

**Discovering Deep Building Blocks  
for Competent Genetic Algorithms  
Using Chance Discovery via KeyGraphs**

**David E. Goldberg  
Kumara Sastry  
Yukio Ohsawa**

IlliGAL Report No. 2002026  
December, 2002

Illinois Genetic Algorithms Laboratory (IlliGAL)  
Department of General Engineering  
University of Illinois at Urbana-Champaign  
117 Transportation Building  
104 S. Mathews Avenue, Urbana, IL 61801  
<http://www-illigal.ge.uiuc.edu>

# Discovering Deep Building Blocks for Competent Genetic Algorithms Using Chance Discovery via KeyGraphs

David E. Goldberg, Kumara Sastry  
Illinois Genetic Algorithms Laboratory (IlliGAL)  
Department of General Engineering  
University of Illinois at Urbana-Champaign  
104 S. Mathews Ave, Urbana, IL 61801  
{deg,ksastry}@uiuc.edu

Yukio Ohsawa  
The Laboratory of Chance Discovery  
Graduate School of Business Sciences  
University of Tsukuba  
Otsuka, Tokyo, Japan  
osawa@gssm.otsuka.tsukuba.ac.jp

## Abstract

In this paper, we see whether chance discovery in the form of KeyGraphs can be used to reveal deep building blocks to competent genetic algorithms, thereby speeding innovation in particularly difficult problems. On an intellectual level, showing the connection between KeyGraphs and genetic algorithms as related pieces of the innovation puzzle is both scientifically and computationally interesting. GAs represent that aspect of human innovation that tries to innovate through the exchange or *cross-fertilization* of notions contained in different ideas; the KeyGraph procedure represents that portion of human innovation that pays special attention to and interprets *salient fortuitous events*. The paper goes beyond mere conjecture and performs pilot studies that show how KeyGraphs and competent GAs can work together to solve the problem of deep building blocks; the work is promising and steps toward a practical computational combine of the two procedures are suggested.

## 1 Introduction

This volume brings together some of the latest work in *chance discovery*, and one of the things that makes chance discovery interesting relative to more common modes of machine learning and data mining is its emphasis on rare and possibly opportune events. Chance discovery is increasingly used (Ohsawa, 2001, 2002a) to mine text and other data for the connection of rare events to more statistically regular events in the hopes of teasing out a meaningful trend. For example, in consumer marketing, the early behavior of market-leading innovative consumers can be unmasked with chance discovery techniques and exploited by savvy companies (Ohsawa, 2002a).

On the other hand, going back almost 20 years (Goldberg, 1983), genetic algorithms—search procedures based on the mechanics of natural selection and natural genetics—have been likened to mechanistic versions of certain modes of human innovation, and a recent monograph (Goldberg,

2002) details the innovation connection, facetwise theory, and algorithmic mechanism that makes this connection tighter than ever.

Here in this paper, we bring these two seemingly disparate ideas together and wonder whether chance discovery in the form of KeyGraphs (Ohsawa, Benson, & Yachida, 1998) can be used to reveal deep building blocks to competent genetic algorithms, thereby speeding innovation in particularly difficult problems. On an intellectual level, showing the connection between KeyGraphs and genetic algorithms as related pieces of the innovation puzzle is both scientifically and computationally interesting. More will be said about the angle on innovation science as the paper progresses, but briefly, GAs represent that aspect of human innovation that tries to innovate through the exchange or *cross-fertilization* of notions contained in different ideas; the KeyGraph procedure represents that portion of human innovation that pays special attention to and interprets *salient fortuitous events*. This understanding of GAs and KeyGraphs is interesting on its face, but the linking of one sort of innovation to the other is a particular contribution of this work. Of course the paper goes beyond mere conjecture and performs pilot studies that should lead to an effective computational combine of the two procedures in the near future.

This paper is structured as follows. We start with brief introductions to genetic algorithms, the innovation intuition, and competent GAs. The discussion of competent GAs leads to the identification of the *problem of deep building blocks*. This is followed by an introduction to KeyGraphs, and then the use of KeyGraphs as a possible solution to the problem of deep building blocks is discussed. Pilot studies of using KeyGraphs to discover deep building blocks in GAs are outlined, and preliminary experiments are presented and discussed. We conclude with a discussion of a number of future research directions and some discussion on the more immediate utility of these results.

## 2 GAs: Innovation, Competence, and Deep Building Blocks<sup>1</sup>

This paper briefly reviews what genetic algorithms are, how they may be connected to human innovation, the recent program for making GAs more *competent* or scalable, and the problem of deep building blocks.

### 2.1 The One Minute Genetic Algorithmist

Genetic algorithms are search procedures based on the mechanics of natural selection and genetics. In this section we briefly review what GAs are, their mechanics, and their power of effect.

Suppose we are seeking a *solution* to some *problem*. The first thing we must do to apply a genetic algorithm to that problem is *encode* it as an artificial *chromosome* or chromosomes. These artificial chromosomes can be strings of 1s and 0s, parameter lists, permutation codes, or even complex computer codes, and one of the surprises found over the last decade is how well selectionist schemes do when faced with widely different codings.

Another thing we must do in solving a problem is to have some means or procedure for discriminating good solutions from bad solutions. This can involve the usual elaborate computer simulation or mathematical model that helps determine what good is (the standard notion of an *objective* function), or it can be as simple as having a human intuitively choose better solutions over worse ones (what we might call a *subjective* function). It can even be an ecology-like process where different digital species *co-evolve* through an intricate mix of competition and cooperation.

---

<sup>1</sup>Portions of this section are excerpted from the first author's new book, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms* (Kluwer Academic Publishers, 2002) and are copied with the permission of Kluwer Academic Publishers.

However it is done, the idea is that *something* must determine a solution's relative *fitness to purpose* (or *context*), and whatever that is will be used by the genetic algorithm to guide the evolution of future generations.

Having encoded the problem in a chromosomal manner and having devised a means of discriminating good solutions from bad ones, we prepare to *evolve* solutions to our problem by creating an initial *population* of encoded solutions. The population can be created randomly or by using prior knowledge of possibly good solutions, but either way a key idea is that the GA will search from a population, not a single point.

With a population in place, *selection* and *genetic operators* can process the population iteratively to create a sequence of populations that hopefully will contain more and more good solutions to our problem as time goes on. There is much variety in the types of operators that are used in GAs, but quite often (1) *selection*, (2) *recombination*, and (3) *mutation* are used.

Simply stated, selection allocates more offspring to better individuals; this is the survival-of-the-fittest mechanism we impose on our solutions. It can be accomplished in a variety of ways. Weighted roulette wheels can be spun, local tournaments can be held, various ranking schemes can be invoked; but, however we do it, the main idea is to *prefer better solutions to worse ones*. Of course, if we were only to choose better solutions repeatedly from the original database of initial solutions, we would expect to do little more than fill the population with the best of the first generation. Thus, simply selecting the best is not enough, and some means of creating new, possibly better individuals must be found; this is where the genetic mechanisms recombination and mutation come into play.

Recombination is a genetic operator that *combines bits and pieces of parental solutions* to form new, possibly better offspring. Again, there are many ways of accomplishing this, and achieving competent performance does depend on getting the recombination mechanism designed properly; but the primary idea to keep in mind is that the offspring under recombination will not be identical to any particular parent and will instead *combine parental traits in a novel manner*. By itself, recombination is not all that interesting of an operator, because a population of individuals processed under repeated recombination alone will undergo what amounts to a random shuffling of extant traits.

Where recombination creates a new individual by recombining the traits of two or more parents, mutation acts by simply modifying a single individual. There are many variations of mutation, but the main idea is that the offspring be identical to the parental individual except that *one or more changes are made to an individual's trait or traits* by the operator. By itself mutation represents a "random walk" in the neighborhood of a particular solution. If done repeatedly over a population of individuals, we might expect the resulting population to be indistinguishable from one created at random.

## 2.2 An Innovation Intuition for GAs

The previous section briefly described the mechanics of a genetic algorithm, but it gives us little idea of why these operators might promote a useful search. To the contrary, individually we saw how the operators acting alone were ineffectual, and it is something of an intellectual mystery to explain why such individually uninteresting mechanisms acting in concert might together do something useful. Starting in 1983 (Goldberg, 1983), the first author developed what has been called the *innovation intuition* to help explain this apparent mystery. Specifically, the innovation intuition likens the processing of selection and mutation together and that of selection and recombination taken together to *different facets of human innovation*, what are here called the *improvement* and *cross-fertilizing* types of innovation. We start first with the combination of selection and mutation

and continue with the selection-recombination pair.

### 2.2.1 Selection + mutation = Continual improvement

When taken together, selection and mutation are a form of hillclimbing mechanism, where mutation creates variants in the neighborhood of the current solution and selection accepts those changes with high probability, thus climbing toward better and better solutions. Human beings do this quite naturally; in the literature of total quality management, this sort of thing is called *continual improvement* or, as the Japanese call it, *kaizen*. Others have had similar thoughts, for example the British author and politician Bulwer-Lytton (Asimov & Shulman, 1988, p. 118):

Invention is nothing more than a fine deviation from, or enlargement on a fine model... Imitation, if noble and general, insures the best hope of originality.

