# Blackbox Optimization: Implications Of SEARCH*

**Hillol Kargupta**[†]**& David E. Goldberg**[‡]
Department of Computer Science & Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA.

**Abstract**

The SEARCH (Search Envisioned As Relation & Class Hierarchizing) framework developed elsewhere (Kargupta, 1995; Kargupta & Goldberg, 1995) offered an alternate perspective toward blackbox optimization—optimization in presence of little domain knowledge. The SEARCH framework investigates the conditions essential for transcending the limits of random enumerative search using a framework developed in terms of relations, classes and partial ordering. The primary motivation comes from the observation that sampling in blackbox optimization is essentially an inductive process (Michalski, 1983) and in absence of any relation among the members of the search space, induction is no better than enumeration (Watanabe, 1969). This paper presents the main implications of SEARCH. The lessons from SEARCH are used to develop a comprehensive approach to identify the role of algorithm, problem, and the user in blackbox optimization. The computational benefit of *implicit parallelism* (Holland, 1975) is analytically identified. This paper also presents the SEARCH perspective of problem difficulty and identifies the class of *order-k delineable* problems that can solved in sample complexity, polynomial in problem dimension, reliability and accuracy of decision making. SEARCH also offers an alternate perspective of natural evolution that emphasizes on the computational role of gene expression (DNA→RNA→Protein). The main arguments of this perspective are briefly discussed.

# 1 Introduction

Blackbox optimization (BBO)—optimization in the presence of little domain knowledge—has become popular in the recent years primarily because of the advent of powerful computers. As the computational capabilities of machines grow the demand for solving large difficult optimization problems increases accordingly. This has resulted the design and development of many new BBO algorithms like simulated annealing (Kirpatrick, Gelatt, & Vecchi, 1983), genetic algorithms (Holland, 1975), and tabu search (Glover, 1989). These algorithms apparently differ from one another on many aspects. Their performance and scope often vary widely from one problem to another. With all these new algorithms, designed based on seemingly different principles, it is quite natural to ask whether any common principle and common ground exist for approaching BBO.

The SEARCH (Search Envisioned As Relation, and Class Hierarchizing) framework, developed elsewhere (Kargupta, 1995; Kargupta & Goldberg, 1995) made an attempt to answer this question. SEARCH offers an alternate perspective of BBO in terms of relations, classes, and partial orderings. It starts by noting that blackbox search is an induction problem, i.e., a matter of guessing based on what is known. Induction is no better than enumerative table look-up when no relation exists among the members of the search space (Mitchell, 1980; Watanabe, 1969). SEARCH realizes this, and it tries to exploit the properties of different relations that can be defined among the members of the search space. It decomposes the search for optimal solution in BBO into three components, searching in

1. relation space
2. class space

*Some portions of this paper are taken verbatim from Kargupta (1995), Kargupta and Goldberg (1995)

†Currently with Computational Science Methods division, Los Alamos National Laboratory, P.O. Box 1663, XCM-F645, Los Alamos, NM 87545, USA.

‡Department of General Engineering, University of Illinois at Urbana-Champaign.

3. sample space

This decomposition lays the foundation of the SEARCH framework. A probabilistic and approximate framework is developed on this foundation to give SEARCH all its capabilities. Detailed qualitative and quantitative analyses of this framework are presented elsewhere (Kargupta, 1995). Kargupta and Goldberg (1995) also showed that different BBO algorithms such as simulated annealing, genetic algorithms, and Bayesian algorithms can be viewed in the light of SEARCH. In this paper we shall first briefly review the main results and then primarily focus on the implications of SEARCH.

The SEARCH framework has many implications on the approach toward solving BBO problems. Solving a BBO problem requires an understanding of the three participants of the process:

1. algorithm
2. problem
3. user

SEARCH identifies the different components of a BBO algorithm. It also provides us with an understanding of problem difficulty in a quantitative manner. It answers many questions like: What are essential components of a BBO algorithm that aspire to transcend beyond random enumerative search? Why searching for appropriate relations is important in BBO? How does SEARCH characterize difficult BBO problems? Is there a class of blackbox problem that is solvable in polynomial sample complexity?[1] What is the role of user in blackbox search? Both qualitative and quantitative arguments are used to answer these questions.

Section 2 presents a brief review of the related works. Section 3 describes the main components of SEARCH. Section 4 discusses the different decision makings in SEARCH. The main analytical results for ordinal decision making are presented in Section 5. Section 7 discusses the lessons from SEARCH in the context of natural evolution. The emphasis on search for appropriate relations in SEARCH is used to understand the computation in gene expression (DNA→RNA→Protein) of natural evolution. Section 8 summarizes the paper. Finally, Section 9 presents the ramifications and conclusions.

# 2 Background

Although optimization has been addressed in both theory and practice for several centuries, the methodology for solving optimization problems have often followed a pattern: Given a very specific class of problems with some known properties, design an algorithm for solving this class. Unfortunately, because of the ever-growing list of different optimization problems, the process of designing new problem-specific algorithms is unlikely to terminate. Designing algorithms for solving blackbox problems—optimization problems with little knowledge available about the problem domain—offers an alternate approach. By assuming little knowledge about the problem, algorithms designed using this approach aspire to solve a more general class of problems.

The purpose of this section is to introduce BBO and to review some earlier works. Section 2.1 introduces BBO and Section 2.2 reviews some existing works on BBO.

## 2.1 Blackbox optimization

Almost every discipline of engineering and science can make use of optimization algorithms. As a result, a large number of optimization algorithms have been developed and applied to different problems. For example, smooth convex functions can be efficiently optimized using gradient search techniques (Papadimitriou & Steiglitz, 1982) The simplex algorithm (Dantzig, 1963) performs well for a large class of linear programming problems. Dynamic programming techniques (Dreyfus & Law, 1977) work well when the optimization problems are stage decomposable. Several analyses have been done for local and global optimization of real functions that are Lipschitz continuous with a known Lipschitz constant (Törn & Žilinskas, 1989; Vavasis, 1991). This approach of operations research is characterized by a pattern: Given a class of problem, find an algorithm to solve it. Unfortunately, this approach of designing algorithms that work only for a specific class of problems does not ever seem to end, as the list of different types of optimization problems continues to grow.

[1] number of samples needed to solve a problem grows polynomialy with the size of the problem, accuracy of the solution, and the reliability.

Moreover, determining the class of problems in which a real-world optimization problem belongs is often as difficult as finding a reasonable solution for the problem. The ever-increasing computing capability has also fueled the desire for solving large-scale problems with little prior knowledge about the objective functions.

This growing demand for algorithms to solve new classes of difficult optimization problems and the never-ending process of designing algorithms that work for a restricted class of problems suggest the need for an alternate approach. The applicability of the previously mentioned optimization algorithms is very restricted, because these algorithms make assumptions about the properties of the objective functions that are often too restrictive. Therefore, one step toward designing optimization algorithms that work for a large class of problems is to reduce assumptions about the objective function. Since these algorithms make little assumption about the objective function, they should be able to solve problems using as little domain knowledge as possible. These problems would fall into the general class of blackbox optimization (BBO) problems, where little knowledge about the objective function is assumed. In this model of optimization, the objective function is often available as a black box, i.e., for a given $x$ in the feasible domain, it returns the function value $\Phi(x)$. No local or global information about the function is assumed. Let us denote the finite input and the output spaces by $\mathcal{X}$ and $\mathcal{Y}$, respectively. The general blackbox optimization problem can be formally defined as follows. Given a blackbox that somehow computes $\Phi(x)$ for an input $x$,

$$\Phi : \mathcal{X} \rightarrow \mathcal{Y} \tag{1}$$

The objective of a maximization problem is to find some $x^* \in \mathcal{X}$ such that $\Phi(x^*) \geq \Phi(x)$ for all $x \in \mathcal{X}$. Performance of an optimization algorithm in this model depends on the information collected by sampling different regions of the search space. The following section presents a brief review of some previous studies related to the work presented in this section.

## 2.2 Brief review of previous works

By definition, a strict blackbox search algorithm must work without any prior information about the structure of the objective function. Although the field of global optimization has a rich volume of literature, many studies are severely restricted because of their assumptions about the properties of the objective function (Schoen, 1991), and therefore it can be questioned whether they can really be called BBO algorithms. The objective of this section is to present a brief account of some previously developed algorithms that make little use of domain information about the problem. First, we present a classification of BBO algorithms based on whether the algorithm is deterministic or non-deterministic. Next, we concentrate on the non-deterministic or stochastic methods. Finally, we present a brief description of some previous efforts to relate different BBO algorithms with one another and to understand them on common grounds.

Although there may be several ways to classify optimization algorithms from different points of view (Törn & Žilinskas, 1989), one natural candidate is classification based on the deterministic or non-deterministic nature of the search algorithm. Several earlier efforts (Archetti & Schoen, 1984; Dixon & Szegö, 1978; Gomulka, 1978) suggested classification of global optimization algorithms using this approach. BBO algorithms can be similarly classified as

- Deterministic approaches
- Stochastic approaches
  - blind random search methods
  - adaptive sampling search methods

Each of these approaches will be briefly described in the following.

Deterministic enumeration of members of the search space is one method. Unfortunately, for most of the interesting optimization problems, deterministic enumeration becomes practically impossible because of the growth in the search space.

On the other hand, the stochastic algorithms introduce some random elements into the algorithm and try to solve the problem by relaxing the guarantee of the deterministic enumerative search. This relaxed nature of stochastic search algorithms makes them more suitable for practical applications.

Blind random search (Schoen, 1991; Törn & Žilinskas, 1989) is probably the simplest class of algorithms within the family of stochastic BBO algorithms. The Monte Carlo and multistart algorithms are examples of this kind of algorithm. The Monte Carlo algorithm generates random samples from the search space according to a fixed distribution. Multistart methods make use of local search techniques in addition to the Monte Carlo sample generation process. Although algorithms of this class are simple in nature, they are likely to be suitable for the worst case when different regions of the search space cannot be qualified and when evaluating a particular member of the search space does not provide information about another member.

Adaptive sampling search techniques try to exploit the information gathered from samples taken from the search space. They try to qualify different regions of the search space in terms of the fitness values of their members and use that information to decide which region to explore next. Bayesian algorithms, clustering methods, simulated annealing (SA) and genetic algorithms (GAs) are examples of this class of algorithms. This dissertation mainly considers this class of algorithms.

Bayesian algorithms (Betrò, 1983) try to develop a statistical model of the objective function. These algorithms do not explicitly construct a function; instead, they use a random variable to minimize the expected deviation of the estimate from from the real global optimum. The expected value of the random variable is set to the best estimate of the function and the variance of the random variable capture the uncertainty about this estimate. The problem of Bayesian algorithms are that they are often complicated and involve fairly cumbersome computations, such as computing the inverse of the covariance matrix (Törn & Žilinskas, 1989).

Clustering methods (Rinnooy Kan & Timmer, 1984; Törn & Žilinskas, 1989) use a Monte Carlo sample generation technique. Cluster analysis algorithms are used to identify local minima. This is followed by a local search for each local optimum. Clustering methods have been found useful for many global optimization problems (Hart, 1994; Törn & Žilinskas, 1989). However they are likely to perform poorly when the objective function is multimodal and there are many local optima (Hart, 1994).

Since the early 80s, the simulated annealing (SA) algorithms (Kirpatrick, Gelatt, & Vecchi, 1983) and their variants have been used for solving blackbox problems. The natural motivation behind SA is the statistical behavior of molecules during the crystallization process in annealing. SA considers one sample at a time and this sample represents the state of the algorithm. A neighborhood generator is used to generate new samples. SA makes use of a probabilistic comparison statistic (the Metropolis criterion) for deciding whether the new sample should be accepted as the state of the algorithm. The Metropolis criterion dynamically changes along with a parameter known as temperature. The temperature takes a high value in the beginning and gradually decreases according to a chosen cooling schedule. The acceptance probability is often very high in the beginning, when the temperature is high. The acceptance probability decreases as the temperature reduces. SA has a proof for asymptotic convergence to the optimal solution (Kirpatrick, Gelatt, & Vecchi, 1983) SA has been applied to a wide range of blackbox problems. Many of them reported very promising results. However, in the recent past several negative results have also come out (Dueck & Scheuer, 1988; Ferreira & Žerovnik, 1993).

