Title: BLACKBOX AND NON-BLACKBOX OPTIMIZATION: A COMMON PERSPECTIVE

Author(s): H. Kargupta

Submitted to: NATO ASI Meeting
Gran de Canaria de Spainia
July 1-5, 1996

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
Blackbox And Non-blackbox Optimization: A Common Perspective

Hillol Kargupta*
Computational Science Methods Group
Los Alamos National Laboratory
Los Alamos, NM, 87545

Abstract

The SEARCH (Search Envisioned As Relation & Class Hierarchizing) framework developed elsewhere (Kargupta, 1995; Kargupta & Goldberg, 1995) offered an alternate perspective toward blackbox optimization (BBO)—optimization in absence of domain knowledge. This paper argues that the fundamental concepts are also applicable to non-blackbox optimization (NBBO)—optimization in presence of information about the search domain and objective function. 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. This paper reviews some of the main results of that work and describes its generality by considering different popular BBO and NBBO algorithms.

1 Introduction

The desire for solving large, high dimensional optimization problems is ever increasing. Developing techniques to solve large linear optimization problems addresses one aspect. There also exists a large class of problems which may not be classified as linear problems. Optimization problems in which sufficient information about the properties of the objective function is available a priori can be called non-blackbox optimization (NBBO). Apart from NBBO problems, there are also many optimization problems in which little knowledge is available about the properties of the search space. Blackbox optimization (BBO) deals with this extreme case in which little domain knowledge about the problem structure is available. Despite the individual advancements of techniques for solving both BBO and NBBO little progress has been been understanding both of them from a common perspective. This paper makes an effort to do that using the SEARCH (Search Envisioned As Relation and Class Hierarchizing) framework introduced elsewhere (Kargupta, 1995).

SEARCH made an effort to capture the fundamental computations in search and optimization in terms of relations, classes and partial ordering. SEARCH is primarily motivated by the observation that searching for optimal solution in a BBO is essentially a pure 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). SEARCH decomposed optimization into (1) relation, (2) class, and (3) sample spaces. SEARCH also identified the importance of searching for appropriate relations in BBO. No BBO algorithm can efficiently solve a reasonably general class of problems unless it searches for relations. Kargupta (1995) also showed that the class of order-k delineable problems

*The author can be reached at, P.O. Box 1663, XCM, Mail Stop F645, Los Alamos National Laboratory, Los Alamos, NM 87545, USA. e-mail: hillol@lanl.gov
can be solved in SEARCH with sample complexity polynomial in problem size, desired quality and reliability of the solution. These results are relevant for any BBO algorithm. Although the analysis is performed for BBO problems we shall see that the underlying concepts remain unchanged for NBBO problems.

In this paper we first present a brief description of the SEARCH effort. We present some of the main results without going through the detailed derivations. Section 2 introduces the fundamental similarities and differences between BBO and NBBO. Section 3 briefly describes the main concepts and the foundation of SEARCH. Section 4 describes what it means to be an optimization algorithm in SEARCH. This is followed by a discussion on problem difficulty in Section 5. Section 6 specializes the results for sequence representation and identifies the class of order-\(k\) delineable problems, that can be solved in polynomial sample complexity in SEARCH. Section 7 further explains the implications of order-\(k\) delineable problems. Section 8 presents few case studies. Finally, Section 11 concludes this paper.

2 Optimization: Blackbox and Non-blackbox

An optimization problem is comprised of a search domain and an objective function to be optimized. The objective function of an optimization problem can be defined as,

\[ \Phi : \mathcal{X} \rightarrow \mathcal{Y} \]  

The objective of a maximization problem is to find some \(x^* \in \mathcal{X}\) such that \(\Phi(x^*) \geq \Phi(x)\) for all \(x \in \mathcal{X}\). In blackbox optimization, the objective function is available as a black box, i.e., for a given decision variable, \(x\) in the feasible domain, it returns the function value \(\Phi(x)\). No local or global information about the function is assumed. For most of the optimization problems with practical importance the size of the search space grows exponentially with the problem size. Since in BBO no problem knowledge is available the only way to do better than enumerative search is to take few samples from the search domain and guess about the possible location of the optimal solution. Therefore, searching for an optimal solution in BBO is essentially an inductive process (Michalski, 1983). It has been well known that in absence of any relation among the members of the search space, induction is no better than enumeration (Watanabe, 1969). A relation is a set of ordered pairs.

