List of Figures

List of Tables

A Perspective On The Foundation And Evolution Of The Linkage Learning Genetic Algorithms

H. Kargupta [a,1] S. Bandyopadhyay [b]

[a] *School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, hillol@eecs.wsu.edu*

[b] *Machine Intelligence Unit, Indian Statistical Institute, Calcutta, India, res9407@isical.ac.in*

Abstract

Intelligent guessing plays a critical role in the success and scalability of a non-enumerative optimization algorithm that primarily relies on the samples taken from the search space to guide the optimization process. Linkage learning deals with the issue of intelligent guessing by exploiting properties of the representation. This paper underscores the importance of linkage learning in genetic algorithms and other adaptive sampling-based optimization algorithms. It develops the foundation, identifies the problems of implicit linkage learning in simple genetic algorithms, reviews some of the early linkage learning efforts, reports some of the recent developments, and identifies the future directions of linkage learning research.

Key words: Linkage learning, messy GAs, fast messy GA, GEMGA

1 Introduction

Optimization deals with the problem of finding solution(s) from a given search space that extremizes the objective function value beyond a given desired level. The suitability of an optimization algorithm depends on the nature of the search space and the objective function. There exist many optimization techniques that are specially designed for classes of objective functions and the search spaces. For example, a gradient search based technique may require sufficient knowledge about the objective function in order to compute

[1] Supported by the US National Science Foundation Grant IIS-9803360.

the gradient; linear programming is designed for linear objective functions, subjected to linear constraints. These algorithms make use of their knowledge about the search space and the objective function for guiding the search directions. However, there also exist many practical problems for which such knowledge is unfortunately not available. In this paper we shall call them Black-Box Optimization (BBO) problems. If sufficient information to guide the search process is not available, the most obvious way to proceed is either to do an enumerative search or to search adaptively based on the samples taken from the search space. Since for most of the interesting problems enumerative search is too expensive, a sampling based adaptive approach is often the choice in practice. There exist many optimization algorithms like the Genetic Algorithms [31], Simulated Annealing [45], and Tabu search [17] that take the latter approach. This approach to BBO is essentially based on the process of guessing or hypothesis formation. Intelligent guessing requires both understanding of patterns from the data and efficient evaluation of guesses. This process plays a fundamental role in sampling based adaptive BBO and this paper addresses this issue in the context of Genetic Algorithms (GAs). Like all other sampling based adaptive BBO algorithms, efficient and intelligent guessing from the collected samples is critical for scalable performance of the GAs.

This paper points out that since sampling based adaptive guessing is fundamentally an inductive process, problem representation plays an important role. Linkage learning, the main theme of this paper, addresses the issue of efficient, intelligent guessing by exploiting the properties of the representation, in the context of genetic algorithms. The objective of this paper is to lay the foundation of linkage learning, offer a perspective of the evolution of linkage learning genetic algorithms over the last decade, and present some of the recent developments.

Section 2 presents a discussion on the role of guessing and induction in BBO. Section 3 gives a flavor of an abstraction of the concepts developed informally in Section 2. Section 4 defines linkage learning. Section 5 illustrates the developed concepts in the light of sequence representation, typically used in most of the evolutionary algorithms. Section 6 discusses the early linkage learning efforts in simple GA, describes their problems, and identifies the need for a well-designed, explicit approach toward linkage learning. Section 7 develops a Walsh representation based approach to understand the underlying mechanism of different algorithms, presented in the following sections. Sections 8—10 present different versions of the so called messy GAs that pay explicit attention to linkage learning. Section 8 describes the messy GA, its strengths, and its weaknesses. Section 9 does the same for the fast messy GA. Section 10 presents the recent developments of the gene expression messy GA. Section 11 discusses some of the recent developments and on going research in this area. Section 12 presents a general discussion regarding many common argu-

ments often raised regarding linkage learning. Finally, Section 13 concludes this paper.

2 Guessing, Induction, and BBO

Guessing in absence of complete knowledge is a common event of our everyday life. In this section we investigate the role of guessing in the context of black-box optimization (BBO). First, let us consider a simple situation that requires guessing. Say there is a large room full of people, possibly with money in their pocket, and we would like to identify the person with the highest amount of money. Let us also stipulate that we cannot search everyone's pocket; rather, we can only search a small fraction of the crowd.

One possible approach to address this problem is to randomly select individuals from the crowd and report the one with the highest amount of money. However, this approach is unlikely to be successful beyond the chance of random events. A more intelligent and possibly effective way to approach this problem is to try to detect possible patterns from the collected data; in other words, we try to identify a relation between the amount of money that a person has and some "intelligently" chosen set of features of that person. For example, we could consider the quality of the dress, the type of watch, and the hair-style as a set of features to determine the group of people who are likely to have a lot of money in their pocket. So, initially we select a possible set of such features to be considered, and then evaluate how good these features are individually, or may be together in groups, for identifying the class of people with lot of money. We may conclude that cheaply dressed people are unlikely to carry a lot of money; therefore the classes (cheaply-dressed people and well-dressed people) defined by the feature quality-of-dress are appropriate for identifying the person with the highest amount of money. Once we note that the class of well-dressed people is likely to contain the person that we are looking for, we can focus the search only on this class of people. Now we may note that a new feature, type-of-watch, can further classify the class of well-dressed people in a useful manner. We note that people who are well-dressed and wear an expensive watch are likely to have a lot of money. At the same time, the feature hair-style may turn out to be a bad choice and different hair-styles may have nothing to do with the money. So we choose to use the features quality-of-dress and type-of-watch together to define a small class of people who are likely to contain what we are looking for. As we see, intelligent guessing in the current context involves formation of hypotheses, consideration of the classes defined by the individual hypothesis, evaluation of hypotheses, and selection of appropriate hypotheses.

We consider hypotheses defined by the feature set, use it to divide the search

space into different classes, and evaluate hypotheses using samples taken from the search domain. The set of features to which we restrict our attention may be pre-determined or dynamically constructed during the course of the search. We started with three features; however, in addition to the individual features, we also noted that considering some of these features together may turn out to be quite useful. We can certainly argue that such subsets of features should be considered as a new feature itself. Therefore dynamic construction of features is a clear possibility. However, it is not difficult to imagine a situation when the number of such possible combinations of different features becomes enormous. For any n features there are 2^n different possible subsets of features. Since 2^n is an exponentially growing number, very soon we shall realize that consideration of all such combinations may not be achievable. Therefore, we need to restrict our scope somehow to a moderate set of hypotheses. We can now summarize the main steps of our intelligent scheme to find the person with the highest amount of money without checking everyone's pocket as follows:

(1) restrict the set of hypotheses to be considered to a moderate size;
(2) consider a hypothesis and divide the search domain into different classes using the hypothesis;
(3) take samples for evaluating the distribution properties of the classes;
(4) compare the hypotheses with each other, create an ordering, and select appropriate hypotheses;
(5) select the better classes defined by the chosen hypotheses for further exploration.

Although we can design different variations of the above scheme, the underlying processes for hypotheses formation, evaluation through the consideration of the class properties, and exploitation of good hypotheses for subsequent search remain invariant. This document emphasizes these fundamental processes of non-enumerative adaptive search.

The problem discussed above illustrates a typical situation in a BBO, where the objective to find the person with highest amount of money is replaced by a general computable objective function. The search space is a well-defined domain; in this paper we shall only consider a discrete search domain. The samples taken from the search domain are generated by the search operators of the BBO algorithm. Despite these superficial differences, the fundamental steps of intelligent guessing about the desired quality solution is essentially the same as what is delineated earlier in this section. Since guessing is essentially an inductive process, at this point it may be appropriate to make the concepts slightly more formal following the standard literature on induction [72] (note that we are referring to inductive learning not to mathematical induction).

The word hypothesis may mean different things in different contexts. There-

fore, let us choose a rather well-defined set-theoretic object called *relations*[2] [9] instead of the word hypothesis. In our previous example, different combinations of the features quality-of-dress, watch-type, and hair-style correspond to different relations. These relations divide the domain into different classes and the class properties are in turn evaluated by taking samples. This decomposition of BBO is illustrated in Figure 1. This figure uses similarity based equivalence relations[3] and classes for the relation and class space respectively. The search domain is the space of all four-bit binary strings. The SEARCH (Search Envisioned As Relation and Class Hierarchizing) framework proposed elsewhere [32, 39] makes use of this decomposition of BBO into (1) *relation*, (2) *class*, and (3) *sample* spaces. SEARCH emphasizes two main important underlying processes of BBO algorithms: (1) evaluations and construction of partial ordering, followed by selection of good relations, and (2) evaluation and construction of partial ordering, followed by selection of good classes. Note that we used the phrase 'partial ordering' since in general there may be cases when construction of a strict ordering among the relations and the classes is not possible.

In SEARCH, relations that are inherently good for decision making are said to *properly delineate* the search space. If we construct a partial ordering among the classes defined by a relation, select the 'top' ranked classes for further exploration, and the class containing the optimal solution is one among those selected classes, then we say that relation *properly delineates* the search space. The function 'top' is typically defined either implicitly or explicitly by the algorithm. Moreover, construction of the partial ordering among the classes requires a comparison statistic that is again provided by the chosen algorithm. Therefore, for a given class comparison statistic and a definition of 'top', we can identify a certain group of delineable relations that help identify the classes containing the desired quality solutions.

Earlier we noted that computation is manageable only when the set of all relations to be considered is moderate. Similar argument can be made in the class space too. Note that evaluation of a relation requires evaluation of the classes it defines. If a relation divides the search space into a very large number of classes, then evaluating those classes is going to be computationally expensive. Therefore, we need to consider only a moderate number of relations, and also only those relations that define a moderate number of classes. Let us introduce

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

[3] An equivalence relation is a relation that is reflexive, symmetric, and transitive. Equivalence classes are the classes defined by an equivalence relation. Similarity based equivalence relations among a space of binary sequences define equivalence based on the similarity among the sequences.

an index associated with the relations for representing the number of classes that it defines. Let us call a relation of order-k if k is the logarithm of the number of classes it defines. We would like to consider only those relations for which k is a small constant. Therefore, a BBO algorithm essentially needs to detect order-k delineable relations where k is a small constant.

This discussion points out that since induction is an essential part of BBO, a search for appropriate relations is critical. Instead of looking for better solutions from the beginning, an 'intelligent' BBO algorithm should

(1) first detect the structure of the search space, induce relations and classes to capture that, and then
(2) identify desired quality solutions by guiding the search following the detected structure.

