Clearly, the overall sample complexity is going to be prohibitive if the cardinality of this subset is very large (not polynomially bounded).

Finally, we need to bound the total number of relations that we can evaluate, that is, $\|S_r\| = O(\rho_1(\ell))$. Now we are set to define a class of problems that can be searched in polynomial time. The following definition presents the class of generalized order-$k$ delineable problems that can be searched in polynomial time in SEARCH.

**Definition 2 (Generalized order-*k* delineable problems).** Let $\Psi_{\{o(r)\le k\}} = \{r_i : o(r_i) \le k \ \ \& \ \ r_i \in \Psi_r\}$. For a given class comparison statistic $\mathcal{T}_i$, a problem is *order-k delineable* if there exists a subset $\Psi' \subseteq \Psi_{\{o(r)\le k\}}$ and at least one member of $\Psi'$ has an order equal to $k$, such that its every member $r_i$ satisfies the delineation constraint with memory size $M_i$, and the size of the intersection set

$$\mathcal{G} = \bigcup_{a_1,a_2,\cdots a_k} \bigcap C_{[a_1],i} C_{[a_2],i+1} \cdots C_{[a_k],\|\Psi'\|}$$

is bounded by a polynomial of $\ell$, $\rho_2(\ell)$; $a_j$ in $C_{[a_j],i}$ can take any value between $N_i$ and $N_i - M_i$.

Note that since every relaton in $\Psi'$ satisfies the delineation constraint, by definition the optimal solution must be a member of any of the top classes that any of these relations produces. The intersection of the classes, one from each relation, defines a possible subset of the domain that may contain the desired solution. Since there are different such subsets for every unique combination of classes, all of them must be searched for the desired solution. In the worst case the algorithm needs

to search all of them. As long as $\mathcal{G}$ is bounded by a polynomial of $\ell$ we can do that. Therefore, for every problem in the class of generalized order-$k$ delineable problems, there exists a class of relations that can be used to appropriately detect the classes and detect the desired solution in polynomial time. This class of problems is important primarly because it is amenable to polynomial-time BBS. However, it has additional importance. The bound on the order of the relations essentially means that the underlying structure of the search domain can be captured using relatively "simple" relations that involve only a constant number of search variables. This requirement has a deep connection to the problem of inductive learnability. The machine learning community has already recognized the need of short, concise hypothesis (relations) capturing the concept space, in the success of the inductive learner. A detailed discussion on the implication of this bounded dependency on the learnability can be found elsewhere [36].

This discussion on the SEARCH framework has pointed out the different facets of decision making in BBS and explained why searching for relations is essential in BBS. It also identified a class of BBS problems that can be solved in polynomial sample complexity in SEARCH. This sets the stage for launching algorithms for solving the $k$-delineable problems. However, this paper intends to address the search in evolutionary computation. Therefore, we shall take a short detour and first establish the implications of SEARCH in the context of a classical BBS algorithm of nature—the evolution of life on earth. The following section examines the computational processes in natural evolution and demonstrates that the lessons of SEARCH may open up some new dimensions in evolutionary computation.

## 5. SEARCH and natural evolution

Natural evolution has evolved fitter organisms during the course of time; some species became extinct and some flourished. The development of functionally complex but efficient organisms like human beings has taken place in about two billion years. The human genome is comprised of around $2.9 \times 10^8$ base pairs, which essentially means that the search space is extremely large. Clearly we evolved in a relatively short period of time and the performance of the natural evolutionary search is quite impressive. Unless the evolutionary search was equipped with prior knowledge about the search domain, we have every reason to believe that the BBS in evolution must be subjected to the restrictions of any other BBS algorithm identified earlier in this paper. This section considers the implications of the SEARCH framework in the natural evolutionary BBS and suggests that the process of gene expression may play a key role in the search for relations in evolution. First, let us briefly overview the flow of information in natural evolution.

### 5.1 Information flow in evolution

Information flow in evolution is primarily divided into two kinds:

- *Extracellular flow*. Storage, exploration, and transmission of genetic information from generation to generation.
- *Intracellular flow*. Expression of genetic information within the body of an organism.

The extracellular flow involves replication, mutation, recombination, and transmission of DNA (deoxyribonucleic acid) from parents to offspring. A 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. A detailed description can be found elsewhere [2]. *Eukaryotes* (most of the developed organisms) have two chromosomes in their cell nucleus, and thus are called *diploid* organisms. On the other hand, *prokaryotes*, such as single-celled bacteria, have only one chromosome and are called *haploid* organisms. The DNA sequence is changed by mutation. Crossing over and subsequent recombination result in exchange of base pairs between the parent DNA sequences. These processes result in the generation of new DNA sequences. DNA is then transmitted from the parents to the offspring. The DNA is responsible for defining the phenotype of an organism and thereby controls the suitability of the organism to the environment. This suitability determines the selective pressures on the organism: Fitter organisms survive, and the rest do not. However, the computation of the phenotype from the DNA—gene expression—is an interesting process in itself.

Expression of genetic information from the DNA to protein takes place through several complicated steps. However, the major distinct phases are identified as follows.

- *Transcription*. Formation of messenger ribonucleic acid (mRNA) from DNA.
- *Translation*. Formation of protein from mRNA.
- Protein folding.

Figure 5 shows the different steps of gene expression. Each of them is briefly described in the following.

Transcription synthesizes mRNA from part of the DNA. The mRNA 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). Transcription constructs a sequence of bases from another sequence of bases—the DNA. Transcription is initiated by some particular sequences of bases in DNA. These are known as *promoter regions*.

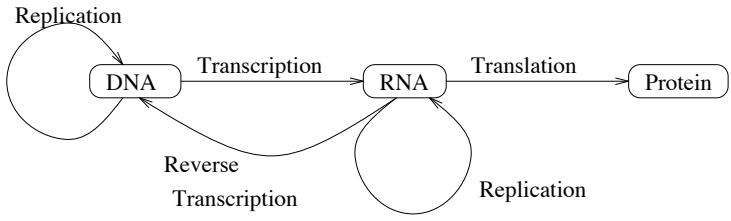


**Figure 5**. Intracellular flow of genetic information.

For example, in many prokaryotes, the *Pribnow box* sequence TATAAT is a common promoter region. Transcription continues until it reaches some particular kind of sequences of bases, known as a *terminator region*. RNA polymerase transcribes the portion of DNA between the promoter and terminator regions. Regulatory proteins of a cell can directly control the transcription of DNA sequences. There are two kinds of regulatory proteins.

- *Gene activator protein*. Enhances transcription of a gene, wherever it binds.
- *Gene repressor protein*. Inhibits transcription of a gene.

These proteins usually bind to specific sequences of DNA and determine whether or not the corresponding gene will be transcribed.

The mRNA act as templates for protein synthesis. Proteins are sequences of *amino acids*, joined by peptide bonds. The mRNA 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. Finally, the sequence of amino acids fold into a three-dimensional structure and this defines a protein. This is called *protein folding* and the performance of a protein mainly depends on its shape. Clearly, the intracellular flow of information is accomplished through a series of changes in information representation (DNA→mRNA→protein). As pointed out earlier in this paper, representation often defines the relation and class spaces. For example, a binary sequence of length $\ell$ defines $2^\ell$ similarity-based equivalence classes that group similar sequences in similar classes. Any process that modifies the representation also transforms the underlying relation space. Clearly, gene expression may hide a mechanism to construct and modify the relations. Moreover, the nature of a protein depends only on the region defined by the promoter and the terminator regions. The set of all DNA sequences that share similarity over this region defines a class that corresponds to a particular

kind of protein. These intriguing manipulations of relations and classes demand further investigation of gene expression. However, first we review the existing work on gene expression in evolutionary computation literature.

### 5.2 Gene expression in existing models of evolutionary computation

Most of the existing computational models of evolution address only the extracellular flow of genetic information. Simple genetic algorithms [7, 16, 20], evolutionary strategy [44], and evolutionary algorithms [10] are some examples. These existing perspectives of evolutionary computation do not assign any computational role to the nonlinear mechanism for transforming the information in DNA into proteins. The same DNA is used for different kinds of proteins in different cells of living beings. The development of different expression control mechanisms and their evolutionary objectives are hardly addressed in these models. They primarily emphasize the extracellular flow. The main difference among these models seems to be the emphasis on crossover compared to mutation or *vice versa*.

Although gene expression is not emphasized very much in most of the popular models of evolutionary computation, several researchers have realized its importance. The importance of the computational role of gene expression was first realized by Holland. He describes in [20] the dominance operator as a possible way to model the effect of gene expression in diploid chromosomes. He also noted the importance of the process of protein synthesis from DNA in the computational model of evolution. Despite the fact that, traditionally, dominance maps are explained from the mendelian perspective, Holland made an interesting leap by connecting it to the synthesis of protein by gene signals, which today is universally recognized as gene expression. He realized the relation between the dominance operator with the "operon" model of the functioning of the chromosome [24] in evolution and pointed out the possible computational role of gene signaling in evolution [20].

Several other efforts have been made to model some aspects of gene expression. Diploidy and dominance have also been used elsewhere [3, 4, 22, 46, 49]. Most of these took their inspiration from the mendelian view of genetics. The underspecification and overspecification decoding operator of messy GA has been viewed as a mechanism similar to gene signaling in [19]. The structured genetic algorithm is proposed in [5]. It uses a structured hierarchical representation in which genes are collectively switched on and off. This implementation also gathered its motivation from gene expression.