Genetic algorithms (GAs) (De Jong, 1975; Goldberg, 1989; Holland, 1975), evolutionary programming (Fogel, Owens, & Walsh, 1966), and evolutionary strategies (Rechenberg, 1973) are also getting increasing attentions for dealing with global optimization in blackbox problems. Design of the simple genetic algorithm (GA) is motivated by natural evolution. Unlike the SA, it emphasizes the role of representation and the interaction between the representation and perturbation operators. GAs use the representation to implicitly divide the search space into several non-overlapping classes often called schemata (Holland, 1975). Unlike SAs, GAs work from a population of samples, with each sample often represented as sequences. This population of sequences is used to evaluate different schemata. New samples are generated by crossover and mutation operators. Crossover also implicitly combines the better schemata while generating new samples. GAs have been successfully applied to different classes of problems (Goldberg, 1989). However, the simple GA suffers from several limitations. Although the simple GA realizes the role of representation that induces relations among members of the search space, the simple GA does not really search for appropriate relations. Moreover, the evaluation of schemata is also very noisy in the simple GA. These issues will be revisited and elaborated in Chapter 4.

With all these different BBO algorithms in our arsenal, it is quite natural to ask whether they can be studied on common grounds using common principles. Several previous efforts have been made to address this question. Although Holland's (1975) framework for adaptive search was primarily motivated by evolutionary computation, the underlying concepts of search based on schema processing and decision making are fundamental issues that are equally relevant in the context of any other adaptive BBO algorithms. In fact, Holland's work (1975) is the root of the current thesis. Davis (1987) made an effort to put literature on

SAs and GAs under a common title. Unfortunately, this book did not make any direct effort to link them; rather, it simply discussed them separately. Sirag and Weisser (1987) combined several genetic operators into a unified thermodynamic operator and used it to solve traveling salesperson problems. However, this paper did not study the fundamental similarities and differences between SAs and GAs. Goldberg (1990) addressed this issue. He presented a common ground to understand the effects of the different operators of SAs and GAs. He also proposed the Boltzmann tournament selection operator, which attempts to achieve Boltzmann distribution over the population. Mahfoud and Goldberg (1992) introduced a parallel genetic version of simulated annealing called *parallel recombinative simulated annealing.* This algorithm attempted to harness the strengths of both SAs and GAs. Recently Rudolph (1994) developed a Markov chain formulation of SAs and GAs for analyzing their similarities and differences. Jones and Stuckman (1992) made an interesting effort to relate GAs with Bayesian approaches to global optimization. They noted the similarities and differences between these two approaches and concluded that they share many common grounds. They also developed hybrid algorithms that try to harness the strengths of both approaches. Recently Jones (1995) proposed a framework to study the correspondence between evolutionary algorithms and heuristic state space search of graph theory. In this approach the search domain of the objective function is viewed as a directed, labeled graph. Jones and Forrest (1995) also proposed the fitness-distance-correlation measure for quantifying search difficulty and applied this measure to several classes of objective functions.

Unfortunately, very few of the previous works actually made a quantitative effort to study the computational capabilities and limitations of BBO. Little attention has been paid to the role of relations in BBO, which is essential for transcending the limits of random enumerative search. We still lack any common framework that describes these different algorithms in terms of the basic concepts of theory of computation. The SEARCH framework, makes an attempts to do that. The following section presents a brief description of SEARCH.

# 3 SEARCH: An Informal Picture

SEARCH presents an alternate picture of blackbox optimization in terms of relations and classes that can be constructed among the members of the search space. SEARCH is also a formal framework that helps us quantify different aspects of BBO, 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 rigourous derivations given elsewhere (Kargupta, 1995). Section 3.1 presents the fundamental motivation behind SEARCH. Section 3.2 presents an overview of SEARCH. A more detailed picture is presented in Section 3.3.

## 3.1 Motivation

Some existing BBO algorithms try to find the optimal solution by directly searching the original domain of optimization variables. Samples are often used to estimate the best solution of the search space. In these approaches, a BBO algorithm always searches for a better solution compared to the current best solution. It takes one or more samples and then decides how to choose the next sample. Although the task is certainly nontrivial, the approach of finding the best solution by iteratively updating the best estimate has a fundamental problem. Sampling one particular point from the search domain does not necessarily tell us anything about another point. When a BBO algorithm makes a decision to sample another member from the domain, it is performing *induction*—the process of hypothesizing the premise from the consequences (Michalski, 1983). This is because we are first observing the objective function values for the members of the sample set and then trying to determine whether an unknown point should have a higher or lower objective function value. In other words, it is guessing; it is a proven fact that induction is impossible when no relation exists between the members (Mitchell, 1980; Watanabe, 1969). If no prior relation is assumed between them, there is little reason to choose one member over others, and the blackbox search 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

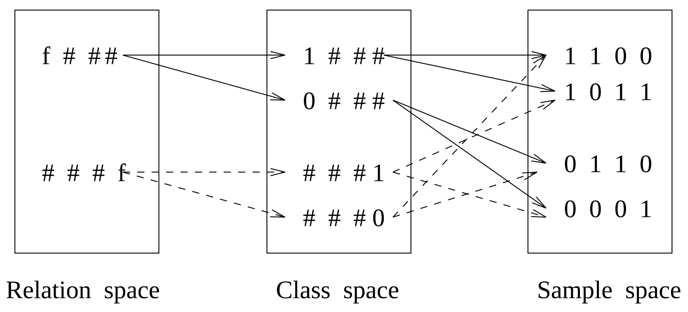


Figure 1: Decomposition of blackbox optimization in SEARCH.

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. SEARCH tries to establish such relations among the members of the search space. Instead of directly searching for the best solution from the beginning, SEARCH tries to find these relations and then use them to locate the classes containing the optimal solution. The following section presents a brief overview of SEARCH.

## 3.2 Overview

The foundation of SEARCH is laid on a decomposition of the blackbox search 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 divide the set of cubes into two subsets—set of white cubes and set of black cubes. Consider a 4-bit binary sequence. There are $2^4$ such binary sequences. This set can be divided into two classes using the equivalence relation[2] $f\#\#\#$, where $f$ denotes position of equivalence; the # character matches with any binary value. This equivalence relation divides up 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 BBO problem, relations among the search space members are often introduced through different means, such as representation, operators, heuristics, and others. The above example of relations in binary sequence 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 define a set of members that can be reached by one or several application 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, Perttunen and Stuckman (1990) proposed a Bayesian optimization algorithm 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 optimization (Ratschek & Voller, 1991), 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 any search problem either implicitly or explicitly. The role of relations in BBO is also very fundamental and important. Although, in a BBO, many relations can be introduced in different ways not all of the relations are approrpaite from the optimization perspective. The objective of sampling based BBO is to detect regions of the domain that are most likely to contain the optimal solutions. In other words, a BBO 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 shall later formally define such relations as those which *properly delineates* 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 BBO all these decision making is done by taking a finite number of samples from the domain. Clearly, all the BBO algorithms often implicitly deal with the three distinct spaces: (1) relation space, (2) class space, and (3) smaple space. SEARCH considers all of them explicitly in

[2] An equivalence relation is a relation that is relexive, symmetric, and transitive.

an effort to understand them rigorously. Figure 1 shows this fundamental decomposition in SEARCH. The major components of SEARCH can be 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 component is discussed in more detail in the following paragraphs. To do so requires some notation that we shall use throughout the remainder of the section. 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 $r_i$. The set of relations $S_r$ actually used by an algorithm to solve the given BBO is a subset of $\Psi_r$. Denote the members of $C_i$ by $C_{1,i}, C_{2,i} \cdots C_{N_i,i}$, where the cardinality of the class $C_i$ is $\|C_i\| = N_i$. Therefore, $C_i$ is a collection of classes.

Once the relation is used to construct $C_i$ the next step is to evaluate the classes in $C_i$. To do that we need samples from the domain of optimization. A perturbation operator $\mathcal{P}$ is defined as an operator that generates new samples. This operator can be either 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 so, we need a way 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 will be called a *class comparison statistic*. This 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 $C_i$. Let us 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 considerations. 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 larger value of $M_i$ may be used. These $M_i$ classes constitute the updated version of the class search space.

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 relation comparison 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 we did 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 logical AND operation. If either of $r_1$ or $r_2$ was earlier found to properly delineate the search space with certain value of $M_i$, then the information about the classes that are found to be bad earlier can be used to eliminate some classes in $r_0$ from further consideration. Blackbox 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 which look better using 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 success in finding proper relations, better classes, and sufficient sampling.

The following section presents a formal description of the different aspects of the SEARCH framework.

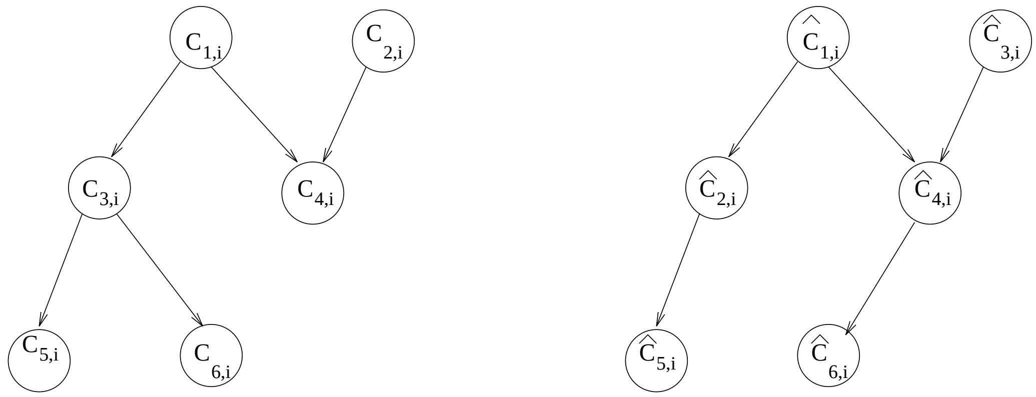


Figure 2: Hasse diagram representation of $C_i$ *(left)* and $\hat{C}_i$ *(right)*.

## 3.3 SEARCH: The detailed picture

The objective of this section is to present a more quantitative picture of SEARCH and formalize the earlier descriptions. The definition of a better relation requires defining what we mean by better classes. Therefore, the decision making in the class space is considered first, in Section 3.3.1. Section 3.3.2 considers the class selection process. This is followed by Section 3.3.3 that discusses the relation search. Finally, Section 3.3.4 presents the resolution process of SEARCH.

### 3.3.1 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 $\Psi_r$, a set of relations, as an abstract entity provided to the search process. However, we continue to give illustrative examples whenever required, using relations defined by sequence representation.

Let $C_i$ be the collection of classes created by some relation $r_i$. Denote the members of $C_i$ by $C_{1,i}, C_{2,i}, \ldots C_{N_i,i}$, where $\|C_i\| = N_i$. Once a relation $r_i$ is used to define $C_i$, the collection of classes, each of its members needs to be evaluated first. Since we are interested in the relative "goodness" of the classes with an ultimate goal to pick up some and reject 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 $\Gamma$ 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 $\leq_{\mathcal{T}}$. In other words, when $C_{j,i}$ and $C_{k,i}$ are compared to each other, then either $C_{j,i} \leq_{\mathcal{T}} C_{k,i}$ or $C_{k,i} \leq_{\mathcal{T}} C_{j,i}$, or they cannot be compared. When $C_i$ is partially ordered on the basis of $\leq_{\mathcal{T}}$, 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 $C_{2,i} \leq_{\mathcal{T}} C_{1,i}$. We can say that $C_{1,i}$ covers $C_{2,i}$ if $C_{1,i}, C_{2,i} \in C_i$, $C_{2,i} \leq_{\mathcal{T}} C_{1,i}$, and no element $C_{3,i} \in C_i$ satisfies $C_{2,i} \leq_{\mathcal{T}} C_{3,i} \leq_{\mathcal{T}} C_{1,i}$. 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, $C_{1,i}, C_{2,i}, \ldots C_{N_i,i}$. Let us 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 ranked class in this ordering is denoted by $C_{[N_i],i}$. he 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.

