1998

Adaptive Linkage Crossover

Ayed A. Salman
Syracuse University, ayed@top.cis.syr.edu

Kishan Mehrotra
Syracuse University, mehrotra@syr.edu

Chilukuri K. Mohan
Syracuse University, ckmohan@syr.edu

Adaptive Linkage Crossover

Ayed A. Salman, Kishan Mehrotra[1], and Chilukuri K. Mohan
2-120 CST, Dept. of EECS
Syracuse University
Syracuse, NY 13244-4100
ayed/kishan/mohan@top.cis.syr.edu

ABSTRACT

Problem-specific knowledge is often implemented in search algorithms using heuristics to determine which search paths are to be explored at any given instant. As in many other AI search methods, utilizing this knowledge will lead a genetic algorithm (GA) faster towards better results. In many problems, crucial knowledge is to be found not in individual components, but in interrelations between those components. For such problems, we develop an interrelation (linkage) based crossover operator that has the advantage of *liberating GAs from the constraints imposed by the fixed representations generally chosen for problems.* The strengths of linkages between components of a chromosomal structure can be explicitly represented in a linkage matrix and used in the reproduction step to generate new individuals. For some problems, such a linkage matrix is known *a priori* from the nature of the problem. In other cases, the linkage matrix may be learned by successive minor adaptations during the execution of the evolutionary algorithm. This paper demonstrates the success of such an approach for several problems.

Keywords Genetic algorithms, crossover operators, deception, linkage probabilities.

[1]Author for correspondence

1 Introduction

What is the role of crossover in a GA? Current explanations mention maintaining diversity in the population while preserving good building blocks. What constitute good building blocks? The Schema Theorem (Holland, 1975) and the Building Block Hypothesis (Goldberg, 1989) imply that successive generations contain an exponentially increasing number of instances of short, low-order, high (sampled) fitness schema, assuming the use of 1-point crossover (1PTX) and fitness-proportionate reproduction selection. However, these results do not say anything about schema whose order or defining length is non-trivially large. If a problem is such that the first and last components of individuals need to evolve together, such linkages are so likely to be disrupted that an operator such as 1PTX becomes useless at maintaining such linkages. As in the case of any other weak (general-purpose) operator, there are as many problems for which 1PTX works well as those for which 1PTX does not work well, paraphrasing the *No Free Lunch Theorems* (Wolpert and Macready, 1995). Some deceptive problems can be solved using linear transformations that change representation (Lippens and Vose, 1991) and hence linkage structure. Harik and Goldberg (1997) have formulated a "Linkage friendly" crossover operator very similar to the two-point crossover operator. However, discovering the linkages of distant components is difficult. By contrast, we develop operators that work well by exploiting known linkages between the components of chromosomes.

The incorporation of *pleiotropy* (one gene affects multiple traits) and *polygeny* (many chromosomal components are responsible for a single trait) remains largely unexplored by researchers in evolutionary computation, although these are believed to exist in "all systems with complex behavior" (Atmar, 1992). Is there any advantage to such complicated many-to-many (gene-to-trait) coding mechanisms? We hypothesize that such mechanisms do serve an additional purpose equivalent to encoding "linkage probabilities" that determine the likelihood with which different traits are inherited from the same parent during crossover.

Distributed representations carry the advantages of fault tolerance and robustness, which partially explains the method behind the madness of pleiotropy and polygeny in biological evolution. In addition, we argue that these mechanisms indirectly code for elaborate representations of linkages between traits, transcending the limitations of the linear sequencing of genes. If we view nature's algorithm in terms of traits rather than genes, what emerges is not a simple string (linear sequence) in which each element (trait) is connected to its immediate neighbors. Instead, we find a complex of multiple connections

between different traits implemented by the connections between multiple genes (for each trait) within the linear chromosomal structure.