Although this qualitative description is distant from an algorithmic one, we can hear the echo of mutation and selection within these words. Certainly, continuing to experiment in a local neighborhood is a powerful means of improvement, although it will have a tendency to be fairly local in scope, unless a means can be found for intelligently jumping elsewhere when a locally optimal solution is found.

### 2.2.2 Selection + recombination = innovation

One way of promoting this kind of intelligent jumping is through the combined effect of selection and recombination, and we can start to understand this if we liken their effect to that of the processes of human cross-fertilizing innovation. What is it that people do when they are being innovative in a cross-fertilizing sense? Usually they are grasping at a notion—a set of good solution features—in one context, and a notion in another context and juxtaposing them, thereby speculating that the combination might be better than either notion taken individually. Again, the first author's thoughts on the subject were introspective ones, but others have written along similar veins, for example, the French mathematician Hadamard (1945, p. 29):

We shall see a little later that the possibility of imputing discovery to pure chance is already excluded....Indeed, it is obvious that invention or discovery, be it in mathematics or anywhere else, takes place by combining ideas.

Likewise, the French poet-philosopher Valéry had a similar observation (Hadamard, 1945, p. 30):

It takes two to invent anything. The one makes up combinations; the other chooses, recognizes what he wishes and what is important to him in the mass of the things which the former has imparted to him.

Once again, verbal descriptions are far from our more modern computational kind, but something like the innovation intuition has been clearly articulated by others.

With a basic understanding of the mechanics of genetic algorithms and an intuitive understanding of the innovation-GA connection, we now consider key aspects of the theory and practice of competent GAs.

## 2.3 Competent Genetic Algorithms

The last few decades have witness great strides toward the development of so-called *competent* genetic algorithms—GAs that solve hard problems, quickly, reliably, and accurately (Goldberg, 1999). Competent GAs are a two-edged sword. From a computational standpoint, the existence of competent GAs suggests that many difficult problems can be solved in a scalable fashion. Moreover, it suggests that the usual casting about for a good coding or a good genetic operator that accompanies many GA applications can be minimized; if the GA can adapt to the problem, there is less reason for the user to have to adapt the problem, coding, or operators to the GA. From the standpoint of human innovation, the existence of competent GAs is tantamount to the creation of a computational theory of cross-fertilizing innovation. This has scientific ramifications that have yet to be explored.

In this section we briefly review some of the key lessons of competent GA design. Specifically, we restrict the discussion to selectorecombinative GAs and focus on the cross-fertilization type of innovation and briefly discuss key facets of competent GA design.

Using Holland’s notion of a building block (BB) (Holland, 1975), the first author proposed decomposing the problem of designing a competent selectorecombinative GA (Goldberg, 1991; Goldberg, 1993; Goldberg, Deb, & Clark, 1992; Goldberg & Liepens, 1991). This design decomposition has been explained in detail elsewhere (Goldberg, 2002), but is briefly reviewed in what follows.

**Know that GAs process building blocks (BBs):** The primary idea of selectorecombinative GA theory is that genetic algorithms work through a mechanism of *decomposition* and *re-assembly*. Holland (1975) called well-adapted sets of features that were components of effective solutions *building blocks* (BBs). The basic idea is that GAs (1) implicitly identify building blocks or subassemblies of good solutions, and (2) recombine different subassemblies to form very high performance solutions.

**Understand BB hard problems:** From the standpoint of cross-fertilizing innovation, problems that are hard have BBs that are hard to acquire. This may be because the BBs are deep or complex, hard to find, or because different BBs are hard to separate, or because low-order BBs may be *misleading*, or in other words *deceptive* (Goldberg, 1987).

**Understand BB growth and timing:** Another key idea is that BBs or notions exist in a kind of competitive *market economy of ideas*, and steps must be taken to ensure that the best ones (1) grow and take over a dominant market share of the population, and (2) the growth rate can neither be too fast, nor too slow.

**Understand BB supply and decision making:** One role of the population is to ensure adequate *supply* of the raw building blocks in a population (Holland, 1975; Goldberg, 1989; Goldberg, Sastry, & Latoza, 2001). Randomly generated populations of increasing size will, with higher probability, contain larger numbers of more complex BBs. Furthermore, decision making among different, competing notions (BBs) is *statistical* in nature, and as we increase the population size, we increase the likelihood of making the best possible decisions.

**Identify BBs and exchange them:** Perhaps the most important lesson of current research in GAs is that the *identification and exchange of BBs* is the critical path to innovative success. First-generation GAs, usually fail in their ability to promote this exchange reliably. The primary design challenge to achieving competence is the need to identify and promote effective BB exchange.

Efforts in principled design of effective BB identification and exchange mechanisms have led to the development of competent genetic algorithms.

Interestingly, the mechanics of competent GAs vary widely, but the principles of innovative success are invariant. Competent GA design began with the development of the *messy genetic algorithm* (Goldberg, Korb, & Deb, 1989), culminating in 1993 with the *fast messy GA* (Goldberg, Deb, Kargupta, & Harik, 1993). Since those early scalable results, a number of competent GAs have been constructed using different mechanism styles:

**Perturbation techniques** include the fast messy GA (fmGA) (Goldberg, Deb, Kargupta, & Harik, 1993), gene expression messy GA (GEMGA) (Kargupta, 1996), and linkage identification by nonlinearity check GA (LINC GA), and linkage identification by monotonicity detection GA (LIMD GA) (Munetomo & Goldberg, 1999).

**Linkage adaptation techniques** such as linkage learning GA (LLGA) (Harik & Goldberg, 1997; Harik, 1999b; Chen & Goldberg, 2002).

**Probabilistic model building techniques** such as the extended compact GA (ECGA) (Harik, 1999a), iterated distribution estimation algorithm (IDEA) (Bosman & Thierens, 1999), Bayesian optimization algorithm (BOA), and hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Goldberg, 2001). For further details on probabilistic model building GAs, the interested reader should look elsewhere (Pelikan, 2002b; Pelikan, 2002a; Larrañaga & Lozano, 2002).

The breadth of capability of competent GAs is difficult to fathom. The usual idea of optimization is restricted to fairly straightforward notions such as hillclimbing and enumeration, but competent GAs zero in on global or near-global solutions in problems that are massively multimodal, are noisy, and have difficult hierarchical and misleading nonlinearities, because they have been constructed to face adversarially designed problems. This gives them striking capability to penetrate difficult problems with astonishing rapidity. Nonetheless, a more careful reading of the decomposition exposes an important Achilles heel of the method, one shared by many data-mining procedures and meta-heuristics. This difficulty is explored in the next section.

## 2.4 The Problem of Deep Building Blocks

Despite the progress made in competent GA design, certain types of problems can require solution times that thwart the best of known competent GAs. To better understand this we consider some details of competent GA population sizing.

The decomposition of competent GA design of the last section examined population sizing in GAs as largely an issue of building block supply and decision making. Regardless of the viewpoint one takes, the simple fact is that high-order building blocks require exponentially larger population sizes. That is, regardless whether the population size is governed on supply grounds or decision grounds, population sizes  $n$  must grow as an exponential function of the building block order  $k$  required:

$$n \propto 2^k. \quad (1)$$

For small  $k$ , this requirement is no problem, but for larger values it is a deal breaker. Without delving into too many specifics, the usual population-sizing equation multiplies the BB-wise exponential term by a function that goes up as a polynomial function of the number of building blocks involved. The end result is a prohibitively expensive, yet theoretically tractable computation, for large- $k$  or what here we shall term *deep* building blocks.

To put some flesh on the bones of our argument, consider a population sizing that goes as  $n = cm2^k$ , where  $m$  is the number of building blocks,  $k$  is the order of the deepest building block that must be revealed whole, and  $c$  a constant that varies with other problem parameters. Here we take  $c = 1$  without loss of generality. We assume that we are solving a modestly sized problem with  $m = 100$  building blocks, and compare population sizes required to solve problems with building blocks of order  $k = 1, 5, 30$ . The resulting population sizes are 100, 3200, and  $1.07 (10^{11})$  (or about 107 billion!). Clearly, the need to discover even modestly deep building blocks whole is a difficult task, and ways need to be found to expedite the solution of such problems.

In a later section, we engage chance discovery through KeyGraphs to try to do just that. In the next section, we briefly review the KeyGraph calculation to better understand how it might help us in this endeavor.