The SEARCH framework captures this perspective in a formal manner. Appreciating linkage learning requires understanding the foundations of this perspective. Therefore, before addressing linkage learning, let us briefly overview this framework.

3 Overview Of SEARCH

This section presents a brief abstract overview of the decomposition of BBO, following the SEARCH framework. Figure 2 presents a process oriented view of the following major components of SEARCH:

(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

Each component is briefly discussed in more detail in the following. A relation is denoted by r_i, where i is the index of the set of all relations, Ψ_r, under consideration of the algorithm. Let C_i be the collection of the classes, 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 Ψ_r. Denote the members of C_i by $C_{1,i}, C_{2,i} \cdots C_{N_i,i}$, where N_i is the total number of classes in C_i.

Once the relation r_i is used to generate C_i the next step is to evaluate the classes in C_i. To do that we need samples from the domain of optimization. We assume that the BBO algorithm is equipped with at least one operator to generate new samples. This operator can be either a random sample generator

or a smarter one that exploits information from the prior relation, class, and sample evaluations.

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 $\mathcal{T}_c$ can be computed for each of the classes, and they may be compared based on this statistic. 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 $\mathcal{T}_c$, 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 $\mathcal{T}_c$ can be used to construct a partial order on C_i. Let us denote this partially ordered collection by $C_{i[\,]}$. Typically, BBO algorithms use either distribution dependent or distribution free statistics for comparing classes. A distribution dependent statistic typically uses mean and variance information for comparing classes. On the other hand, a distribution free approach like order statistics [10] uses properties of the orderings of class members. 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. Let $\mathrm{TOP}(C_{i[\,]}, M_i)$ be a function that returns the 'top' M_i classes of $C_{i[\,]}$. The exact choice of M_i is likely to depend 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. Choosing the parameter M_i is a responsibility of the BBO algorithm.

Next, this ordering among the classes is used to evaluate the relation r_i itself. Different kinds of statistics can be used to compare relations with one another. We denote this relation comparison statistic by $\mathcal{T}_r$ and call it a *relation comparison statistic*. 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. The choice of a good $\mathcal{T}_r$ is not quite obvious. This is one among the main issues that this paper considers.

Not all the relations are appropriate for a given BBO problem. A relation is not appropriate with respect to the chosen $\mathcal{T}_c$ and the BBO problem if the class $(C_{*,i})$ containing the desired 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. Let us define a function $DC(r_i, \mathcal{T}, M_i)$ that returns a one if $C_{*,i} \in \mathrm{TOP}(C_{i[\,]}, M_i)$; otherwise, it returns a zero. For a given BBO problem, a relation r_i, a class comparison

statistic $\mathcal{T}_c$, and a parameter M_i, if $DC(r_i, \mathcal{T}_c, M_i) = 1$, we say that r_i properly delineates the search space. If all the properly delineating relations needed to solve the problem in polynomial time (along problem size, quality of solution and reliability of the success probability) are in the given relation space Ψ_r, then we call the *problem delineable* with respect to Ψ_r. Recall from the previous section that having all the required delineable relations may not be sufficient since we can only evaluate a polynomial number of classes in polynomial time. Therefore, we need the problem to be order-k delineable.

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 were found to be bad earlier can be used to eliminate some classes in r_0 from further consideration. Black-box algorithms often implement a resolution-like process to take advantage of any such possible decomposability. 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 whether or not the problem is delineable, success in finding proper relations, better classes, and sufficient sampling. A detailed description of each of these processes can be found elsewhere [32].

As noted earlier, choice of a good relation comparison statistic is not quite obvious. This question goes to the heart of the so called linkage learning problem in genetic algorithms. The following section initiates the discussion.

4 Relation Evaluation And Linkage Learning

This paper considers the issue of relation evaluation and comparison for problems that are order-k delineable with respect to the given relation space. Since the problem is order-k delineable the objective of the search in the relation space is to detect relations that are very likely to properly delineate the search space.

Detecting delineable relations requires first defining a good measure to evaluate relations. The SEARCH framework assumes existence of such a measure; it does not, however, provide us a good measure that can be used to evaluate relations. Given a set of relations, their respective classes, and a sample set, we need to identify an effective way to decide whether or not the relation is order-k delineable. This problem is traditionally called the *linkage learning* problem in the genetic algorithm literature. Although linkage learning is a fundamental task for every inductive optimization algorithm independent of the relation space considered, typically GAs are designed to consider similarity based equivalence relations in a sequence representation. Therefore, the rest of this paper will focus on this special case. The following section describes this special case in further details.

5 Illustration: Sequence Representation and Similarity Based Equivalence Relations

Consider an ℓ-bit binary sequence representation similar to the case shown in Figure 1. Let us consider Similarity Based Equivalence Relations (SBERs) in this sequence space. There exist 2^ℓ such SBERs. Clearly this is exponential in ℓ and as a result we cannot consider all of them in a reasonable time for large values of ℓ. We need to somehow choose a polynomial number of relations from them; moreover, the every chosen relation must have an order value less than or equal to some constant k. One way to do that is to restrict the number of f's (i.e. the positions of similarity based equivalence) to at most k. This is because an SBER with k positions of similarity based equivalence defines exactly 2^k different classes. For example, f### (f denotes position of equivalence, and the # character matches with any binary value) divides the space 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. Similarly ff## defines classes 00##, 10##, 01##, and 11##. SBERs in the sequence space and the corresponding classes are sometimes called partitions and schemata respectively in the genetic algorithm literature. Therefore, our relation space is now comprised of order-k relations. The objective of the search in the relation space is essentially the search for detecting the delineable ones from the set of $\sum_{i=0}^{i=k} \binom{\ell}{i}$ relations. Any member of this relation space has at most 2^k classes.

The rest of this paper explores linkage learning for order-k delineable problems with respect to a relation space defined by SBERs in sequence space. However, before moving any further we would like to explain the rationale behind choosing this special case.

First of all, representations are typically not chosen randomly. In case of our

toy problem of finding the person in the crowd with lot of money, we started with a sensible set of features. For most of the real life cases, the chosen representation is often much better than a randomly chosen one. Although there may exist interesting problems that are not delineable with respect to their most obvious representation, we believe that we should take one step at a time—first develop accurate algorithms for delineable problems and then explore representation construction. We also assume that orders of the required delineable relations are bounded by some constant k. This has a similar rationale. Most of the real world problems do not exhibit complete non-linearity where every feature non-linearly interacts with every other feature. Typically, non-linearity is bounded; in other words only a relatively small number of features (with respect to ℓ) interact with each other. Simon's article on the *architecture of complexity* [67] offers a stimulating discussion on this issue. Although delineability and non-linearity are two separate but related issues, bounded non-linearity in many practical problems often suffices considering relations, defined by only a bounded number of features together.

The next issue is regarding the special case where the relation space is comprised on only SBERs. Although SBERS are certainly simple in nature, they are of popular choice for many inductive learning algorithms such as Decision trees [64], Version space algorithms [56] and others [55]. However, it is easy to construct a problem where more complex relations are needed to capture certain subsets of the search space. Given their popularity, simplicity, and historical role in the development of GAs, the authors believe that using the SBER based relation space is a good choice as a starting point for designing and testing linkage learning algorithms.

Although the relation space contains 2^ℓ SBERs, the order-k delineability assumptions restrict the number of effectively considered relations to a number polynomial in ℓ. Clearly we can come up with a polynomial time enumerative algorithm to evaluate relations, provided we have a good measure to do so. The objective of the rest of this paper is to offer a perspective on the efforts to develop good relation evaluation measures and efficient polynomial time algorithms that make use of them for detecting good relations. The importance of linkage learning has been realized since the dawn of the GA research. The following section describes the early work in simple GA [11, 20] based linkage learning.

6 Linkage Learning In Simple GA

The simple GA (sGA) does not explicitly process the partitions (SBERs) and schemata (classes). Therefore in sGA, there is no explicit effort to detect significant partitions. However, the design of the GAs has been traditionally

motivated by different aspects of the partition and schema processing [31]. Nevertheless, the efficacy of the implicit processing has been questioned since the inception of the GAs.

Several efforts have been made for designing simple GAs that try to detect significant partitions and schemata. The history of linkage learning efforts dates back to Bagley's dissertation [2]. Bagley used a representation in which the gene explicitly contains both the position and the allele value. For example, string ((0 1)(2 0)(1 1)) will correspond to the string 110 in a fixed-locus representation of the simple GA. Bagley used the so called *inversion* operator for adaptively clustering the related genes that define good partitions and schemata. The inversion operator works by reversing the order of the genes lying in between a pair of randomly chosen points along the chromosome. Although this mechanism was intended to generate new tightly coded partitions, Bagley's work provided no mechanism for accurate evaluation of the partitions. Moreover, introduction of the inversion operator restricted the use of GA crossover operator and Bagley did not conclude in favor of the use of inversion. Rosenberg [65] also investigated the possibility of learning linkage by evolving the probability of choosing a location for crossover. Although this approach does not rigorously search for appropriate partitions, adaptive crossover point may be able to process schemata, with widely separated fixed bits, better than a single point crossover. Frantz [15] investigated the utility of the inversion operator and like Rosenberg reported that inversion is too slow and not very effective. Holland [31] also realized the role of linkage learning and suggested the use of inversion operator despite its reported failure in earlier studies. Goldberg and Lingle [27] introduced a new PMX crossover operator that could combine the ordering information of the selected regions of the parent chromosomes. They concluded that this approach has more potential than the earlier approaches. Schaffer and Morishima [66] introduced a set of flags in the representation. These flags were used for identifying the set of genes to be used for crossover points. For different test problems, they noted the formation of certain favorite crossover points in the population, that corroborated their hypothesis regarding the need for detecting gene linkage. Goldberg and Bridges [22] confirmed that lack of linkage knowledge can lead to failure of GAs for difficult classes of problems, such as deceptive problems. Additional efforts on linkage learning GAs can be found elsewhere [47, 62].

In addition to the growing empirical evidence of the need for well-designed explicit linkage learning algorithms in the GAs, theoretical advancements have also started corroborating these observations. Efficacy of such implicit processing of relations has been seriously questioned on theoretical grounds elsewhere [26, 32, 70]. Thierens and Goldberg [70] showed that simple GA fails to scale up for the class of problems with only order-k significant partitions, unless information about the appropriate partitions is provided by the user.