An interesting perspective of the natural evolution that realizes the importance for gene expression is offered in [35]. However, Kauffman's work does not explain the process in basic computational terms on

analytical grounds and does not relate the issue to the complexity of search process. The "neutral network" theory was developed in [45]. It considers a sequence-to-structure mapping from the perspective of random graph construction. Although this work provides interesting insights into the physical process of gene expression, its contribution towards polynomial-complexity BBS is not clear.

As we see, the computational role of gene expression in BBS has mostly been unrecognized. Even when duly recognized, little argument has been made to explain its specific role in making search efficient. The following sections will try to develop an understanding of gene expression in light of the SEARCH framework. However, we need to explore why the lessons of the SEARCH should even be relevant in natural evolutionary search.

### 5.3 Evolution of life: Some questions

The problems of our existing computational understanding of evolutionary search will become more clear when we ask some hard questions and demand answers in rigorous computational terms. The objective of this section is to do so and to demonstrate the need for the alternate perspective of evolution that SEARCH offers.

First, in section 5.3.1 the issue of adequate time [35] in evolution is discussed and it is argued that evolution is efficient because it directly searches for relations in *gene expression*. Section 5.3.2 investigates the possible computational role of genetic recombination.

#### 5.3.1 The problem of adequate time

The evolution of living organisms, comprised of a large number of mutually interacting components with an amazing degree of coordination is undoubtedly impressive. This naturally leads us to think about the time that might have been needed to evolve such organisms from primitive ingredients. Some biologists think that there was enough time for evolution to succeed [35] and some of them [23, 48] do not. This is the classical question of "adequate time" in evolution.

John Holland came to an interesting conclusion in [21]. Using the so-called $\alpha$-universe model, he argued that emergence of life on earth in such a short time is only possible if evolutionary search could detect the appropriate equivalence classes or schemata. This was an important argument; unfortunately, this line of argument was largely neglected by biologists and the debate continued primarily because of the lack of analytical results supporting Holland's argument.

Those who believe the inadequate time theory depend on the observation that the search space is too large to be dealt with by random enumeration. Two billion years may be a "long time" compared to our lifetime; it may not be quite so compared to the size of the evolutionary search space. Wald conjectured in [53] that two billion years

of time is sufficient and sampling different organisms during the course of this long time made evolution successful. He even concluded "Time is in fact the hero of the plot." Shapiro criticized Wald's perspective in [48] and presented a convincing argument demonstrating that there was not sufficient time for evolution to succeed. Shapiro estimated the total number of samples that evolution could have taken during the last two billion years. He then computed a conservative bound on the joint probability of finding the set of functional enzymes of a primitive bacterium from this set of samples and showed that the probability of success is extremely low. The joint success probability is so low that it has been compared elsewhere [23] with the chance that "a tornado sweeping through a junkyard might assemble a Boeing 747 from the materials therein." The foundation of this line of argument is based on the assumption that evolution searches by random enumeration. Once we accept this premise their argument makes sense.

In [14] Goldberg indirectly addressed this question on computational grounds following Holland's idea of schema processing. Although his arguments were primarily directed toward computational limitations of evolutionary algorithms such as GAs, their implications for the biological context were equally important. He introduced in [13] order-$k$ deceptive functions which essentially accepted that a BBS algorithm can only efficiently solve problems having a certain degree of decomposability.

Kauffman offered a different way of answering this question in [35]. He argued against the idea of computing the joint success probability for all the different enzymes of living beings. He writes "We should instead be concerned with the probability of finding any one of possibly very many properly coupled sets of enzymatic activities which might constitute a living proto-organism." He proposes that the development of the individual components in the RNA and protein spaces lead to the emergence of the whole in a time shorter than that corresponding to the joint probability computed by Shapiro in [48]. This is an interesting break. Although Kauffman presents his arguments in terms of phase transitions, autocatalysis, and percolation principles, in the opinion of the author his arguments against computing the joint probability make sense only when the protein space is decomposable. Even if the search problem in DNA space is not decomposable, the DNA→RNA→protein transformation may convert the problem into a decomposable one.

As we see, the arguments in favor of adequate time are threefold: (1) Holland's idea of equivalence class processing, (2) Goldberg's argument about problem decomposability, and (3) Kauffman's emphasis on gene expression. However, none of them alone describes the complete picture of the computational processes in natural evolution. In the coming sections we put them together in the light of SEARCH to offer a more complete picture of efficiency in evolutionary search. Before that, we

consider another important factor in evolution—natural selection—and see whether or not existing models of evolutionary computation capture the complete picture.

#### 5.3.2 Natural selection: Some questions

The role of natural selection in evolution is almost universally acknowledged. Natural selection has been identified as one of the main factors defining the evolution and self-organization of many complex systems.

An immediate question that may come to mind is: What does natural selection select? Clearly, living organisms have DNA space and protein space. The DNA sequence defines the set of proteins in an organism. The proteins are in turn responsible for the phenotypic features of the organism. The performance of an organism in its environment may act as an index of the selective pressure. However, the question is: How does this selective pressure effect the organism? Evolution of the DNA and the mechanism for its expression requires different selective pressures and there must be some mechanism to distribute the selective pressure in each of these two spaces.

Unfortunately, existing evolutionary algorithms do not consider the apportionment of selective pressure in these two different spaces. As we saw earlier, evolutionary search algorithms mainly contend with selection in the sample space, corresponding to the effect of natural selection in the DNA space. Clearly, the lack of consideration of the selective pressure in gene expression is a missing feature from the modeling perspective. The question is: Does selective pressure in gene expression affect even the computational modeling of evolutionary search? The answer is yes. However, let us again resist ourselves from explaining the answer until we discuss another puzzle of natural evolution that appears from the role of genetic recombination and crossing-over.

#### 5.3.3 Recombination of what?

Recombination among homologous pairs of chromosomes results in exchanging sets of genes among the parent chromosomes and produces offspring with new chromosomes. A good deal of controversy exists about the utility of recombination. In fact, the field of evolutionary computation appears to be divided into two camps, one supporting the utility of recombination and the other dismissing it. The basic question that we need to ask first is: Recombine what? If we consider the parent chromosome together as a tuple, then recombination is nothing but a permutation operator among the $2\ell$ genes. There are $(2\ell)!$ ways to permute that tuple of $2\ell$ genes. Therefore, searching using recombination is no more efficient than mutative search.

However, recombination is good if we know what to exchange. If we know what relations are good then we only need to exchange the classes that belong to those relations. In an order-$k$ delineable representation

recombination can be used to combine the classes to produce classes of higher order relations.

In natural evolution, the recombination process is controlled by different proteins. For example in *E. coli*, recombination is mediated by products of *rec* genes [50]. After the single-stranded DNA is created by *recBCD* protein, the *recA* protein directly controls the process of binding the duplex DNA, base pairing, and the exchange of strands. Clearly the working of recombination depends on these proteins, and recombination will reduce to a random permutation operator in the absence of these proteins. Therefore, the evolution of the correct proteins appears to be important for the efficient working of recombination.

Unfortunately, most of the existing evolutionary algorithms do not recognize this. One and multipoint crossover [7, 20] and uniform crossover [51] are some examples of artificial crossovers widely used in GAs. In one and multipoint crossovers the points of crossing-over are randomly chosen. In uniform crossover, individual gene swapping is decided randomly. Clearly none of them has any controlling feature. An interesting effort was made in [47] where the use of adaptive crossover that gradually biases toward better classes is suggested. A domain knowledge-based nonuniform crossover is proposed in [37]. Other efforts on adaptive crossovers can be found elsewhere [25, 55].

### 5.4 Evolutionary computation: The SEARCH perspective

Previous sections have clearly explained the need for understanding the processing of relations in natural evolution. In this section we take one step forward by drawing a one-to-one correspondence between the evolutionary search mechanisms and decomposition of BBS in SEARCH.

- *Sample space.* DNA constitutes the sample space. Crossover and mutation generate new samples of DNA. A population of organisms defines the sample space for the evolutionary search.

- *Class space.* Base sequences of mRNA transcribed in a cell correspond to only a part of the complete DNA. The sequence of amino acids in protein in turn corresponds to a base sequence in mRNA. The genetic code tells us that there is a unique relationship between the nucleotide triplets of the DNA and the amino acids in the protein. Therefore, if we consider the DNA as a representation defined over the evolutionary search space for life and different forms of life, then the amino acid sequence of a protein corresponds to a class of different DNA; every DNA in this class must have a certain sequence of nucleotides that can be transcribed to that particular sequence of amino acids. Since the genetic code is unique, a particular sequence of amino acids can only be produced by a certain sequence of nucleotides. In other words, the sequence of amino acids in a protein defines an equivalence class over the DNA space.

| SEARCH | Natural evolution |
|---|---|
| Relation space | gene regulatory mechanism |
| Class space | amino acid sequence in protein |
| Sample space | DNA space |

**Table 1.** Counterparts of different components of SEARCH in natural evolution.

- *Relation space.* Recall that amino acid sequences in protein are translated from the nucleotide sequences of mRNA. The construction of mRNA is basically controlled by the transcription process. Since an equivalence relation is an entity that defines the equivalence classes, the transcription regulatory mechanism can be viewed as the relation space that defines classes in terms of the nucleotide sequences in mRNA and finally in terms of the amino acid sequences in proteins. Among the different components of this mechanism regulatory proteins, promoter regions, and terminator regions play a major role. Regulatory proteins exist as a separate entity from the DNA, but the promoter and terminator regions are defined on the DNA. It appears that there is a distinct relation space comprised of the different regulatory agents, such as activator and inhibitor proteins. However, it is quite interesting to note that this space also directly makes use of information from the sample space—the DNA. Expression of genetic information in eukaryotic organisms is more interesting than that in prokaryotes.