### 3 The KeyGraph Procedure: Overview & Intuition

The *KeyGraph* procedure is a graphical method for data mining originally developed for indexing a document (Ohsawa, Benson, & Yachida, 1998; Ohsawa, 2002b), and recently utilized for chance discovery (Ohsawa, 2001, 2002a) including discovery of risky active faults of earthquakes (Ohsawa & Yachida, 1999). Like other text- and data-mining algorithms, KeyGraph identifies relationships between terms and term clusters in a document. In particular, KeyGraph focuses on co-occurrence relationships, but one thing that sets KeyGraph apart is its emphasis on both high probability and low probability events. The feature that attracts us to KeyGraph in this paper, is its identification of low probability terms and their linkage to high probability terms and clusters of terms. The idea here is to see if KeyGraph can identify deep building blocks, and if so, under what conditions.

Here, we review the basic KeyGraph algorithm. We start by putting forth an innovation intuition for KeyGraphs, and we continue by reviewing the steps of KeyGraph processing. The next section follows up this discussion by suggesting how we might use the KeyGraph procedure to discover deep building blocks.

#### 3.1 An Innovation Intuition for the KeyGraph Procedure

Previously, we likened the processing of genetic algorithms to different facets of human innovation, facets we called cross-fertilizing and continual improvement. Here we make an analogous connection for KeyGraphs. In particular, we suggest the KeyGraph procedure is a computational analog of the human tendency to mine and interpret *salient fortuitous events*.

The literature of the history of science and technology has lavished considerable attention on the role of chance events in the discovery of new ideas (Beveridge, 1957; Roberts, 1989; Fine & Deegan, 1996). The Archimedes' bathtub moment, the discovery of penicillin, and the invention of Post-It notes were all accompanied by a Eureka moment in which the scientist or inventor paid attention to a chance occurrence that seemed odd, but interesting. We will call such events *salient fortuitous events*, and we suggest that one of the reasons that KeyGraphs (and other chance discovery computations) are interesting is because they focus on the *unexpected* and how it is linked to that which is regular and well anticipated.

We will examine the details of the KeyGraph procedure in a moment, and the first part of the procedure has much in common with other data- and text-mining techniques. The interesting part of KeyGraph for our purposes (and for most chance discovery applications) is its emphasis of *rare events*. In the usual chance discovery application of KeyGraphs, everyone usually agrees upon the statistically significant data. The value of the procedure comes from interpretation of the low probability events that link clusters of high probability events. In other words, KeyGraph users

usually find that the statistically regular terms (so-called *high-frequency terms*) need no discussion or interpretation. Instead, practitioners come to understand the need to focus on the *key terms* and try to understand how they may be exploited in the particular application.

Clearly we are using this innovation intuition for KeyGraphs in our study. In particular, we are attempting to use KeyGraphs to point out deep building blocks as salient fortuitous events. If we are successful, we will be able to solve problems with deeper building blocks more easily.

### 3.2 The KeyGraph Procedure: An Overview

Here we assume that a document,  $D$ , is composed of sentences and each sentence is composed of words. The main steps of the KeyGraph algorithm can be outlined as follows.

**Document preprocessing**, which consists of two tasks:

1. **Document compaction:** Insignificant words from the document is removed using a user supplied list of words and word stems. Word stems are used to reduce related words to the same root. For example, words like “innovate”, “innovates”, and “innovating” are reduced to “innovate” using a method proposed by Porter (1980).
2. **Phrase construction:** Here preference is given to longer phrases with higher frequency. A subset of  $\ell_{\text{phrase}}$  words are chosen from the document and all possible phrases out of those words are constructed. A phrase that occurs with the highest frequency in the document is retained.

It should be noted that in this study we don’t use any preprocessing. After preprocessing, the document  $D$  is reduced to  $D'$  which consists of unique terms  $w_1, w_2, \dots, w_\ell$ , where a term  $w_i$  refers to either a word or a phrase.

**Extracting high-frequency terms:** Terms in  $D'$  are sorted by their frequencies in the document,  $D$  and top  $n_{\text{nodes}}$  high-frequency terms are retained. These high-frequency terms are represented as nodes in a graph  $G$ . A set of the high-frequency terms is denoted by  $N_{hf}$ .

**Extracting links:** Links represent *co-occurrence*—term-pairs that often occur in the same sentence. A measure for co-occurrence of terms  $w_i$  and  $w_j$  is defined as

$$assoc(w_i, w_j) = \sum_{s \in D} \min(|w_i|_s, |w_j|_s), \tag{2}$$

where  $w_i$  and  $w_j$  are elements of the set  $N_{hf}$ , and  $|w_i|_s$  is the number of times a term  $w_i$  occurs in a sentence  $s$ .

The *assoc* values are computed for all pairs of high-frequency terms in  $N_{hf}$ . The term-pairs are sorted according to their *assoc* values and top  $N_{hf} - 1$  tightly associated term-pairs are taken to be the links. The links between term-pairs are represented by the edges in  $G$ .

**Extracting key terms:** Key terms are terms that connect clusters of high-frequency terms together. To measure the tightness with which a term  $w$  connects a cluster, the following function is defined:

$$key(w) = 1 - \prod_{g \in G} \left[ 1 - \frac{based(w, g)}{neighbors(g)} \right], \tag{3}$$

where  $g$  is a cluster, and

$$based(w, g) = \sum_{s \in D} |w|_s |g - w|_s, \quad (4)$$

$$neighbors(w) = \sum_{s \in D} \sum_{w \in s} |w|_s |g - w|_s. \quad (5)$$

$$|g - w|_s = \begin{cases} |g|_s - |w|_s, & \text{if } w \in g \\ |g|_s & \text{if } w \notin g \end{cases}, \quad (6)$$

where  $|g|_s$  is the number of times a cluster  $g$  occurs in a sentence  $s$ .

Qualitatively,  $key(w)$  gives a measure of how often a term  $w$  occurs near a cluster of high-frequency terms.

The  $key$  values are computed for all the terms in  $D$  and  $n_{key}$  top  $key$  terms are taken as *high-key terms*. These high-key terms are added as nodes—if they are not already present—in  $G$  and are elements of a set  $K_{hk}$ .

**Extracting key links:** For each high-frequency term  $w_i \in N_{hf}$  and each high-key term  $w_j \in K_{hk}$ , the  $assoc(w_i, w_j)$  is calculated. Links touching  $w_j$  are sorted by their  $assoc$  values for each high-key term  $w_j \in K_{hk}$ . A link with highest  $assoc$  values connecting  $w_j$  to two or more clusters is chosen as a key link. Key links are represented by edges—if they are not already present—in  $G$ .

**Extracting Keywords:** Nodes in  $G$  are sorted by the sum of  $assoc$  values associated with the key links touching them. Terms represented by nodes of higher values of these sums than a certain threshold are extracted as keywords for the document  $D$ .

More details are available in other references (Ohsawa, Benson, & Yachida, 1998; Ohsawa & Yachida, 1999; Fukuda & Ohsawa, 2001; Ohsawa, 2002b). In the next section, we examine how KeyGraph may be helpful in solving the problem of deep building blocks.

## 4 Can KeyGraphs Discover Deep Building Blocks?

We have likened selectorecombinative GAs to cross-fertilizing innovation and we have likened the KeyGraph procedure to keeping an eye open for salient fortuitous events. Here we wonder whether we can use the KeyGraph procedure as a preprocessor to competent GAs to help identify deep building blocks in a population, where the deep BBs are not yet present in large numbers. This is an important question, because most GAs explicitly or implicitly assume that good building blocks are present in sufficiently large numbers that they constitute a statistically significant sample. As discussed earlier, this assumption results in population-sizing requirements that grow exponentially with deep (high order) BBs. From the standpoint of a GA, the quest for less costly methods of identifying potentially useful deep BBs takes us from a point where searching for solutions to problems with deep building blocks might be intractable to one where it may be practical.

In this section, we explain the design of a series of experiments to test the use of the KeyGraph procedure in this manner. The results from those experiments are also presented. As it turns out, the results are mixed, but they are interesting, and they lead to a number of reasonable continuations of this work.

## 4.1 Design of Pilot Experiments

Our approach in designing pilot experiments for investigating the BB identification capability of KeyGraph is to *design* bounding *adversarial problems* that exploit one or more dimensions of problem difficulty. Particularly, our pilot test problems should possess the following properties:

- Building-block identification should be critical for successful innovation. That is, if the BBs of the problem are not identified and exchanged, it should be impossible to attain the global solution. We are interested in solving such class of search problems where BB identification is essential for cross-fertilization type of innovation.
- Building-block structure and interactions of the problem should be known to the researchers, but not to the problem solver (search method). Knowing the BB-structure and their interactions a priori, makes it easy to analyze if a BB-identification technique—be it a competent GA, or KeyGraph—has successfully discovered individual building blocks. Furthermore, ensuring that the BB identification methods work on such problems, provides assurance that they would also identify BBs of real-world problems, where the BBs are not known a priori.
- The properties such as building-block size, problem difficulty should be tunable without significantly changing the function. For example, we should be easily able to change a building block of low order into a deep building block, by changing one or two parameters.

This adversarial and systematic design method contrasts sharply with the common practices of using historical, randomly generated, or ad hoc test functions (Goldberg, 2002).