Among others, merger of the distinct decision making processes in relation, class, and sample spaces into a single selection process over the population and the lack of adequate efforts to methodically search for the appropriate order-k partitions makes the SGA less scalable. The SGA also has some additional problems in the context of efficient partition search. A single sample from the search space can be used for the evaluation of all the relations under consideration. This is because that sample must belong to some schema defined by any partition. This is often called *implicit parallelism* in the GA literature. Although this can be exploited in a very systematic manner when relations are methodically processed, implicit processing of schemata makes this quite noisy in the sGA. These observations regarding the problems of simple GA in searching appropriate partitions and schemata resulted in the development of a new class of genetic algorithms that explicitly search for the delineable relations and classes. However, before discussing these algorithms we would like to develop a Walsh representation based perspective of partition evaluation. The following section discusses this.

7 Linkage Learning, And Walsh representation

As we noted in the previous section, implicit detection of delineable partitions through the simple GA selection does not appear to be a scalable solution to difficult linkage learning problem. However, explicit detection of linkage requires an appropriate measure for detecting linkage. It turns out that, although we cannot determine delineability of relation without the knowledge of the desired solution, we can still define a quite accurate measure to construct a partial ordering among the partitions based on their contribution to the objective function value. This can be done in a straight forward manner by representing the objective function using a set of suitable basis functions. Walsh representation [5] provides one possible way to do so for SBERs in sequence representation. In the Walsh representation, the objective function can be viewed as a linear combination of the fitness contributions from the different partitions. This provides a nice approach for associating a notion of "significance" with the different partitions. The absolute magnitude of the individual partition contribution to the objective function can be used for evaluating and comparing the partitions.

Although a linkage learning algorithm does not necessarily have to use Walsh representation for evaluating relations, in the rest of this paper we shall use this approach to understand the underlying mechanism of different explicit linkage learning genetic algorithms. We shall do so because the authors believe that Walsh representation can serve this purpose well for SBERs in a sequence representation, at least until a better approach is developed. The following section offers a brief review of the Walsh representation. This will be

followed by a discussion on the suitability of this chosen approach to compare partitions.

7.1 Brief review of Walsh representation

Walsh functions [5] are orthogonal functions that found applications in many different fields such as signal processing, image analysis, and others. Like Fourier, Laplace, and other transformations, Walsh functions are often used to transform the representation into a convenient form. Application of Walsh transformation (WT) in understanding Genetic Algorithms was first noted by Bethke [7]. Further investigation of this approach can be found elsewhere [14, 18, 19, 48]. Traditionally the Walsh functions are designed for binary sequences. However, they can easily be extended to higher cardinality representation, as shown elsewhere [61]. Therefore all the arguments to be made in the coming sections using WT can be extended for higher cardinality representations. For a string ($\overline{x}$) of ℓ binary variables, WT makes use of 2^ℓ Walsh functions as a basis set, where each basis function corresponds to a unique partition $\overline{j}$. They can be defined as follows:

$$\psi_{\overline{j}}(\overline{x}) = (-1)^{(\overline{x}.\overline{j})} \tag{1}$$

Where $\overline{j}$ and $\overline{x}$ are binary strings of length ℓ. In other words $\overline{j} = j_1, j_2, \cdots j_\ell$ and $\overline{x} = x_1, x_2, \cdots x_\ell$. $\psi_{\overline{j}}(\overline{x})$ can either be 1 or -1. A function $f(\overline{x})$ can be written using the Walsh basis functions as follows:

$$f(\overline{x}) = \sum_{\overline{j}} w_{\overline{j}} \psi_{\overline{j}}(\overline{x}) \tag{2}$$

where $w_{\overline{j}}$ is the Walsh Coefficient (WC) corresponding to the partition $\overline{j}$ as defined in the following,

$$w_{\overline{j}} = \sum_{\overline{x}} f(\overline{x}) \psi_{\overline{j}}(\overline{x}) \tag{3}$$

We note from Equation 2 that the fitness function can be expressed as a linear sum of the Walsh functions, each weighted by the corresponding Walsh coefficients. The Walsh coefficient $w_{\overline{j}}$ can be viewed as the relative contribution of the partition $\overline{j}$ to the function value of $f(\overline{x})$. Therefore, the absolute value of $w_{\overline{j}}$ can be used as the "significance" of the corresponding partition $\overline{j}$. The following section explains this proposition.

The absolute value of the Walsh function reflects the relative importance of a partition with respect to others in terms of its contribution to the objective function. Consider an ℓ-bit maximization problem and a partition $\overline{j}$. Without loss of generality let us assume that $w_{\overline{j}}$ is a positive number. Let Ω_j be the set of all partitions that involve at least one variable used in the partition $\overline{j}$. Let $C_{\overline{j}}^{(+)}$ and $C_{\overline{j}}^{(-)}$ be the collections of schemata for which the Walsh function $\psi_{\overline{j}}(\overline{x})$ takes a value of 1 and -1 respectively. Now if we take a member of any of the classes in $C_{\overline{j}}^{(-)}$ and modify its values only over the fixed positions of the partition $\overline{j}$ in such a way that it becomes a member of one of the classes in $C_{\overline{j}}^{(+)}$, the overall objective function value may be increased by an amount $2w_{\overline{j}}$ if the change in feature values does not cause some other partitions in Ω_j to decrease the objective function value. However, in general, fixing the values of partition $\overline{j}$ may restrict the choice of feature values in some other partitions in $\Gamma_j \subseteq \Omega_j$ and as a result partitions in Γ_j may increase or decrease the objective function value. The overall change in objective function value because of the changes in partition $\overline{j}$ can therefore be computed as $\eta_{\overline{j}} = 2(w_{\overline{j}} + \sum_{\overline{k} \in \Gamma_j} w_{\overline{k}} \psi_{\overline{k}}(.))$. By no means does the above discussion claim that the suggested relation measure can be efficiently computed and used for constructing a partial ordering in the relations for any general class of objective functions without adopting any approximation. Selection of a few (polynomialy bounded) good relations requires construction of a partial ordering among the relations. In other words, we should be able to quickly identify the relations that significantly contribute to the objective function and discard the rest. This fundamentally means that there exists a large number of relations whose contribution to the objective function is negligible. The order-k delineability property satisfies this requirement. Clearly, this property of the representation depends on the choice of the basis functions to define the relations, and Walsh representation is unlikely to be a universally good choice of basis for representing the relations. However, we believe that Walsh representation works quite well for understanding the linkage learning of SBERs.

Partitions with non-zero Walsh coefficients also reflect the underlying non-linearity of the given problem. For example, consider an objective function $f(x_1, x_2, x_3, x_4) = f_1(x_1, x_2) + f_2(x_3, x_4)$. In the Walsh representation the value of $w_{(j_1,j_2,j_3,j_4)}$ and $w_{(j_1,j_2,j_3)}$ and any such other Walsh coefficient corresponding to partitions involving a pair of variables, one each from the two linearly decomposable partitions, is zero. However, note that having $w_{(j_1,j_2,j_3,j_4)} = 0$ does not necessarily mean that x_1, x_2, x_3, x_4 there does not exist any low order interaction among these variables. For example, if the WCs coefficients corresponding to every order-2 partitions have a non-zero, significant WC, then fixing the value for some (x_i, x_j) is going to effect every other partition

(x_i, x_k) and (x_j, x_m). As result, we may have to consider higher-order partitions, subsuming overlapping order-2 partitions. An example of this case can be found elsewhere [21], that constructed a class of order-ℓ deceptive problems using only up to order-3 non-zero WCs. However, this by no means suggests that partitions, higher than order-3 but less then order-k are not needed to be considered for learning linkage information. This is because non-linear interactions among more than three variables can always be contributed by non-zero WCs for higher order partitions.

The following section describes the first effort to develop the so called messy GA that tries to detect linkage in an explicit manner.

8 The Messy GA: Early Efforts

The messy GA (mGA) [12, 26, 24, 25] is one among the few early efforts that was specifically developed for linkage learning. The mGA took at least two important steps:

(1) Separated the partition and schema spaces from the sample space and thereby paid explicit attention to the partition and schema processing.
(2) Focused on only the order-k delineable problems.

The mGA uses a population that contained all (deterministically enumerated) order-k schemata defined by the chosen representation. So the population size was $\binom{\ell}{k}2^k$, where ℓ is the problem length. The population is cleverly used to represent the partition and schema spaces during the initial stage of the algorithm. It gradually switches the population to the sample space during the following stages. This switching process will be more obvious once we discuss the messy representation later in the section. The search process is distinctly divided into two stages, namely:

- Primordial stage: Detects appropriate partitions and schemata; population represents the partition and schema spaces.
- Juxtapositional stage: Computes intersection among the better schemata to find the optimal solutions; once the good partitions and schemata are detected, population gradually switches to the mode representing the sample space.

The complexity of the mGA is $O(2^k \ell^k)$. The following section describes the mGA representation.

8.1 Messy representation

The mGA uses a richer representation compared to sGA in order to learn the appropriate partitions and also to be able to use the population for representing both the schema and sample space. Just like Bagley's GA representation [2], the mGA gene is an ordered pair, (*locus, value*). The *locus* gives the actual position of the gene in the decoded string and the *value* is any letter from the alphabet set.

The messy GA strings can be under-specified, exactly specified or over-specified. For example, in a 3-bit problem the string ((0 1)(2 0)(1 1)(3 1)(2 1)) is over-specified, the string ((0 1)(2 1)(3 0)) is exactly specified, and the string ((0 1)(2 1)) is under-specified. Over-specified strings are mapped to a string containing ℓ unique genes by left-to-right scanning of the strings on the basis of first-come first-served preference. The over-specified and exactly specified strings are samples from the search space. On the other hand, an under-specified string with k unique genes defines a schema of order-k. In mGA, partitions are evaluated based on the quality of the schemata it defines. The schemata defined by the under-specified strings are evaluated using a local search template string. This template is a locally optimal string that remains unchanged during a particular iteration of the messy GA. The template is always exactly specified. The missing genes of an under-specified string are filled in by the template. The incompletely specified string ((0 1)(2 1)) produces the string ((0 1)(1 0)(2 1)) once it is expressed in the context of the template ((0 1)(1 0)(2 0)).

8.2 Messy operators

The messy GA uses two main operators: (1) *thresholding selection* and (2) *the cut and splice* operator. The following sections describe these operators.