On the other hand in non-blackbox optimization some knowledge about the objective function and/or search domain is available. For example in linear programming problem the objective function is known to be linear; meaning, for any two given points \(x_1\) and \(x_2\) in the search domain, we know that their objective function values are linearly related to each other. If an objective function is convex we know that for some \(0 < \lambda < 1\), the convexity condition \(\Phi((1 - \lambda)x_1 + \lambda x_2) \leq (1 - \lambda)\Phi(x_1) + \lambda\Phi(x_2)\) is satisfied. If the local gradient information is available then we know that how the function value changes locally between any two points. As we see, in NBBO we have access to some additional information that relates a set of points to each other. This information about the relative properties of different points can be interpreted as relations using the set theoretic terminology. Therefore, we can say that in NBBO the search for relations is made easier depending on the amount of available domain knowledge.

There exists a huge body of literature for NBBO algorithms (Törn & Zilinskas, 1989). Some popular NBBO algorithms are Bayesian methods (Bétró, 1983), Branch and bound methods. Although the amount of work done on BBO algorithms is relatively smaller, many new algorithms like Genetic algorithms (GAs) (Holland, 1975), simulated annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983), tabu search (Glover, 1989) became popular recently for solving BBO problems. Despite the
emergence more and more new algorithms, little work has been done to understand the fundamental processes in optimization for both BBO and NBBO from a common ground. The following section introduces the SEARCH framework that takes a small step toward that.

3 The SEARCH Framework

This section presents a brief review of the SEARCH framework. As we noted in the previous section that the main difference between the BBO and NBBO problems is the availability of information about relations among the members of the search space. The search for relations decreases as the availability of such information increases. In order to maintain its generality the SEARCH framework considers relation search as a component of optimization. SEARCH offers a perspective of optimization in a probabilistic and approximate sense in terms of relations, classes and partial ordering. Section 3.1 describes the general concepts. Section 3.2 discusses the bound on sample complexity in SEARCH.

3.1 Foundation

The foundation of SEARCH is laid on a decomposition of the blackbox search problem into relation, class, and sample spaces. As mentioned earlier, a relation is a set of ordered pairs. For example, in a set of cubes, some white and some black, the color of the cubes defines a relation that divides the set of cubes into two subsets—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 $f###$, 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. The total number of classes defined by a relation is called its index. The order of a relation is the logarithm of its index with some chosen base. 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 operators 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

---

1 An equivalence relation is a relation that is reflexive, symmetric, and transitive.
search space as more promising than others often based on some domain specific knowledge. Clearly this can be a source of relations.

On the other hand in NBBO, the relations can be introduced in a more direct way by using domain knowledge about the search space and the objective function. 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 by every search algorithm, either implicitly or explicitly. The role of relations in optimization is very fundamental and important.

Relations divide the search space into different classes and the objective of sampling based non enumerative optimization is to detect those classes that are most likely to contain the optimal solutions. To do so requires constructing a partial ordering among the classes defined by a relation. The classes are evaluated using samples from the search domain and a class comparison statistic is used for comparing different classes. For a given class comparison statistic \( \leq_T \) and some number \( M \), a relation is said to properly delineate the search space if the class containing the optimal solution is within the top \( M \) classes, when the set of all classes defined by the relation are ordered using \( \leq_T \). This basically means that if a relation satisfies the delineation constraint then, given sufficient samples, the relation will pick up the class containing the optimal solution within the top \( M \) ranked classes. If a relation does not satisfy this, then the relation leads to wrong decision and as a result success in finding the optimal solution is very unlikely.

A particular relation may not satisfy the delineation constraint for different problems, different class comparison statistics, and different values of \( M \). One relation may work for a particular case and may fail to do so for a different setting. Therefore, any algorithm that aspires to be applicable for a reasonably general class of problems, must search for appropriate relations. Determining whether or not a relation satisfies this delineation constraint requires decision making in absence of complete knowledge. For a given relation space \( \Psi_r \), an optimization algorithm must identify the relations that properly delineate the search space with certain degree of reliability and accuracy. This requires comparing one relation with another using a relation comparison statistic and constructing a partial ordering among them.