One such class of adversarial problems is the additively separable, *deceptive* function (Goldberg, 1987). Deceptive functions are designed to thwart the very mechanism of selectorecombinative search by punishing any localized hillclimbing and requiring mixing of whole building blocks at or above the order of deception. Using such *adversarially* designed functions is a stiff test—in some sense the stiffest test—of algorithm performance. The idea is that if an algorithm can be an adversarially designed test function, it can solve those problems and any easier than the adversary. Furthermore, if the building blocks of such deceptive functions are not identified and respected by a selectorecombinative GAs, then they almost always converge to the local optimum.

In this study, we use deceptive *trap* functions (Deb & Goldberg, 1993; Deb & Goldberg, 1994). A fully-deceptive  $k$ -bit trap function is defined as follows:

$$f(u(x_1, x_2, \dots, x_k)) = \begin{cases} 1.0 & u = k \\ (1.0 - \delta) \left(1 - \frac{u}{k-1}\right) & u < k \end{cases}, \quad (7)$$

where  $u$  is the unitation, or the number of ones in a portion of the bit string:

$$u(x_1, x_2, \dots, x_k) = \sum_{i=1}^k x_i, \quad (8)$$

where,  $x_i$  is the value of the  $i$ th bit,  $\delta$  is the difference in the functional value between the good BB (all ones) and its deceptive attractor (all zeros). The difficulty of a trap function can be adjusted by modifying the values  $k$ , and  $\delta$ . The problem becomes more difficult as the value of  $k$  is increased and that of  $\delta$  is decreased. A 4-bit deceptive trap function is illustrated in figure 1.

Given that the string length is  $\ell$ , an additively separable test function is defined as

$$F(\mathbf{x}) = f(u(x_1, x_2, \dots, x_k)) + f(u(x_{k+1}, \dots, x_{2k})) + \dots + f(u(x_{(m-1)k+1}, \dots, x_{mk})), \quad (9)$$

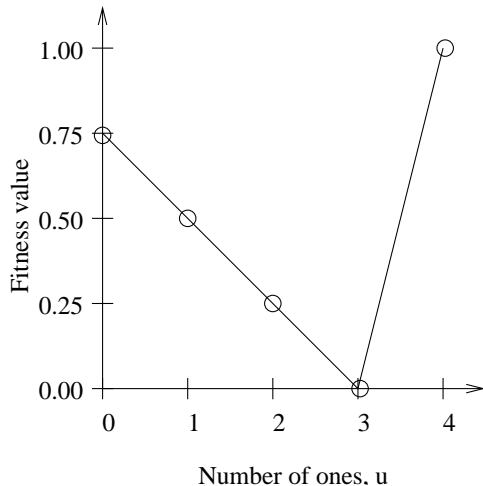


Figure 1: A Fully deceptive trap function with  $k = 4$ , and  $\delta = 0.25$ .

where,  $F$  is the fitness function,  $\mathbf{x}$  is the bit string (individual), and  $m$  is the number of BBs.

The important feature of additively separable trap functions is that if the good BB (all ones) in any particular partition is not identified, then the GA tends to converge to the deceptive attractor (all zeros) in that partition. Therefore, BB identification and mixing is critical to innovation success. Furthermore, notice that the problems are of bounded difficulty  $k < \ell$ . If the trap was of length  $\ell$ , nothing could work better than enumeration or random search without replacement. However, given that the difficulty is bounded, a GA that identifies BBs and mixes them well has the opportunity to solve the problem in polynomial time (Goldberg, 2002).

In this study, we utilize two test problems:

1. **The  $3 \times 4$  Problem:** This problem consists of three uniformly-scaled low-order building blocks (BB size,  $k = 4$ ). That is, the  $3 \times 4$  problem consists of three 4-bit fully deceptive trap functions ( $m = 3$ ,  $k = 4$ ):

$$F_{3 \times 4}(\mathbf{x}) = f(u(x_1, x_2, x_3, x_4)) + f(u(x_5, x_6, x_7, x_8)) + f(u(x_9, x_{10}, x_{11}, x_{12})). \quad (10)$$

Here, all the BBs are of low order and therefore they have statistically significant market share in typically (subquadratically or linearly) sized population. In a randomly generated initial population, each building block will approximately have a market share of  $(1/2^4)$ th or 6.25% of the population.

2. **The  $2 \times 4 + 12$  Problem:** This problem consists of two uniformly-scaled building blocks of low order (BB size,  $k = 4$ ) and a deep building block (BB size,  $k = 12$ ). That is, the  $2 \times 4 + 12$  problem is composed of 3 deceptive trap functions, two of which have low-order BBs of size,  $k = 4$ , and the other is a deep BB of size  $k = 12$ :

$$F_{2 \times 4 + 12}(\mathbf{x}) = f(u(x_1, x_2, x_3, x_4)) + f(u(x_5, x_6, x_7, x_8)) + f(u(x_9, x_{10}, \dots, x_{20})). \quad (11)$$

Even though the regular BBs have a market share of 6.25% in the population, the deep BB has a market share of only  $(1/2^{12})$ th, or 0.024% in the population. Therefore, the probability of a deep BB occurring is insignificantly small in a typically sized population. Moreover, if a deep BB does not occur in a population, we have almost no chance of creating it from smaller pieces, because the very nature of a deceptive problem is to encourage the emphasis

of the *wrong* smaller pieces. Yet, when a deep BB does occur in very few numbers (one or two individuals in the population of thousand individuals have the deep building block), we need to identify it.

Here, for the sake of better visualization and understanding, we consider relatively short string lengths (12-20 bits) and modest BB counts (3 BBs). By way of comparison, modern competent GAs can solve problems of bounded difficulty with hundreds and even thousands of decision variables to global optimality with high reliability and accuracy. Our use of modest string lengths and BB counts is justified by our immediate goal of understanding the possible role of KeyGraphs in solving the problem of deep BBs for competent GAs. Additional work will be necessary to more fully integrate KeyGraphs and working competent GAs if we are successful, but the pilot studies here are important first steps along the way to a working hybrid.

In the subsequent sections, we discuss the following pilot studies. First, we test the capability of competent GAs in identifying deep BBs. We consider two state-of-the-art competent GAs, the Bayesian optimization algorithm (BOA), and the extended compact genetic algorithm (ECGA). Next, we investigate whether KeyGraphs can identify low-order building blocks. Finally, we analyze the potential of KeyGraph in deep BB identification.

## 4.2 Competent GAs Fail to Identify Deep Building Blocks

Previous studies on competent GAs have demonstrated that they are highly capable of identifying and exchanging BBs of low order, and often do so in polynomial (usually subquadratic) time (Goldberg, Deb, Kargupta, & Harik, 1993; Pelikan, 2002a; Harik, 1999a; Bandyopadhyay, Kargupta, & Wang, 1998; Kargupta & Bandyopadhyay, 1998; Sastry & Goldberg, 2000). However, their capability in identifying deep building blocks, to the best of our knowledge, has not been investigated. Therefore, it was logical for us to test if the competent GAs could identify deep BBs. Specifically, we investigated the deep-BB-identification capability of two state-of-the-art competent GAs—the Bayesian optimization algorithm (BOA) (Pelikan, 2002b), and the extended compact genetic algorithm (ECGA) (Harik, 1999a).

We used both BOA and ECGA to solve the  $2 \times 4 + 12$  problem. The BOA and ECGA procedure employed can be broadly outlined as follows:

1. Initialize the population randomly.
2. Evaluate the fitness values of individuals in the population.
3. Perform selection to make more copies of better individuals. Selection biases the population towards highly fit individuals and therefore facilitates the identification of BBs.
4. Construct the probabilistic model of the selected individuals (Bayesian networks in the case of BOA and marginal product models (MPMs) in the case of ECGA). The probabilistic model refers to the building-block structure and interaction of the search problem.
5. Sample new individuals based on the probabilistic model.
6. Repeat steps 2–5 till one or more convergence criteria is satisfied.