8.2.1 Thresholding selection

As noted earlier, the mGA population represents the complete order-k schema space during the primordial stage. However, there must be a mechanism to pay attention to the partitions for making sure that the schemata from different partitions are not compared to each other. Since selection is the primary mechanism for comparing different schemata, the mGA imposes certain restrictions on the selection operator. It introduces an operator called *thresholding selection* that tries to minimize selective competition among schemata that do not share more than a certain threshold number (called the *thresholding parameter*) of genes with same *locus*. Consider the strings ((1 0)(0 0)), ((1 1)(0 1)),

and ((1 0)(2 1)). The first two strings define equivalence classes 00#, 11# over the relation $ff\#$. On the other hand, the last string defines the class #01 over the relation $\#ff$. Clearly, comparing the first two makes sense, because they are from the same relation; however, the last string must be restricted from competing with the other two strings, since it belongs to a different relation. The thresholding selection avoids such competition to some extent. Typical mGA implementations use thresholding based tournament selection [8, 26]. The following section describes the mGA cut and splice operator.

8.2.2 Cut and Splice operation

The cut and splice operation simulates the behavior of crossover for strings of different lengths. Consider the strings ((1 1)(0 1)(2 0)) and ((2 1)(1 0)(0 1)(2 1)). The cut operation randomly picks two points, one for each string. Let us say that it picks 2 and 3 for the first and second string, respectively. The cut operation then splits the first string into ((1 1)(0 1)) and ((2 0)). The second string is also divided into the strings ((2 1)(1 0)(0 1)) and ((2 1)). The splice operation swaps the split parts and generates new strings. In the current example, splice operation generates the strings ((1 1)(0 1)(2 1)) and ((2 1)(1 0)(0 1)(2 0)). The following section describes the overall organization of the mGA.

8.3 Organization of the messy GA

The messy GA works by iterating within two loops—the outer and inner loops. The variable of the outer loop is the order of the partitions under consideration. The inner loop searches for appropriate schemata of order defined by the outer loop variable and produces solutions by combining these schemata using the cut and splice operations. The overall mechanism is described in the following.

- *Outer loop begins:* The outer loop iterates over the order of the partition κ. At the initial iteration of the outer loop, κ may be chosen as 1 if no other prior information is available. Initially, the template is randomly generated.
- *Inner loop:*

(1) *Initialization:* The population is initialized deterministically with strings of length κ. It contains all the $2^{\kappa}\binom{\ell}{\kappa}$ order-κ schemata. The objective function values of all these strings are evaluated by first filling up the missing genes from the template.

(2) *Primordial phase:* The primordial phase applies thresholding selection alone for detecting the appropriate schemata. Once the good schemata are detected by applying selection for certain number of generations, this phase stops.

(3) *Juxtapositional phase:* During this phase both thresholding selection and cut-&-splice operators are used. Good strings are picked, cut, and then spliced to generate better strings. As string lengths continued to grow and become either exactly specified or over-specified strings, the population accordingly started to represent more the sample space, rather than the schema space.
(4) *Inner loop ends:* The template is set to the best solution found at the end of the juxtapositional stage of the previous iteration of the inner loop.

- *Outer loop ends.* The algorithm stops when the order of partitions considered exceeds a certain value or some other stopping criterion is satisfied.

The mGA organization was very different from the previous linkage learning efforts in the sense that it decomposed the search into two explicit stages, one for detecting better schemata and the other for exploiting them for searching the problem domain. Although the mGA did not explicitly use the Walsh framework to detect the significant partitions, interpreting the mGA schema evaluation in terms of the Walsh representation may provide us useful insight. Let us consider evaluation of schemata defined over a certain partition $j_{(p)}$. Let us write all the partitions that do not involve any variable contained in the partition $j_{(p)}$ in a column matrix form and denote it by the symbol $W_{\overline{p}}$. Let W_p be the column matrix of all other Walsh coefficients that are not in $W_{\overline{p}}$. We can now write the "goodness" of a schema H_1 evaluated in the context of a template (G) as,

$$f(H_1, G) = W_p^T \Psi_p(H_1, G) + W_{\overline{p}}^T \Psi_{\overline{p}}(H_1, G) \tag{4}$$

where Ψ_p and $\Psi_{\overline{p}}$ are the corresponding column matrices of the Walsh functions. The term (H_1, G) denote the string generated by schema H_1 with the rest of the genes filled using the template G. Let us now consider the expression of another competing schema, H_2, defined over the same partition $j_{(p)}$. Since both H_1 and H_2 are defined over the same partition, only the sign of Ψ_p changes. Therefore

$$f(H_1, G) - f(H_2, G) = W_p^T [\Psi_p(H_1, G) - \Psi_p(H_2, G)] \tag{5}$$

The thresholding selection operator of the mGA essentially makes a decision based on this difference when schemata from the same partition are compared to each other. If the expression on the right side of the equation 5 is greater than zero, schema H_1 is considered better than H_1. Although this does not tell us specifically about the significance of the specific partition $j_{(p)}$, it tells us the contribution of all the partitions that involve variables from $j_{(p)}$. Schemata that effectively manipulate the Walsh functions for optimizing the objective functions grow during the primordial stage. If none of the partitions involving

any variable from $j_{(p)}$ has significant Walsh coefficients then the term W_p^T of equation 5 will also be insignificant compared to that of the significant partitions. However, note that since the Walsh coefficients may have different signs, the mGA schema evaluation will mistakenly conclude a W_p to be insignificant if the large WC-s cancel themselves out.

Several efforts have been made to extend this work on both abstract and application grounds. Merkle [50] developed a parallel implementation of original messy GA. He [54] also addressed data distribution strategies for the parallel implementation of mGA. Additional work on parallel mGA can be found elsewhere [16]. Although the population size in the primordial stage grows polynomially with problem size ℓ, $O(\ell^k)$ is a fairly large number for any reasonable value of k. Plevyak [63] investigated the possibility of smaller population size in the primordial stage. Mohan [57] applied mGAs for clustering. A hierarchical controller based on messy GAs is reported elsewhere [30]. Whitley [73] reported an application of mGA for feature subset selection.

However, the mGA have some problems as described in the following:

(1) *Template based schema evaluation:* The mGA used the template guided schema fitness for comparing classes and relations. As noted earlier, this template based evaluation does not necessarily correctly tell us about the significant partitions and it can be misleading.
(2) *Expensive enumerative search for schemata:* The explicit enumeration of all order-k schemata is very expensive ($O(\lambda^k \ell^k)$).
(3) *Lack of mechanism for exploiting implicit parallelism:* Since different relations simply divide the sample space in different classes, the same sample set can be used to evaluate different relations. Traditionally in the GA literature, it is called implicit parallelism. The mGA does not exploit the computational benefits of implicit parallelism.

The following section describes a major extension of the mGA that tries to address the second and third problems listed above.

9 The Fast Messy GA

The fast messy GA (fmGA) proposed elsewhere [23, 32] made an effort to reduce the cost of deterministic initialization by using the so called *probabilistically complete initialization* (PCI) and *building-block filtering* (BBF) techniques. Each of them is briefly described in the following sections.

9.1 Probabilistically complete initialization

The basic idea behind the probabilistically complete initialization is that all the $2^k\binom{\ell}{k}$ schemata can be defined using a much smaller number of strings, when the string length is higher than k. In other words, strings represent multiple schemata over different partitions at the same time. The number of ways a string of size $\ell' > k$ contains a gene combination of size k may be calculated by assigning k genes to the string and then choosing the total number ways $\ell' - k$ genes can be created from $\ell - k$ genes. This is simply $\binom{\ell-k}{\ell'-k}$. Note that the total number of strings of size ℓ' created with ℓ genes is $\binom{\ell}{\ell'}$. Thus if we take $n_g = \binom{\ell}{\ell'}/\binom{\ell-k}{\ell'-k}$ string samples each of length ℓ' randomly, on average there will be one string that will have the desired gene combination of size k; n_g decreases exponentially as the string length ℓ' increases. Further details can be found elsewhere [23].

9.2 Building-Block filtering

During the primordial phase of fmGA, thresholding selection increases the proportion of the better strings. However, in addition to that, the string length needs to be gradually reduced from length ℓ' to k. Gradual reduction of string length is accomplished by increasing the number of copies of the strings by applying thresholding selection, followed by random deletion of genes. This process of detecting the good schemata by thresholding selection and gene deletion is called building-block filtering. Consider the sequences of string lengths generated by successive applications of gene deletion, denoted by $\lambda^{(0)}, \lambda^{(1)}, \cdots \lambda^{(i)} \cdots \lambda^{(\mathcal{N})}$. The initial string length $\lambda^{(0)} = \ell'$ and the final string length $\lambda^{(\mathcal{N})}$ are chosen to be some number close to k. Selection continues for some number of generations with constant string length $\lambda^{(i)}$ to produce more copies of the better strings. Note that these are selection-only generations and therefore no new function evaluation is needed. This is followed by random deletion of genes which reduce the string length to $\lambda^{(i+1)}$. These shorter strings are then evaluated and the same process of thresholding selection and gene deletion continues until $\lambda^{(i)} = \lambda^{(\mathcal{N})}$. Since gene deletion is applied only after a certain number of applications of the thresholding selection, the exact schedule for string reduction needs to be carefully designed. Kargupta [32] proposed one way to do that. However, this process turned out to be somewhat unstable and success depended on the tuning of the schedule. In order to overcome this problem, Kargupta [32] proposed an iterative process that applied the fmGA multiple times for every order in order to assure better performance. The following section presents the overall organization.

9.3 Organization of the fmGA

Like the original messy GA, the fmGA goes through a level-wise processing of schemata. The selection-only primordial stage is replaced by a primordial stage that uses thresholding selection and gene deletion for filtering out the good schemata along with the probabilistic initialization of population. The working of the fmGA is very similar to that of the original messy GA. The main differences of the fmGA are the following: (1) probabilistically complete initialization, (2) gradual reduction of string length by random deletion of genes during primordial stage, and (3) multiple iteration within the each level. The following section presents some experimental results.

9.4 Experimental results

The fmGA was tested on different classes of problems [32]. This paper reports only a small fraction of the results. It reports the performance for boundedly deceptive problems that require only k-bit interactions using the modified BBF scheduling technique proposed elsewhere [32]. The deceptive trap [1] function is defined as, $f(x) = k$ if $u = k$; $f(x) = k - 1 - u$ otherwise; where u is the unitation variable, or the number of 1-s in the string x, and k is the length of the sub-function. This function is widely reported to be difficult for simple GA since low order partitions lead sGA toward the wrong direction. If we carefully observe this trap function, we shall note that it has two peaks. One of them corresponds to the string with all 1-s and the other is the string with all 0-s. For $\ell = 200$, and $k = 5$, the overall function contains 40 sub-functions; therefore, an order-5 bounded 200-bit problem has 2^{40} local optima, and among them, only one is globally optimal. As the problem length increases, the number of local optima exponentially increases.