### 3.3.2 Selection of better classes

Once the classes are partially ordered based on $\leq_{\mathcal{T}}$, the next immediate objective is to select $M_i$ "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 $\text{TOP}(C_{i[\,]}, M_i)$ 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 $\text{TOP}(\hat{C}_{i[\,]}, M_i)$. Unfortunately, this is very unlikely, unless $C_{*,i}$ itself is not within $\text{TOP}(C_{i[\,]}, M_i)$. This sets the stage for introducing the notion of inherently better or worse relations with respect to a given problem, a class comparison statistic, and memory size. This is considered in the following section.

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

A relation is not appropriate with respect to the chosen class comparison statistic and the BBO problem if the class containing the optimal solution is not one among some top-ranked classes, ordered based on this statistic. 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 $DC(r_i, \mathcal{T}, M_i)$. This function returns a one if $C_{*,i} \in \text{TOP}(C_{i[\,]}, M_i)$; otherwise, it returns a zero. This will be denoted by $DC()$ in short (DC stands for Delineation Constraint), unless otherwise required.

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

This *delineation constraint* plays an important role in SEARCH processes. 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 the search space. Therefore, determining whether or not a relation properly delineates 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 $\text{TOP}(C_{i[\,]}, M_i)$. 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 relations. 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 $\leq_{\mathcal{T}_r}$. 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 $r_j \leq_{\mathcal{T}_r} r_i$, then it is not necessarily true that $\hat{r}_j \leq_{\mathcal{T}_r} \hat{r}_i$; we denote a relation $r_i$ when compared based on limited sampling by $\hat{r}_i$. This process of relation selection involves decision making in absence of complete knowledge and it is therefore susceptible to decision errors. The following section describes the resolution process.

### 3.3.4 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

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

where the index $b$ varies over the all $M_j$ 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 the a member of the unordered collection of classes $C_i$. The following sections present a brief description of the analysis of the different decision makings in SEARCH.

# 4 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 in absence of 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 $\text{TOP}(C_{i[\,]}, M_i)$. 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. Let us call this error the *relation selection error*.

2. Even when $C_{*,i}$ is in $\text{TOP}(C_{i[\,]}, M_i)$, 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 $\text{TOP}(\hat{C}_{i[\,]}, M_i)$. 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 BBO 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.

## 4.1 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 whether a relation satisfies this requirement or not a priori, 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 the 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 $\hat{r}_j \leq_{\mathcal{T}_r} \hat{r}_i$ over all possible relations is denoted by, $Pr(\hat{r}_j \leq_{\mathcal{T}_r} \hat{r}_i)_{min}$, the success probability that $\hat{r}_i$ will be one among the top $\|S_r\|$ relations is

$$Pr(CRS \mid r_i) \geq Pr(\hat{r}_j \leq_{\mathcal{T}_r} \hat{r}_i)_{min}^{\|\Psi_r\| - \|S_r\|}, \tag{2}$$

where $CRS$ stands for *correct relation selection*. The following section considers the decision making in class selection process.

## 4.2 Class selection success

Let us now consider the class selection problem. The probability that the best solution is in any of the selected subsets will be denoted by $Pr(CCS|r_i)$. $CCS$ stands for *correct class selection* and conditional to $r_i$, reflects its association with relation $r_i$. Let $Pr(\hat{C}_{j,i} \leq_{\mathcal{T}} \hat{C}_{*,i})$ denote the success probability given that $C_{j,i} \leq_{\mathcal{T}} C_{*,i}$, and let $Pr(\hat{C}_{j,i} \leq_{\mathcal{T}} \hat{C}_{*,i})_{min}$ be the minimum value of $Pr(\hat{C}_{j,i} \leq_{\mathcal{T}} \hat{C}_{*,i})$ 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,

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

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

### 4.3 Overall success

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

$$Pr(CS \mid \forall r_i \in S_r) = \prod_{\forall r_i \in S_r} Pr(CRS \mid r_i) Pr(CCS \mid r_i). \tag{3}$$

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 blackbox search algorithm depends on

1. the success probability in finding relations that properly delineate the search space and
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 statistic.

## 5 Ordinal Class and Relation Selection

Constructing a total order and selection of some $M_i$ top subsets from that order have been studied using both parametric and non-parametric approaches (Gibbons, Sobel, & Olkin, 1977). 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 non-parametric, ordinal approach (David, 1981) 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 a quite general ordinal relation and class comparison statistics.

Section 5.1 considers an ordinal class comparison statistic and the SEARCH framework is specialized for this statistic. Section 5.2 further specializes SEARCH for an ordinal relation comparison statistic. Section 5.3 combines the decision making for both better classes and relations; it also bounds the overall success probability. Finally, Section 5.4 derives the overall sample complexity and discusses its properties.

### 5.1 Ordinal class selection

As we argued in the previous section, BBO can be viewed as a combined process of search for better relations and better classes defined by each of these relations. Let us first consider the class comparison process from an ordinal perspective. In order statistics any two classes will 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, let us assume that the $\alpha$ quantile is unique. We should note that such quantile-based class comparison will always produce a total order on the collection $C_i$.

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 shall denote $n$ samples from the class $C_{j,i}$ by $\hat{C}_{1,j,i}, \hat{C}_{2,j,i}, \ldots \hat{C}_{n,j,i}$; 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:

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

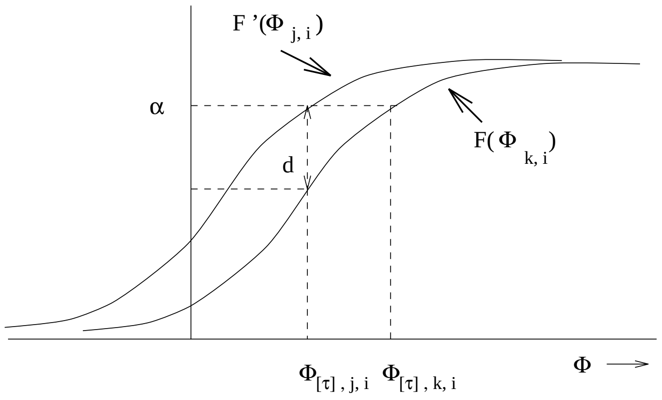


Figure 3: Fitness distribution function of two classes $C_{[j],i}$ and $C_{[k],i}$.

where, $\hat{C}_{[\omega],j,i} \leq_{\Phi} \hat{C}_{[\beta],j,i}$ if $\hat{\Phi}_{[\omega],j,i} \leq \hat{\Phi}_{[\beta],j,i}$. $\hat{\Phi}_{[k],j,i}$ denotes the $k$-th order statistic. 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 cumulative distribution function $F'$ 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 $C_{j,i} \leq_{\alpha} C_{k,i}$, 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. Let us define

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

The variable $d$ defines 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 $C_{j,i}$ (David, 1981), $\Phi_{[\tau],j,i}$,

$$Pr(\hat{\Phi}_{[\tau],j,i} \leq c^{'}) = \sum_{w=\tau}^{n} \binom{n}{w}(F(c^{'}))^{w}(1 - F(c^{'}))^{n-w}. \tag{4}$$

The probability of correct selection among these two classes can be written as (Kargupta, 1995)

$$Pr(\hat{\Phi}_{[\tau],j,i} \leq_{\alpha} \hat{\Phi}_{[\tau],k,i}) \quad \geq \quad 1 - 2^{nH(\alpha)}(\alpha - d)^{\alpha n}. \tag{5}$$

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

$$Pr(CCS \mid r_i) \geq [1 - 2^{nH(\alpha)}(\alpha - d^{''})^{\alpha n}]^{N_i - M_i}. \tag{6}$$

Given relation $r_i$ that properly delineates the search space, Equation 6 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.

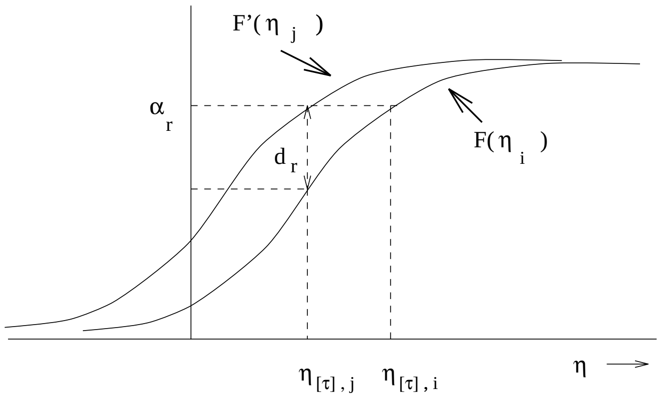


Figure 4: Cumulative distribution function of two relations $r_i$ and $r_j$.

## 5.2 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 delineation constraint. Let us define a measure $\eta : \Psi_r \times 2^{\mathcal{C}} \times 2^{\mathcal{X}} \longrightarrow \Re$. $2^{\mathcal{C}}$ denotes the collection of classes and $2^{\mathcal{X}}$ denotes the sample set. For a given relation $r_i$, the corresponding set of classes $C_i$, and a sample set $\mathcal{S}$, this measure $\eta(r_i, C_i, \mathcal{S})$ returns a real value that corresponds to the chances of $r_i$ to satisfy 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 $r_i, r_j \in \Psi_r$. Let us again adopt an ordinal approach to compare different relations, just as we did for 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 that, here in the relation space the random variable is the measure $\eta$, which is defined over a collection of classes and a sample set for a given relation relation. Since, for a given $r_i$, the computation of $\eta_i$ depends on a tuple from $(2^{\mathcal{C}} \times 2^{\mathcal{X}})$, 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 is computed in exactly the same way that we just did for class comparisons. If $\alpha_r$ be the corresponding percentile,

$$Pr(\hat{r}_{[\tau_r],j} \leq_{\alpha_r} \hat{r}_{[\tau_r],i}) \geq 1 - 2^{n_r H(\alpha_r)}(\alpha_r - d_r^{'})^{\alpha_r n_r} \tag{7}$$

where

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

where $F$ 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 the all possible relations in $\Psi_r$. However, in reality, it may be true that only a 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

$$Pr(CRS \mid r_i \in \Psi_g) \geq [1 - 2^{n_r H(\alpha_r)}(\alpha_r - d_r^{'})^{\alpha_r n_r}]^{\|\Psi_r\| - \|\Psi_g\|}. \tag{8}$$

This is the success probability in choosing one good 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 (Kargupta, 1995):

$$n_r > \frac{\log(1 - q_r^{1/(\|S_r\|(\|\Psi_r\| - \|\Psi_g\|)})}{-d_r^*} \tag{9}$$

Inequality 9 can be further rearranged. Define *delineation-ratio*,

$$\begin{aligned} \Omega &= \frac{\text{Number of relations in } \Psi_r \text{ that properly delineates}}{\text{Total number of relations in } \Psi_r} \\ &= \frac{\|\Psi_g\|}{\|\Psi_r\|} \end{aligned} \tag{10}$$

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 definition 10 and Ineqality 9 we can write

$$n_r > \frac{\log(1 - q_r^{1/(\|S_r\| \, \|\Psi_r\|(1-\Omega))})}{-d_r^*}. \tag{11}$$

This bounds the overall computational complexity in the relation space. Inequality 11 can be further simplified using the approximation $\log(1-x) \approx -x$ for $x << 1$,

$$n_r > \frac{q_r^{1/(\|S_r\| \, \|\Psi_r\|(1-\Omega))}}{d_r^*}. \tag{12}$$

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.

The decision making in the relation and class spaces are combined in the following section.

### 5.3 Overall selection success

Let us now combine the search for better relation and better classes together 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 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 3. 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 $M_{\min}$ and $q_r$ be the minimum value of memory size $M_i$ and bound on success probability in choosing a relation respectively over all the relations in $S_r$. In formal notation,

$$\begin{aligned} N_{\max} &= \max\{N_i | \forall r_i \in S_r\} \\ M_{\min} &= \min\{M_i | \forall r_i \in S_r\} \end{aligned}$$

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 (Kargupta, 1995):

$$\begin{aligned} &[(1 - 2^{nH(\alpha)}(\alpha - d^*)^{\alpha n})^{(N_{\max} - M_{\min})}]^{\|S_r\|} q_r \geq q \\ &n > \frac{\log\left(1 - \left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}}\right)}{\alpha \log(\alpha - d^*)} \end{aligned} \tag{13}$$

The denominator of Inequality 13 can be simplified to,

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

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; $q_r$ gives the given level of success probability in choosing $\|S_r\|$ relations correctly. The cost of increasing the bound $q_r$ can be realized using inequality 11.