The population size in each case was chosen in accordance with appropriate population-sizing models (Pelikan, Goldberg, & Cantú-Paz, 2000; Sastry & Goldberg, 2000). However, in determining the population size, we assumed a problem with low-order BBs ( $5 \times 4$  problem). The population sizes resulting from the above assumption are less than that is required to ensure the presence of

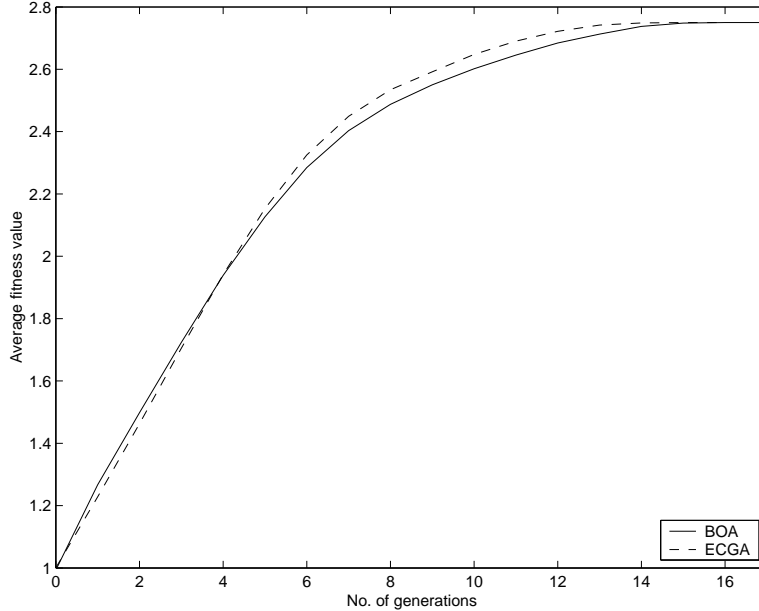


Figure 2: Convergence of BOA and ECGA on the  $2 \times 4 + 12$  problem. The average fitness of the population is plotted as a function of number of generations. The results are averaged over 50 independent runs. The results show that both BOA and ECGA fail to identify the deep BB and therefore converge to the local optimum (fitness value 2.75) in all 50 runs. Similar results are obtained even when the population is injected with a few copies of the deep BB.

significant number of raw deep BBs in the population. However, they are large enough to ensure a statistically significant presence of low-order BBs. Therefore, the competent GAs should be able to identify building blocks of low order without much difficulty.

If the deep BB of the  $2 \times 4 + 12$  problem is correctly identified, then the individuals in the population should converge to the global optimum and the average fitness of the population should be 3.0 (fitness value of the global solution). On the other hand, if the deep BB is not identified then the population converges to the local optimum with a fitness value of 2.75.

First, we used BOA and ECGA to solve the  $2 \times 4 + 12$  on randomly initialized population, where the chances of the deep BB occurring in the population are slim to none. The performance of BOA and ECGA on the  $2 \times 4 + 12$  problem is shown in the figure 2. The figure plots average fitness of the population as a function of the number of generations. The results are averaged over 50 independent runs. Results indicate that both BOA and ECGA, while successful in identifying low-order BBs, fail to discover deep BBs.

Usually in a typically-sized (non-exponential) random population, there are no individuals with the deep BB, and due to the deceptive nature of the  $2 \times 4 + 12$  problem, we have no chance of creating the deep BB from smaller pieces. This makes it impossible for competent GAs to identify the deep BB. Therefore, a more interesting (and perhaps a more fair) question is to ask ourselves whether the competent GAs can identify the deep BB, if it is present in the population in small numbers (one or two individuals in a population of one or two thousand).

Therefore, we inject a couple of individuals into the selected population (after step 3 of the experimental procedure) in the first generation. We randomly choose two individuals from the selected population and for each individual replace the 12 bits,  $x_9-x_{20}$ , with the best BB (all ones). The remaining 8 bits ( $x_1-x_8$ ) of both individuals are not modified during the injection. Note that

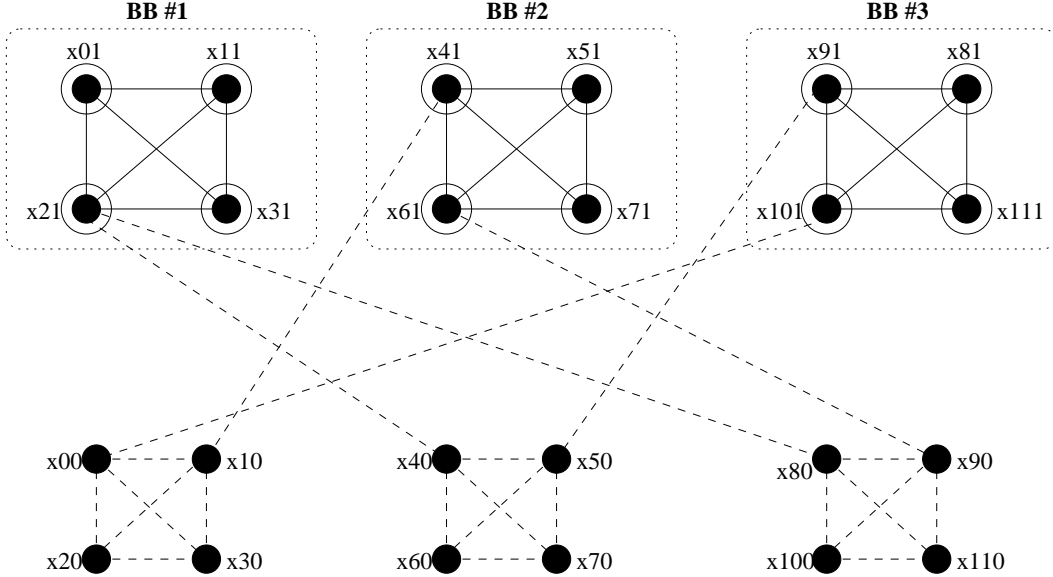


Figure 3: KeyGraph results of a 3 4-bit trap functions. KeyGraph clearly identifies BBs as a cluster of high-frequency terms. The deceptive attractors are identified as weak clusters.

the deep BBs are injected only once—after the first selection—in a GA run. The doping procedure ensures that the deep BB is at least present—albeit in very small numbers—in the population during the BB identification and exchange phase of the competent GAs.

Similar to the randomly initialized population, the competent GAs discovered the low-order BBs but failed to identify the deep BB. The injected deep BBs were lost in the subsequent—usually one, or two—generations. Similar results were obtained even when as high as 5% of the population was doped with the deep BB.

The results presented in this section clearly demonstrate that both BOA and ECGA fail to identify the deep BB and that they lose the deep BB even when the population is doped with a very few number of individuals with the deep BB. Therefore, we need to utilize other techniques to identify deep BBs when they occur in very few numbers in the population. One possible technique for so doing is chance discovery through KeyGraph which is discussed in the following sections.

### 4.3 KeyGraph and Identification of Low-Order Building Blocks

The previous section demonstrated that the competent GAs are unable to identify deep BBs when they are present in very few numbers in the population. One of the primary objectives of this study is to investigate the potential of employing KeyGraph for the identification and discovery of deep BBs. Though KeyGraph has been successful in solving a wide range of problems related to chance discovery and data mining, to the best of our knowledge, it has not been applied to identify BBs of a search problems. Therefore, we need to test whether KeyGraph can first identify low-order BBs. If KeyGraph is unable to identify BBs of low order, which is a much easier task than identifying deep BBs, then it would be less likely to succeed in identifying deep BBs. Therefore we tested the capability of KeyGraph in identifying building blocks of the  $3 \times 4$  problem.

The algorithmic procedure employed in using KeyGraph for identifying the building blocks of  $3 \times 4$  problem is as follows:

1. Initialize the population randomly.

2. Evaluate the fitness values of individuals in the population.
3. Perform tournament selection (Goldberg, Korb, & Deb, 1989) to make more copies of better individuals. Selection biases the population towards highly fit individuals and therefore facilitates the identification of BBs.
4. Convert the population of selected individuals into a document with a sentence referring to an individual and each word represents a bit value. For example the following individual 10101100 is converted to “x01 x10 x21 x30 x41 x51 x60 x70.”. This step is necessary because KeyGraph operates on a text document containing sentences constructed of words and phrases. A straight forward method of translating bit strings (individuals) is to convert each bit of an individual into a word, and each individual forms a sentence in the text document.
5. Run KeyGraph on the text document for building-block identification.

The above procedure refers to a single generation of a GA run. Therefore, the success of KeyGraph depends on whether it can identify BBs in the first generation. This requirement imposed on KeyGraph can be stringent, however, if the BBs are not identified in the early stages of a GA run, it becomes difficult, if not impossible, to identify BBs in the later stages of a GA run.

The results of KeyGraph on the  $3 \times 4$  problem is shown in figure 3. In the  $3 \times 4$  problem, each of the building blocks has a fair market share in the initial population (6.25% of the population). Selection further increases the market share of the BBs. Therefore, KeyGraph should identify each BB as a cluster of high-frequency terms. From figure 3, we can clearly see that KeyGraph indeed identifies individual BBs as a cluster of high-frequency terms (bits). Furthermore, attached with each BB is a cluster (with dashed lines) of the deceptive attractor of other two BBs. The deceptive attractor is also identified by the KeyGraph because in the initial stages of the GA run, many individuals of the population get one BB right and the others are the deceptive attractors. Therefore, if an individual has the first BB, most probably the second and third partitions would have the deceptive attractor. Results shown in figure 3 indicate that KeyGraph is not only effective in identifying the BBs, but also can be used to visualize a GA population. It should be noted that results on larger problems (up to  $m = 10$ ) were qualitatively similar to figure 3.

In this section, we demonstrated that KeyGraph is capable of identifying low-order building blocks using a simple  $3 \times 4$  problem. Next, we discuss deep-BB identification using KeyGraphs.