An optimization algorithm in SEARCH cannot be efficient if it needs to consider relations that divide the search space in classes, with the total number of classes growing exponentially with the problem dimension. For example, in an \( \ell \)-bit sequence representation, if there is a class of problems which requires considering the equivalence relations with \( (\ell - 1) \) fixed bits then there is a major problem. This relation divides the search space into \( 2^{\ell-1} \) classes and we cannot solve this problem in complexity polynomial in \( \ell \). However, in optimization the ultimate objective is to identify the optimal solution which basically defines a singleton class. The smaller the cardinality of the individual classes, the larger the index of the corresponding relation. So we need the higher order relations for finally identifying the optimal solution, but we cannot directly evaluate them since their index is large. The solution is to limit our capability and realize that we can only solve those problems which can be addressed using low order relations and when high order relations are decomposable to those low order relations. This means that the information about low order relations can be used to evaluate the higher order relations. Consider the following example. Let \( r_0 \) be a relation that is logically equivalent to \( r_1 \land r_2 \), where \( r_1 \) and \( r_2 \) are two different relations; the sign \( \land \) denotes logical AND operation. If either of \( r_1 \) or \( r_2 \) was earlier found to properly delineate the search space, 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. This process in SEARCH is called resolution. Resolution basically evaluates the relations of higher order using the information gathered by direct evaluation of lower
Figure 2: Fitness distribution function of two classes $C_{j,i}$ and $C_{k,i}$.

order relations.

The above description gives a brief informal overview of the SEARCH framework. As we saw, SEARCH addresses optimization on three distinct grounds: (1) relation space, (2) class space, and (3) sample space. Figure 1 shows this fundamental decomposition in SEARCH. The major components of SEARCH can be summarized as follows:

1. classification of the search space using relations;
2. sampling;
3. evaluation, ordering, and selection of better classes;
4. evaluation, ordering, and selection of better relations;
5. resolution.

A detailed description of each of these processes can be found elsewhere (Kargupta, 1995). In the following part of this section we consider the expression for sample complexity in SEARCH derived elsewhere (Kargupta, 1995) and define the class of order-$k$ delineable problems that can be efficiently solved in SEARCH.

3.2 Sample complexity in SEARCH

For a given relation space, and an algorithm in SEARCH, it is possible to derive the bound on sample complexity for the desired quality and reliability of decision making. Defining an algorithm in SEARCH first requires specifying class and relation comparison statistics. Although, most of the existing optimization algorithms do not explicitly define them, SEARCH does so in order to quantify and understand the role of decision making in the relation and class spaces. Kargupta (1995) considered distribution free ordinal comparison statistics. In an ordinal comparison statistic, two distributions are compared on the basis of some chosen percentile. Figure 2 shows the cumulative distribution function $F'$ and $F$ of two arbitrary subsets $C_{j,i}$ and $C_{k,i}$, respectively. Indices $j$, $k$ represent the two classes defined by some relation $r_i$. 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 $\Phi_{[r],j,i} \leq \Phi_{[r],k,i}$; $\Phi_{[r],j,i}$ and $\Phi_{[r],k,i}$ are the solutions of $F'(\Phi_{j,i}) = \alpha$ and $F(\Phi_{k,i}) = \alpha$, respectively. Let us define

$$d = F(\Phi_{[r],k,i}) - F(\Phi_{[r],j,i}).$$
The variable $d$ defines the \textit{zone of indifference}, which is basically the difference in the percentile value of $\Phi_{[r],i,j,i}$ and that of $\Phi_{[r],j,i,j}$ computed from the same cdf $F$. Figure 2 clearly explains this definition. We can quantify the decision making process using such ordinal class and relation comparison statistics. Let $\Psi_r$ be the given relation space and $S_r \subseteq \Psi_r$ be the set of relations needed to solve the given problem. We denote the index of a relation $r_i$ by $N_i$. Define,

$$N_{\text{max}} = \max\{N_i| \forall r_i \in S_r\}$$

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

where $F(\Phi_{[r],*,i})$ is the cdf of the class containing the optimal solution. The index $j$ varies over all the classes defined by a relation $r_i$. Index $i$ varies over all the relations in $\Psi_r$. If $d^*$ is a constant such that $d' \geq d^*$, that corresponds to the desired quality of decision making in the class space, the bound on overall success probability is,

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

where $H(\alpha)$ is the binary entropy function, $H(\alpha) = -\alpha \log_2 \alpha - (1 - \alpha) \log_2 (1 - \alpha)$. From this we can derive the bound on overall sample complexity,

$$\text{SC} \leq \frac{N_{\text{max}}||S_r|| \log(1 - \frac{q_r}{q})}{d^*}$$ (3)

where $q$ is the overall desired success probability in SEARCH and $q_r$ is the desired success probability in the relation space. $M_{\text{min}}$ is a constant that depends on the memory used by the algorithm. The success probability in the relation space depends on the appropriate decision making and the statistic defined for comparing relations with each other. Similar expression for the computational complexity in the relation space can be derived as shown elsewhere (Kargupta, 1995).