These possible relationships between the different spaces of SEARCH and natural evolution are summarized in Table 1. Now that we have drawn a correspondence between the different components of natural evolution and the SEARCH framework, we are ready to answer the questions raised earlier.

### 5.5 Evolution of life: Some possible answers

In this section we revisit the questions raised in section 5.3 in the light of SEARCH and present some possible explanations.

#### 5.5.1 The issue of adequate time

As shown earlier, the arguments favoring the adequate time theory are threefold: (1) Holland's idea of equivalence class processing, (2) Goldberg's argument about problem decomposability, and (3) Kauffman's argument favoring the importance of evolution in the protein space. Now if we look at the adequate time problem and these arguments in the light of SEARCH we can come to an interesting conclusion—all of them are correct when we put them together. When we do so, the hypothesis appears as follows: *evolution can be successful in such a short period of time if and only if it searches for appropriate equivalence classes defined by the representation and the search problem was either originally decomposable or transformed to a decomposable one*

*in the protein space*. The following part of this section corroborates this hypothesis on the analytical grounds offered by SEARCH.

First of all, SEARCH proved that polynomial complexity BBS is not possible unless some relation among the members of the search space is exploited. Relations define classes and exploiting relations requires processing classes. Therefore, evolutionary search cannot be of polynomial complexity (i.e., efficient) unless it processes classes defined over the genetic representation. The second point is about decomposability. Was the evolutionary search problem decomposable in the initial genetic representation? No one knows, but it is unlikely. The genetic representation and the expression of genetic information (a representational transformation) evolved during the course of evolution. If the evolutionary search landscape were really decomposable and solvable in an efficient manner even at an early stage, such evolution and transformation of representation would not be required. It seems that nature had to search for an appropriate transformation which expressed the genetic expression in such a way that the search problem becomes decomposable at a certain level. The need for problem decomposability and the possible mechanism of gene expression for accomplishing such decomposability can again be corroborated using SEARCH. The SEARCH framework proved that a BBS algorithm can only solve problems that need considerations of relations up to a bounded order—the class of order-$k$ delineable problems. In other words, there must be some degree of decomposability in the relation space. If the given relation space is not order-$k$ delineable, the relation space $\Psi_r$ must be transformed to introduce delineability. Evolutionary search in nature uses a sequence representation. DNA sequence defines the primary representation. Expression of this information using the DNA→RNA→protein defines a new relation space. Searching for an appropriate regulatory mechanism can be viewed as the search for the right relation space that makes the problem order-$k$ delineable. Clearly all three components of the hypothesis supporting adequate time theory can be put in proper perspective in the light of the analytical foundation offered by SEARCH.

#### 5.5.2 Apportionment of selection pressure

Section 5.4 identified the DNA space as the sample space, the protein space as the explicit class space, and the gene regulatory control mechanisms as the relation space. SEARCH clearly points out that no algorithm can surpass the limits of random enumerative search if it guides itself by applying selection in the sample space. Therefore, evolutionary search in nature cannot surpass this computational limit by simply applying the selective pressure in the DNA space. The effect of natural selection must also be distributed in the relation and class spaces of evolutionary search. In other words, the effect of natural selection must show up in the evaluation of parts of DNA sequences into proteins

in different cells of an organism and also the gene regulatory mechanism space.

#### 5.5.3 Recombination of classes

From the SEARCH perspective, recombination serves the purpose of *resolution*. Once the right relations are detected, the corresponding better classes can be *resolved* using a recombination-like operator. Therefore the purpose of recombination in natural evolution is not at all clear unless we introduce the relations as possible controlling agents.

### 5.6 Representation construction

Evolution of the gene regulatory mechanism can be viewed as construction of new representations. Eukaryotic organisms have a richer way of constructing new representations, since most of the eukaryotic organisms are diploid. At a particular gene one allele is recessive and the other is dominant. The expression of a dominant gene takes place during transcription and translation. When a diploid chromosome is viewed as a sequence of dominant and recessive tuples, the set of dominant alleles can be interpreted as a new representation for the set of recessive alleles. The gene regulatory control mechanism determines what gets expressed in a particular cell. The evolution of this regulatory control mechanism is computationally equivalent to the construction of a new relation space. There is existing biological evidence that these settings for the intracellular expression of genetic information evolved during the course of evolution [2]. As noted in SEARCH, such transformation is needed when the original relation space is not order-$k$ delineable. Therefore, one reason why eukaryotic organisms became more successful in the evolutionary race could be their ability to construct new representations. On the other hand, prokaryotes are primarily haploid (i.e., one chromosome only) and are deprived of this capability.

## 6. Messy genetic algorithms: Why and where they stand

So far in this paper it has been argued that searching for appropriate relations and classes is important for efficient BBS and there may exist a mechanism in natural evolution to do exactly this. This section presents preliminary development of a class of evolutionary algorithms that draws motivation from the gene expression process at a higher level for detecting appropriate relations and classes. The proposed algorithm does not necessarily try to replicate the biological process of gene expression; rather it tries to accomplish efficient search for relations and classes which we believe to be one of the main roles of gene expression. This work has its root in the messy genetic algorithms [19] that paved the development of the SEARCH framework. The following sections

review the existing work on messy GAs, their accomplishments and bottlenecks.

### 6.1 Why messy genetic algorithms?

The messy GA was developed in order to address the inadequate processing of relations and classes in traditional models of evolutionary computation. The following discussion summarizes these problems.

1. *Relation, class, and sample spaces are all combined.* Almost every paradigm of evolutionary algorithms uses a single population of samples. This essentially means that the relation, class, and sample spaces are combined and as a result the decision making processes in each of them can affect others in an undesirable way. Only one selection operator is used for deciding in both relation and class spaces.

2. *Poor search for relations.* As shown in the previous sections, none of the evolutionary algorithms emphasizes the role of search for relations. Although in [20] GAs are designed along the right direction using the perspective schemata and partitions, simple GAs unfortunately fails to process relations in an adequate fashion. Simple GAs with single point crossover do not exploit the complete relation space defined by the representation. They evaluate and process only those relations that are defined over positions close to one another. Uniform crossover does not allow proper evaluation of relations unless selection produces more copies. Since the relation space and the sample space are combined, random perturbation of the sample strings also results in disrupting proper evaluation of the relations. Uniform crossover should also fail to accomplish proper search in the relation space. In fact, this is exactly what is reported in [52]. Their analysis and experimental results showed that the sample complexity grows exponentially with the problem size for solving bounded deceptive problems [52] using a simple GA with uniform crossover.

Clearly, these are very fundamental bottlenecks; evolutionary algorithms cannot be used successfully for wide classes of problems unless these bottlenecks are eliminated.

### 6.2 Previous versions of messy genetic algorithms

Long before the development of the SEARCH framework, Goldberg and his students [8, 18, 19, 26] realized the importance of detecting appropriate relations and so proposed a unique class of algorithms known as messy genetic algorithms (MGAs). The original version of the MGA [8, 19] took at least two important steps.

1. It separated the relation and class spaces from the sample space.

2. It deterministically processed all order-$k$ relations and classes.

The original MGA used a population that contained all (deterministically enumerated) order-$k$ classes defined by the chosen representation. This population defined the relation and class spaces together. These classes were evaluated using a template string (a locally optimal solution), which defined the sample space. In other words, in the MGA the sample space was comprised of the locally optimal solutions, found by the algorithm in different iterations. In a particular iteration the sample space was, however, defined by only one locally optimal solution, called template. This decomposition and the emphasis on search for relations made decision making in MGA more accurate compared to other evolutionary algorithms.

Different versions of MGAs studied different aspects of BBS by decomposing it along different dimensions. These investigations have directly influenced the development of SEARCH and the design of the GEMGA. Undoubtedly, the authors of the original version of MGA [19] deserve the credit for first realizing the importance of detecting appropriate relations among the members of the search space.

Another interesting aspect of their work was the class of problems they wanted to solve. The class of bounded deceptive problems [15] captures the essence of order-$k$ delineability developed in SEARCH. They took the right steps in both designing algorithms and identifying a class of problems that can be solved efficiently when the representation is fixed.

However, the MGA had many problems. Some of them are listed in the following.

1. *Combined relation and class spaces.* In MGA the relation space is implicitly defined together with the class space. The SEARCH framework pointed out that these two different spaces require different decision making and they should be processed separately in order to avoid undesirable errors.

2. *Sample space comprised of one template.* A locally optimal solution, the template, defines the sample space. Evaluating and comparing different classes on the basis of one member does not make sense.

3. *Lack of implicit parallelism.* The explicit enumeration of all order-$k$ classes is very expensive ($O(|\Lambda|^k \ell^k)$). SEARCH pointed out that the same sample set can be used for evaluating different relations, which, in the author's opinion, defines the quantitative benefits of implicit parallelism. The MGA completely lacked such computational benefits.