By analogy with nature, a GA with a "one-gene-per-trait" (or a "one-sequence-of-genes-per trait") representation must not rely merely on a sequential arrangement of the genes with implied strong linkages only among neighboring genes. In order to be effective,

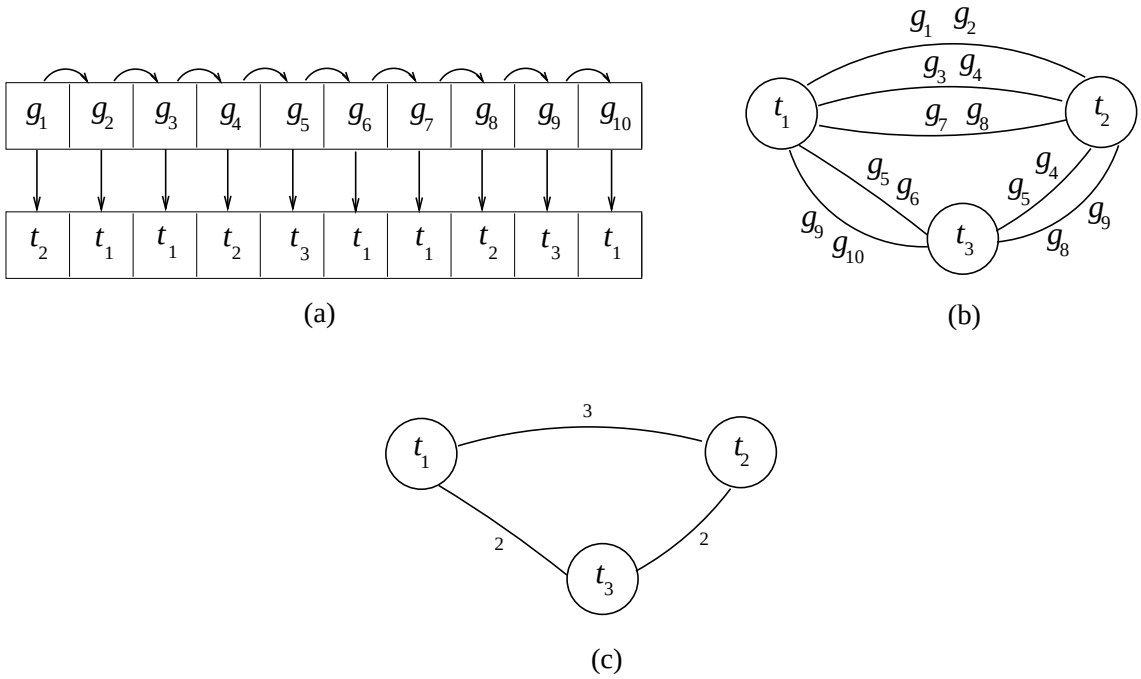


Figure 1: Abstract interpretation of pleiotropy, polygeny, and linkages: (a) Simple gene-level linkages; pleiotropy and polygeny trait representation, (b) Trait-level linkages; adjacencies in gene level representation, (c) Abstraction of linkage strengths between traits.

a GA must instead allow for an adaptation for the representations of multiple connections between traits. We suggest that this be accomplished via "linkage probabilities": each individual in the population is a collection of traits whose "sequencing" is irrelevant. The population consists of a number of individuals as well as a representation of the species-specific linkages between traits. For simplicity of implementation and computational considerations, we propose that a two-dimensional array of first-order linkage values is adequate for this purpose. In keeping with this understanding, we consistently use the term "individual" instead of the traditional "chromosome" in reference to the members of the population.

If we use a "one-gene-per-trait" encoding, then such linkage probabilities must be explicitly stated and used by operators of the GA.

Problem-specific knowledge can be utilized by genetic search in several ways, one of which addresses the "move-generation" step: how can we define genetic operators that utilize knowledge specific to each class of problems? Maini, Mehrotra, Mohan & Ranka (1994) have addressed two ways of doing this: incorporating allele-specific biases, and utilizing information about the history of genetic search. Another simple approach is found in the 'particle swarm optimization' system (see Kennedy and Eberhardt, 1995). This paper explores a third, crucial approach, the development of operators that exploit linkages between components of problem representations.