Inequality 3 presents the upper bound on the sample complexity in optimization. As we increase the 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. This also shows that SC decreases with increase in $q_r$. Note that the ratio $\left(\frac{q_r}{q}\right)^{\frac{1}{||S_r||(N_{\text{max}} - M_{\text{min}})}}$ approaches 1 in the limit as $||S_r||(N_{\text{max}} - M_{\text{min}})$ approaches infinity. Therefore, SC grows at most linearly with the maximum index value $N_{\text{max}}$ 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. Kargupta (1995) also showed that when no relations are considered, this expression points out that the sample complexity will be of the order of the size of search space; in other words search will be no better than enumeration. For a given relation space and a class of problems that can be solved considering a bounded number of relations from that space, inequality 3 gives the bound on sample complexity for desired quality and reliability of the decision making. This expression is valid for any finite blackbox search space. The following section identifies the different components of an optimization algorithm in SEARCH.

4 An Optimization Algorithm In SEARCH

The SEARCH framework decomposed optimization in a given search domain 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 an optimization 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 \( M_r^2 \) and \( M_i^2 \) determine the size of the memory for storing relations and classes respectively. In a 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.


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. In SEARCH, this is introduced in an non parametric way. Examples of parametric approaches may be found elsewhere Goldberg, Deb, and Clark (1993, Holland (1975).

SEARCH provides a common ground for developing new sampling based adaptive optimization algorithms in the future. Regardless of the motivation and background, any blackbox optimization algorithm should clearly define each of the above listed components. An optimization algorithm should define how it processes relations, classes, and samples. It should state its relation and class comparison statistics. The following section addresses this important aspect of optimization—problem difficulty.

5 Problem Difficulty

SEARCH presents an alternate perspective of problem difficulty in optimization. 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 3 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 \( 2M_r \), the relation space memory comes into the picture when the memory is not large enough to process all the relations in \( \Psi_r \).
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 1 (Problem difficulty in SEARCH)** Given an optimization function \( \Phi : X \rightarrow \mathbb{R} \) 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.

This gives a general description of the SEARCH perspective of problem difficulty. The following section brings us closer to the ground by specializing the framework for sequence representation. We identify a class of problems in sequence representation that can be solved in polynomial sample complexity in SEARCH.

### 6 Sequence representation and the class of order-\( k \) delineable problems

Sequence representation is used in many evolutionary optimization algorithms. We therefore choose this for exploring the class of problems that can be efficiently solved.

A sequence representation can be defined as \( I : X \rightarrow \Lambda^\ell \), where \( \Lambda \) is the alphabet set. This sequence representation induces a set of equivalence relations, \( \Psi_r = \{f, \#\}^\ell \), where \( f \) indicates values that must match for equivalence and \( \# \) is a wild character that matches any value. The cardinality of the set of all such equivalence relations \( ||\Psi_r|| = 2^\ell \).

**Definition 2 (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 \( \leq \pi \), 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,

\[
\mathcal{G} = \bigcup_{a_1,a_2,...,a_k} C_{[a_1]} \cap C_{[a_2]} \cap ... \cap C_{[a_k]},
\]

is bounded by a polynomial of \( \ell, \rho(\ell) \). The indices \( a_1, a_2, \ldots, a_k \) can take any value in between 1 and \( M_i \).
It has been shown elsewhere (Kargupta, 1995) that this class of problems can be solved in sample complexity polynomial in $q$, $q_r$, $1/d^*$, $1/d^*_r$, and the problem size $\ell$. To achieve an overall success probability of $q$, the required sample complexity is,

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

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

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

This basically says that the problems that can be solved using a polynomially bounded number of relations can be efficiently solved in SEARCH. Note that this class of problems is fundamentally defined in terms of relation space. Note that, a problem may be order $k$-delineable in one relation space but fail to be for another relation space. Therefore when the relation space is already chosen by fixing the sequence representation, an optimization problem can be solved efficiently if it is order-$k$ delineable in that relation space.