Problems comprised of order-five trap functions are constructed by concatenating non-overlapping order-5 sub-functions. Figures 3 and 4 show the performance of the fmGA for 100-bit and 150-bit problems respectively. The population is kept to 7,500 and 8,500 respectively. This was done following the population sizing equation for fmGA described elsewhere [23, 32]. The total number of function evaluations for the 100-bit and 150-bit problems are $100,5000$ and $425,000$ respectively. Note that the number of function evaluations for the 100-bit problem is larger than that for the 150-bit problem because in the former case, the fmGA found the best solution after several iterations; on the other hand, the fmGA could not improve the best solution of the first iteration using additional iterations. Since these are relatively big problems and population sizes are quite large, only order-5 level of the fmGA is run with a locally optimal template in order to reduce the computation time.

The following section summarizes the conceptual strengths and weakness of the fmGA.

9.5 Strengths and weaknesses of the fmGA

The fmGA replaced the enumerative initialization ($O(\ell^k)$) of the original messy GA by an $O(\ell)$ probabilistically complete initialization. Since the building-block filtering schedule can have at most $O(\ell)$ steps, the overall sample complexity of the primordial phase is $O(\ell^2)$. This computational benefit is fundamentally based on the fact that the fmGA exploits the power of implicit parallelism, defined earlier. This is the main conceptual contribution of the fmGA.

However, the fmGA has some weak points too. The main problem is that the fmGA assumes that the thresholding selection is perfectly capable of restricting the schema competition to only those that belong to a single partition. It turns out that thresholding selection cannot maintain that reliably for the period of time needed to reduce the string length to $O(k)$ from ℓ' [32]. Although Kargupta [32] suggested a multiple iteration based approach to reduce this effect, this resulted in an increased number of function evaluations. So, although the population sizing in fmGA is drastically reduced by using strings of length $O(\ell)$, the detection of schemata from the strings using thresholding selection and gene deletion appears to be difficult. Merkle [51] has proposed a framework to analyze fmGA-like linkage learning algorithms that use tournament selection with thresholding, and building-block filtering. He posed the problem of selecting BBF filtering schedule and other parameters as a constrained optimization problem in which the objective function is directly related to the expected effectiveness. He also derived the Kuhn-Tucker conditions for the optimality of a parameter set of a generalized version of the fmGA. Parallel implementations of the fmGA have also been developed and discussed elsewhere [52, 53].

The fmGA can also be understood in the light of Walsh representation. The fmGA uses an initial string of length close to the problem length. Therefore the contribution of the templates is initially minimal and, as the BBF process continues, the role of the template in schema evaluation continues to increase. Since the string length is greater than that of the schema of length k (as in the case of mGA), the size of $W_{\bar{p}}^T$ decreases and that of W_p^T increases. However for any pair of two different strings the number of Walsh functions that take different value is exactly $2^{\ell-1}$. Therefore the total number of non-zero terms in equation 5 remains the same. As the string length gets reduced by gene deletion, different subsets of elements in the Walsh difference matrix (right hand side of equation 5) take non-zero values. As a result the fitness of a

schema, embedded in a string, changes. Clearly, this results in evaluation of a schema based on different members of the schema; however, it is not clear how this approach can be used for more precise information regarding significant partitions.

The following section describes a gene expression based messy GA that keeps the string length to the problem size ℓ; however it replaces fmGA schema detection technique by a more efficient and scalable technique.

10 Gene Expression Based Messy GAs

The recent past has witnessed a series of efforts on the development of a new class of messy GAs, called the Gene Expression Messy Genetic Algorithm (GEMGA) [4, 34, 40, 35, 33, 36, 37, 38, 42]. The GEMGA tries to detect significant partitions using a scalable approach motivated by the natural process of gene expression. This paper reports the latest version of the GEMGA described in detail elsewhere [4, 37, 38, 42].

10.1 Population sizing and representation

In order to detect a schema, the GEMGA requires that the population contain at least one instance of that schema. The population size in GEMGA is therefore, $m = c\lambda^k$, where c is a constant and λ is the alphabet size. Although we treat c as a constant, c is likely to depend on the variation of fitness values of the members of the schema. Note that the population size is independent of the problem size ℓ. For all the experiments reported in this paper, the population size is kept constant.

In GMEGA, the strings are always exactly specified. A GEMGA gene is a data structure that contains the *locus*, *value*, and *capacity*. The *capacity* field is used facilitating the schema detection process in the GEMGA. The chromosome also contains a dynamic list of lists called the *linkage set*. It is a list of weighted lists. Each member of this collection of lists, called *locuslist*, defines a set of genes that are related. Each *locuslist* also contains three factors, the *weight*, *goodness*, and *trials*. The weight is a measure of the number of times that the genes in *locuslist* are found to be related in the population. The *goodness* value indicates how good the linkage of the genes is in terms of its contribution to the fitness. The *trial* field indicates the number of times the linkage set has been tried.

10.2 Linkage learning in the GEMGA

Linkage learning in the GEMGA is accomplished using three processes, namely: (1) *Transcription* and (2) *PreRecombinationExpression*, and (3) *RecombinationExpression*.

The GEMGA *Transcription* operator detects *local* symmetry in the fitness landscape by noting the relative invariance of the fitness values of chromosomes under transformations that change the value of one dimension, one at a time. It changes the current value of a gene to a different value, randomly chosen from the alphabet set, and notes the change in fitness value. If the fitness deteriorates because of the change in gene value, that gene is identified as the symmetry breaking dimension. On the other hand, if the fitness improves or does not change at all, the gene is marked as a symmetry preserving [4] dimension. Finally, the value of that gene is set to the original value and the fitness of the chromosome is set to the original fitness. This process continues for all the genes and finally all the genes that are tentatively marked as symmetry breaker are collected in one set, called the initial linkage set. The mechanism of the transcription operator can be understood in terms of the WCs. Since any two unique binary strings have exactly $2^{\ell-1}$ Walsh functions with different values, identifying the contribution of half of all the Walsh coefficients requires only one bit difference between the strings. In other words, if we subtract the fitnesses of two strings X_1 and X_2 (where they differ only in one position) the difference gives us the contribution of all partitions involving that position. The GEMGA explicitly notes the fitness difference by changing the value only along that dimension. If $(f(X_1) - f(X_2))$ is very close to zero, we can neglect the contribution of all the Walsh coefficients in the matrix W_p to the fitness of string X_1. If $(f(X_1) - f(X_2))$ is a large number, then W_p significantly contributes to the fitness value of the string. This partitions can be implicitly listed to be significant by simply marking the dimension under observation. The GEMGA only marks the fitness symmetry breaking dimensions; therefore, only those dimensions that decrease the fitness upon changing the value in only one dimension are marked. Since this information is imprecise, the GEMGA adopts two other steps for identifying the significant partitions precisely.

The PreRecombinationExpression finds the significant partitions and schemata more precisely by collecting population wide statistics. An $\ell \times \ell$ conditional probability matrix is formed by collecting initial linkage set information from

[4] Increase in fitness for maximization problem is not considered "symmetry-breaking" since it does not contradict the objective of the optimization. Moreover, we choose to use the word symmetry since its definition directly alludes to transformations of the state.

different randomly selected chromosomes of the population. The i, j-th entry of this matrix indicates the probability of the occurrence of gene i, when gene j is present in a linkage set. This matrix is usually constructed using a user given *NoOfLinkageExpt* times. For each row i of the conditional matrix, its maximum value is computed, and the genes that have their probability values close to the maximum value are included in the linkage set for i.

After the PreRecombinationExpression phase, the GEMGA Recombination operator is applied iteratively on pairs of chromosomes. The GEMGA recombination uses the linkage sets for selecting regions of the parent chromosomes to be exchanged during the crossover. Linkage sets of the offsprings are modified based on the change in fitness from the parent to the children chromosomes.

Details about the algorithm can be found elsewhere [4, 42]. The population size in GEMGA is required to be $O(\lambda^k)$. The overall complexity of the GEMGA is estimated to be $O(\lambda^k \ell)$. The following section demonstrates the linear time performance of the GEMGA for different classes of additively decomposable problems, where each subproblem is comprised of 5 variables.

10.3 Organization of the GEMGA

The overall structure of the GEMGA is summarized below:

(1) Randomly initialize the duly sized population.
(2) Execute *primordial expression* stage: Detect schemata that capture local fitness symmetry by the so called *transcription* operator. Since population size $m = c\lambda^k$, this can be done in time $O(\lambda^k \ell)$.
(3) *PreRecombinationExpression:* Identify schemata that capture fitness symmetry over a larger domain. This only requires comparing the chromosomes with each other and no additional function evaluation is needed.
(4) Execute *recombination expression* stage:
    (a) GEMGA recombination: The GEMGA uses a recombination operator, designed using motivation from cell meosis process that combines the effect of selection and crossover. Reconstruct, modify schema linkage sets and their parameters.
    (b) Mutation: Low probability mutation like simple GA. All the experiments reported in this paper used a zero mutation probability.

The following section presents the test results.

10.4 Test Results

The performance of GEMGA is tested for five different problems, namely *i) Deceptive trap (Trap), ii) Mühlenbein (MUH), iii) Goldberg-Wang function 1 (GW1), iv) Goldberg-Wang function 2 (GW2)* and *v) Massively-Multimodal function (MULTI).* Each of the functions is constructed by concatenating order-5 sub-functions (both overlapping and non-overlapping versions are considered). Functions *MULTI* and *GW1* are defined in Table 1. *MULTI* is a massively multimodal function of unitation where the global optima is a string of all 1-s (assuming that length of the sub-function is odd). Functions *GW2* and *MUH* are defined in Table 2. Functions *GW1* and *GW2* are discussed in detail elsewhere [40]. In *MUH* The global optima is the string of all 0-s while all the strings having a number of trailing 1-s constitute the local optima. Unlike the case for *Trap*, here the building block corresponding to the global optima has a significant amount of overlap with the local optimas.

10.4.1 Results : Uniform scaling and non-overlapping sub-functions

Figures 5—7(left) show the number of sample evaluations needed to find the globally optimal solution for problem sizes ranging from 100 to 1000. The results are the average values obtained over ten runs when the problem sizes range from 100 to 500, and over five runs for problem sizes beyond 500 to 1000. For these test problems the sub-functions are uniformly scaled and non-overlapping. The population size is 200, chosen as described earlier in this paper. It is kept constant for all the problem sizes. In each case we see that the number of function evaluations required for attaining the optimal value linearly depends on the problem size.