In order to show the effectiveness of our operators, we tested them against the well-known one-point, two-point and uniform crossover operators. Our test set was composed of three problems, two of which (*30bits, order-three problem* (Goldberg, Korb & Deb, 1989) and *bipolar, order-six problem* (Goldberg, Deb & Horn, 1992)) are theoretically constructed to be GA-deceptive, yet have a well-known linkage structure, and the third of which is the well-known bipartitioning problem, whose exact linkage structure is unknown *a priori*. Experiments show the superiority of the linkage crossover over other operators.

The proposed framework is described in Section 2. Section 3 defines a class of crossover operators using this approach and the relation of these to crossover operators traditionally used in GAs. Section 4 elucidates our algorithms, Section 5 presents experimental results that support our new approach, Section 6 evaluates the adaptive procedure, and Section 7 concludes.

2 Framework

This section describes the overall framework establishing the relation between crossover and probabilistic inference. In our notation, $i \leftarrow p_j$ denotes the event that the ith gene in the offspring comes from the jth parent when crossover occurs, π denotes a partial inheritance assignment, and $\pi(i)$ denotes the parent p_1 or p_2 from which an offspring inherits the ith gene. We assume that there is no *a priori* bias toward either parent, p_1 or p_2, i.e., symmetry is assumed.

Without loss of generality we assume that parents p_1 and p_2 generate only one offspring.[2] In examining which genes are inherited from which parent, we restrict attention

[2]Operators that generate multiple offspring can easily be viewed as the results of multiple applications of operators that generate single offspring. For instance, one-point crossover applied to p_1, p_2 at position k

to those genes where the parents differ. Also, let $\{x_1, x_2, \ldots, x_n\}$ denotes a permutation of $\{1, \ldots, n\}$, where n is the number of genes (components) in a chromosome.

Classical crossover operators break linkages among some genes, irrespective of potential dependence among them, whereas it would be desirable to use a crossover operator that honors special attractions between sets of alleles. This dependence can be reformulated in terms of problem-specific conditional probabilities. Specifically, the crossover operator should address: if the ith gene is inherited from parent p_1, then what is the probability that the jth gene is also inherited from parent p_1? More generally, if $x_1, x_2, \ldots, x_i$ are the positions of genes inherited from $\pi(x_1), \pi(x_2), \ldots, \pi(x_i)$, respectively, then what is the probability that the x_{i+1}th element of the offspring is inherited from p_1? We view crossover as accomplishing this probabilistic inference task, where the probability depends on the problem, and is "hard-coded" into biological chromosomal structures via the mechanisms of pleiotropy and polygeny.

Special cases:

One-Point Crossover

In one-point crossover (1PTX), the structure of linkages is linear, similar to probabilistic inference with a chain structure. The only linkage between the $(i+1)$th gene and the $(i-1)$th gene is through the ith gene. The linkage probabilities associated with 1PTX may be described in the following manner:

$$P(i \leftarrow p_1 \mid (i+a) \leftarrow p_1 \ \& \ (i-b) \leftarrow p_1) = 1, \text{ where } a > 0, b > 0,$$

$$P(i \leftarrow p_1 \mid (i+a) \leftarrow p_2 \ \& \ (i-b) \leftarrow p_2) = 0, \text{ where } a > 0, b > 0,$$

and

$$P(i \leftarrow p_1 \mid (i+1) \leftarrow p_1 \ \& \ (i-1) \leftarrow p_2) = 0.5.$$

One-point crossover is expected to work well when the linkages in the problem are of a similar linear nature, e.g., when the desirable "building blocks" consist of alleles for physically proximate genes.

is generally defined to produce two offspring, but can be viewed as equivalent to two separate applications of an operator that generates only one offspring; the second application is obtained by reversing the order of the parents.