#### 4.4 KeyGraph and Identification of Deep Building Blocks: A Naïve Approach

The previous section showed that KeyGraph was indeed capable of identifying the building blocks of low order. This section investigates whether the KeyGraph procedure is capable of identifying deep BBs. In particular, we consider the  $2 \times 4 + 12$  problem. As mentioned in section 4.2, the chances of creating a deep BB from smaller pieces is highly improbable and similarly its identification is also impossible if it has no market share in the population. Therefore, we inject one or two individuals with the deep BB in the selected population (after step 3 in the algorithm procedure mentioned in previous section (Section 4.3). Note that the injection procedure is similar to that of section 4.2.

The KeyGraph on such a population is shown in figure 4. A population size of 2000 and tournament selection with tournament size of 10 is used. Similar to the case of  $3 \times 4$  problem, the KeyGraph successfully identifies low-order BBs as clusters of high-frequency terms. The KeyGraph also discovers the deceptive attractor of the low-order building blocks. However, the KeyGraph does not identify the deep building block. Instead, it discovers the deceptive attractor of the deep

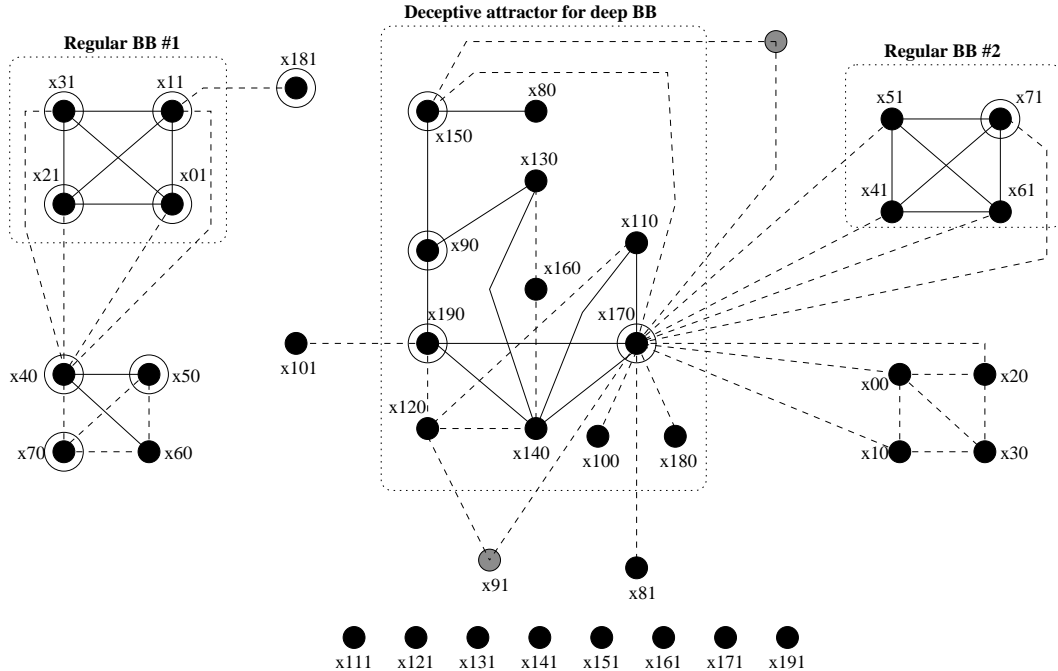


Figure 4: Identification of the deep BB through KeyGraph using the naïve approach. Low-order Building blocks are discovered as clusters of high-frequency terms. However, the KeyGraph fails to identify and discover the deep BB. The population was injected with two individuals and a population size of 2000 individuals was considered.

building block as a cluster of high-frequency terms. Qualitatively similar results were obtained for other independent runs and for different population-size and tournament-size values.

A detailed examination of the internal mechanisms of the KeyGraph technique provides the reason for its failure in discovering the deep BB. We find that KeyGraph is effective in discovering rare events as a single node cluster that are associated with clusters of high-frequency events. In other words, KeyGraph is effective if the key rare events occur in association with high-frequency events and if each of the rare events are independent of one another. However, in our case the deep BB is a rare cluster of events which may or may not be associated with other low-order building blocks. That is individuals events of the deep BB (genes belonging to the deep building block) are themselves not rare, but the cluster of genes with correct gene values is. Such a cluster of genes with correct gene values (ones) is the deep BB. Therefore, for the  $2 \times 4 + 12$  problem the KeyGraph identifies parts of the deep BB as independent clusters of single node, where each node represents the genes belonging to the deep BB.

Based on the above analysis, we reasoned that in order to use KeyGraph in its current form, we should either convert the deep BB into a single node or convert the cluster representing the deep BB into a relatively higher frequency one. In the following sections we propose two different techniques for circumventing the problem with the naïve KeyGraph approach.

#### 4.5 Two Approaches for Identifying Deep Building Blocks

The two methods we propose to aid KeyGraph in identifying deep BBs are an *aggregation* method, where we change the methodology of converting a population of individuals to the text document, and a *doping* method, where the individuals with deep BBs are injected into the population before

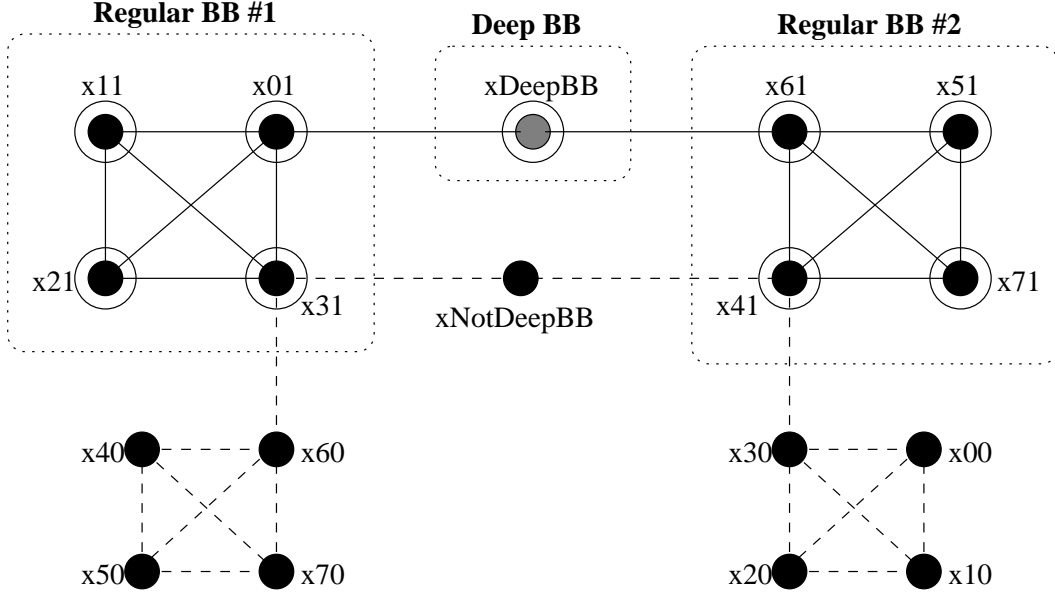


Figure 5: Identification of the deep BB through KeyGraph using aggregation. The regular building blocks are represented by clusters of high-frequency terms and the deep BB is discovered as a low probability high key event. Two individuals with the deep BBs were injected into a population of 2000 individuals.

selection. Each of these methods are discussed in the following subsections.

#### 4.5.1 Aggregation

In aggregation, instead of converting each bit into a word, we convert bits of the deep BBs into a phrase. Furthermore, the phrase indicates whether the individual has a deep BB or not. For example, the following individuals 1111 0000 010110101000, 0000 1111 1010 0001 1100, and 0000 1111 111111111111 are converted to “x01 x11 x21 x31 x40 x50 x60 x70 xNotDeepBB.”, “x00 x10 x20 x30 x41 x51 x61 x71 xNotDeepBB.”, and “x00 x10 x20 x30 x41 x51 x61 x71, xDeepBB.” respectively. Qualitatively, aggregation transforms a deep BB from being a rare cluster of genes with correct values to a rare event “xDeepBB”. Likewise all the other possible events (bit-value sequences) are converted to a single high-frequency event “xNotDeepBB”.

The careful reader will recognize aggregation as cheating, but we do so to understand whether the KeyGraph procedure will work under the circumstances that better match its capability than those of the naïve tests. We consider how the aggregation might be accomplished after considering experimental results with the mechanism enabled.

The KeyGraph result on the population processed through aggregation is shown in figure 5. The figure shows that KeyGraph identifies the regular BBs as a cluster of high-frequency terms and the deep BB is discovered as a low-probability, high-key term (rare event). Figure 5 indicates that KeyGraph effectively discovers the deep BB even though it had only two copies in a population of 2000 individuals.