The fast messy GA (FMGA) proposed elsewhere [18, 26] made an effort to reduce the cost of deterministic initialization by using the probabilistically complete initialization (PCI) and building-block filtering (BBF) techniques. Instead of explicitly generating all the order-$k$ classes, the

PCI technique initialized the population with order-$\ell$ classes. This is followed by the BBF process in which the good order-$k$ classes are gradually detected using random gene deletion and thresholding selection. The FMGA indeed reduced the cost of initialization but it succumbed to a fundamental problem that all the versions of MGAs had—implicit definition of the relation space in the class space. The primary objective of the thresholding selection of MGAs was to select good classes defined by the same relation. However, thresholding selection was also implicitly responsible for selecting good relations. This was simply because the chosen value of the thresholding parameter was always less than the string length. The main problem of the FMGA was that thresholding selection could not satisfactorily maintain the growth of strings which are instances of good classes. Undesirable cross-competition among classes from different relations usually eliminated some of the good classes and, as a result, the algorithm required several expensive iterations for solving large problems [26]. Apart from this problem, the FMGA also faced the same questions regarding the use of the single local template. Another problem of the FMGA was that it was slow because of the thresholding selection and it required $O(\ell)$ population sizing since selection took place only at the string or sample level.

Despite several practical applications [9, 38, 41], different versions of the MGAs primarily addressed different fundamental issues and were used for verifying theoretical observations. According to the opinion of this author, MGAs should get credit for taking the following conceptual leaps: (1) realizing the need of searching for relations, (2) decomposing the search space into sample space and class space (with implicit definition of relation space), and (3) paying careful attention to the decision making of a search algorithm.

## 7. The gene expression messy genetic algorithm

In previous sections of this paper, we noted the essential ingredients of efficient, general BBS algorithms and identified a class of BBS problems that can be solved efficiently. We also observed these conclusions in the light of natural evolutionary search. Now it is time to put them together and propose a realization of the theoretical observations along with biological plausibility.

This section introduces the gene expression messy GA (GEMGA)—an $O(|\Lambda|^k(\ell + k))$ sample complexity algorithm for order-$k$ delineable problems in sequence representation of length $\ell$ and alphabet $\Lambda$. Section 7.1 discusses the representation in GEMGA. Section 7.2 explains the population sizing in GEMGA. This is followed by section 7.3 that describes the main operators, transcription, selection, and recombination. Section 7.4 presents the overall mechanisms.

### 7.1 Representation

The GEMGA uses a sequence representation. Like other evolutionary algorithms, the population in GEMGA is comprised of a set of such sequences. Each sequence is called a *chromosome*. Every member of this sequence is called a *gene*. A gene is a data structure, containing the *locus*, *value*, and *weight*. The locus determines the position of the member in the sequence. The locus does not necessarily have to be the same as the physical position of the gene in the chromosome. For example, the gene with locus $i$, may not necessarily be at the $i$th position of the chromosome. When the chromosome is evaluated, however, the gene with locus $i$ gets the $i$th slot. This positional independence in coding was introduced elsewhere [8, 19] to enforce proper consideration for all relations defined by the representation. The GEMGA does not depend on the particular sequence of coding. For a given $\ell$ bit representation, the genes can be placed in an arbitrary sequence. A gene also contains the value, which could be any member of the alphabet set $\Lambda$. The relation space is explicitly evaluated using the weights associated with each member. Weights take a positive real number except at the initial stage when all weights are initialized to $-1.0$. No two members with the same locus are allowed in the sequence. In other words, unlike the original MGA [8, 19] no under- or over-specification is allowed. A population in GEMGA is a collection of such chromosomes.

### 7.2 Population sizing

GEMGA requires at least one instance of the optimal order-$k$ class in the population. For a sequence representation with alphabet $\Lambda$, a randomly generated population of size $\Lambda^k$ is expected to contain one instance of an optimal order-$k$ class. The population size in GEMGA is therefore $n = c\Lambda^k$, where $c$ is a constant. When the signal from the relation space is clear, a small value for $c$ should be sufficient. However, if the relation comparison statistic produces a noisy signal, this constant should statistically take care of the sampling noise from the classes defined by any order-$k$ relation. Since GEMGA uses sequence representation, the relation space contains a total of $2^\ell$ relations. However, GEMGA processes only those relations with order bounded by a constant, $k$. In practice, the order of delineability [26] is often unknown. Therefore, the choice of population size in turn determines what order of relations will be processed. For a population size $n$, the order of relations processed by GEMGA is $k = \log(n/c)/\log|\Lambda|$. If the problem is order-$k$ delineable [26] with respect to the chosen representation and class comparison statistics, then GEMGA will solve the problem, otherwise it will not. In that case a higher population size should be used to consider higher order relations.

## 7.3 Operators

GEMGA has four primary operators, namely: (1) *transcription*, (2) *class selection*, (3) *string selection*, and (4) *recombination*. They are described in the following.

### 7.3.1 Transcription

As mentioned before, the weight space in GEMGA chromosomes is used to process relations. The transcription operator detects the appropriate order-$k$ relations. Comparing relations requires relation comparison statistics. GEMGA does not process the relations in a centralized global fashion; instead it evaluates relations locally in a distributed manner. Every chromosome tries to determine whether or not it has an instance of a good class belonging to some relation. In GEMGA, the quality of a relation is determined by the quality of its good classes distributed over the population. Again, no centralized processing of relations is performed. The transcription operator is a deterministic one that considers one gene at a time. The value of the gene is randomly flipped to note the change in fitness. For a minimization problem, if that change causes an improvement of the fitness (i.e., fitness decreases) then the original instance of the gene certainly does not belong to the instance of the best class of a relation, since fitness can be further improved. Transcription sets the corresponding weight of the gene to zero. On the other hand, if the fitness worsens (i.e., fitness increases) then the original gene may belong to a good class; at least that observation does not say otherwise. The corresponding weight of the gene is set to the absolute value of the change in fitness. Finally, the value of that gene is set to the original value and the fitness of the chromosome is set to the original fitness. In other words, ultimately transcription does not change anything in a chromosome except the weights. For a maximization problem the conditions for the weight change are just reversed. The same process is continued deterministically for all $\ell$ genes in every chromosome of the population. Figure 6 shows the pseudocode for the transcription operator. For genes with a higher cardinality alphabet set ($\Lambda$) this process is repeated for some constant $C < |\Lambda|$ times.

### 7.3.2 Selection

This section describes the two kinds of selection operators used in GEMGA, which correspond to the selective pressures in protein and DNA spaces of natural evolution. Once the relations are identified, a selection operator is applied to make more instances of better classes. GEMGA uses two kinds of selections, class and string selection.

- *Class selection*. The class selection operator is responsible for selecting individual classes from the chromosomes. Better classes detected by the transcription operator are explicitly chosen and given more copies at the expense of bad classes in other chromosomes. Figure 7 describes

```
// pick is the currently considered gene
Transcription(CHROMOSOME chrom, int pick)
{
  double phi, delta;
  int dummy;
  double dwt;

  dwt = chrom[pick].Weight();
  if(dwt > 0.0 OR dwt == -1.0) {
    phi = chrom.Fitness();
    dummy = chrom[pick].Value();
    // Change the value randomly
    chrom[pick].PerturbValue();
    // Compute new fitness
    chrom[pick].EvaluateFitness();
    // Compute the change in fitness
    delta = chrom[pick].Fitness() - phi;
    // For minimization problem
    if(delta < 0.0)
      delta = 0.0;
    // Set the weight
    if(dwt < delta OR delta == 0.0)
      chrom[pick].SetWeight(delta);
    // Set the value to the original value
    chrom[pick].SetValue(dummy);
    // Set the original fitness
    chrom[pick].SetFitness(phi);
  }
}
```

**Figure 6.** Transcription operator for a minimization problem. For a maximization problem, if `delta`$< 0$ the absolute value of `delta` is taken, otherwise `delta` is set to 0.

```
ClassSelection(chrom1, chrom2)
CHROMOSOME chrom1, chrom2;
{
int i;

  for(i=0; i<Problem_length; i++) {
    if(chrom1[i].Weight() >
        chrom2[i].Weight() )
      chrom2[i] = chrom1[i];
    else if(chrom2[i].Weight() >
        chrom1[i].Weight() )
      chrom1[i] = chrom2[i];
  }
}
```

**Figure 7.** Class selection operator in GEMGA. A consistent coding (where `chrom1[i]` and `chrom2[i]` have a common *locus*) is used in place of messy coding for the sake of illustration.

```
Recombination(chrom1, chrom2, pcg)
CHROMOSOME chrom1, chrom2;
double pcg;
{
int i;
GENE dummy;

  for(i=0; i<Problem_length; i++) {
    if(chrom1[i].Weight() >=
        chrom2[i].Weight()
        AND Random() < pcg) {
      dummy = chrom1[i];
      chrom1[i] = chrom2[i];
      chrom2[i] = dummy;
    }
  }
}
```

**Figure 8**. Recombination operator in GEMGA. A consistent coding (where `chrom1[i]` and `chrom2[i]` have a common *locus*) is used in place of messy coding for the sake of illustration. `Random()` generates a random number between 0 and 1; `pcg` is a number between 0 and 1.

the operator. Two chromosomes are randomly picked, the weights of the genes are compared and the gene with higher weight overwrites the corresponding gene in the other chromosome with lower weight.

- *String selection*. This selection operator gives more copies of the chromosomes. A standard binary tournament selection operator [4, 19] is used. Binary tournament selection randomly picks up two chromosomes from the population, compares their objective function values, and gives one additional copy of the winner to the population at the expense of the loser chromosome.

#### 7.3.3 Recombination

Figure 8 shows the mechanism of the recombination operator in GEMGA. It randomly picks up two chromosomes from the population and considers all genes in the chromosomes for possible swapping. It randomly marks one among them. If the weight of a gene from the marked chromosome is greater than that of the corresponding gene from the other chromosome it swaps the genes.