The cases of two-point and k-point crossovers can be derived in a similar manner.

Uniform Crossover

In uniform crossover (UX), no linkages are preserved.

$$P(i \leftarrow p_1 | j \leftarrow \pi(j)) = 0.5, \forall j \neq i.$$

Whether 1PTX or uniform crossover works better on a problem depends on whether the problem itself has implicit linkages of the kind preserved by 1PTX.

3 Linkage Crossover

A general class of crossover operators can be formulated using the framework of linkage probabilities. This class is referred to as *General Linkage Crossover* (GLinX), and is described below. We use the following additional notation:

- $P(x_{i+1} : x_1, \ldots, x_i; \pi)$ denotes the conditional probability that the x_{i+1}th position in the child chromosome comes from parent p_1, given that the x_jth position comes from parent $\pi(j); j = 1, \cdots, i$, i.e.,

$$P(x_{i+1} : x_1, \ldots, x_i; \pi) = P\Big(x_{i+1} \leftarrow p_1 | x_1 \leftarrow \pi(x_1) \& \ldots \& x_i \leftarrow \pi(x_i)\Big).$$

- For the special case when $x_1, \ldots, x_i$ are all inherited from p_1, the "linkage probability"

$$L(x_{i+1} : x_1, \ldots, x_i) \text{ denotes } P\Big(x_{i+1} \leftarrow p_1 \mid x_1 \leftarrow p_1 \ \& \ \ldots \ \& \ x_i \leftarrow p_1\Big).$$

Computation of offspring components using GLinX: Suppose p_1 and p_2 differ from each other in k locations. Let $x_1, \ldots, x_k$ denote these locations.

- Alleles for the first two locations, x_1 and x_2, of the offspring are inherited from p_1 and p_2 respectively.

- Alleles for the remaining $(k-2)$ locations are successively assigned as follows: Suppose i additional locations, $x_3, \ldots, x_{i+2}$, have been assigned alleles from p_1 or p_2. Then. the $(i+3)$th component of the offspring is inherited from parent p_1 with probability $P(x_{i+3} : x_1, \ldots, x_{i+2}; \pi)$.

Example 1 *Consider* $p_1 = (0,0,1,0,0), p_2 = (0,1,0,1,1)$, *differing in the last three positions* $(k = 3)$. *The first position in the offspring is assigned 0, common to both parents. Let* $(x_1, x_2, x_3, x_4) = (2,4,3,5)$. *The second* ($x_1$*th) position in the offspring is chosen from parent* p_1 *and the fourth* (x_2*th) position in the offspring is chosen from* p_2. *Next, the third* (x_3*th) position in the offspring is chosen from* p_1 *with probability* $P(x_3 : x_1, x_2; \pi) = P(x_3 \leftarrow p_1 | x_1 \leftarrow p_1 \& x_2 \leftarrow p_2)$. *Suppose it is chosen from* p_1. *Finally, the fifth* (x_4*th) position in the offspring is chosen from* p_1 *with probability* $P(x_4 : x_1, x_2, x_3 : \pi) = P(x_4 \leftarrow p_1 | x_1 \leftarrow p_1 \& x_2 \leftarrow p_2 \& x_3 \leftarrow p_1)$.

Only in the ideal case, would probabilities $P(x_{i+1} \leftarrow p_1 | x_1 \leftarrow \pi(x_1) \& \ldots \& x_i \leftarrow \pi(x_i))$ be available for each i. There are far too many joint linkage probabilities to be specified and these would be impossible to specify even for problems whose nature is relatively well understood. In practice, these have to be estimated or approximated based on limited information, a task similar to that of probabilistic reasoning with uncertainty in expert systems while making conditional independence assumptions. For instance, the expert systems literature addresses the estimation of $P(A|B \ \& \ C)$ given only $P(A|B)$ and $P(A|C)$ in addition to the priors. For specific problems, a dependency structure may be available, enabling calculations of such quantities. This is the approach we have taken, described in later sections.