## 5.4 Sample complexity

The previous subsection derived closed-form bounds on the overall success probability. This can be directly used to bound the overall **sample complexity**,

$$\mathrm{SC} \leq \frac{N_{\max}\|S_r\| \log(1 - \left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}})}{-d^*}. \tag{15}$$

This inequality gives the overall sample complexity when the probability to find the globally optimal solution is at least $q$. This expression can be further simplified using reasonable approximations to clearly explain its physical significance. Since $\left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}} \leq 1$ and $\log(1-x) \approx -x$ for $x << 1$, we can approximate inequality 15 as follows:

$$\mathrm{SC} \leq \frac{N_{\max}\|S_r\|}{d^*}\left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}}. \tag{16}$$

Inequality 16 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 inequality 11. However, as we increase our success probability in the relation space, the overall success probability in the combined relation and class spaces increases. The sample complexity should therefore decrease as success probability in the relation space increases. Inequality 16 clearly shows that SC decrease with increase in $q_r$. Note that the ratio $\left(\frac{q}{q_r}\right)^{\frac{1}{\|S_r\|(N_{\max} - M_{\min})}}$ 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. The sample complexity decreases as the $d^*$ increases.

This bound on sample complexity establishes an insight introduced earlier in this section. In the beginning of Section 3, we argued that BBO 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 4-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 same as the caridinality of the search space. So, $N_{\max} = \|\mathcal{X}\|$, where $\|\mathcal{X}\|$ denotes the size of the search space $\mathcal{X}$. Substituting these in Inequality 16 we get

$$\mathrm{SC} \leq \frac{\|\mathcal{X}\| q^{\frac{1}{\|\mathcal{X}\| - M_{\min}}}}{d^*}. \tag{17}$$

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 BBO, the importance on relations is fundamental, and it should be emphasized in all possible models of BBO that aspire to guide designing BBO algorithms that perform better than random enumerative search. No BBO algorithm can transcend the limit of random enumerative search without inducing relations among the members.

# 6 Implications of SEARCH

The SEARCH framework reviewed in the previous section shows an alternate way to view BBO. Different facets of this framework have been addressed from an abstract quantitative perspective. However, those formalisms lead toward meaningful, physical aspects of BBO. In this section we present the main physical implications of SEARCH. A framework like SEARCH should contribute to each of the following aspects of problem solving:

1. How do we define an algorithm in this framework? How can we make them efficient?
2. How can we characterize problems that can be solved efficiently?
3. What is the user's role in solving a BBO?

The primary objective of this section is to answer these questions.

We start the discussion by describing what it means to be a BBO algorithm in SEARCH. This is presented in Section 6.1. This is followed by Sections 6.2 and 6.3, in which two possible ways to make a BBO algorithm efficient are pointed out. Section 6.2 describes the benefits of a bottom-up approach of blackbox search, in which low-order relations are considered first. Section 6.3 discusses implicit parallelism—evaluating relations in parallel at no additional sample complexity. Next, in Section 6.4, We introduce the SEARCH perspective of problem difficulty in BBO. Defining difficult problems sets the background for introducing the class of problems that can be solved in polynomial sample complexity. Section 6.5 presents the class of *order*-k *delineable* problems that can be solved in polynomial sample complexity in SEARCH. Section 6.6 discusses one important role of the user in solving a BBO—defining the source of relations. Section 6.7 discusses some issues regarding the representation-operator interaction. Finally,

## 6.1 Blackbox Optimization Algorithm in SEARCH

The SEARCH framework decomposed BBO into different components such as the relation space, class space, and the sample space. Searching in these spaces requires some fundamental tools, such as some comparison statistics and a perturbation operator for generating samples. In this section, we list them together and project a complete picture of what it means to be a BBO algorithm in SEARCH.

1. SEARCH views the solution domain through relations, classes, and samples. An algorithm in SEARCH should be provided with a set of relations $\Psi_r$. Representation in genetic algorithms (GAs), perturbation operators of simulated annealing (SA), and the neighborhood heuristic in k-opt algorithm are some examples of different sources of relations.

2. SEARCH also needs explicit storage for processing relations, classes, and samples. The maximum possible values of $M_r$[3] and $M_i$ determine the size of the memory for storing relations and classes respectively. In genetic algorithm the population serves as the memory for all three of these spaces. In SA the evaluation of different classes defined using the perturbation operators is distributed over time and only one sample is taken at a time. The state of the SA algorithm serves as the memory for the sample space.

3. Two statistic measures for comparing classes and relations are required. A Selection operator is used for comparing classes in the simple GA. The Simple GA does not really search for better relations. On the other hand, in simulated annealing, the Metropolis criterion is used for comparing two states; this can be viewed as a class comparison statistic.

[3] $M_r$, the relation space memory comes into the picture when the memory is not large enough to process all the relations in $\Psi_r$

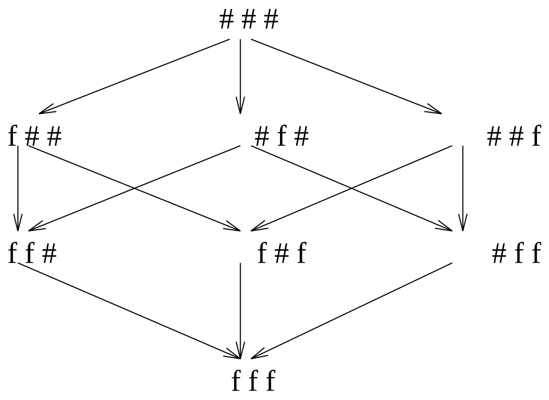


Figure 5: Hasse diagram representation of the set of equivalence relations ordered on the basis of the relation $<_o$.

4. A perturbation operator, $\mathcal{P}$ is required for generating samples. In GA, crossover and mutation generate new samples. SA make use of a neighborhood generator for sample generation.

5. Accepting criterion of success probability, $q$, and $q_r$ are necessary. Almost every practical application of GA and SA either implicitly or explicitly makes use of an acceptance criterion for success. Neither simple GA nor SA actually searches for better relations. Neither of them has any explicit criterion like $q_r$.

6. Required precision in solution quality, $d^*$ is required. Again, in practice, both GA and SA somehow introduce the factor controlling the desired solution quality. Goldberg, Deb, and Clark (1993, Holland (1975) quantified the effect of desired solution quality on the success probability using a parametric approach.

In many algorithms, some of these parameters are not explicitly specified. For example, in random enumerative search, no importance on relations is given. However, even this kind of search can be shown as a special case of SEARCH. As we explained earlier in Section 7.4 during the discussion on sample complexity, the random enumerative search is essentially a search with one and only one relation in consideration that divides the search space into singleton classes. In GAs, the class comparison statistic is not explicitly specified. The selection operator implicitly defines all the comparison statistics together. One of the main objectives of this whole effort is to develop a more systematic approach toward the design and use of BBO algorithms by explicitly identifying different components of a BBO algorithm.

SEARCH provides a common ground for developing new blackbox algorithms in the future. Regardless of the motivation and background, any blackbox optimization algorithm should clearly define each of the above listed components. A BBO algorithm should define how it processes relations, classes, and samples. It should state its relation and class comparison statistics. Apart from defining each of them properly, searching in the relation can take advantage of different properties of the relation space. Moreover, the sample generation operator should comply to the decision making of the relation and class spaces.

The following sections consider two possible ways to exploit the structure in relation space.

## 6.2 Bottom-up Organization of Search

During the development of the SEARCH framework, no prior ordering among the relations in $\Psi_r$ is assumed. We did not care, as long as a relation satisfied the delineation constraint. In this section, we show that imposing a certain kind of ordering among the relations may lead to some computational benefit. First, we define an ordering among the relations based on their *order* and then show that considering relations according to this ordering is advantageous.

The order of an equivalence relation $r_i \in \Psi_r$ can be defined as the logarithm of its index with some base $\alpha$. We shall denote this by $o(r_i)$. The index of $r_i$ is then $N_i = \alpha^{o(r_i)}$. Clearly, the index grows exponentially with $o(r_i)$. The set of all these equivalence relations $\Psi_r$ can be partially ordered on the basis of a $<_o$, defined as follows. If $o(r_i) < o(r_j)$, then we say $r_i <_o r_j$. Figure 5 shows a Hasse diagram representation of such an ordering in a $\ell = 3$ representation scheme, which has a depth of 3. Choosing a low-order relation at the initial stage of search when no regions are pruned out from future consideration is computationally advantageous. The following discussion rationalizes this conclusion.

The fundamental idea is quite simple. Comparing classes can be relaxed as we did in the previous section; two classes may be ordered with high confidence from a sample set. However, completely discarding a class without taking a single sample can hardly be justified. Note that construction of an ordering among $N_i$ classes requires $\binom{N_i}{2}$ comparisons. Since $N_i$ grows exponentially with the order of the relation, considering a relation of *order* in $O(\ell)$ demands exponential complexity computation. Therefore it is essential that the indices of the relations considered during the initial stages of the search when no regions are discarded are bounded by some constant. The argument can be driven home using an example. Consider an aribitrary problem in 3-bit sequence representation The order one relation $f\#\#$ has an index value of 2. On the other hand, the index of an order two relation $ff\#$, is $2^2$. If a search algorithm starts its search with relation $ff\#$ then the number of comparisons needed to construct an ordering is going to be higher.

Therefore, the order of the relation used at the initial stage of the search must be bounded by a constant, $k_c$; otherwise the complexity of the ordering process will itself be exponential in $\ell$. Let us denote the set of relations $S_r$, when ordered on the basis of the sequence, they are considered by the algorithm by $\gamma(S_r)$; also define $\gamma(\Psi_r)_{<_o}$ to be the set $\Psi_r$ partially ordered based on the order of a relation.

One possible way to keep the complexity under control is to make $\gamma(S_r)$ and $\gamma(\Psi_r)_{<_o}$ order-preserving. In other words, if $r_1, r_2 \in S_r \subseteq \Psi_r$, then if $r_2$ comes after $r_1$ in $\gamma(S_r)$, then $r_2 <_o r_1$ in $\gamma(\Psi_r)_{<_o}$. This gradual transition from low order relations to higher-order ones is what we call bottom-up organization of the search. This points out a computational justification of a widely reported, interesting observation about both natural and artificial complex systems. Complex evolving systems are characterized by the gradual growth of patterns from smaller to larger. A small pattern is the one in which the number of features in common to the members of the observed set is less. This bottom-up approach, often observed in nature, seems to have a fundamental computational foundation.

The following section considers another possibility of exploiting the structure of the set of relations when ordered based on the *order* of the relations.

## 6.3 Implicit Parallelism: Parallel Evaluation of Equivalence Relations

The discussion on bottom-up organization of search led to the solution of considering the relations with low index values first. The structure of $\Psi_r$, when partially ordered based on order of the relations, can be further exploited to make the relation evaluation process more efficient. In this section we show that when the poset $\gamma(\Psi_r)_{<_o}$ is not linearly ordered, relations can be evaluated in parallel at no additional sample evaluation. This observation appears to be the underlying principle in the so called *implicit parallelism* (Holland, 1975).

The perspective of the blackbox search as an exploration through $\gamma(\Psi_r)_{<_o}$ opens up an interesting possibility. Parallel exploration along different branches in $\gamma(\Psi_r)_{<_o}$ can be done at no additional cost compared to that along a single branch. *Such parallel evaluation is possible as long as* $\gamma(\Psi_r)_{<_o}$ *is not a totally ordered set.* When $\gamma(\Psi_r)_{<_o}$ is partially ordered, there will be relations of the same order. Therefore, all these relations of the same order can be evaluated using the same set of samples.