### 7.4 The algorithm

GEMGA has two distinct phases: primordial stage and juxtapositional stage. The primordial stage simply applies the transcription operator for $\ell$ generations, deterministically considering every gene in each generation. During this stage the population of chromosomes remains unchanged, except that the weights of the genes change. This is followed by

```
void GEMGA() {
POPULATION Pop;
int i, j, k, C, k_max;

// Initialize the population at random
Initialize(Pop);
i = 0;
// Primordial stage
While(i < C) { // C is a constant
  j = 0;
  Repeat {
    // Identify better relations
    Transcription(Pop, j);
    // Increment generation counter
    j = j + 1;
  } Until(j == Problem_length)
  i = i + 1;
}
k = 0;
// Juxtapositional stage
  Repeat {
    // Select better strings
    Selection(Pop);
    // Select better classes
    ClassSelection(Pop);
    // Produce offspring
    Recombination(Pop);
    Evaluate(Pop); // Evaluate fitness
    // Increment generation counter
    k = k + 1;
  // k_max is of O(log(Problem_length))
  } Until ( k > k_max )
}
```

**Figure 9**. Pseudocode of GEMGA. The constant C < |Λ|, where |Λ| is the cardinality of the alphabet set.

the juxtapositional stage, in which the selection and recombination operators are applied iteratively. Figure 9 shows the overall algorithm. The length of the juxtapositional stage can be roughly estimated as follows. If $t$ is the total number of generations in the juxtapositional stage, then for binary tournament selection, every chromosome of the population will converge to the same instance of classes when $2^t = n$, that is, $t = \log n / \log 2$. Substituting $n = c|\Lambda|^k$ we get $t = (\log c + k \log |\Lambda|)/\log 2$. A constant factor of $t$ is recommended for actual practice. Clearly the number of generations in the juxtapositional stage is $O(k)$. We now compute the overall sample complexity of GEMGA. Since the population size is $O(|\Lambda|^k)$ and the primordial stage continues for $C\ell = O(\ell)$ generations, the overall sample complexity is

$$\mathrm{SC} = O(|\Lambda|^k(\ell + k)).$$

GEMGA is a direct realization of the lessons from the SEARCH framework. Following SEARCH, it can be recognized that the sample complexity is also a function of the desired quality of the solution and the reliability of the process. However, the implementation of GEMGA through distributed local evaluation of relations and classes outweighs the satisfaction of quantifying the success probability that is straightforward in the case of centralized comparison (as it was in SEARCH) from the practical perspective. Therefore, the reader must realize the dependence of the sample complexity on the desired accuracy of the solution and reliability, implicit in the above arguments.

## 8. Test results: Part I

Designing a test set up requires careful consideration. An ideal set up should contain problems with different dimensions of problem difficulty, such as multimodality, bounded inappropriateness of relation space (BIRS), problem size, scaling, and noise. In this section, we present the performance of GEMGA for problems with controlled degrees of difficulty along all these dimensions.

For all functions three performance indexes are measured. They are, namely, the average number of function evaluations per success (AFPS), the best value reached (BV), and the relative time for nonfunction evaluations (RTNE). For every function we choose the desired solution value (DSV) *a priori*. We say the algorithm is successful if it reaches the DSV. BV is the first solution value found by the algorithm that crosses the DSV. RTNE is the ratio of the computation time excluding the time for function evaluations to the total time.

Section 8.1 considers bounded deceptive problems that have massive multimodality and BIRS. Section 8.2 considers the same problems with nonuniform scaling. Section 8.3 studies the effect of noise in objective function evaluation.

### 8.1 TF1: Massive multimodality and bounded inappropriateness of relation space

Deceptive trap functions [1] are used as basic building blocks for designing this test suite. A trap function can be defined as follows:

$$\begin{aligned} f(x) &= \ell' \quad \text{if} \quad u = \ell' \\ &= \ell' - 1 - u \quad \text{otherwise,} \end{aligned}$$

where $u$ is the number of 1s in the string $x$ and $\ell'$ is the length of the sequence used for representing the variable $x$. In [17] it is shown that such deceptive problems can be used to design problems of bounded difficulty. In a trap function defined over a sequence of length $\ell'$ the order of delineability is $\ell'$ with respect to the class average comparison

| | |
|---|---|
| Recombination probability | 0.0 |
| Gene exchange probability (pcg) | 0.0 |
| Class selection probability | 1.0 |
| String selection probability | 0.0 |

**Table 2.** GEMGA parameters for TF1.

statistics. Although GEMGA does not work using the class average comparison statistic (i.e., when classes are compared with respect to the distribution means) this gives us a simple way of capturing the main essence. When multiple numbers of such functions are concatenated with each other, a problem defined over a sequence of length $\ell$ with order-$\ell'$ delineability can be designed. Since the order of delineability directly controls the BIRS, such concatenated functions can be effectively used for designing problems with BIRS by controlling $\ell'$. Such functions have only $\ell/\ell'$ proper relations among the $\binom{\ell}{\ell'}$ order-5 relations that must be detected in order to find the global solution. Therefore, searching for the appropriate relations is not a trivial job in this class of problems. Apart from BIRS, such functions also offer multimodality. If carefully observed, it can be seen that a trap function has two peaks. One of them corresponds to the string with all 1s and the other is the string with all 0s. If we design a problem by concatenating $m$ such functions, it will have a total of $2^m$ local optima and among them only one will be the globally optimal solution. Clearly this class of problems is massively multimodal and has BIRS, defined by the representation.

For testing the GEMGA, a test function is constructed by concatenating multiple numbers of trap functions, each with $\ell' = 5$. Therefore the order of delineability is five. As we increase the number of functions, in other words the overall problem length $\ell$, the degree of BIRS remains constant, but the degree of multimodality increases exponentially. For $\ell = 200$, the overall function contains 40 subfunctions; therefore, an order-5 bounded 200-bit problem has $2^{40}$ local optima, and among them, only one is globally optimal.

The GEMGA is tested against order-5 deceptive problems of different sizes. Table 2 shows the GEMGA parameters used for all of them. Figure 10 (top) shows the average number of function evaluations from five independent runs needed to find the DSV for different problem sizes. For all problems, the DSV is set to the globally optimal solution, which is equal to the problem size $\ell$. The population size is chosen as described in Section 7.2. Figure 10 (bottom) shows the population size used for different problem sizes. The sample complexity clearly grows linearly and the population size is constant. Figure 11 shows the RTNE, that is, the ratio of the computation time excluding the time

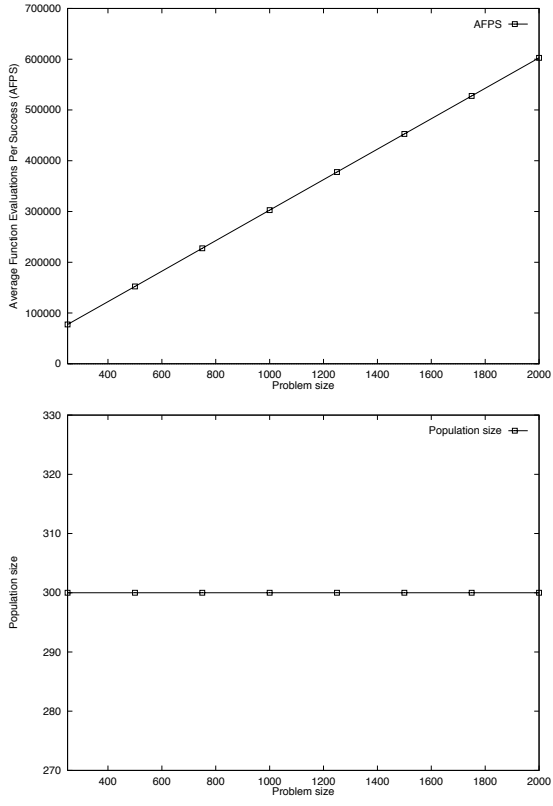


**Figure 10**. Growth of the number of function evaluations with problem size (top). Population size for different problem sizes (bottom). All the results are average of five independent runs.

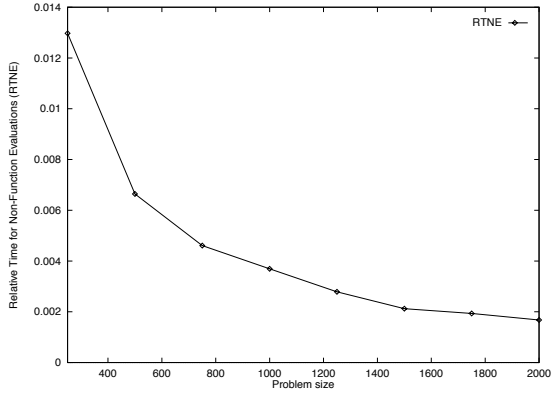


**Figure 11**. Ratio of the time needed to run the algorithm, excluding the time for function evaluations, to the total time (RTNE) is plotted against the problem size.

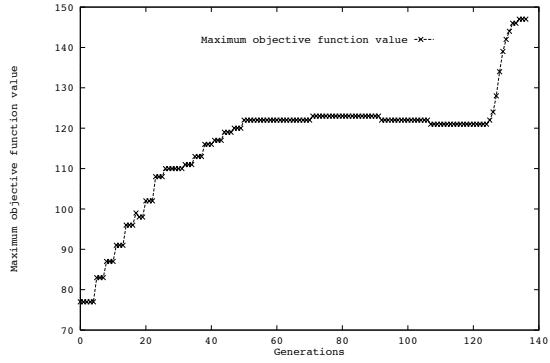


**Figure 12.** Maximum objective function value in different generations for a 150-bit, order-5 deceptive *Trap* function using the fast messy GA. The FMGA found the correct solution for 27 out of the 30 subfunctions. Population size, $n = 8500$.