Before we conclude this review on SEARCH, let us revisit the issue of order-$k$ delineability in order to clear up our current objectives and future directions for designing optimization algorithms.

## 7 Implications Of Order-$k$ delineability

As we saw in the previous section, the notion of order-$k$ delineability presents a picture of the general class of optimization problems from the perspective of an algorithm. In SEARCH, defining an optimization algorithm requires specifying the relation space, class comparison statistic, and the constant $M$ that defines how many “top” classes will be picked up. Therefore, by definition an algorithm in SEARCH specifies the class of order-$k$ delineable problems. For a chosen class comparison statistic and $M$, the relation space restricts the class of order-$k$ delineable problems for an algorithm. Changing the relation space by constructing new relations may convert a non-order-$k$ delineable problem to an order-$k$ delineable one. For some problems finding such transformation by constructing new relations may be possible in sample complexity, polynomial in problem size, reliability, and accuracy of the solution. Clearly, there may exist a class of non-order-$k$ delineable
/* Initialization */
T = High_temperature; // Initialize the temperature to a high value
Initialize(x); // Randomly initialize the state
Evaluate(x); // Evaluate the objective function value
{
  Repeat
  {
    Generate(x'); // Generate new state
    Evaluate(x'); // Evaluate the objective function value
    If ( Metropolis_criterion(x, x') TRUE )
      x = x' // Change state to x'
  }
Until (Equilibrium is reached)
Decrease(T); // Decrease the temperature
}
Until ( T < T_{min} Or (termination criterion TRUE ) )

Figure 4: A pseudo-code for simulated annealing.

problems, that can be transformed to order-k delineable problems in polynomial sample complexity. Figure 3 shows a schematic description of this classification of optimization problems.

It is important to note that, membership of a problem in the class of order-k delineable problems does not necessarily guarantee that the algorithm will solve that problem. It only says that the problem is "efficiently solvable" in the chosen relation space, class comparison statistic, and $M$. The algorithm needs to perform adequate sampling and make decisions with high confidence in the relation and class spaces in order to find the desired quality solution. Therefore, the first step of an algorithm should be to make sure it can solve its own order-k delineable class of problems. That will define the first milestone. The next step should be to introduce mechanism for new relation construction and investigate what kind of problems can be dynamically transformed to order-k delineable class of problems. More work is needed to develop optimization algorithms that follow the systematic decomposition of SEARCH.

In the following sections we shall consider some popular BBO and NBBO algorithms and describe their components from the SEARCH perspective to demonstrate the generality of this approach.

8 SEARCH And Simulated Annealing

Like many other algorithms, simulated annealing (SA) algorithm does not explicitly consider the relations. Therefore, the projection of SA into the SEARCH framework depends on our perspective toward SA as well. Since relations can be defined in many ways, when the relation space is not explicitly specified, identifying it leaves room for speculation. The original version of SA does not emphasize representation. Moreover, the random neighborhood generation operator does not pay enough consideration to the relations and classes defined by the chosen representation.

In this section, we therefore choose to view SA as a processor of relations and classes defined by the neighborhood generation operator. The following part of this section briefly discusses different counterparts of SEARCH in the SA.

- **Relation space**: A state $x_i$ and the neighborhood generation operator ($P$) are the two ingre-
Different classes

Starting state of SA

Final State

Figure 5: The SEARCH perspective of SA.

dients of the relations processed by the SA. For a given state \( x_i \), the neighborhood generation operator defines a set of states that can be reached in certain number of steps (s) from \( x_i \). This defines a relation among a certain subset of the search space. Therefore, a relation \( r_i \) in SA can be specified by \( (x_i, P, s) \).

- **Class space:** The relation \( (x_i, P, s) \) divides the search space into two classes—(1) the set of states that can be reached from \( x \) by applying \( P \) for \( s \) number of times and (2) the rest of the search space. This defines the class space for a given relation. Let us denote the first class by \( C_{1,i} \) and the second by \( C_{2,i} \).

- **Sample space:** The SA processes only one sample at a time. The sample represents the state of the algorithm.

Searching for the optimal solution in SEARCH also requires different comparison statistics and resolution for combining the features of different classes from different relations. The following discussion points out their counterpart in SA.

- **Relation and class comparison statistics:** Since SA does not explicitly defines the relations and classes, only one statistic, defined by the *Metropolis criterion*, is used for serving both purposes. This comparison statistic varies as the *temperature* changes.