For example, the evaluation of $\#\#\#f, \#\#ff$ can be performed at no additional computational cost in terms of function evaluations when the relations $f\#\#\#, ff\#\#$ are already evaluated. Little attention will make it obvious. Both $f\#\#\#$ and $\#\#\#f$ divide the complete search space into two different ways. Similarly, the relation $ff\#\#$ divides the same search space in a different way than the one by $\#\#ff$ does. Clearly, the same set of samples used to evaluate classes $1\#\#\#$ and $0\#\#\#$ can be used for evaluating classes $\#\#\#1$ and $\#\#\#0$. No additional function evaluation is needed for a constant confidence requirement; the samples are just needed to be differently partitioned. In general, the sample set needed to evaluate a particular relation $r_i$ of order $o_i$ can be used for all other relations of the same order. This computational leverage can make an algorithm very efficient in solving the class of order $k$ bounded, delineable problems. As stated earlier, these problems can be solved by evaluating a subset of all order $k$ relations, whose intersection set is a singleton set. Since the globally optimal solution can be found by simply taking an intersection among the top ranked classes of this subset of all order $k$ relations, the overall computational cost remains polynomial in the problem dimension and the success probabilities.

At this point one must take a moment to put this argument into proper perspective. Our definition of computational cost has been solely focused on the number of function evaluations, i.e., the number of distinct samples taken from the search space. According to this definition, parallel evaluations of several equivalence relations do not incur any additional cost. However, consideration of every different relation required partitioning the same set of samples in a different way, followed by the computation of the class comparison statistic. Although the sample complexity remains the same, the overall time complexity increases.

The previous three sections described different components of BBO algorithm and pointed two possible ways to exploit the structure in the set of relations provided to the algorithm. These discussions naturally lead the reader to wonder about the computational limits of BBO algorithms from the SEARCH perspective. The following section addresses this important aspect of BBO—problem difficulty.

## 6.4 Problem Difficulty in SEARCH

SEARCH presents an alternate perspective of problem difficulty in BBO. In this section we identify the main dimensions of problem difficulty in SEARCH and precisely define a characterization of difficult problems in SEARCH.

The expression for the sample complexity developed in the previous section immediately leads to identifying different facets of problem difficulty in SEARCH. As we saw from Inequality 15 the sample complexity grows linearly with the size of the set of relations considered to solve the problem, $S_r$. Often this size depends on the "size" of the problem; the word "size" defines a parameter $\ell$ that bounds the search domain. In a sequence representation with constant alphabet size, the length of the sequences needed to represent the search space may be an example of such a size parameter. This finally sets the stage for introducing problem difficulty in SEARCH.

**Definition 2 (Problem difficulty in SEARCH)** *Given an optimization function $\Phi : X \rightarrow \Re$ and a set of relations $\Psi_r$, we call a problem difficult for an algorithm if the total number of samples needed to find the globally optimal solution grows exponentially with $\ell$, $q$, $q_r$, $1/d^*$, and $1/d^*_r$.*

The size of the problem is represented by $\ell$; $q$ denotes the bound in the overall decision success probability in choosing the right classes; $1/d^*$ defines the quality of the desired solution. Both $q$ and $1/d^*$ together can be viewed as representing the overall accuracy and the quality of the solution found; $q_r$ is the bound in success probability in choosing the right relations, and $1/d^*_r$ represents the desired quality of the relations.

The above definition of problem difficulty in SEARCH can be physically interpreted into the following items:

1. growth of the search space along problem dimension
2. inadequate source of relations and decision making in relation space
3. inaccurate decision making in choosing classes
4. quality of the desired solution and relations

Each of these is briefly discussed in the following.

The search space increases as the problem dimension increases. For many interesting classes of problem the growth rate is exponential. This creates difficulty in solving the problem.

Solving a BBO in SEARCH requires a set of relations that properly delineates the search space. If the number of relations in $\Psi_r$ that properly delineate the search space is small, then success probability in solving the BBO is very small. The previous section defined a simple but meaningful measure of the overall appropriateness of $\Psi_r$, the delineation-ratio. This is the ratio of the number of relations in $\Psi_r$ that properly delineates and the total number of relations in $\Psi_r$. 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. This point will be illustrated in a later section of this section using some example problems.

The overall decision error in choosing the classes that contain the globally optimal solution will be large unless the sampling is sufficiently rigorous. This is a source of decision error and can cause difficulty.

The quality demanded for the solution can also make a problem difficult. If the desired quality is very high, search in BBO may be difficult. Similarly, if the required measure of appropriateness of relations needs to be very high, searching for relations may be difficult.

In a way this basically defines what it means to be efficient search in SEARCH. Therefore, now it is quite natural to ask whether we can identify some classes of BBO problems that can be solved in polynomial sample complexity. The answer is yes. However, the relation between the set of relations $S_r$ and the problem size $\ell$ cannot be quantified until we specifically consider relations in physical terms. In the following section we consider the set of relations defined by sequence representation and introduce the class of *order-k delineable* problems that can be solved efficiently in SEARCH.

In the following section we introduce the class of *order-k delineable* problems in sequence representation.

## 6.5 Order-k Delineable Problems in Sequence Representation

The foregoing analysis treated the source of relations in a general way as an abstract entity. However, defining a specific class of problems and studying its sample complexity in SEARCH require relating $S_r$ with the problem dimensions. In this section we consider the set of relations induced by a sequence representation for defining the class of order-k delineable problems that can be solved in polynomial sample complexity.

Although so far we have frequently used sequence representation for illustrating the concepts with examples, in this section we consider specific properties of the set of relations induced by sequence representation. Therefore, it may be appropriate to review the properties of this representation first.

A sequence representation can be defined as

$$I : X \rightarrow \Lambda^{\ell}, \tag{18}$$

where $\Lambda$ is the alphabet set. A particular position in a sequence may sometimes be called a *locus* (*loci* in plural). The particular letter of the alphabet in a locus is called the *value* of the locus. We will also assume a convention of naming in which the leftmost position is called the first locus, increasing toward the right. This sequence representation induces a set of equivalence relations,

$$\Psi_r = \{f, \#\}^{\ell} \tag{19}$$

where $f$ indicates values that must match for equivalence and $\#$ is a wild character that matches any value at a locus. The cardinality of the set of all such equivalence relations $\|\Psi_r\| = 2^{\ell}$.

In sequence representation the order of an equivalence relation $r_i \in \Psi_r$ is defined in the same way as we did before; we just choose $\alpha = \Lambda$, as the base of the logarithm. The index of $r_i$ is then, $N_i = \Lambda^{o(r_i)}$.

As we see from Figure 5, $\Psi_r$ can be structured as a lattice based on the relation $<_o$. For an $\ell$ loci representation, the depth of the diagram is always $\ell$. The lowest level at order $\ell$, the relation divides the search space into singleton sets. Each of these singleton sets can be reached in at most $\ell$ steps. However, the problem is that when $o(r_i) = O(\ell)$, $N_i = O(2^{\ell})$. Here $O$ is used to denote the order of complexity. Clearly, if we want to solve any arbitrary problem using a $\ell$-loci sequence representation, in the worst case, the complexity is exponential.

It will be interesting to know what classes of problems can be solved in polynomial sample complexity. In the following we define a class of order $k$ (a constant) delineable problems which can be solved in sample complexity polynomial in $q$, $q_r$, $1/d^*$, $1/d_r^*$, and the problem size $\ell$.

**Definition 3 (Class of order $k$ delineable problems)** *: Let us define a subset of $\Psi_r$ containing every order-*k *relation as follows: $\Psi_{\{o(r)\leq k\}} = \{r_i : o(r_i) \leq k \ \ \& \ \ r_i \in \Psi_r\}$. For a given class comparison statistic $\mathcal{T}_i$, a problem is order-*k *delineable if there exists a subset $\Psi' \subseteq \Psi_{\{o(r)\leq k\}}$ 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,*

$$\mathcal{G} = \bigcup_{a_1, a_2, \cdots a_k} \bigcap C_{[a_1],i} C_{[a_2],i} \cdots C_{[a_k],i},$$

*is bounded by a polynomial of $\ell$, $\rho(\ell)$. Indices $a_1, a_2, \ldots a_k$ can take any value in between 1 and $M_i$.*

These are the problems that can be solved by considering at most order-$k$ equivalence relations and enumerating the intersection set $\mathcal{G}$. If a problem is order-$k$ delineable in the chosen representation, then once the good classes are detected within each of the relations in $\Psi^{'}$, a simple intersection should give us a set of ground classes of size $\mathcal{G}$. Since by definition the cardinality of this set is bounded by a polynomial in $\ell$, this set can be explicitly enumerated to find the best solution among them. Since all the members in $\Psi^{'}$ properly delineate the search space, the best solution of $\mathcal{G}$ is guaranteed to be the optimal solution. For this class of problems, the set of relations needed to solve is $S_r \subseteq \Psi^{'}$. Let us now find the bound on $\|S_r\|$. The best case scenario is when $S_r$ contains $\ell/k$ non-overlapping relations. Since the problem is order-$k$ delineable, there is at least one relation in $S_r$ that has an order equal to $k$. In the worst case, there are only one order $k$ relation and $\ell - k$ order one relation in $S_r$. Therefore, $\ell/k \leq \|S_r\| \leq \ell - k + 1$. The maximum possible index value of any relation in $S_r$ is $\|\Lambda\|^k$. Although $S_r$ may need at most $\ell - k + 1$ number of relations, we do not need separate sample sets for each of them. The structure of the relation space can be exploited to gain computational benefits. The order-$k$ relation need a sample set for explicit evaluation. The same sample set can be used to evaluate other order one relations. This is basically what we discussed earlier as benefits of implicit parallelism. While solving this class of problem, the set $S_r$ is not explicitly given. The search for $S_r$ from the set $\Psi_{\{o(r)\leq k\}}$ requires evaluation of all relations of order less than $k$. Again the benefits of implicit parallelism can be exploited for this purpose. Therefore, the $k$ relations along any one particular path from the top node of figure 5 down to order-$k$ node need explicit evaluations. Relations along the other paths from the top can be evaluated using the same sample set.

To achieve an overall success probability of $q$, the computational complexity,

$$SC \leq \|\Lambda\|^k k \frac{\log(1 - \left(\frac{q}{q_r}\right)^{\frac{1}{(\ell-k+1)(\|\Lambda\|^k - M_{\min})}})}{-d^*} + \rho(\ell). \tag{20}$$

When $q/q_r << 1$, this can be approximated as before,

$$SC \leq \|\Lambda\|^k k \frac{\left(\frac{q}{q_r}\right)^{\frac{1}{(\ell-k+1)(\|\Lambda\|^k - M_{\min})}}}{d^*} + \rho(\ell)$$

This says that for fixed k, the class of order-$k$ delineable problems can be solved with complexity growing polynomially in $q$, $1/d^*$, and $\ell$.

Since by definition $S_r \subseteq \Psi_{\{o(r)\leq k\}}$, the search for appropriate relations is also restricted to $\Psi_{\{o(r)\leq k\}}$. The cardinality of $\Psi_{\{o(r)\leq k\}}$ is

$$\begin{aligned} \|\Psi_{\{o(r)\leq k\}}\| &= \sum_{j=1}^{j=k} \binom{\ell}{j} \\ &< k\binom{\ell}{k} \\ &< k\left(\frac{e\ell}{k}\right)^k \\ &< \frac{(e\ell)^k}{k^{k-1}}. \end{aligned} \tag{21}$$

Inequalities 11 and 21 can be used to find the computational complexity in the relation space for this class of problems,

$$n_r > \frac{\log(1 - q_r^{k^{k-1}/((\ell-k+1)(e\ell)^k(1-\Omega))})}{-d_r^*}. \tag{22}$$

This clearly shows that the computational complexity in the relation space is polynomial in $q_r$, $1/d_r$. We shall also clearly demonstrate that it is polynomial in $\ell$ too. Note that $\frac{-x}{1-x} \leq \log(1-x)$. Therefore,

$$\log(1 - q_r^{k^{k-1}/((\ell-k+1)(e\ell)^k(1-\Omega))}) \geq \frac{-q_r^{k^{k-1}/((\ell-k+1)(e\ell)^k(1-\Omega))}}{1 - q_r^{k^{k-1}/((\ell-k+1)(e\ell)^k(1-\Omega))}}.$$

For constant $k$ and $\Omega$, this can be written as

$$\log(1 - q_r^{k^{k-1}/((\ell-k+1)(e\ell)^k(1-\Omega))}) \geq \frac{-q_r^{b/\ell^v}}{1 - q_r^{b/\ell^v}}$$

where $b$ and $v$ are appropriate constants. Consider the ratio of this bound of the numerator of inequality 22 and an exponential growth function $a^\ell$, where $a$ is a constant, and take the limit as $\ell \to \infty$.

$$\lim_{\ell\to\infty} \frac{-q_r^{b/\ell^v}}{a^\ell} \frac{1}{1 - q_r^{b/\ell^v}} = 0$$

This proves that the the expression in Inequality 22 is also polynomial in $\ell$.