Aggregation is an effective method to discover deep BBs in a search problem, however, further investigation is required to develop efficient methods of aggregation. Specifically, we need to answer how one can aggregate bits belonging to a building-block into a phrase indicating whether an individual has a deep building-block or not. Perturbation techniques such as the gene expression

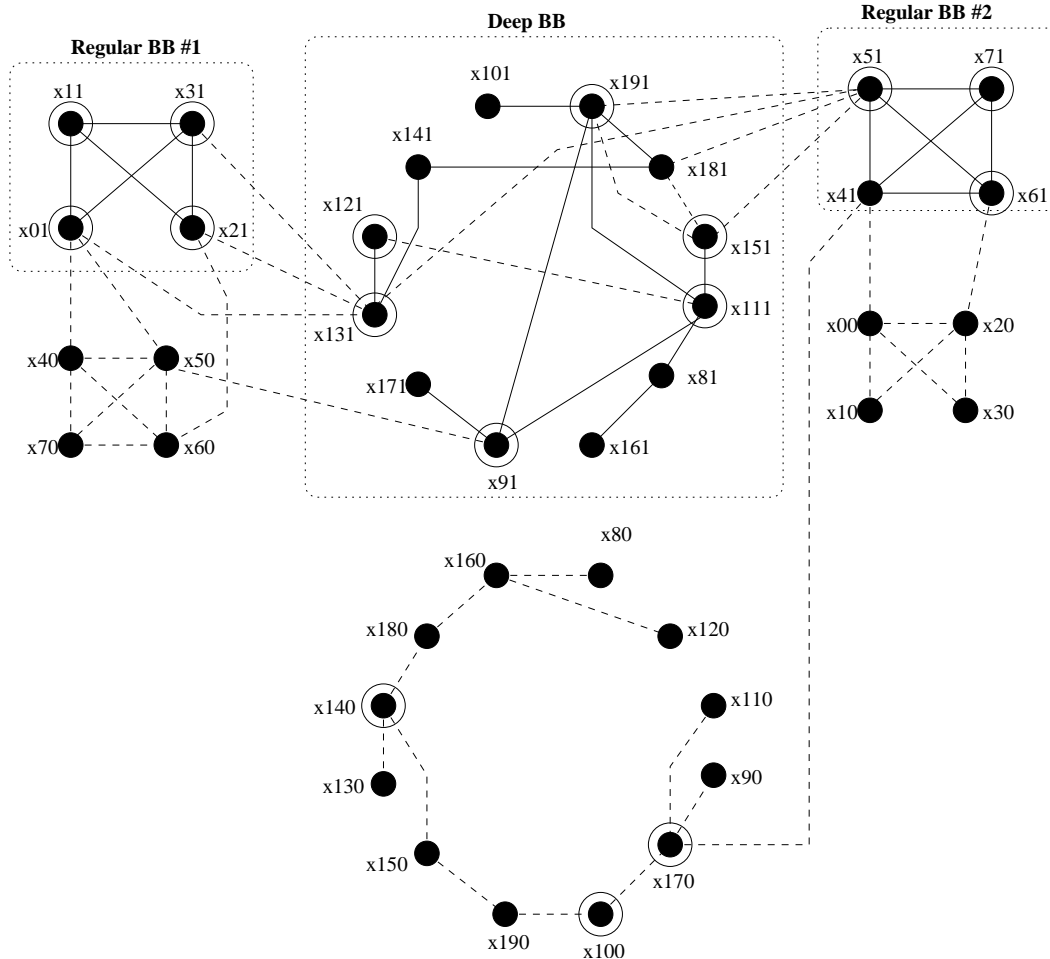


Figure 6: Identification of the deep BB through KeyGraph using the doping method. The regular building blocks are represented by clusters of high-frequency terms and the deep BB is discovered as a cluster of relatively low-frequency. Approximately 0.1% of the population was injected with individuals containing deep BBs and after selection the market share of the deep BB grew to about 3% of the population size (2000 individuals).

messy GA (GEMGA) (Kargupta, 1996), LINC and LIMD GAs (Munetomo & Goldberg, 1999) can potentially be used to differentiate a bit-string partition as being either a building-block or not. In other words, the perturbation techniques can provide with candidates for the deep BB and thus perform the aggregation process. However, the results demonstrate that aggregation helps KeyGraph discover the deep building block.

#### 4.5.2 Doping

The previous section investigated the utility of aggregation in aiding the discovery of the deep BB. This section discusses another technique, called *doping*, and its effect on discovering deep BBs.

In doping, a small number of individuals with deep BBs are injected into the population prior to selection. The injected population is then subjected to selection with high selection pressure (tournament size of 10–20), so that the deep BBs have relatively high market share as compared to the case of aggregation, but a low market share as compared to regular BBs.

The KeyGraph result on a doped population is shown in figure 6. About 0.1% of the population was injected with individuals with deep BBs. After selection about 3% of the population had individuals with deep building blocks. The figure shows that KeyGraph discovers regular building blocks as clusters of high-frequency terms and that it discovers the deep BB as a cluster of relatively low-frequency terms. Similar results were observed with different population sizes and tournament size values and for the deep BBs to be discovered about 2-5% of the population had to be doped with individuals containing deep BBs. It is interesting to note that competent GAs do not discover deep BBs even though up to 5% of the population is doped with the deep BB.

Like aggregation, doping helps KeyGraph discover deep building blocks. As with aggregation, further investigation is required to design effective doping procedures. Specifically, we need to answer how to selectively dope deep BBs either by more injection of good BBs or by increasing selection pressure as soon as deep BBs appear in the population. However, the objective of this study is to demonstrate the utility of KeyGraph in the discovery of deep BBs.

## 5 Future Work

This study demonstrates the utility of KeyGraphs in identification and visualization of deep building blocks in search and optimization problems. However, it has just scratched the surface, and much work is needed to develop practical competent GAs that capture both low-order BBs, and deep BBs. Here we have mentioned some of the key areas of future study which will enable us to do so.

**Effective aggregation techniques:** This work showed that aggregating the bits of the deep BB partition into a phrase is helpful. Perturbation techniques such as GEMGA, LINC GA, and LIMD GA can be used to preprocess the population and convert the deep partitions into a set of potential deep BB candidates.

**Efficient doping methods:** This study also showed that effective doping of the deep BB in the population can lead to its successful identification. Further studies are required to investigate how best to perform doping, especially in a real-world problems, where the deep BBs are not known a priori.

**Principled hybrids with existing competent GAs:** Existing competent GAs are very effective in identifying and exchanging regular building blocks. They do so in polynomial time (usually subquadratic). Therefore, an effective procedure to achieve innovation success for a search problem would combine KeyGraphs and existing competent GAs in a principled way (Goldberg & Voessner, 1999; Sinha & Goldberg, 2001). The result of such a hybrid should be the development of a solver which can identify both regular, and rare notions.

These are not trivial challenges, but the pilot results of this study suggest that the investment is a potentially worthy one in that it should permit deep BBs to be exploited sooner than would otherwise be possible in procedures that rely on statistically significant samples of the better substructures.

## 6 Conclusions

This paper has suggested how genetic algorithms and chance discovery may represent computational analogs of different facets of human innovation. In particular, selectorecombinative GAs, those that use selection and recombination mechanisms, have been discussed as being similar to that facet of human innovation that cross-fertilizes different notions or subsolutions to create new, possibly better

solutions to some problem at hand. Chance discovery, particularly chance discovery as exhibited by the KeyGraph procedure has been considered as that aspect of human innovation where human beings pay more attention to salient fortuitous events, thereafter trying to understand or use the happy happenstance to effect useful change. The paper has reviewed GAs, KeyGraphs, basic mechanisms, and the innovation connection in some detail, and a combine of chance discovery and GAs has been suggested and partially tested.

In particular, KeyGraphs have been tried as a solution to the problem of deep building blocks. Large subsolutions of difficult problems are hard to evolve from lower order building blocks, and sizing populations to include the statistically significant samples of high-order BBs is an exponentially growing nightmare. These facts place a premium on developing mechanisms to identify rare, salient fortuitous events when they occur, and to determine whether or not they may be used reliably as parts of a better overall solution to the problem at hand.

Pilot experiments were run on two test functions. A simple test function with uniformly sized and scaled BBs shows that KeyGraphs can identify and help visualize the usual low-order BB that competent GAs regularly detect. A test function with two low-order BBs ( $k = 4$ ) and one high-order BB ( $k = 12$ ) was used to show that (1) ordinary competent GAs fail to exploit the deep BB, even when the population is seeded with a substantial number of copies of it and (2) a naive implementation of KeyGraphs for deep BBs also fails. This led to two methods to get KeyGraphs to identify the deep BB. One technique called aggregation, assumed that the deep BB could be identified whole as a unit. Under these circumstances, KeyGraphs are able to identify and visualize the deep BB, but assuming the existence of an aggregation procedure is tantamount to assuming local solutions to the linkage learning problem. The other technique, called doping, assumes that the deep BB is present in significant numbers. Under this assumption, KeyGraphs are able to identify the deep BB without the aggregation assumption. Doping may be obtained in practice through significant selection pressures, and using high selection pressures may form a means of extracting chance occurrences of deep BBs from relatively small populations.