- **Resolution:** Consider the two relations \( (x_1, P, s) \) and \( (x_2, P, s) \), where \( x_1 \) and \( x_2 \) are two arbitrary states from the search space. Let us denote the set of states that can be reached from \( x_1 \) and \( x_2 \) by applying \( P \) for \( s \) times by \( C_{1,1} \) and \( C_{1,2} \), respectively. Let \( x_i \) be the current state of SA and \( x_{i+1} \) be the next state. Now if \( x_1 \) and \( x_2 \) are such that the \( x_i \in C_{1,1} \) and \( x_i+1 \in C_{1,2} \), then the next state, \( x_{i+1} \), is basically a sample from the intersection set of the two classes \( C_{1,1} \) and \( C_{1,2} \). Generating samples from the intersection set of classes is essentially what resolution does.

The above discussion presents a perspective of SA in the light of SEARCH. Figure 5 pictorially depicts this perspective of SA. This figure schematically shows the trajectory of SA within the overlapping classes. As we mentioned earlier, this presents only one possible way to define classes and relations in SA. Since SA does not explicitly define them, different possibilities may be speculated. The following section considers another popular BBO algorithm, namely the genetic algorithms and describe the underlying computations in the light of SEARCH.
9 Simple Genetic Algorithm: The SEARCH Perspective

Genetic algorithms (De Jong, 1975; Goldberg, 1989; Holland, 1975) are BBO algorithms designed primarily motivated by natural evolutionary search. Despite their biological motivation, Holland (1975) presented the concepts based on computational arguments. In this section we revisit the relevant issues in the light of SEARCH.

9.1 Schema analysis: Why bother?

Holland (1975) introduced genetic algorithms (GAs) with the spirit of searching by equivalence class processing. He defined the schemata—similarity based equivalence classes in the sequence space (Radcliffe, 1991). Although the algorithm he proposed did not explicitly consider equivalence classes, he visioned GAs as an implicit processor of schemata at an abstract level. Since, relations play a critical role in making BBO efficient, it is important to study the processing of classes in any BBO algorithm. Clearly, Holland's approach using schema analysis complied with the emphasis on the relations defined by the representation. During the last decade as GAs gained popularity many different models of SGA (Srinivas & Patnaik, 1993; Nix & Vose, 1992) came up. Although these models can present an exact picture of the dynamics of the population members from generations to generations, the notion of class processing needs to be incorporated in order to use these models for designing efficient GAs. Jones (1995) offered a different kind of relation source in GAs. He defined a landscape based on crossover perturbation. Although it does not use any schema analysis, counterparts of schemata in the so called crossover landscape needs to be introduced for comparing the performance of GAs with that of random enumerative search. As we see, since relations and classes are what make a BBO algorithm transcend the limits of random search, any model of GAs that aspire to provide insight for designing efficient GAs must project a picture of the class processing. The following section discusses another rule of thumb believed by a portion of the GA research community and study its utility in the light of SEARCH.

9.2 Building block hypothesis: The bottom-line

The schema theorem clearly says that low order, short, above average equivalence classes are expected to be explored more. The term "short" captures the idea that one point crossover can efficiently generate samples and compute set intersections only for those equivalence relations, in which the fixed(f) bits are closer to each other. Schema theorem has been extrapolated to hypothesize that the success of a GA may depend on the increase in the proportion of members in low order, short, better classes (termed building blocks in (Goldberg, 1989)) and the computation of the intersection of these good classes using the crossover operator. This is often called as building block hypothesis. Again, this hypothesis does not necessarily describe the behavior of a GA, since it is partially based on the schema theorem, which is a simple first order bound. However, we need to pay little more attention to appreciate the rationale behind this hypothesis. It has been shown elsewhere (Kargupta, 1995) that the bottom-up organization of the blackbox search has computational significance. Although algorithms, which do not satisfy this criterion, do not necessarily have to be of exponential complexity, but they will have a large constant in front of the bounding polynomials. This desirable feature of the polynomial complexity search exactly resonates with the essence of building block hypothesis. The bottom-up organization requirement of the blackbox search says that the low order equivalence relations need to be evaluated first, which is also suggested by building block hypothesis. Therefore, the building block hypothesis should be viewed as a desirable feature for a polynomial complexity GA rather than something that describes the behavior of all GAs.