10.4.2 Results : Non-uniform scaling and non-overlapping sub-functions

Scaling offers difficulty to any BBO algorithm that uses a selection like operator for selecting better solutions from the search space. The problem is that any such sample is an instance of many different classes defined over the search space, and the contribution of different classes in the overall objective function value may be different. Some classes may contribute higher than other classes. For large problems with a large degree of scaling effect, this can lead to a suboptimal convergence for the less scaled optimization variables.

The effect of scaling on the performance of GEMGA is investigated for *Trap, MUH* and *GW1* functions, for problem sizes of 100, 200, 300, and 500. As earlier, each function is a concatenation of order 5 sub functions. A linearly increasing scaling factor for each set of 5 sub-functions is taken. For example - a 500 bit problem has 100 sub-functions. The first 5 sub functions (bits 0 - 24)

are scaled by 1, next 5 sub-functions (bits 25 - 49) are scaled by 2, and so on. Figures 7(right)—8 show the results obtained for ten independent runs of the algorithm for the non-uniformly scaled Trap, MUH, and the GW1 functions. The GEMGA is found to solve all the problems successfully in linear time.

10.4.3 Results : Uniform scaling and overlapping sub-functions

This section presents preliminary test results for two overlapping test functions Fc2 and Fc3 developed elsewhere [59]. These functions are constructed as follows: Fc2(x) = $\sum_{j=0}^{L-1}$ MUH(s_j) where the s_j-s are overlapping 6-bit substrings of x. The first bit does not contribute to the fitness value. Function Fc3 is defined as follows: Fc3(x) = $\sum_{j=0}^{L-1}$ MULTI(s_j) Figure 9 presents the experimental results for Fc2 and Fc3. Population size is kept at 200. The results are averaged over ten runs.

10.5 Strengths and Weaknesses of the GEMGA

The main contribution of the GEMGA is the new scalable technique for detecting good partitions and schemata. Although the GEMGA transcription operator does not precisely identify the significant WCs, it does filter out the significant bits individually that participate in significant partitions. Moreover, the PreRecombinationExpression and Recombination Expression stages further filter out the good schemata using population-wide statistics.

The GEMGA representation is also rich enough to handle this partial information about the significant partitions and gradually improve the linkage estimation. The experimental results presented in the previous section clearly demonstrate the linear time performance of the GEMGA for a wide range of problems. A comparison between the number of function evaluations needed for solving the trap function by the fmGA and the GEMGA will make the progress obvious.

However, the GEMGA performance can be further improved by establishing the significant partition detection process on a solid foundation. The current GEMGA primordial stage does not provide precise information about the significant partitions. This issue needs to be formally investigated. The following section suggests one possible approach to do that.

11 Other Recent Efforts And Future Directions

The fundamental importance and potential of linkage learning algorithms have gradually started drawing serious attention of several researchers. There exists several dimensions of the current linkage learning research. This section presents a brief overview of the current efforts.

Harik recently introduced the Linkage Learning GA (LLGA) [29, 28]. The LLGA tries to learn linkage by exploiting the so called *exchange crossover operator* and the *probabilistic expression* based representation. The LLGA appears to work nicely particularly for problems in which construction of a linear ordering among the partitions is not so difficult. In a recent work [49] relations between compressed introns and the LLGA representation is discussed. Smith and Fogarty [68] reported a technique for evolving genetic linkage and to exploit it for adapting the recombination strategy. They reported superior performance of their linkage learning evolutionary algorithm over traditional GA.

A linkage learning approach similar to the GEMGA can be found elsewhere [44]. This approach makes use of a GEMGA-like filtering approach to detect the good partitions and schemata. In a recent work [60] a new linkage learning algorithm LINC is proposed. The LINC algorithm works by checking second order non-linearity. It considers feature pairs and performs $O(\ell^2)$ experiments. Let us define $\delta f_i = f(...\overline{x}_i...) - f(...x_i...)$, $\delta f_j = f(....\overline{x}_j..) - f(....x_j..)$, and $\delta f_{ij} = f(...\overline{x}_i\overline{x}_j..) - f(...x_ix_j..)$; where x_i is a boolean variable and $\overline{x}_i = 1 - x_i$. The LINC algorithm detects linkage by exploiting the fact that if two features i and j have pair-wise non-linearity then $|\delta f_{ij} - \delta f_i - \delta f_j| > \epsilon$, where ϵ is a constant.

A principal component analysis based construction of representation for performing evolutionary optimization has been developed elsewhere [71]. Kazadi [43] proposed a similar approach to explicitly detect generalized schemata, that he calls conjugate schemata. His approach proposed a second order non-linearity detection technique in the continuous space, similar to what the recently proposed LINC algorithm uses in the discrete boolean space. A different technique for detecting the significant relations using a dependency tree based approach has been proposed elsewhere [3]. This approach also exploits the second order non-linearity and estimates a second order approximation of the underlying relations; they reported superior performance of their algorithm over other techniques that do not explicitly try to search and exploit relations. A new algorithm called RGDA is developed elsewhere [58, 59] that again considers only second order non-linearity and offers $O(\ell^2)$ performance. Although linkage learning by detecting second order interaction may work well for many problems, it is not difficult to construct a problem where this

may not be sufficient and in fact can be misleading. Therefore, more comprehensive approaches for detecting most of the significant partitions need to be investigated.

One possible approach is to detect the significant partitions by explicitly approximating the WCs. This approach is currently being investigated by the first author and his students. Note that the WCs can be grouped into different subsets by defining schema like equivalence classes over the space of all indices of the WCs. For example, we can define the sets $w_{0\#} = \{w_{00}, w_{01}, w_{02}, w_{03}\}$, $w_{1\#} = \{w_{10}, w_{11}, w_{12}, w_{13}\}$, and similarly $w_{2\#}, w_{3\#}$. Define, $S_\alpha = \sum_{\beta \in \Lambda^{\ell-k}} w_{\alpha\beta}^2$, where Λ is the alphabet set of the representation. Now note that if any of the individual $w_{\alpha\beta}$-s has a magnitude greater than some threshold value θ, then S_α must have a value greater than θ^2. Therefore, if $S_\alpha < \theta^2$, then none of the Walsh coefficients with an index string starting with α has a significant magnitude. Figure 10 schematically illustrates the flow of the algorithm. At every node of the tree we compute S_α, and if S_α at the i-th node is less than θ^2 then none of its children can have an S_α value greater then θ^2 and therefore the subtree can be discarded. If the number of non-zero Walsh coefficients is bounded by a polynomial, we should be able to discard many such sub-trees just by checking the S_α at the root of the sub-tree. Using this idea, a polynomial time algorithm has been developed elsewhere [46] for learning certain classes of boolean functions. Their approach is based on approximate computation of S_α using a randomly chosen sample set. A similar approach is also being explored by Thierens [69]. The following section answers some of the common criticisms against linkage learning algorithms.

12 Discussions

Linkage learning or, in general, searching for relations and classes, plays an important role in blackbox optimization. Unfortunately, there has been a tendency to consider linkage learning as an esoteric research issue that may not have any relevance to the GA practitioners. This paper clearly pointed out that this is not true. Detection of linkage in GAs plays a critical role since GAs are fundamentally based on induction. Linkage learning research sometimes also faces another line of criticism. Sometimes it is argued that linkage learning may be important in GAs but not in other evolutionary algorithms. It is true that the linkage learning problem has been traditionally defined in the context of a relation space, comprised of similarity based equivalence relations. However, the need for detecting relations and classes is universal for all BBO algorithms. So if any other model of evolutionary computation aspires to perform scalable optimization, it has to define its relation space and there must exist a counterpart of linkage learning for that relation space.

Yet another common criticism is that the linkage learning GAs are time consuming, complex, and do not give satisfactory performance. Before addressing this issue, we must realize that even approximate detection of the significant partitions is a difficult problem, and a good algorithm for that will revolutionize many different fields such as signal processing, machine learning, cryptography, and others. There do not exist very many fields of science and engineering that have addressed this issue of approximate and efficient detection of significant partitions. While the early linkage learning efforts were primarily designed for breaking the ground and understanding the basic research issues, linkage learning evolutionary algorithms are starting to get more realistic and more efficient. A comparison among the performances of the different messy GAs will clearly demonstrate the evolution of one class of linkage learning GAs. Although the recent linkage learning GAs typically offer sub-quadratic performance for problems that can be solved using order-k delineable relations and several real-life applications in large scale data mining [41] and optimization [42] are currently under way, the authors believe that the linkage learning algorithms are still in the exploratory stage. We need a lot of hard work on both theoretical and experimental ground. However, there is no doubt that the scalable evolutionary algorithms of the 21st century are going to need efficient mechanisms for relation and class search. The following section concludes this paper.

13 Conclusions

This paper presented a conceptual foundation of linkage learning, described some of the earlier linkage learning GAs, and identified the relatively new state-of-the-art efforts. It started by noting that intelligent guessing plays an important role in non-enumerative inductive BBO. Scalable success of such algorithms require inducing patterns from the sampled data. Relations defined through the representation offer one way to capture such patterns. Linkage learning addresses the issue of efficient, scalable detection of appropriate relations. Therefore the designers of future evolutionary algorithms should pay careful attention to linkage learning. Since the relation space for even moderately interesting cases is extremely large and designing a measure for relations evaluation is quite difficult, there are plenty of tough challenges in this research area. So far the research has resulted in: (1) sub-quadratic-time approximate algorithms for problems, that can be solved using order-k delineable similarity based relations, (2) different techniques for evaluating relations, (3) different approaches to preserve and disseminate the information regarding good relations and classes among the population members of a GA. These results await immediate applications and they can be incorporated in the current GA practice for enhanced, scalable performance.

We should realize that the simple GA is unlikely to offer scalable performance unless the related feature variables defining good delineable relations are represented adjacently in the chromosome. This essentially implies availability of significant knowledge about the problem. If such knowledge is not available a priori and the problem is reasonably large enough then linkage learning techniques should be employed in order to adaptively detect the appropriate delineable relations. A spectrum of different sub-quadratic techniques listed in this paper may serve as the starting point for such purpose. In addition, techniques for preserving, updating, and propagating the linkage information should be employed. The current state-of-the-art linkage learning algorithms can also offer help in doing so. Although we have just began to scratch the surface, an increasing number of linkage learning techniques for improving the scalability of the existing evolutionary algorithms are starting to be available.

Acknowledgements

This work is also currently supported by National Science Foundation Grant IIS-9803360. The first author would like to like to acknowledge the support from US Army, and Air Force Office of Scientific Research for a portion of the work presented here, that was performed at Illinois genetic Algorithm Laboratory.

References

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

[2] J. D. Bagley. *Dissertations Abs. Intl.*, 28-12 (1967) 5106B. (Univ. Microfilms No. 68-7556).