The following section considers the user's role in solving a BBO. We consider two primary roles of the user regarding the source of relations and the interactions among the relations and the search operators.

## 6.6 Defining The Relation Space

In the SEARCH framework an algorithm searches for appropriate relations. However, this requires first defining the relation space, in other words, the set of relations $\Psi_r$. Defining this source of relations is one of the primary responsibilities of the user of a blackbox optimization algorithm. Doing so requires some understanding about what it means to be an appropriate source of relations. The objective of this section is to discuss this, using the concepts developed earlier in this thesis.

In section 6.6.1 we note at least three possible ways to define relations over the search domain. Section 6.6.2 considers the issue regarding proper delineation. This is followed by Section 6.6.3 which discusses the utility of the structure of the set of relations.

### 6.6.1 Defining relations

As we noted repeatedly in the previous section, relations can be defined in many different ways. Some of the possible ways could be

1. representation,
2. perturbation operators, and
3. search heuristics.

I discuss each of them briefly in the following part of this section.

Representation can be directly used to define a similarity measure, and that can be exploited to induce classes over the search domain. We already gave examples of such similarity measures for the sequence representation. Let us consider another example using a different kind of representation—disjunctive normal form (DNF). This kind of representation is sometimes used in machine learning. For the sake of simplicity, consider binary variables $x_1, x_2, \ldots x_n$. A k-DNF representation is defined as follows:

$$(x_1 \wedge x_2 \cdots \wedge x_k) \vee (x_{k+1} \wedge x_{k+2} \cdots \wedge x_{k+k}) \vee \cdots \vee (x_{n-k} \wedge x_{n-k+1} \cdots \wedge x_n)$$

Relations and classes can be defined using this representation in a way exactly similar to what we did in case of sequence representation. Replace some variables with the fixed position corresponding to which equivalence is to be defined and substitute the wild character for rest of the variables. For example, in 3-DNF representation, $(f \wedge f \wedge f) \vee (\# \wedge \# \wedge \#) \vee \cdots \vee (\# \wedge \# \wedge \#)$ defines a relation. The classes within this relation can also be defined in a similar way, described in the context of sequence representation.

Perturbation operators can also be used to define neighborhood and classes. In this approach, a state, $x$ and an operator define a set of states that can be reached from state $x$ by applying the operator to it. This defines a class of states. Therefore, an operator can be used to divide the search domain into a set of overlapping classes. Many search algorithms use this approach. Graph search algorithms like depth-first and breadth-first search (Rich & Knight, 1991) are some examples that can be viewed from this angle. Simulated annealing can also be viewed from this perspective. Even GAs can also be viewed from this angle (Jones,

Table 1: A trap ($f_d$) and one-max ($f_e$) function in 3-bit representation.

| x | $f_d$ | $f_e$ |
|---|---|---|
| 111 | 3 | 3 |
| 110 | 0 | 2 |
| 101 | 0 | 2 |
| 011 | 0 | 2 |
| 100 | 1 | 1 |
| 001 | 1 | 1 |
| 010 | 1 | 1 |
| 000 | 2 | 0 |

1995). Unfortunately, most of these algorithms consider very restricted sets of of fixed operators, in practice it is often like—one algorithm, one operator. This may severaly constrain the relation space of the algorithm.

Search heuristics are another possible way to go. The basic idea is to use some a priori chosen rules for assigning a priority in the order different regions of the search space is explored. The use of a heuristic rule can be illustrated using a simple example. Let $h(x)$ be a heuristic rule that computes the set of states that satisfy the heuristic criterion, given that the current state is $x$. A heuristic-based algorithm computes $h(x)$ and decides which state to visit next, based on heuristic decision. Clearly, such heuristics divide the search space into a set of classes based on heuristic preferences. This approach is quite popular in the Artificial Intelligence community, where sometimes it is taken to an extreme end. The need for finding appropriate relations is often replaced by assertions of relations using domain knowledge. Such extensive domain knowledge is unlikely to be available in blackbox optimization problems. The field of combinatorial optimization also makes use of heuristics. For example many heuristics have been suggested for solving the traveling salesperson problem; the k-opt algorithm is one such example. $A^*$ (Hart, Nilson, & Raphael, 1968) algorithm is another example from the body of graph search algorithms. Just like the previous approach, heuristic based relations must satisfy the delineation requirement. Therefore, use of a set of heuristics is more appropriate and search for appropriate heuristics is essential.

This section described three possible ways to define relations. The following section considers what makes a set of relations appropriate.

### 6.6.2 Proper delineation

Proper *delineation* of the search space is the most important requirement of a relation. Regardless of how the relations are defined, they need to satisfy this fundamental requirement. The greater the number of relations in $\Psi_r$ that satisfy this requirement, the better it is. We have explained these concepts in the previous section. In this section, we restrict myself only to illustrate the use of the delineation-ratio for quantifying the appropriatness of a source of relations using a simple example.

Searching for relations that properly delineate the search space may not be that useful, in the case that most of the relations in $\Psi_r$ defined by the chosen representation do not satisfy this requirement. Earlier in this section, we defined the delineation-ratio as the ratio of the number of relations in $\Psi_r$ that properly delineates to the total number of relaions. For a given class comparison statistic, memory size, and a BBO, the delineation-ratio reflects how easy the search for better relation will be. Let us illustrate this observation by using the following example. Consider the two functions defined in Table 1. Function $f_d$ is a trap function (Ackley, 1987) in a deceptive representation (Goldberg, 1987), and $f_e$ is an one-max function. The one-max function is known to be very easy to solve; on the other hand, the trap function offers a great degree of difficulty to many algorithms beacuse of its isolation of the optimal solution in the hamming space. Let us again consider class average as the comparison statistic. The delineation-ratio of $f_e$ and $f_d$ are 1 and 1/8, respectively. Although this is for a particular class comparison statistic, the qualitative distinction is likely to be invariant for other reasonable statistics.

If the delineation-ratio is very small, then the representation is inappropriate, and we may be better off choosing a new representation. One possible way is to adaptively construct a new representation that redefines

$\Psi_r$.

The inequality bounding the sample complexity in SEARCH clearly tells us that the index of a relation contributes to the sample complexity. The higher the value of the index, the larger is the required number of function evaluations. Therefore, it is computationally advantageous if the relations with low index, or low order, satisfy the delineation requirement.

The following section reminds us that implicit parallelism should be given consideration while choosing the set of relations.

### 6.6.3 Ordering the set of relations

Earlier in this section we noted one interesting possibility—implicit parallelism—that exploits the stucture of $\Psi_r$ when organized based on the *order* (log of the index) of the relations. This opens up the possibility of evaluating relations in parallel at no additional sample evaluation cost. This is another factor that should influence the choice of the set of relations. Computational benefits of implicit parallelism increase as the lattice formed by ordering $\Psi_r$ based on the *order* becomes more bushy. This is a property of the chosen set of relations and it therefore deserves due consideration.

The following section considers the role of operator-representation interaction in SEARCH.

## 6.7 Search Operators And Representation

Although in some BBO algorithms, search operators may be used to define the set of relations, generating new samples from the search space for evaluating classes is one of their main purposes. Since SEARCH treats relation and sample space as two seperate entities, in this section We shall focus only on the sample generation aspect.

In SEARCH samples are always generated for some particular set of classes. A random perturbation operator does not satisfy this requirement. This is discussed in Section 6.7.1. Once the ordering among the classes is constructed, and once seemingly bad classes are rejected with acceptable level of confidence, there is no reason to generate samples for them again. Therefore, the perturbation operators need to follow the decisions made earlier. This is discussed in Section 6.7.2.

### 6.7.1 Sample generation for equivalence classes

As we just mentioned, the primary objective of a search operator is to generate new samples. Why does a search algorithm need new samples? The perspective of the search, as an ordering process among equivalence classes, says that we need new samples in order to compute the class comparison statistic. Therefore, an algorithm does not just need *any* sample, that a sample from some particular class. To accomplish this, the perturbation operators should be told about the class for which the sample is needed. Clearly, a random perturbation operator falls short of requirements.

Consider an example. To evaluate the class 11##, only those samples that have two consecutive 1s in first two bit positions are needed. Similarly for evaluating 1##1, we need samples with 1s in the first and last bit positions. Clearly the search operators need to know what class is currently under evaluation. It should accordingly change its region of perturbation. In other words, the search operators need to be adaptive to the direction of the search process. A little more thought should make it clear that what we are really talking about is adaptive operator-representation interaction. Sample generation for specific classes can also be accomplished with a non-adaptive search operator when representation itself is adaptive. A simple example may clear up the rationale behind this argument. Consider a 4-bit binary representation and a perturbation operator which generates new samples by randomly changing the third and fourth bit values. Such an operator could never generate samples from class ##11. Now consider a 4 position representation in which every position has a data structure, which in turn comprises a tuple $(locus, value)$. The data *locus* tells the position where *value* should be placed in a sequence. For example, $[(3,0)(0,1)(2,0)(1,1)]$ can be interpreted to the binary string 1100. Now let us reconsider the same search operator that generates samples by perturbing only the third and fourth positions. In the new representation scheme, the sample generation of class ##11 is quite straightforward, since *locus* is explicitly tagged with a value. The class ##11 can be represented as $[(3,1)(2,1)\#\#]$ or $[(2,1)(3,1)\#\#]$. This simple example illustrates that non-adaptive search operators can also serve the purpose, provided that the representation itself is adaptive. This section noted

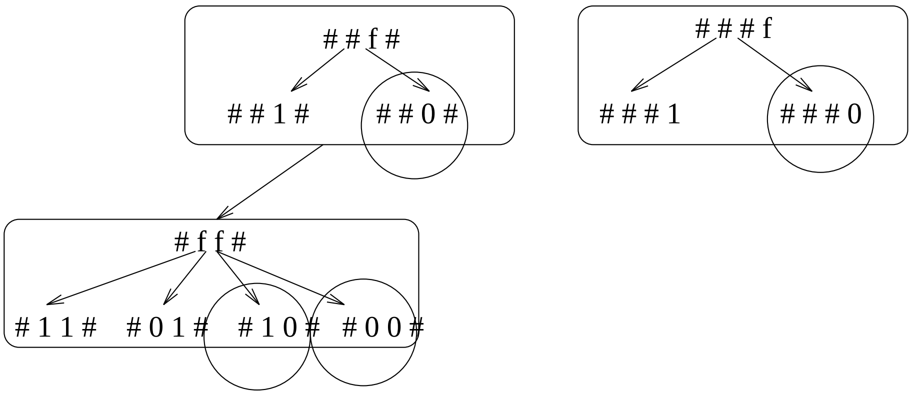


Figure 6: Effect of low-order equivalence class pruning on future exploration. The classes marked by circles need not be considered.

that the representation-operator interaction needs to be adaptive to ensure proper sample generation. The following section points out that the perturbation operators need to be controlled to ensure that they do not generate samples for the classes that are already discarded.