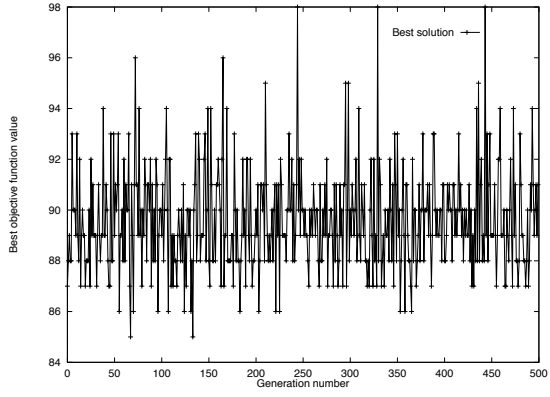


**Figure 13.** Maximum objective function value in different generations for a 150-bit, order-5 deceptive *Trap* function using a naïve mutation based GA. The GA could not find the optimal solution. Population size, $n = 500$.

for function evaluations to the total time, for different problem sizes. RTNE decreases as the problem size increases. The performance of the GEMGA is much better when compared to that of earlier versions of MGAs. For example, consider Figure 12 that shows the performance of the FMGA [18] for 150-bit order-5 deceptive problems. Note that a population size of 8500 is required for solving the 150-bit problem. The overall number of objective function evaluations is about 1,181,500 which is much higher when compared to that of the GEMGA (see Figure 10 (top)). It is also important to note that a naïve mutation-based GA

fails to work for this problem. Figure 13 shows the performance of a mutation-only GA (0.8 mutation probability) with binary tournament selection and a population size of 500.

Figures 14 and 15 show the gradual detection of the relations during the primordial and juxtapositional stages for a 30-bit order-5 deceptive problem. Each figure represents the relation space of the whole population at a certain generation. The $x$-axis denotes the weights in the genes, ordered on the basis of the locus of the gene. In other words, the values along the $x$-axis correspond to the actual value of the locus of a gene in a chromosome, while the $y$-axis corresponds to the different members in the population. The $z$-axis, perpendicular to the page, denotes the real-valued weights of the corresponding gene in the corresponding chromosome. Since the test function is comprised of order-5 trap functions, for any particular gene in a chromosome, there are only four other genes that are related with it. The complete relation space has a cardinality of $2^{30}$. Among $\binom{30}{5}$ order-5 relations there are only six relations that correctly correspond to the actual dependencies defined by the problem. GEMGA needs to detect the relations that relate genes with loci ranging from 0 to 4 together, from 5 to 9 together, and so on. Figure 14 shows that these relations are gradually detected in different chromosomes that contain good classes from those relations. Finally, at the end of the primordial stage (Figure 15 (top)) all of the relations are detected. Figure 15 (middle and bottom) shows the processing of classes during the juxtapositional stage. More instances of good classes are produced by selection and are exchanged among different strings to create higher order relations that finally lead to the optimal solution.

### 8.2 TF2: Adding difficulty due to scaling

The scaling problem presents difficulty to any BBS algorithm that uses a selection-like operator for choosing better solutions from the search space. The problem is that any such sample is an instance of many different classes defined over the search space, and the contribution of different classes in the overall objective function value may be different. Some classes may contribute higher than others. For example, we can find a problem in which having a 1 in the first position instead of a 0 can change the objective function value by, say, 100; on the other hand, the same in the last bit position can increase the objective function value by, say, 1. Since selection usually picks up better samples rather than individual better classes, samples that have a 1 in the first position are likely to be selected regardless of their value in the last position. In other words, a suboptimal class defined by a relation with a fixed bit ($f$) over the last position can get selected in a sample just because it is accompanied by a 1 in the first position. Clearly this is not appropriate

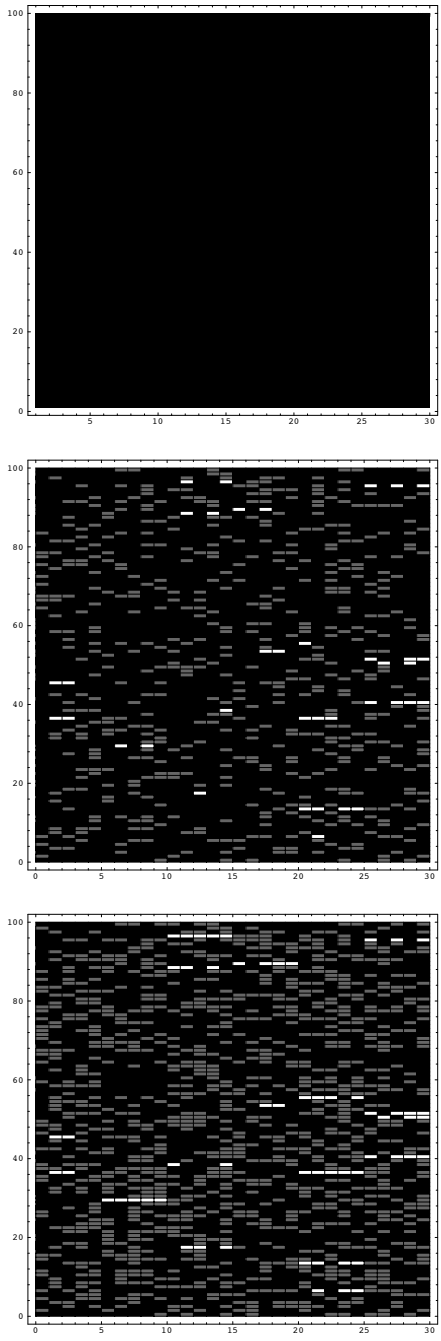


**Figure 14.** The relation space during primordial geneation 1 (top), 10 (middle), and 20 (bottom).

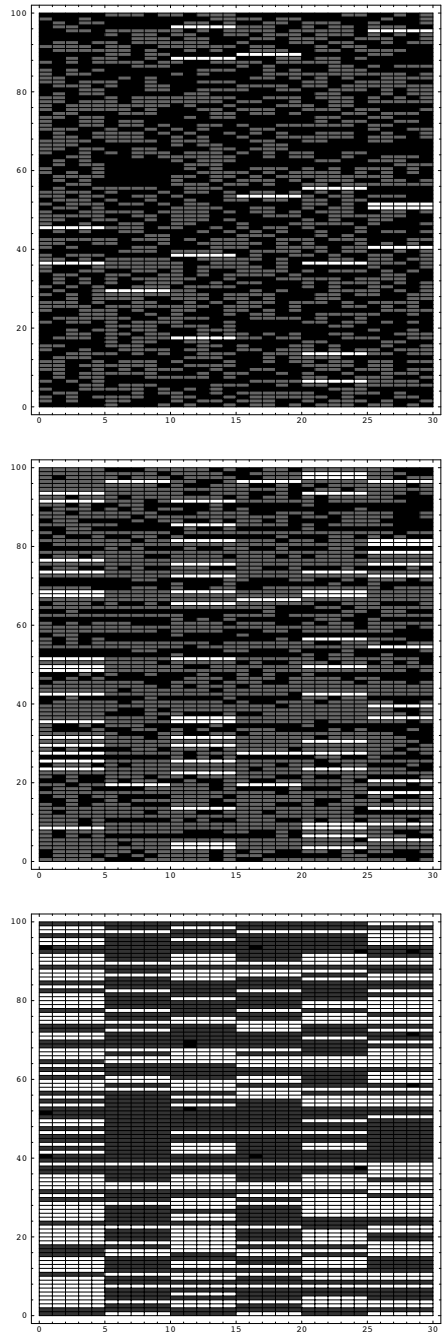


**Figure 15.** The relation space at the end of primordial geneation (top), juxtapositional generations 1 (middle) and 4 (bottom).

and for problems with a large degree of scaling effect, this can lead to suboptimal convergence for the less scaled optimization variables.

The fundamental problem is the complete reliance on selecting samples instead of classes. Fortunately, scaling is not a problem for GEMGA since it also uses a direct class selection operator. The GEMGA is tested against differently scaled trap functions. Concatenated trap functions are used, as in section 8.1. An $\ell = 500$ problem is constructed by concatenating trap functions each with $\ell' = 5$. There are 100 subfunctions and each of them is scaled by a linearly increasing factor. The first subfunction is multiplied by 1, the second one by 2, the third one by 3, and so on. The 100th subfunction is scaled by factor of 100. In this problem the optimal solution for the first $\ell' = 5$ subfunction has an optimal solution with an objective function value of 5; on the other hand the same for the 100th subfunction is 500. Clearly this is a highly scaled problem. The same set of GEMGA parameters as given in Table 2 is used. The population size is 300 as before. The AFPS from five runs was 152,400. The BV and RTNE were 25,250 and 0.0064 respectively. For all the runs the DSV was the globally optimal solution value 25,250.