[3] S. Baluja and S. Davies. Tech. Rep. CMU-CS-97-107, Dept. of Comp. Sc., Carnegie Mellon Univ., Pittsburgh, (1997).

[4] S. Bandyopadhyay, H. Kargupta, and G. Wang. In *Proc. of the IEEE Intl. Conf. on Evolutionary Computation*, IEEE Press, (1998) 603.

[5] K. G. Beauchamp. *Applications of Walsh and Related Functions.* Academic Press, USA, 1984.

[6] R. Belew and M. Vose, editors. *Foundations of Genetic Algorithms*, San Mateo, CA, (1996).

[7] A. D. Bethke. Tech. Rep., Univ. of Michigan, Logic of Computers Group, Ann Arbor, 197 (1976).

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

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. Massachussets: MIT Press ; New York: McGraw-Hill, (1990).
[10] H. A. David. *Order statistics*. John Wiley & Sons, Inc., New York, (1981).
[11] K. A. De Jong. *Dissertation Abs. Intl.*, 36/10, (1975) 5140B. (Univ. Microfilms No. 76-9381).
[12] K. Deb. Univ. of Ill. at Urbana-Champaign, Ill. IlliGAL Report no. 91004, Genetic Algorithms Laboratory, Urbana, (1991). and Doctoral dissertation (1991), Univ. of Alabama, Tuscaloosa.
[13] S. Forrest, editor. *Proc. of the Fifth Intl. Conf. on Genetic Algorithms*, San Mateo, CA, (1993).
[14] S. Forrest and M. Mitchell. *Proc. of the Fourth Intl. Conf. on Genetic Algorithms*, (1991) 182.
[15] D. R. Frantz. Non-linearities in genetic adaptive search. *Dissertation Abs. Intl.*, 33/11 (1972) 5240B–5241B. (Univ. Microfilms No. 73-11,116).
[16] G. Gates. Parallel messy genetic algorithms for the modelling the protein folding structure. Master's thesis. (1994), Air Force Inst. Of Tech.
[17] F. Glover. *ORSA J. on Computing*, 1 (1989) 190.
[18] D. E. Goldberg. *Complex Systems*, 3/2 (1989) 129. (Also TCGA Report 88006).
[19] D. E. Goldberg. *Complex Systems*, 3/2 (1989) 153. (Also TCGA Report 89001).
[20] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. (1989), Addison-Wesley, New York.
[21] D. E. Goldberg. IlliGAL Report No. 90002, (1990), Univ. of Ill. at Urbana-Champaign, Ill. Genetic Algorithms Laboratory, Urbana.
[22] D. E. Goldberg and C. L. Bridges. *Bio. Cyber.*, 62 (1990) 397. (and TCGA Report No. 88005).
[23] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik. *Proc. of the Fifth Intl. Conf. on Genetic Algorithms*, (1993) 56.
[24] D. E. Goldberg, K. Deb, and B. Korb. *Complex Systems*, 4 (1990) 415.
[25] D. E. Goldberg, K. Deb, and B. Korb. Don't worry, be messy. *Proc. of the Fourth Intl. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, (1991) 24.
[26] D. E. Goldberg, B. Korb, and K. Deb. *Complex Systems*, 3/5 (1989) 493. (And TCGA Report 89003).
[27] D. E. Goldberg and R. Lingle. In J. J. Grefenstette, editor, *Proc. of an Intl. Conf. on Genetic Algorithms and Their Applications*, (1985) 154.
[28] G. Harik. *Learning Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms*. PhD thesis, (1997), Dept. of Comp. Sc., Univ. of Michigan, Ann Arbor.
[29] G. Harik and D. E. Goldberg. In Belew and Vose [6].
[30] F. Hoffmann and G. Pfister. Presented on IFSA July, (1995), Sao Paulo.
[31] J. H. Holland. *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, Ann Arbor, (1975).
[32] H. Kargupta. *SEARCH, Polynomial Complexity, and The Fast Messy*

Genetic Algorithm. PhD thesis, (1995), Dept. of Comp. Sc., Univ. of Ill. at Urbana-Champaign, Urbana, IL 61801, USA. And IlliGAL Report 95008.

[33] H. Kargupta. Presented in SIAM Annual Meeting, (1996) as the winner of the 1996 SIAM Annual Best Student Paper Prize.

[34] H. Kargupta. In *Proc. of the IEEE Intl. Conf. on Evolutionary Computation*, IEEE Press, (1996) 814.

[35] H. Kargupta. In *Proc. of the IEEE Intl. Conf. on Evolutionary Computation*,(1996) 631. IEEE Press.

[36] H. Kargupta. In C. Poloni D. Quagliarella, J. Periaux and G. Winter, editors, *Genetic Algorithms in Engineering and Comp. Science.*, Chapter 4. (1997), John Wiley & Sons Ltd.

[37] H. Kargupta. In *Computational Aerosciences in the 21st Century*, (1998), Kluwer Academic Publishers.

[38] H. Kargupta and S. Bandyopadhyay. In *Lecture Notes in Comp. Sc.: Parallel Problem Solving from Nature*, Springer-Verlag, (1998) 315.

[39] H. Kargupta and D. E. Goldberg. In Belew and Vose [6], 291.

[40] H. Kargupta, D. E. Goldberg, and L. W. Wang. LAUR-96-27-48 (1996).

[41] H. Kargupta, E. Johnson, E. Riva Sanseverino, H. Park, L. D. Silvestre, and D. Hershberger. Tech. Rep. EECS-98-001, School of Electrical Engineering and Comp. Sc., Washington State Univ. (1998).

[42] H. Kargupta, E. Riva Sanseverino, E. Johnson, and S. Agrawal. To be published in Intelligent Data Analysis in Sc.: A Handbook by Cartwright, H., Oxford Univ. Press, (1998).

[43] S. Kazadi. In *Proc. of the Intl. Conf. on Genetic Algorithms*, (1997), 10. Morgan Kaufmann.

[44] C. Kemenade. Explicit filtering of building blocks and genetic algorithms. Personal communication, (1996).

[45] S. Kirpatrick, C. D. Gelatt, and M. P. Vecchi. *Science.*, 220/4598 (1983) 671.

[46] S. Kushilevitz and Y. Mansour. In *Proc. 23rd Annl. ACM Symp. on Theory of Computing*, (1991), 455.

[47] J. R. Levenick. In R. K. Belew and L. B. Booker, editors, *Proc. of the Fourth Intl. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, (1991) 123.

[48] G. E. Liepins and M. D. Vose. *Complex Systems*, 5/1 (1991) 45.

[49] F. Lobo, K. Deb, D. Goldberg, G. Harik, and L. Wang. In *Genetic Programming: Proc. of the Third Annual Conf.*, San Francisco, CA, (1998) 551.

[50] L. D. Merkle. Master's thesis, (1992), Air Force Inst. Of Tech., WPAFB OH 45433.

[51] L. D. Merkle. PhD. dissertation (1998), Air Force Inst. Of Tech., WPAFB OH 45433.

[52] L. D. Merkle and Lamont. G. B. In *Appl. Computing (1994), Proc. of the 1994 Symposium on Applied Computing*, New York, The Assoc. for

Computing Machinery.
[53] L. D. Merkle, G. H. Gates, and Lamont. G. B. In *Appl. Computing, (1998), Proc. of the 1998 Symp. on Applied Computing*. New York: The Assoc. for Computing Machinery.
[54] L. D. Merkle and G. B. Lemont. In Forrest [13] 191.
[55] R. S. Michalski. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning: An artificial intelligence approach*, Tioga Publishing Co, (1983) 323.
[56] T. Mitchell. *Machine Learning*. McGraw-Hill, USA, 1st edition, (1997).
[57] C. K. Mohan. In C. H. Dagli, L. I. Burke, Fernádez, and J. Ghosh, editors, *Intelligent Engineering Systems Through Artificial Neural Networks*, ASME Press, New York, (1993) 831.
[58] H. Muhlenbein and G. Paab. In *Parallel Problem Solving from Nature - PPSN IV*, Berlin, Springer, (1996), 178.
[59] H. Mühlenbein and A. O. Rodriguez. Schemata, distributions and graphical models in evolutionary optimization. Personal Communication., (1997).
[60] M. Munetomo and D. E. Goldberg. IlliGAL report 98012, Univ. of Ill. at Urbana-Champaign, (1998).
[61] C. K. Oei. Walsh function analysis of genetic algorithms of nonbinary strings. Unpublished master's thesis, (1992). Univ. of Ill. at Urbana-Champaign, Dept. of Comp. Sc.
[62] J. Paredis. In L. Eshelman, editor, *Proc. of the Sixth Intl. Conf. on Genetic Algorithms*, San Mateo, CA, (1995) 359.
[63] J. Plevyak. A messy GA with small primordial population (1992).
[64] J. R. Quinlan. *Machine Learning*, 1/1 (1986), 81.
[65] R. S. Rosenberg. Simulation of genetic populations with biochemical properties. *Dissertation Abs. Intl.*, 28/7 (1967) 2732B. (Univ. Microfilms No. 67-17, 836).
[66] J. D. Schaffer and A. Morishima. In J. J. Grefenstette, editor, *Proc. of the Second Intl. Conf. on Genetic Algorithms*, (1987) 36.
[67] H. A. Simon. In *The Sciences of the Artificial*, MIT Press, Cambridge, Massachusetts, USA, (1981) 192.
[68] J. Smith and T. Fogarty. In *Proc. of the IEEE Intl. Conf. on Evolutionary Computation*, IEEE Press, (1996) 826.
[69] D. Thierens. Personal communication. (1998).
[70] D. Thierens and D. Goldberg. In Forrest [13], 38.
[71] H. Voigt and H. Muhlenbein. In *Proc. of the Second Intl. Conf. on Evolutionary Computation*, (1995) 172.
[72] S. Watanabe. *Knowing and guessing - A formal and quantitative study*. John Wiley & Sons, Inc., New York, (1969).
[73] D. Whitley, R. Beveridge, C. Guerra, and C. Graves. In B. Punch, editor, *Proc. of the Seventh Intl. Conf. on Genetic Algorithms*, San Mateo, CA, (1997) 568.

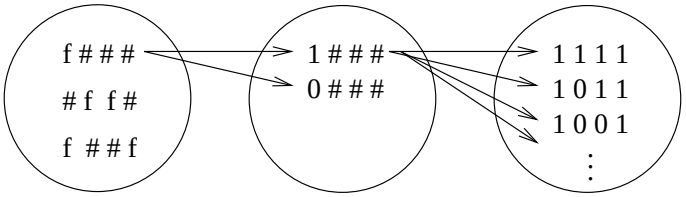

Fig. 1. Guessing by exploiting the relations and classes defined by the representation.