### 6.7.2 Controlling the perturbation

Better control of operator perturbation is also needed for another reason. Consider the subgraph shown in Figure 6. Let us say that the class ##0# is evaluated to be worse than other order one classes of relation ##$f$# and that it is eliminated from further consideration; also assume that the class ###0 is rejected. This essentially means that all higher-order equivalence classes such as #10#, and #00#, which are subsumed by the class ##0#, can also be pruned out. Moreover, since ###0 is already pruned out, the samples generated for evaluating classes #11# and #01# should be restricted from being a member of ###0. The search operators need to keep track of the equivalence classes that are eliminated and accordingly adapt themselves to generate appropriate samples. Again, a simple randomized perturbation operator does not satisfy this requirement. Radcliffe (1993) discussed this desirable feature of the perturbation operator in details.

The following section takes the discussion in a different direction. We present an alternate perspective of evolutionary computation using the lessons from SEARCH.

# 7 SEARCH And Natural Evolution

The primary goal of the SEARCH framework has so far been to capture the essential computational aspects of BBO needed for transcending beyond the random enumerative search. On the other hand, 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 a relatively shorter period of time. If the premise of evolutionary theory is true then the performance of natural search for fitter organisms appears very impressive. Although, the current evolutionary geneticists debate whether or not there was enough time for evolution to succeed by mere chance (Wald, 1954; Shapiro, 1986; Kauffman, 1993) in our opinion evolutionary search is not very interesting computationaly if it solely relies on the power of random enumeration. In this section, we shall argue against this view and sketch an alternate perspective of natural evolution that identifies gene expression as a mechanism for relation construction. This section hypothesizes that the natural evolution might have transcended the limits of random enumerative search by first detecting the appropriate relations. The objective of this section is not to present a comprehensive discusiion on this issue. We rather restrict ourself to present only a possible correspondence between evolutionary mechanisms and different components of SEARCH.

Section 7.1 briefly discusses the flow of information in natural evolution. Section 7.2 points out the main problem of the existing models of evolutionary computation—lack of emphasis on gene expression. Finally, Section 7.3 draws the correspondence between SEARCH and evolution and presents an alternate perspective.

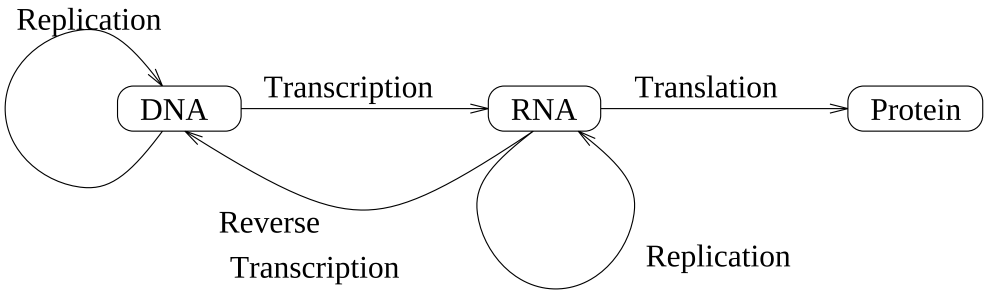


Figure 7: Intra-cellular flow of genetic information.

## 7.1 Information flow in evolution

Information flow in evolution is primarily divided into two kinds:

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

Each of these will be discussed in the following two paragraphs.

The extra-cellular 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. For more detailed description, the reader should refer to Alberts, Bray, Lewis, Raff, Roberts, and Watson (1994, Stryer (1988). *Eukaryotes* (most of the developed organisms) have two chromosomes in their cell nucleus, and thus called *diploid* organisms. On the other hand, in *prokaryotes*, such as single-celled bacteria, only one chromosome is present. These 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 generation of new DNA sequences. DNA is then transmitted from the parents to the offspring. The DNA is responsible for defining the phenotype of organism and thereby controls the suitability of the organism in the environment. This suitability determines the selective pressure 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. The following paragraph briefly describes the main steps of gene expression, that define the intra-cellular flow of evolutionary information.

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

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

Figure 7 shows the different steps of gene expression. Each of them is briefly described in the following.

Transcription synthesizes messenger RNA (mRNA) from part of the DNA. RNA (ribonucleic acid) consists of four types of bases joined to a ribose-sugar-phosphodiester backbone. The four bases are *adenine* (A), *uracil* (U), *guanine* (G), and *cytosine* (C). Transcription is basically constructing a sequence of bases from another sequence of bases—the DNA. Transcription is initiated by some particular sequences of bases in DNA. They are known as *promoter regions*. 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, which enhances transcription of a gene, wherever it binds.
- gene repressor protein, which 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.

Messenger RNA acts as the template for protein synthesis. Proteins are sequence of *amino acids*, joined by peptide bonds. Messenger RNA is transported to the cell cytoplasm for producing protein in the ribosome. There exists a unique set of rules that define the correspondence between nucleotide triplets (known as codons) and the amino acids in proteins. This is known as the *genetic code*. Each codon, comprised of three adjacent nucleotides in a DNA chain, produces a unique amino acid.

Most of the existing models of evolutionary computation do not provide any understanding about the computational role of the intracellular flow of genetic information. The following section points this out.

## 7.2 Problems of the existing views of evolution

Unfortunately, many of the existing computational models of evolution address only the extracellular flow of genetic information. Simple genetic algorithms (De Jong, 1975; Goldberg, 1989; Holland, 1975), evolutionary strategie (Rechenberg, 1973), and evolutionary algorithms (Fogel, Owens, & Walsh, 1966) 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 realized its importance. The importance of the computational role of gene expression was first realized by Holland. He described (Holland, 1975) 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 (Jacob & Monod, 1961) in evolution and pointed out the possible computational role of gene signaling in evolution (Holland, 1975).

Several other efforts have been made to model some aspects of gene expression. Diploidy and dominance have also been used elsewhere (Bagley, 1967; Brindle, 1981; Hollstien, 1971; Rosenberg, 1967; Smith, 1988). Most of them 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 Goldberg, Korb, and Deb (1989). Dasgupta and McGregor (1992) proposed the so-called structured genetic algorithm, which uses a structured hierarchical representation in which genes are collectively switched on and off. This implementation also gathered its primary motivation from gene expression.

Kauffman (1993) offered an interesting perspective of the natural evolution that realizes the importance for gene expression. However, Kauffman's work does not explain the computational aspects and does not relate the issue to the complexity of search process.

Most of the above efforts really addressed the major steps in gene expression primarily identified by the biologists during the last 30 years, and instead quantify their purpose in terms the basic computational principles. Many questions involving the computational benefit from the collective switching of genes or the regulatory mechanisms controlling the transcription of DNA to RNA remain unanswered. In the following section I present a brief discussion to fill in this lacuna using the lessons from the SEARCH framework. The central argument that evolves out of the following discussion is that the intracellular expression of genetic information plays an important role in evolutionary computation that can be quantified from the SEARCH perspective.

Table 2: Counterparts of different components of SEARCH in natural evolution.

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

### 7.3 Evolutionary computation: The SEARCH perspective

Despite the fact that nowhere during the development of SEARCH did we consider any idea from biology, it will be interesting to see how it views the evolutionary search in nature. The following part of this section briefly discusses a possible correspondence between SEARCH and natural evolution.

- **Sample space**: DNA constitute 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 correspond to 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.

- **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 regulatory mechanism, regulatory proteins, promoter 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. Apart from more sophisticated transcriptional control system, the diploid chromosome can be viewed as an additional process of new relation construction.

These possible relationships between the different spaces of SEARCH and natural evolution are summarized in Table 2.

- **Relation and class comparison statistics**: Classes are defined by amino acid sequences in protein. Proteins are directly responsible for almost every functional and organizational behavior of an organism. Proteins are sometimes called the phenotype of an organism because of this reason. The efficacy of a protein is evaluated in terms of the organism's performance, and natural selection assigns a certain selective measure to this process. This can be viewed as a process of evaluating classes. On the relation front, there is already evidence that the settings for the intracellular expression of genetic information evolves during the course of evolution (Alberts, Bray, Lewis, Raff, Roberts, & Watson, 1994). Therefore, there must be some selective pressure toward the appropriate regulatory setting and that can be held responsible for the decision making in the relation space.

- **Perturbation operators**: Crossover, mutation, and gene deletions appear to be the perturbation operators in natural evolution. They change the participating DNA sequence(s) and thereby generate new samples.
- **Resolution**: Crossover and recombination swap different portions of two DNAs. Although the exact mechanism of crossover is quite involved, the main outcome of the process is exchange of DNA subsequences between the parent chromosomes. The offspring DNAs are therefore comprised of the features of classes containing the parent DNAs. Therefore, the offspring are samples from the intersection set of the parent classes.

The following section summarizes the major points discussed in this paper.

# 8 Summary

This paper started by noting that a BBO algorithm is not likely to perform better than a random search unless some relations among the members of the search space are exploited. This led to the development of the SEARCH framework, which tries to exploit the role of relations in BBO. SEARCH presents an alternate perspective toward blackbox optimization. It views blackbox optimization as a composition of the following different processes:

1. classification of the search space;
2. sampling from the search space;
3. searching for appropriate relations that divide the search space in a suitable way;
4. searching for better classes defined by each of these relations, so that the desired solution is a member of one of these chosen classes;
5. resolution.

Searching along relations and the class spaces has three steps: (1) evaluation, (2) ordering, and (3) selection. Evaluation detects good relations or classes; ordering constructs a ranking based on the evaluation; selection picks up the good ones and discards the bad ones.

A relation is appropriate if it can classify the search space in such a way that the class containing the globally optimal solution can be detected using the class comparison statistic. This requirement is called proper delineation of the search space. Determining whether a relation satisfies this or not requires decision making in absence of complete knowledge. This is a source of decision error in BBO.

Detecting a good relation in turn requires approximate identification of good and bad classes defined by the relation. This introduces the other dimension of the SEARCH framework, searching for better classes. Even if the relation properly delineates the search space, decision error can be made during the comparison between two different classes. This is again viewed as a decision problem.

Success in blackbox optimization is tied with the successes along these two different dimensions. After presenting a general description of search from this perspective, we incorporated ordinal class and relation comparison statistics into the framework. A closed form bound in sample complexity is developed for any BBO that can be solved by considering a finite set of relations defined among the members of the search space. The sample complexity grows linearly with the cardinality of the set of relations, the maximum index value of these relations, the desired solution quality (described using order statistics), the desired degree of appropriateness of the relations, and the overall success probability demanded. This expression for sample complexity established our earlier argument that consideration of relations among the members of the search space is unavoidable if an algorithm aspires to perform better than random enumerative search. We specifically showed that the sample complexity approaches the size of the search space when we consider no relations that put at least two ground members into the same class. In this case, the only reduction in sample complexity comes from the relaxation in the desired solution quality and the overall success probability. Therefore, emphasizing the role of relations is fundamental and should be a part of any model of BBO.

This paper also laid out a clear approach toward designing new BBO algorithms by identifying the basic principles for understanding the three major entities in BBO,

- search algorithm
- problem
- user's role in defining the set of relations

The main points of this part are recapitulated in the following.

First we defined what it means to be an algorithm in SEARCH. Different components that are required for solving a BBO problem in SEARCH are listed together and discussed. We hope that this will provide a common ground for developing new blackbox optimization algorithms in the future. Before designing any new algorithm, we should ask several questions: how does this algorithm process relations? How does it treat the classes? Does the operator for sample generation follow the decision making in the relation and class spaces? What are the relation and class comparison statistics? How does it take advantage of implicit parallelism?

Then we discussed two important issues that may help make a BBO algorithm computationally more efficient. It is also noted that the index of the relation considered in the beginning of the search process should be bounded by a constant. The smaller the constant, the better it is. This requirement is called *bottom-up* organization of search, which may have some biological confirmation. It is noted that parallel evaluation of different relations at no additional function evaluation is possible by exploiting the structural property of the set of relations, when ordered based on the *order* of relations. We identified this as implicit parallelism (Holland, 1975).

This section also rigorously defined problem difficulty in SEARCH. The notion of problem difficulty introduced here is philosophically very similar to the PAC-learning theory (Natarajan, 91). The SEARCH framework decouples problem difficulty along these dimensions:

1. problem size
2. success probability in making the right decision to choose a relation that properly delineates the search space
3. success probability in selecting the class containing the optimal solution
4. desired quality of relations and the solution

All of the above dimensions of difficulty, except problem size, came as a part of the development of SEARCH. Identifying the exact quantitative dependence upon the problem size requires relating the size of the set of relations with problem size. We considered sequence representation as the source of relations and identified the class of order-$k$ delineable problems that can be solved in polynomial sample complexity.