Since GAs work from a population of samples, relations can be evaluated in parallel. The following
section points out that the relation, class, and sample spaces are defined by a single population in GA.

9.3 Implicit definition of relation, class and sample spaces

In simple GA the relation, class, and the sample spaces are defined together in a single population. As we noted earlier during the development of SEARCH, relation and class spaces require distinct decision makings and interference among them may cause decision error. The SGA uses a single selection operator for making decisions in each of these spaces. As a result decision making is very noisy in SGA. This is one among the major bottlenecks of the SGA.

9.4 Crossover as resolution of SEARCH

In SGA, crossover exchanges parts of parent strings. If two partially good solutions can be detected and combined to produce a better solution crossover can be effective in finding good solutions. However, the condition of detecting partially good solution can only be satisfied when the optimization problem is delineable at a certain order in the chosen representation. The resolution operation of SEARCH that computes intersection of different classes can also be viewed as the process of combining class features. Therefore crossover can be fundamentally viewed as a way to implement resolution. However, in simple GA the relations and classes are processed implicitly and crossover is randomized with no direct bias toward good classes. The bias toward better classes is introduced through selection and random operation of crossover probabilistically hopes for accomplishing the right resolution among better class features.

9.5 Linkage: Search for relations

As we saw earlier, search for proper relations that satisfy the delineation requirement plays an important role in the blackbox search.

The simple GA considers only a small fraction of relations defined by the representation. A simple GA with one-point crossover (De Jong, 1975) favors those relations in which positions in sequence space defining equivalence are closer to each other and neglects those relations that contain equivalence defining positions far apart. One-point crossover also fails to generate samples for the intersection set of two equivalence classes in which fixed bits are widely separated. For example, in a 20-bit problem, single-point crossover is very unlikely to generate a sample from the intersection set of 1##···# (first bit is fixed) and #···1 (last bit is fixed). In biological jargon, this is called the linkage problem. Unfortunately, this is a major bottleneck of SGA. Although Holland (1975) realized the importance of solving this problem and suggested use of the inversion operator (Holland, 1975), it has been shown elsewhere (Goldberg & Lingle, 1985) that inversion is very slow and unlikely to solve this problem efficiently. One-point crossover is not the only type to suffer from this problem. Uniform crossover is another kind of crossover (Syswerda, 1989) often used in the simple GA. In uniform crossover, the exchange of bit values among the two parent strings takes place based on a randomly generated binary mask string. If the value of this mask string at a particular locus is 1, the corresponding bits in the parent strings get exchanged; otherwise they do not. Unlike one-point crossover, uniform crossover does not have any preference bias toward the closely spaced partitions. Since the relation space and the sample space are combined, random perturbation of the sample strings also result in disrupting proper evaluations of the relations. Uniform crossover should also fail to accomplish proper search in the relation space. In fact, this is exactly what Thierens and Goldberg (1993) reported. Their analysis and experimental results showed that the sample complexity grows exponentially with the problem size for solving bounded deceptive problems (Thierens & Goldberg,
1993) using a simple GA with uniform crossover. This discussion clearly points out that the search in the relation space is very poor in the case of a simple GA with either one-point or uniform crossover.

9.6 Discussion

The simple GA has many interesting features. The first important aspect is that GA emphasizes the role of representation in search and representation is one possible way to define a rich source of relations. Searching by constructing and ordering equivalence relations is very natural in GA. Apart from this, crossover provides an interesting tool to implement sample generation process from the intersection set of different equivalence classes. Probably one of the most interesting observations that Holland made in his book (1975) is the idea of implicit parallelism. Although this conjecture was not thoroughly laid in terms of computational arguments, the SEARCH framework confirms this observation in a quantitative manner. Despite these interesting features, the simple GA has several major problems. The main bottlenecks of the simple GA are listed below:

- Poor search for relations;
- Lack of precise mechanism for implicit parallelism;
- Relation, class, and sample spaces are combined.

Now that we have discussed the computational aspects of two BBO algorithms like SA and SGA in the light of SEARCH, it will be interesting to consider an NBBO in the light of SEARCH. The following section presents the SEARCH perspective of Bayesian optimization algorithms.