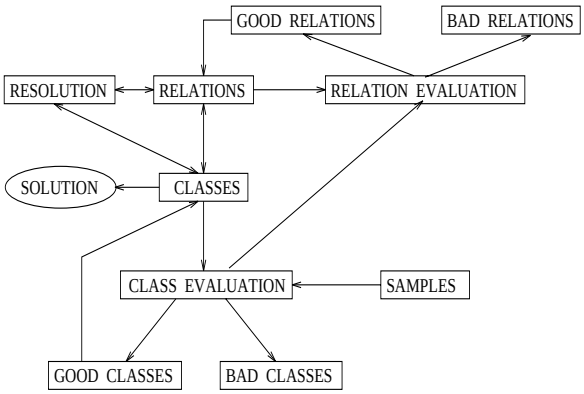

Fig. 2. A process oriented perspective of the SEARCH.

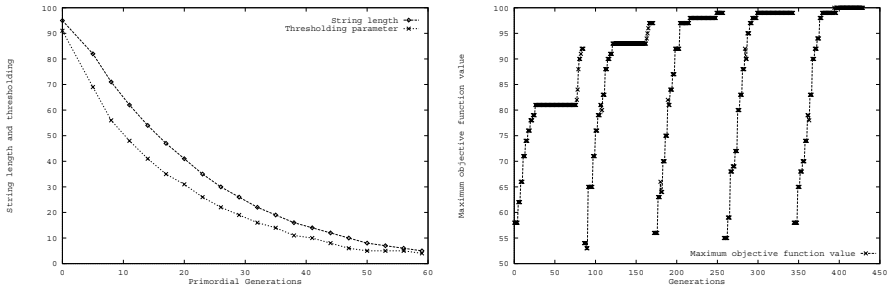

Fig. 3. *(left)* Building-block filtering schedule for order-5 level of a 100-bit problem. *(right)* Maximum objective function value in different generations for a 100-bit, order-5 deceptive *Trap* function. The fast messy GA found the best solution. Population size, $n = 7500$.

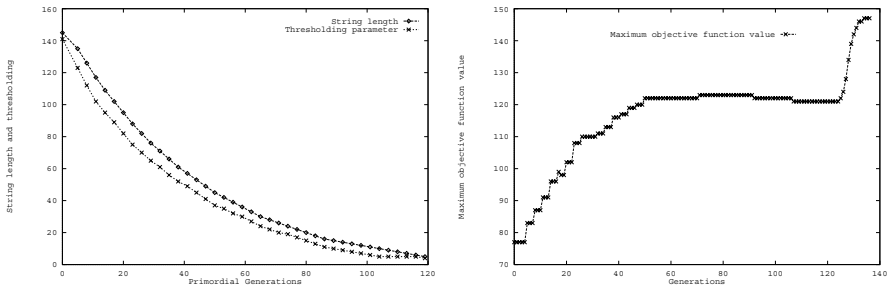

Fig. 4. *(left)* Building-block filtering schedule for order-5 level of a 150-bit problem. *(right)* Maximum objective function value in different generations for a 150-bit, order-5 deceptive *Trap* function. The fast messy GA found the correct solution for 27 out of the 30 subfunctions. Population size, $n = 8500$.

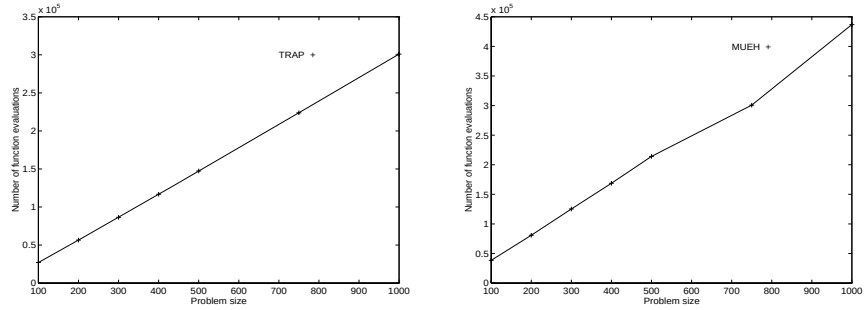

Fig. 5. Number of function evaluations vs. problem size for attaining the optimum solution by the GEMGA in case of *uniformly scaled*, *non-overlapping* (left) *Trap* and (right) *MUH*.

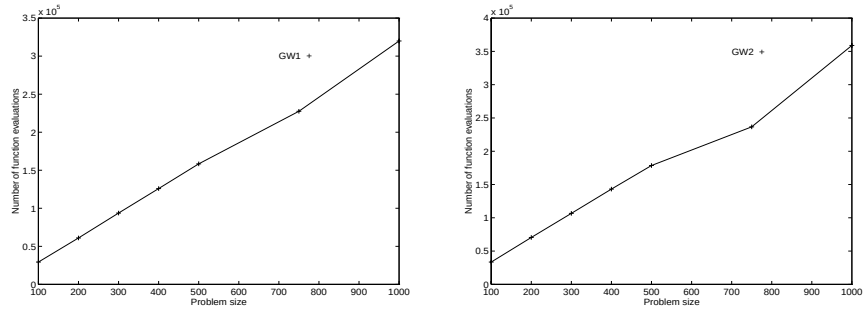

Fig. 6. Number of function evaluations vs. problem size for attaining the optimum solution by the GEMGA in case of *uniformly scaled*, *non-overlapping* (left) *GW1* and (right) *GW2*.

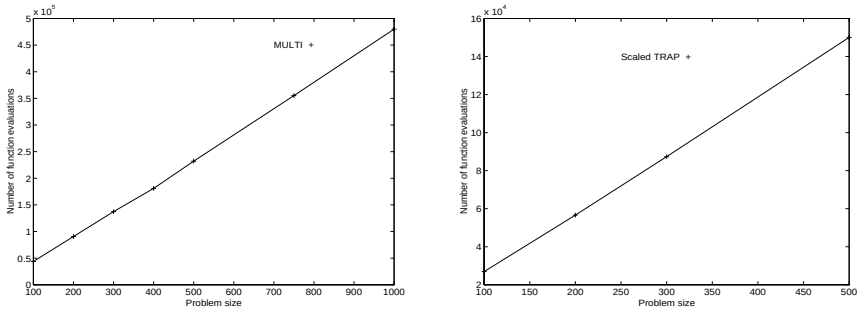

Fig. 7. Number of function evaluations vs. problem size for attaining the optimum solution by the GEMGA in case of *non-overlapping* (left) *uniformly scaled*, *Massively Multimodal* and (right) non-uniformly scaled *Trap*.

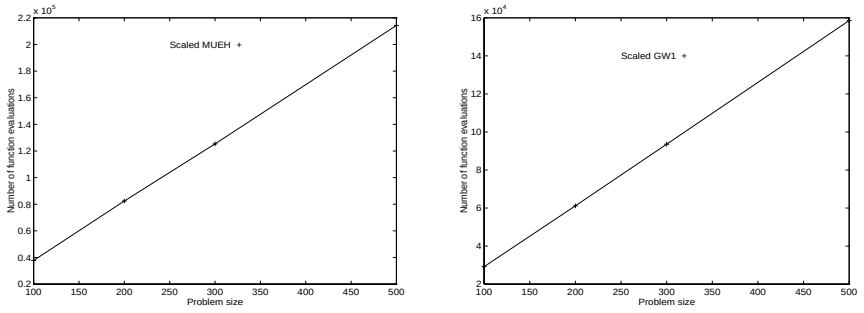

Fig. 8. Number of function evaluations vs. problem size for attaining the optimum solution by the GEMGA in case of non-uniformly scaled, *non-overlapping* (left) *MUH* and (right) *GW1*.

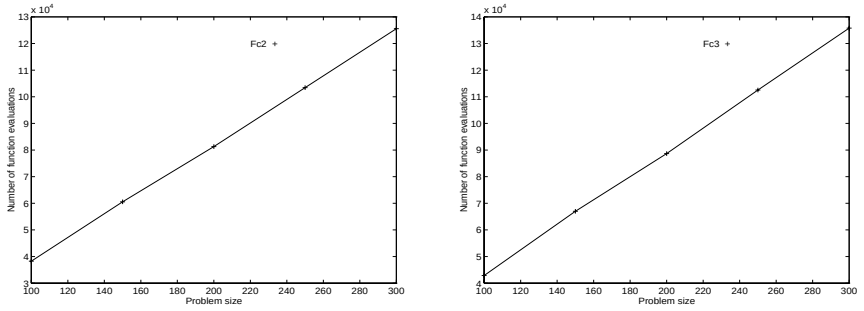

Fig. 9. Number of function evaluations vs. problem size for attaining the optimum solution by the GEMGA in case of (left) *Fc2* and (right) *Fc3*.

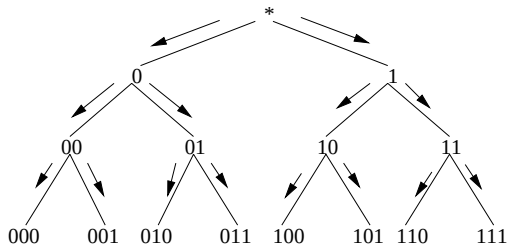

Fig. 10. Flow of the S_α computation for different α values.

Table 1
(left) Massively multimodal function and (right) GW1; u denotes the number of 1-s in the string. The symbol # denotes the don't care position.

Multi-modal	GW1
MULTI(x) = u+2 × f'(x)	GW1(x) = 4 if x = 1 # 1 # 0
where,	= 8 if x = 1 # 0 # 0
f'(x) = 1 if odd(u)	= 10 if x = 0 # 1 # 0
= 0 otherwise	= 0 if x = 0 # 1 # 0

Table 2
(left) MUH and (right) GW2; functions odd(0) and even(0) return true if the number of 0-s in x are odd and even respectively. odd(1) and even(1) are analogously defined.

Mühlenbein	GW2
MUH(x) = 4 if x = 00000	GW2(x) = 10 if u=0
= 3 if x = 00001	= 8 if u=k
= 2 if x = 00011	= 7 if u=1 and odd(0)
= 1 if x = 00111	= 2 if u=1 and even(1)
= 0 if x = 01111	= 4 if u=k-1 and odd(1)
= 3.5 if x = 11111	= 3 if u=k-1 and even(1)
= 0 otherwise.	= 0 otherwise.