The primary roles of the user of a BBO algorithm are to define the set of relations and to choose the perturbation operators based on the chosen set of relations. We first note three of the many possible ways to define relations over the search domain. The appropriateness of a set of relations is quantified in terms of the delineation-ratio. Computational benefits can be achieved by choosing a set of relations in which low-order relations satisfy the delineation requirement and also parallel evaluation of relations is possible by exploiting the structure of $\Psi_r$ when organized based on the *order* of the relations. All these factors contribute to the efficacy of the set of relations. At the end, the role of perturbation operators is viewed from the relation construction perspective. Perturbation operators need to generate samples for those classes that are currently being evaluated. Therefore, search operators clearly need to know about the classes for which they are going to generate the samples.

SEARCH also provides insights in natural evolution. Understanding evolution in the light of SEARCH requires paying attention to the intra-cellular flow of genetic information—gene expression. An alternate computational model of evolution is sketched that extends the traditional model of evolutionary computation (Holland, 1975) by incorporating steps of gene expression. A direct correspondence between different components of gene expression process and SEARCH is hypothesized.

# 9 Future Work and Conclusions

In this section we present the future extensions and the main conclusions of this paper.

## 9.1 Future work

There are several possible ramifications of this work. Some of them are listed in the following:

1. Design new algorithms in a more constructive manner, by designing new instances of the components of the laundry list that defines an algorithm in SEARCH.
2. Find classes of BBO problems, other than the order-$k$ delineable problems, that can be solved in polynomial complexity.
3. Explore the possibility of new relation construction.

The following paragraphs present a brief account of them.

This work outlined the main components that an algorithm needs in order to transcend the limits of random enumerative search. Now that we have a listing of the different components, development of new algorithms should be more constructive. Design of a new algorithm is now reduced to design of new instances of one or more items from this laundry list. Designing a new class or relation comparison statistic, designing a new ordering construction algorithm in either of these spaces, and defining a new order to consider relations are some examples.

The class of order-$k$ delineable problems has been identified as solvable in polynomial sample complexity. Other such classes of problems need to identified.

Most of the blackbox optimization algorithms rely on the user-provided source of relations, such as representation, operators. SEARCH also analyze BBO when such a source of relation exists. An immediate extension is to explore the possibility of using new relation construction. When the set of relations $\Psi_r$ does not have enough relations that properly delineate the search space, BBO is destined to fail. When we do not know this beforehand, the search will be executed anyway. However, the question is whether or not we can use the information gathered from this bad set of relations to construct a new set of relations that is more appropriate for the problem. One possible way to approach this may be to solve assuming some order-$k$ delineability and then use that information to construct a new source of relations and observe the improvement of performance if any. The following section lists the main conclusions of this paper.

## 9.2 Conclusions

The main conclusions of this paper are as follows:

1. It is possible to view blackbox optimization from a common unified perspective. The SEARCH framework offers one possible way to do that in an ordinal, probabilistic, and approximate sense.
2. Unless we assume relations among the members of the search space, blackbox search cannot be any better than random search. Therefore the role of relations in BBO is fundamental.
3. The appropriateness of relations in solving a BBO problem can be quantified in terms of delineation, and it can be taken into account while deriving bounds on success probability and sample complexity in BBO.
4. Relations can be evaluated in parallel from a common set of samples by exploiting some structural properties of the set of relations provided to the algorithm. This observation can be used to quantify *implicit parallelism.*
5. Bounds on sample complexity and success probability can be derived for the given relation space, class space, desired quality of solution and reliability.
6. Problem difficulty in BBO can be rigorously quantified using the SEARCH framework.
7. The class of order-$k$ delineable problems can be solved with sample complexity growing polynomialy along $\ell$, $q$, $q_r$, $1/d^*$, and $1/d_r^*$. This class of problems can be solved by considering a bounded number of relations.

8. SEARCH offers an alternate perspective of natural evolution that establishes the computational role of gene expression in evolution. Many of the existing computational models of evolution emphasize only on the extra-cellular flow of genetic information. SEARCH appears to suggest that the intra-cellular expression of genetic information is the primary aspect of evolutionary search, since unless the correct relations are detected first, the search is unlikely to be efficient.

The SEARCH framework and its implications offered an alternate approach toward different components of BBO. We hope that this work will lead to a more constructive approach toward designing new algorithms for BBO and remove some of the myth, folklore, and black magic from this field.

# 10 Acknowledgments

This work has been supported by AFOSR Grant F49620-94-1-0103, and the UIUC Department of General Engineering. We also thank Jeffrey Horn and George Harik for providing very useful feedbacks. The first author also acknowledges the support from the Computational Methods Division, Los Alamos National Laboratory.

# References

Ackley, D. H. (1987). *A connectionist machine for genetic hill climbing.* Boston: Kluwer Academic.

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

Archetti, F., & Schoen, F. (1984). A survey on the global optimization problem: General theory and computational approaches. *Annals of Operations Research*, *1*(1), 87–110.

Bagley, J. D. (1967). The behavior of adaptive systems which employ genetic and correlation algorithms. *Dissertation Abstracts International*, *28*(12), 5106B. (University Microfilms No. 68-7556).

Betrò, B. (1983). A bayesian nonparametric approach to global optimization. In Stähly, P. (Ed.), *Methods of Operations Research* (pp. 45–47). Atenäum Verlag.

Brindle, A. (1981). *Genetic algorithms for function optimization.* Unpublished doctoral dissertation, University of Alberta, Edmonton, Canada.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to algorithms.* Massachussets: MIT Press ; New York: McGraw-Hill.

Dantzig, G. B. (1963). *Linear programming and extensions.* New Jersey: Princeton University Press.

Dasgupta, D., & McGregor, D. R. (1992). Designing neural networks using the structured genetic algorithm. *Artifical Neural Networks*, *2*, 263–268.

David, H. A. (1981). *Order statistics.* New York: John Wiley & Sons, Inc.

Davis, L. (Ed.) (1987). *Genetic algorithms and simulated annealing.* Los Altos, CA: Morgan Kaufmann.

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems. *Dissertation Abstracts International*, *36*(10), 5140B. (University Microfilms No. 76-9381).

Dixon, L. C. W., & Szegö, G. P. (1978). The global optimization problem: an introduction. In Dixon, L. C. W., & Szegö, G. P. (Eds.), *Towards global optimization 2* (pp. 1–15). Amsterdam: North-Holland.

Dreyfus, S. E., & Law, A. M. (1977). *The art and theory of dynamic programming.* New York: Academic Press.

Dueck, G., & Scheuer, T. (1988). *Threshold accepting–a general purpose optimization algorithm appearing superior to simulated annealing* (Technical Report No. 88.10.011). IBM Heidelberg Sci. Center.

Ferreira, A. G., & Žerovnik, J. (1993, 10/11). Bounding the probability of success of stochastic methods for global optimization. *Computers, Mathematics, Applications*, *25*, 1–8.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution.* New York: John Wiley.

Gibbons, J. D., Sobel, M., & Olkin, I. (1977). *Selecting and ordering populations: A new statistical methodology*. New York: John Wiley & Sons, Inc.

Glover, F. (1989). Tabu search - part I. *ORSA Journal on Computing*, *1*, 190–206.

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal, deceptive problem. In Davis, L. (Ed.), *Genetic Algorithms and Simulated Annealing* (pp. 74–88). San Mateo, CA: Morgan Kaufmann. (Also TCGA Report 86003).

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. New York: Addison-Wesley.

Goldberg, D. E. (1990). A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, *4*(4), 445–460. (Also TCGA Report No. 90003).

Goldberg, D. E., Deb, K., & Clark, J. H. (1993). Accounting for noise in the sizing of populations. In Whitley, L. D. (Ed.), *Foundations of Genetic Algorithms 2* (pp. 127–140). San Mateo, CA: Morgan Kaufmann.

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

Gomulka, J. (1978). Deterministic vs probabilistic approaches to global optimization. In Dixon, L. C. W., & Szegö, G. P. (Eds.), *Towards global optimization* (pp. 19–29). Amsterdam: North-Holland.

Hart, P. E., Nilson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, *4*(2), 100–107.

Hart, W. E. (1994). *Adaptive global optimization with local search*. Doctoral dissertation, Department of Computer Science, University of California, San Diego.

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

Hollstien, R. B. (1971). Artificial genetic adaptation in computer control systems. *Dissertation Abstracts International*, *32*(3), 1510B. (University Microfilms No. 71-23,773).

Jacob, F., & Monod, J. (1961). Genetic regulatory mechanisms in the synthesis of proteins. *Molecular Biology*, *3*, 318–356.

Jones, D. R., & Stuckman, B. E. (1992). Genetic algorithms and the Bayesian approach to global optimization. *Proceedings of the 1992 International Fuzzy Systems and Intelligent Control Conference*, 217–235.

Jones, T. (1995). *Evolutionary algorithms, fitness landscapes and search*. Doctoral dissertation, Department of Computer Science, University of New Mexico, Albuquerque, NM.

Jones, T., & Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Eshelman, L. (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 184–192). San Mateo, CA: Morgan Kaufmann.

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

Kargupta, H., & Goldberg, D. E. (1995, October). *Search: An alternate perspective toward blackbox optimization*. In Communication.

Kauffman, S. (1993). *The origins of order*. New York: Oxford University Press.

Kirpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Mahfoud, S. W., & Goldberg, D. E. (1992). A genetic algorithm for parallel simulated annealing. In Männer, R., & Manderick, B. (Eds.), *Parallel Problem Solving from Nature* (pp. 301–310). Amsterdam: Noth-Holland.

Michalski, R. S. (1983). Theory and methodology of inductive learning. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.), *Machine learning: An artificial intelligence approach* (pp. 323–348). Tioga Publishing Co.

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

Papadimitriou, C. L., & Steiglitz, K. (1982). *Combinatorial optimization.* New Jersey: Prentice Hall.

Perttunen, C., & Stuckman, B. (1990). The rank transformation applied to a multiunivariate method of global optimization. *IEEE Transactions on System, Man, and Cybernetics*, *20*, 1216–1220.

Radcliffe, N. J. (1993). Genetic set recombination. In Whitley, L. D. (Ed.), *Foundations of Genetic Algorithms* (pp. 203–219). San Mateo, CA: Morgan Kaufmann.

Ratschek, H., & Voller, R. L. (1991). What can interval analysis do for global optimization. *Journal of Global Optimization*, *1*, 111–130.

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

Rich, E., & Knight, K. (1991). *Artificial intelligence* (2 ed.). New York: McGraw Hill.

Rinnooy Kan, A. H. G., & Timmer, G. T. (1984). Stochastic methods for global optimization. *American Journal of Mathematics and Management Sciences*, *4*(1), 7–40.

Rosenberg, R. S. (1967). Simulation of genetic populations with biochemical properties. *Dissertation Abstracts International*, *28*(7), 2732B. (University Microfilms No. 67-17,836).

Rudolph, G. (1994). Massively parallel simulated annealing and its relation to evolutionary algorithms. *Evolutionary Computation*, 361–383.

Schoen, F. (1991). Stochastic techniques for global optimization: A survey of recent advances. *Journal of Global Optimization*, *1*, 207–228.

Shapiro, R. (1986). *Origins: A skeptic's guide to the creation of life.* New York: Summit Books.

Sirag, D. J., & Weisser, D. J. (1987). Toward a unified thermodynamic genetic operator. In Grefenstette, J. J. (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 116–122). Hillsdale, NJ: Lawrence Erlbaum Associates.

Smith, R. E. (1988). *An investigation of diploid genetic algorithms for adaptive search of nonstationary functions* (TCGA Report No. 88001). Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms.

Stryer, L. (1988). *Biochemistry.* New York: W. H. Freeman Co.

Törn, A., & Žilinskas, A. (1989). *Global optimization.* Berlin: Springer-Verlag.

Vavasis, S. A. (1991). *Nonlinear optimization: Complexity issues.* New York: Oxford University Press.

Wald, G. (1954, August). The origin of life. *Scientific American.*

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