We think the results of the paper are interesting computationally, graphically, and scientifically. Computationally, the aggregation and doping pilot results are promising and deserve further inquiry to (1) understand the mechanisms involved theoretically (2) develop practical deep BB procedures integrating the best of competent GA practice and KeyGraphs. Graphically, one of the unintended benefits of this work has been the usefulness of graphical BB visualization using KeyGraphs. Highly dimensional optimization problems are notoriously difficult to visualize and the KeyGraph procedure has proved helpful in understanding graphically what is going on inside a run. Even if KeyGraphs were not useful for finding deep structure, this benefit would be worth exploring on its own. Scientifically, better understanding of different facets of human innovation, especially understanding that is computational and quantitative, is a useful thing in and of itself. The scientific contributions of this paper in this arena are largely conjectural, and we certainly do not claim to have found the fingerprint or smoking gun of human innovation in all its manifest complexity in our pilot experiments. But we believe that the connections we suggest are plausible, appeal to our own intuitions (and qualitative theories) about innovation, and deserve closer scrutiny from psychologists and other professionals who study these matters for a living. Regardless whether these latter suggestions are widely accepted, we believe the computational and graphical merits of the work deserve immediate scrutiny by those who are interested in chance discovery and genetic algorithms.

## 7 Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163, and the National Science Foundation under grant DMI-9908252. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

Kumara Sastry was also supported by the Computational Science and Engineering (CSE) fellowship, University of Illinois at Urbana-Champaign.

## References

- Asimov, I., & Shulman, J. A. (Eds.) (1988). *Isaac Asimov's book of science and nature quotations*. New York, NY: Weidenfeld & Nicolson.
- Bandyopadhyay, S., Kargupta, H., & Wang, G. (1998). Revisiting the GEMGA: Scalable evolutionary optimization through linkage learning. *Proceedings of the IEEE International Conference on Evolutionary Computation*, 603–608.
- Beveridge, W. I. B. (1957). *The art of scientific investigation*. New York, NY: W. W. Norton.
- Bosman, P., & Thierens, D. (1999). Linkage information processing in distribution estimation algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference*, 60–67.
- Chen, Y.-P., & Goldberg, D. E. (2002). Introducing start expression genes to the linkage learning genetic algorithm. *Parallel Problem Solving from Nature*, 7, 351–360. (Also IlliGAL Report No. 2002007).
- Deb, K., & Goldberg, D. E. (1993). Analyzing deception in trap functions. *Foundations of Genetic Algorithms*, 2, 93–108. (Also IlliGAL Report No. 91009).
- Deb, K., & Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, 10, 385–408. (Also IlliGAL Report No. 92001).
- Fine, G. A., & Deegan, J. G. (1996). Three principles of Serendip: insight, chance, and discovery in qualitative research. *Qualitative Studies in Education*, 9(4), 434–447.
- Fukuda, H., & Ohsawa, Y. (2001). Discovery of rare essential food by community navigation with KeyGraph - An introduction to data-based community marketing. *Proceedings of the Fifth International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES2001)*, –.
- Goldberg, D. E. (1983). *Computer-aided gas pipeline operation using genetic algorithms and rule learning*. Doctoral dissertation, University of Michigan, Ann Arbor, MI. University Microfilms No. 8402282.
- Goldberg, D. E. (1987). Simple genetic algorithms and the minimal, deceptive problem. In Davis, L. (Ed.), *Genetic algorithms and simulated annealing* (Chapter 6, pp. 74–88). Los Altos, CA: Morgan Kaufmann.

- Goldberg, D. E. (1989). Sizing populations for serial and parallel genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, 70–79. (Also IlliGAL Report No. 88004).
- Goldberg, D. E. (1991). *Six steps to GA happiness*. Paper presented at Oregon Graduate Institute, Beaverton, OR.
- Goldberg, D. E. (1993). The Wright Brothers, genetic algorithms, and the design of complex systems. In Schaffer, J. D. (Ed.), *Proceedings of the Symposium on Neural-Networks: Alliances and Perspectives in Senri 1993* (pp. 1–7). Osaka, Japan: Senri International Information Institute.
- Goldberg, D. E. (1999). The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In Bentley, P. (Ed.), *Evolutionary Design by Computers* (Chapter 4, pp. 105–118). San Mateo, CA: Morgan Kaufmann.
- Goldberg, D. E. (2002). *Design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer Academic Publishers.
- Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6, 333–362. (Also IlliGAL Report No. 91010).
- Goldberg, D. E., Deb, K., Kargupta, H., & Harik, G. (1993). Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the International Conference on Genetic Algorithms*, 56–64. (Also IlliGAL Report No. 93004).
- Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5), 493–530. (Also IlliGAL Report No. 89003).
- Goldberg, D. E., & Liepens, G. (1991). Theory tutorial. (Tutorial presented at the 1991 International Conference on Genetic Algorithms, La Jolla, CA).
- Goldberg, D. E., Sastry, K., & Latoza, T. (2001). On the supply of building blocks. *Proceedings of the Genetic and Evolutionary Computation Conference*, 336–342. (Also IlliGAL Report No. 2001015).
- Goldberg, D. E., & Voessner, S. (1999). Optimizing global-local search hybrids. *Proceedings of the Genetic and Evolutionary Computation Conference*, 220–228. (Also IlliGAL Report No. 99001).
- Hadamard, J. (1945). *The psychology of invention in the mathematical field*. Princeton, NJ: Princeton University Press.
- Harik, G. (1999a, January). *Linkage learning via probabilistic modeling in the ECGA* (IlliGAL Report No. 99010). Urbana, IL: University of Illinois at Urbana-Champaign, General Engineering Department.
- Harik, G., & Goldberg, D. E. (1997). Learning linkage. *Foundations of Genetic Algorithms*, 4, 247–262. (Also IlliGAL Report No. 96006).
- Harik, G. R. (1999b). *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. Doctoral dissertation, University of Michigan, Ann Arbor, MI. (Also IlliGAL Report No. 97005).
- Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
- Kargupta, H. (1996). The gene expression messy genetic algorithm. *Proceedings of the International Conference on Evolutionary Computation*, 814–819.

- Kargupta, H., & Bandyopadhyay, S. (1998). Further experimentations on the scalability of the GEMGA. *Lecture Notes in Computer Science*, 1498, 315–324.
- Larrañaga, P., & Lozano, J. A. (Eds.) (2002). *Estimation of distribution algorithms*. Boston, MA: Kluwer Academic Publishers.
- Munetomo, M., & Goldberg, D. E. (1999). Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 7(4), 377–398.
- Ohsawa, Y. (2001). The scope of chance discovery. In Terano, T., et al. (Eds.), *New Frontiers in Artificial Intelligence* (pp. 413). Berlin, Germany: Springer-Verlag.
- Ohsawa, Y. (2002a). Chance discoveries for making decisions in complex real world. *New Generation Computing*, 20, 143–163.
- Ohsawa, Y. (2002b). KeyGraph as risk explorer from earthquake sequence. *Journal of Contingencies and Crisis Management*, 10(3), 119–128.
- Ohsawa, Y., Benson, N. E., & Yachida, M. (1998). KeyGraph: Automatic indexing by co-occurrence graph based on building construction metaphor. *Proceedings of Advanced Digital Library Conference (IEEE ADL '98)*, 12–18.
- Ohsawa, Y., & Yachida, M. (1999). Discover risky active faults by indexing an earthquake sequence. *Proceedings of International Conference on Discovery Science*, –.
- Pelikan, M. (2002a). *Bayesian optimization algorithm: From single level to hierarchy*. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL.
- Pelikan, M. (2002b). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21, 5–20. (Also IlliGAL Report No. 99018).
- Pelikan, M., & Goldberg, D. E. (2001). Escaping hierarchical traps with competent genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference*, 511–518.
- Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (2000). Bayesian optimization algorithm, population sizing, and time to convergence. *Proceedings of the Genetic and Evolutionary Computation Conference*, 275–282. (Also IlliGAL Report No. 2000001).
- Porter, M. F. (1980). An algorithm for suffix stripping. *Automated library and information systems*, 14(3), 130–137.
- Roberts, R. M. (1989). *Serendipity: Accidental discoveries in science*. New York, NY: John Wiley & Sons.
- Sastry, K., & Goldberg, D. E. (2000). On extended compact genetic algorithm. *Late Breaking Paper in the Genetic and Evolutionary Computation Conference*, 352–359. (Also IlliGAL Report No. 2000026).
- Sinha, A., & Goldberg, D. E. (2001). Verification and extension of the theory of global-local hybrids. *Proceedings of the Genetic and Evolutionary Computation Conference*, 591–597. (Also IlliGAL Report No. 2001010).