### 8.3 TF3: Adding noise

Another facet of problem difficulty is noisy objective function evaluation. We use trap functions with additive gaussian noise (mean = 0, variance = 1.0) for testing GEMGA against this aspect of problem difficulty. The same $\ell' = 5$ trap functions are used, except that all the subfunctions are scaled by a factor of 2 to make sure that the function does not become too noisy. The chosen problem size was 100. The noise added to the objective function also makes the signal in the relation and class spaces noisy. Therefore, the GEMGA parameters are made somewhat nondeterministic. Table 3 shows the parameters. The population size was increased to 1000 to take care of the added noise to the decision making processes during the class comparison process. The AFPS from five runs was 141,750. The BV and RTNE were 208.326 and 0.055 respectively. For all the runs the DSV was 200, the best solution value for the deterministic version of the problem.

| | |
|---|---|
| Recombination probability | 0.7 |
| Gene exchange probability (pcg) | 1.0 |
| Class selection probability | 0.3 |
| String selection probability | 1.0 |

**Table 3**. GEMGA parameters for TF3.

| | |
|---|---|
| Recombination probability | 1.0 |
| Gene exchange probability (pcg) | 0.8 |
| Class selection probability | 0.6 |
| String selection probability | 1.0 |

**Table 4**. GEMGA parameters.

## 9. Test results: Part II

This section presents the results for several real test functions. For all these test functions the algorithm parameters are kept constant. Table 4 presents these parameters. As before, three indexes, AFPS, BV, and RTNE, are measured.

Since these problems are in the real domain an iterative version of GEMGA is used for obtaining solutions with a high degree of accuracy. Section 9.1 describes this iterative application process. Section 9.2 presents the test results.

### 9.1 Iterative application of the gene expression messy genetic algorithm with local search

All the test problems used in this section are real functions. Since the sequence representation can only sample a finite subset of the real domain, GEMGA is iteratively applied on a gradually shrinking domain for obtaining solutions with a high degree of accuracy. The initial domain is the same as the given domain of the variables. After GEMGA finds the global solution $\mathbf{x} = \{x_1, x_2, \ldots x_n\}$, the new search domain for the next iteration is defined by $\{(x_1 + \delta, x_1 - \delta), (x_2 + \delta, x_2 - \delta), \ldots (x_n + \delta, x_n - \delta)\}$. The initial value of $\delta$, $\delta_0$ is chosen to be a fraction of the whole domain. For each successive iteration the value of $\delta$ is gradually decreased. A simple rule $\delta = \delta - \delta/\kappa$ is used for decreasing $\delta$, where $\kappa$ is a constant such that $\kappa > 1$.

After several iterative applications of GEMGA, a simple local hillclimber is used to fine tune the solution. The role of this local hillclimber is very minimal. Most of the time GEMGA alone would have been able to find the solution with the desired degree of accuracy. This is used primarily for efficiency reasons.

### 9.2 Real function results

The experiments used $\kappa = 2$ for the domain reduction rule $\delta = \delta - \delta/\kappa$. For every problem the GEMGA is applied for, at most, 15 iterations. 20 different trials are made for each of the test functions. Each GEMGA trial is terminated when either one of the following conditions is satisfied.

| Functions | $N$ | AFPS | BV | RTNE |
|---|---|---|---|---|
| The sphere model | 5 | 1522.6 | 0 | 0.6670 |
| | 10 | 6392.2 | 0 | 0.2381 |
| Griewank's function | 5 | 511797 | 6.38E-05 | 0.2595 |
| | 10 | 890683 | 4.75E-05 | 0.1699 |
| Shekel's foxholes | 5 | 451992 | −9.98 | 0.1115 |
| | 10 | 1.49E07 | −9.62 | 0.0821 |
| Michalewicz's function | 5 | 60219 | −4.6876 | 0.1872 |
| | 10 | 234698 | −9.66 | 0.1510 |
| Langerman's function | 5 | 45783.2 | −1.4976 | 0.0956 |
| | 10 | 443436 | −1.4872 | 0.1029 |

**Table 5**. Test results. Note that the BV is the first value found that crossed the DSV. For all problems the GEMGA reached the DSV.

1. DSV is reached.
2. The solution quality did not change for the last three iterations of GEMGA.

If the DSV is reached, that particular trial is finished. If not, the solution is given to the local hillclimber for fine tuning. It stops when either the desired solution quality is reached or it cannot increase the solution quality. Table 5 presents the performance measures for five real test functions. Every class of test problem is tested for dimensions $N = 5$ and $N = 10$. For all problems a sequence representation with a set of integers as the alphabet is used. Only one gene is assigned for each variable. For example, in a five-dimensional ($N = 5$) problem the chromosome length is 5. The value of the gene is linearly mapped within the search domain as usual. Specific experimental details about each test function are given next.

#### 9.2.1 The sphere function

The sphere function is defined as:

$$\phi(\overline{x}) = \sum_{i=1}^{N} (x_i - 1)^2$$

where $x_i \in [-5, 5]$. A sequence representation with base 11 is used for this class of problems. Other related parameters are: $C = 3$, $\delta_0 = 1.0$, DSV=1.0E-6, and population size = 100. Note that for this function GEMGA reached the desired solution quality without any aid from the local hillclimber.

#### 9.2.2 Griewank's function

Griewank's function is defined as:

$$\phi(\overline{x}) = \frac{1}{d}\sum_{i=1}^{N}(x_i - 100)^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1$$

where $d = 4000$ and $x_i \in [-600, 600]$. A sequence representation with base 301 is used for this class of problems. Other related parameters are: $C = 25$, $\delta_0 = 30$, DSV=1.0E-4, and population size = 300.

#### 9.2.3 Shekel's foxholes

Shekel's foxholes function is defined as:

$$\phi(\overline{x}) = -\sum_{i=1}^{m}\frac{1}{\|\overline{x} - A(i)\|^2 + c_i}$$

where $m = 30$ and $x_i \in [0, 10]$. $A(i)$ is a $30 \times 10$ matrix; $c$ represents a $1 \times 30$ row matrix. Both $A$ and $c$ are not presented here because of length. A sequence representation with base 11 is used for this class of problems. Other related parameters are: $C = 11$, $\delta_0 = 3$, DSV=−9.0, and population size = 3000.

#### 9.2.4 Michalewicz's function

Michalewicz's function is defined as:

$$\phi(\overline{x}) = \sum_{i=1}^{N}\sin(x_i)\sin^{2m}\left(\frac{i\,x_i^2}{\pi}\right)$$

where $m = 10$ and $x_i \in [0, \pi]$. A sequence representation with base 121 is used for this class of problems. Other related parameters are: $C = 11$, $\delta_0 = 1$, DSV= −4.687 (for $N = 5$), DSV= −9.0 (for $N = 10$), and population size = 500.

#### 9.2.5 Langerman's function

Langerman's function is defined as:

$$\phi(\overline{x}) = -\sum_{i=1}^{m} c_i\left(e^{-\frac{1}{\pi}\|\overline{x} - A(i)\|^2}\cos\left(\pi\|\overline{x} - A(i)\|^2\right)\right)$$

where $x_i \in [0, 10]$ and $A(i)$ is a $30 \times 10$ matrix; $c$ represents a $1 \times 30$ row matrix. Both $A$ and $c$ are not presented here because of length. A sequence representation with base 121 is used for this class of problems. Other related parameters are: $C = 5$, $\delta_0 = 3$, DSV=−1.4, and population size = 500.

## 10. Conclusion

Pure blackbox search (BBS) is essentially search in absence of knowledge. In absense of any knowledge about the search domain one can

do hardly any better than unguided random search over the class of all problems. However, if we restrict the domain of problems under consideration, restricted search for certain kinds of relations may allow a BBS algorithm to take the inductive leap over random unguided search. The SEARCH framework observes that. This paper makes use of this observation to identify the prospect of polynomial-complexity BBS for the class of order-$k$ delineable problems by detecting the structure of the search space using up to order-$k$ relations. The class of order-$k$ delineable problems imposes the restriction of bounded variable interaction in the relations under consideration. This requirement has a deep connection with the learnability issues of polynomially-descriptive hypothesis space. Without a similar restriction polynomial-complexity BBS for the general class of all problems is quite hopeless.

Since search for relations in order-$k$ delineable problems is important for making the inductive leap, BBS for this class of problems must have efficient and accurate algorithms to do so. Evolutionary search algorithms are no exception. This paper notes that natural evolution has been quite successful in evolving complex organisms in a relatively short period of time. If that is true, and we interpret this as an outcome of a polynomial-time search, then nature must also face the same restrictions unless there was prior knowledge about the search domain. In the absence of adequate domain knowledge, nature must face the problem of efficient detection of relations in order to evolve highly adaptive organisms in a short period of time. Since doing that in polynomial time without any restriction on the class of problems and relations is impossible, natural evolutionary search must have some restriction on the class of problems and relations. This paper imposes the restriction of order-$k$ delineability in the context of natural evolution and explores the implications. It also suggests that the process of gene expression may play a significant role in the search for appropriate relations. The suggested relation with the gene expression is argued only at a higher level from a functional perspective. Although analysis of specific mechanisms of gene expression is beyond the scope of this paper, efforts along that direction are currently being carried out [32, 33, 34].

Finally this paper combines the observations of the SEARCH framework and its implications in the context of evolutionary computation to develop the preliminary structure of an experimental evolutionary algorithm. This algorithm, called the gene expression messy genetic algorithm (GEMGA), is developed and tested against a large number of different optimization problems. Test results for large problems with millions of local optima and bounded variable interaction demonstrated promising performance. Although the proposed relation search technique is heuristic-based and better technqiues are currently under development [29, 30, 32], the material in this paper presents the overall

philosophical approach towards a new class of well-grounded evolutionary algorithms.

## 11. Acknowledgements

The early stage of this work was supported by AFSOR Grant F49620-94-1-0103 and the Illinois Genetic Algorithm Laboratory. The following stages of the design and experimentation have been performed at Los Alamos National Laboratory under the auspices of the U.S. Department of Energy. The author would also like to acknowledge the support from National Science Foundation Grant IIS-9803360. The author would also like to thank Professor David E. Goldberg for his input in this work.