10 SEARCH and Bayesian Optimization

Bayesian algorithms try to solve an optimization problem by gradually developing an approximation of the objective function in an implicit way. The approximation of the objective function is captured by using a random variable. Like any other adaptive sampling-based search techniques Bayesian algorithms make use of sample points from the search space. These algorithms can also be viewed in the light of SEARCH and the computational results derived in this section are also valid for this approach. In the following part of this section we discuss this possibility.

Although different Bayesian algorithms exists (Betro, 1983) in the literature, I shall confine myself to the method by Kushner (1963) for illustrating the fundamental characteristics of this class of algorithms, following Jones and Stuckman (1992). Kushner’s (1963) univariate method divide the entire search space into different line segments. It then tries to capture the distribution of the objective function values within these line segments using the mean and variance of a random variable $X(s)$. The objective function is evaluated at the end points of a segment and a point within the segment, $s'$ is chosen that maximizes the probability of improving the current maximum. Computation of this point is based on second order approximation and requires the mean $E[X]$ and the variance $\text{Var}[X]$. For example, if $[a, b]$ defines a segment, then the probability that $X(s)$ will exceed the current maximum, $\Phi_{\text{max}}$ by an amount $\epsilon$ can be expressed as (Jones & Stuckman, 1992),

$$Pr[X(s) \geq \Phi_{\text{max}} + \epsilon] = 1 - \Xi \left( \frac{\Phi_{\text{max}} + \epsilon - E[X(s)]}{\sqrt{\text{Var}[X(s)]}} \right)$$

where $s \in [a, b]$ and $\Xi(\cdot)$ is a monotonically increasing function. Once a point is chosen that maximizes this probability, the segment can be further divided based on this new point. The process is continued in the same way, each iteration including a new point and further subdividing the segment. Following (Jones & Stuckman, 1992) we summarize this process in Figure 6.
/* Initialize */
Set the complete search space to a single segment.
Repeat
{
1. Evaluate the objective function at the endpoints of the segment.
2. Compute the point, $s'$ that maximizes the probability of increasing
   the objective function value by more than $\varepsilon$.
3. Evaluate $\Phi(s')$ and update the estimate $\Phi_{\text{max}}$.
4. Define the new set of segments based on previous end points and $s'$.
} Until (termination criterion TRUE)

Figure 6: A pseudo-code for Kushner’s Bayesian optimization algorithm.

In the following part of this section we identify the main correspondence between Bayesian optimization with SEARCH.

- **Relation space:** The relation space depends on how the algorithm chooses to classify the search space. Kushner’s (1963) method chooses to construct the line segments using the point $s'$ that maximizes the probability of increasing the objective function value. Perttunen and Stuckman (1990) extended Kushner’s (1963) method to multi-dimension case and suggested Delaunay triangulation of the search space for classification. Again, Delaunay triangulation defines the relation space for the algorithm by Perttunen and Stuckman (1990).

- **Class space:** The class space is defined by the segments and Delaunay triangles in case of Perttunen’s algorithm.

- **Sample space:** Kushner’s algorithm evaluates the objective function at the end points of the segments. This defines the sample space.

- **Relation and Class comparison statistics:** The classes in Bayesian optimization are compared based on the probability of increase in objective function value. The maximum probability values are computed and the class with the highest value is selected. The search for relations is not very much emphasized. Segmentation of the search space into lines or Delaunay triangles provide the fundamental source of relations. Like many other algorithms, search for relations is loosely defined and combined with class selection.

- **Resolution:** In Bayesian optimization previously eliminated classes are explicitly rejected from future considerations. Considerations in Bayesian optimization.

The following section concludes this paper.

11 Conclusion

The SEARCH framework provided insightful quantitative results relevant to any BBO algorithms. This paper showed that the underlying concepts are also applicable to NBBO algorithms that make use of domain knowledge. The availability of domain knowledge in NBBO contributes to defining the relation space and class comparison statistic. Given sufficient domain knowledge the search for
relations may be trivial and the class comparison statistic may become accurate. On the other hand a BBO algorithm may require more involved search for appropriate relations. Unfortunately, most of the popular BBO algorithms like simulated annealing, genetic algorithms do not properly search for relations. Therefore, we need algorithms that are more constructively designed following the decomposition offered by SEARCH. Hopefully this work will also provide a systematic approach to introduce domain knowledge in optimization algorithms.

12 Acknowledgment

This work was supported by US. Department of Energy and AFSOR Grant F49620-94-1-0103. The author acknowledges many insightful suggestions, discussions, and help from David E. Goldberg.

References


DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.