## References

[1] D. H. Ackley, *A Connectionist Machine for Genetic Hill Climbing* (Kluwer Academic, Boston, 1987).

[2] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson, *Molecular Biology of the Cell* (Garland Publishing Inc., New York, 1994).

[3] J. D. Bagley, "The Behavior of Adaptive Systems which Employ Genetic and Correlation Algorithms," *Dissertation Abstracts International*, **28**(12) (1967) 5106B. University Microfilms Number 68-7556.

[4] A. Brindle, *Genetic Algorithms for Function Optimization* (Unpublished doctoral dissertation, University of Alberta, Edmonton, Canada, 1981.)

[5] D. Dasgupta and D. R. McGregor, "Designing Neural Networks Using the Structured Genetic Algorithm," *Artifical Neural Networks*, **2** (1992) 263–268.

[6] H. A. David, *Order Statistics* (John Wiley and Sons, Inc., New York, 1981).

[7] K. A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," *Dissertation Abstracts International*, **36**(10) (1975) 5140B. University Microfilms Number 76-9381.

[8] K. Deb, "Binary and Floating-point Function Optimization Using Messy Genetic Algorithms," IlliGAL Report Number 91004, University of Illinois at Urbana-Champaign; and doctoral dissertation, University of Alabama, Tuscaloosa, 1991.

[9] A. Dymek, "An Examination of Hypercube Implementations of Genetic Algorithms," masters thesis and Report Number AFIT/GCS/ENG/92M-02, Air Force Institute of Technology, Wright-Patterson Air Force Base, 1992.

[10] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution* (John Wiley, New York, 1966).

[11] S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms* (Morgan Kaufmann, San Mateo, CA, 1993).

[12] J. D. Gibbons, M. Sobel, and I. Olkin, *Selecting and Ordering Populations: A New Statistical Methodology* (John Wiley and Sons, Inc., New York, 1977).

[13] D. E. Goldberg, "Simple Genetic Algorithms and the Minimal, Deceptive Problem," in *Genetic Algorithms and Simulated Annealing*, edited by L. Davis, (Morgan Kaufmann, San Mateo, CA, 1987). Also TCGA Report 86003.

[14] D. E. Goldberg, "Genetic Algorithms and Walsh Functions: Part I, a Gentle Introduction," *Complex Systems*, **3** (1989) 129–152. Also TCGA Report 88006.

[15] D. E. Goldberg, "Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis," *Complex Systems*, **3** (1989) 153–171. Also TCGA Report 89001.

[16] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, New York, 1989).

[17] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic Algorithms, Noise, and the Sizing of Populations," *Complex Systems*, **6** (1992) 333–362.

[18] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik, "Rapid, Accurate Optimizaiton of Difficult Problems Using Fast Messy Genetic Algorithms," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993.

[19] D. E. Goldberg, B. Korb, and K. Deb, "Messy Genetic Algorithms: Motivation, Analysis, and First Results," *Complex Systems*, **3** (1989) 493–530. Also TCGA Report 89003.

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

[21] J. H. Holland, "Studies of the Spontaneous Emergence of Self-replicating Systems Using Cellular Automata and Formal Grammars," in *Automata, Languages, Development*, edited by A. Lindenmayer and G. Rozenberg (Numberth-Holland, New York, 1976).

[22] R. B. Hollstien, "Artificial Genetic Adaptation in Computer Control Systems," *Dissertation Abstracts International*, **32**(3) (1971) 1510B. University Microfilms Number 71-23,773.

[23] F. Hoyle and N. C. Wickramasinghe, *Evolution from Space* (Dent, London, 1981).

[24] F. Jacob and J. Monod, "Genetic Regulatory Mechanisms in the Synthesis of Proteins," *Molecular Biology*, **3** (1961) 318–356.

[25] P. Jog, J. Y. Suh, and D. Van Gucht, "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem," in *Proceedings of the Third International Conference on Genetic Algorithms*, edited by J. D. Schaffer, 1989.

[26] H. Kargupta, *SEARCH, Polynomial Complexity, and The Fast Messy Genetic Algorithm*, PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, October 1995. Also available as IlliGAL Report 95008.

[27] H. Kargupta, "The Gene Expression Messy Genetic Algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation* (IEEE Press, 1996).

[28] H. Kargupta, "Performance of the Gene Expression Messy Genetic Algorithm on Real Test Functions," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, (IEEE Press, 1996).

[29] H. Kargupta and S. Bandyopadhyay, "A Perspective on the Foundation and Evolution of the Linkage Learning Genetic Algorithms," accepted in the special issue in genetic algorithms: *The Journal of Computer Methods in Applied Mechanics and Engineering*, edited by D. E. Goldberg and K. Deb.

[30] H. Kargupta and S. Bandyopadhyay, "Further Experimentations on the Scalability of the GEMGA," in *Lecture Numberes in Computer Science: Parallel Problem Solving from Nature* (Springer-Verlag, 1998).

[31] H. Kargupta and D. E. Goldberg, "SEARCH, Blackbox Optimization, and Sample Complexity," in *Foundations of Genetic Algorithms*, edited by R. Belew and M. Vose (Morgan Kaufmann, San Mateo, CA, 1996).

[32] H. Kargupta, I. Hamzaoglu, and B. Stafford, "Web Based Parallel/Distributed Data Mining Using Software Agents," in *Proceedings of the American Medical Association Fall Symposium* (AMIA, 1997).

[33] H. Kargupta and K. Sarkar, "Function Induction, Gene Expression, and Evolutionary Representation Construction," in *Proceedings of the Genetic and Evolutionary Computation Conference, volume 1*, edited by D. Goldberg and J. Koza (AAAI Press, Orlando, FL, July, 1999).

[34] H. Kargupta and B. Stafford, "From DNA to Protein: Transformations and their Possible Role in Linkage Learning," in *Proceedings of the Seventh International Conference on Genetic Algorithms*, July 1997.

[35] S. Kauffman, *The Origins of Order* (Oxford University Press, New York, 1993).

[36] S. Kushilevitz and Y. Mansour, "Learning Decision Trees Using Fourier Spectrum," in *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing* (ACM Press, 1991).

[37] H. Maini, K. Mehrotra, C. Mohan, and S. Ranka, "Knowledge-based Nonuniform Crossover," in *Proceedings of the First IEEE Conference on Evolutionary Computation, volume 1* (IEEE Service Center, Piscataway, NJ, 1994).

[38] L. D. Merkle and G. B. Lemont, "Comparison of Parallel Messy Genetic Algorithm Data Distribution Strategies," in Forrest [11], pages 191–205.

[39] R. S. Michalski, "Theory and Methodology of Inductive Learning," in *Machine learning: An Artificial Intelligence Approach*, edited by R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Tioga Publishing Co., 1983).

[40] T. M. Mitchell, "The Need for Biases in Learning Generalizations," Rutgers Computer Science Technical Report CBM-TR-117 (Rutgers University, New Brunswick, NJ, 1980).

[41] C. K. Mohan, "A Messy Genetic Algorithm for Clustering," in *Intelligent Engineering Systems Through Artificial Neural Networks*, edited by C. H. Dagli, L. I. Burke, Fernádez, and J. Ghosh (ASME Press, New York, 1993).

[42] C. Perttunen and B. Stuckman, "The Rank Transformation Applied to a Multi-univariate Method of Global Optimization," *IEEE Transactions on System, Man, and Cybernetics*, **20** (1990) 1216–1220.

[43] H. Ratschek and R. L. Voller, "What Can Interval Analysis Do for Global Optimization?" *Journal of Global Optimization*, **1** (1991) 111–130.

[44] I. Rechenberg, "Bionik, evolution und optimierung," *Naturwissenschaftliche Rundschau*, **26** (1973) 465–472.

[45] C. Reidys and S. Fraser, "Evolution of Random Structures," Technical Report 96-11-082 (Santa Fe Institute, Santa Fe, 1996).

[46] R. S. Rosenberg, "Simulation of Genetic Populations with Biochemical Properties," *Dissertation Abstracts International*, **28**(7) (1967) 2732B. University Microfilms Number 67-17,836.

[47] J. D. Schaffer and A. Morishima, "An Adaptive Crossover Distribution Mechanism for Genetic Algorithms," in *Proceedings of the Second International Conference on Genetic Algorithms*, edited by J. J. Grefenstette, 1987.

[48] R. Shapiro, *Origins: A Skeptic's Guide to the Creation of Life* (Summit Books, New York, 1986).

[49] R. E. Smith, "An Investigation of Diploid Genetic Algorithms for Adaptive Search of Nonstationary Functions," TCGA Report Number 88001 (University of Alabama, The Clearinghouse for Genetic Algorithms, Tuscaloosa, 1988).

[50] L. Stryer, *Biochemistry* (W. H. Freeman Co., New York, 1988).

[51] G. Syswerda, "Uniform Crossover in Genetic Algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, edited by J. D. Schaffer, 1989.

[52] D. Thierens and D. Goldberg, "Mixing in Genetic Algorithms," in Forrest [11], pages 38–45.

[53] G. Wald, "The Origin of Life," *Scientific American*, August 1954.

[54] S. Watanabe, *Knowing and Guessing—A Formal and Quantitative Study* (John Wiley and Sons, Inc., New York, 1969).

[55] T. White and F. Oppacher, "Adaptive Crossover Using Automata," in *Parallel Problem Solving from Nature—PPSN III*, edited by Y. Davidor, Hans-Paul Schwefel, and R. Männer (Springer-Verlag, Berlin, 1994).