A first step toward using linkage information would be to develop a crossover operator that makes use of *pairwise linkage*, $L(x_i : x_j)$. Pairwise linkages among genes are considered to be "first order" linkages. Information about such linkages is most likely to be available as domain knowledge for practical problems. For instance, in the graph partitioning problem, the connection weight between nodes suggests a choice for the corresponding linkage probability. Note that $P(x_j \leftarrow p_i | x_k \leftarrow p_i) = L(x_j : x_k)$ We assume that conditional symmetry prevails, i.e.,

$$P(x_j \leftarrow p_1 | x_k \leftarrow p_2) = P(x_j \leftarrow p_2 | x_k \leftarrow p_1) = 1 - L(x_j : x_k). \tag{1}$$

We assume that the problem description specifies the first order linkage probabilities, $L(i : j)$, for each i, j; no other information is available. Other probabilities, such as $P(x_{i+1} : x_1, \ldots, x_i; \pi)$ need to be estimated from $L(x_{i+1} : x_1), \ldots, L(x_{i+1} : x_i)$.

This is analogous to the expert system's task of combining the conclusions obtained from multiple sources of uncertain knowledge. We examine two heuristics used in the expert systems literature. For any two events A and B, Bayes' rule gives

$$\begin{aligned} P(A|B) &= \frac{P(B \cap A)P(A)}{P(B \cap A)P(A) + P(B \cap \overline{A})P(\overline{A})} \\ &= \frac{P(B \cap A)}{P(B \cap A) + o(A)P(B \cap \overline{A})} \end{aligned} \tag{2}$$

where $o(A) = P(\overline{A})/P(A)$ represents the odds (of the prior probabilities of occurrence) of A. Applying Equation (2) to the problem of interest gives:

$$\begin{aligned} P(x_{i+1} : x_1, \ldots, x_i; \pi) &= P(x_{i+1} \leftarrow p_1 | \cap_{j=1}^{i} x_j \leftarrow \pi(x_j)) \\ &= \frac{P(\cap_{j=1}^{i} x_j \leftarrow \pi(x_j) | (x_{i+1} \leftarrow p_1)}{P(\cap_{j=1}^{i} x_j \leftarrow \pi(x_j) | (x_{i+1} \leftarrow p_1) + o(x_{i+1} \leftarrow p_1) P(\cap_{j=1}^{i} x_j \leftarrow \pi(x_j) | (x_{i+1} \leftarrow p_2))} \end{aligned} \tag{3}$$

It would be reasonable to replace the odds ratio $o(x_{i+1}) = P(x_{i+1} \leftarrow p_1)/P(x_{i+1} \leftarrow p_2)$ by 1; there is no *a priori* preference that the allele in the $i+1$th position of the offspring should come from parent p_1 or p_2. In the rest of the development, we assume that the prior probabilities are the same for inheriting any component from either parent.

Conditional independence assumption. Using this assumption, one writes

$$P(\cap_i A_i | C) = \prod_i P(A_i | C)$$

for arbitrary events $(C, A_1, A_2, \ldots)$. In the present context it is assumed that

$$P(x_j \leftarrow p_1 \ \& \ x_k \leftarrow p_1 \mid x_{i+1} \leftarrow p_1) = P(x_j \leftarrow p_1 \mid x_{i+1} \leftarrow p_1) P(x_k \leftarrow p_1 \mid x_{i+1} \leftarrow p_1),$$

and

$$P(x_j \leftarrow p_1 \ \& \ x_k \leftarrow p_1 \mid x_{i+1} \leftarrow p_2) = P(x_j \leftarrow p_1 \mid x_{i+1} \leftarrow p_2) P(x_k \leftarrow p_1 \mid x_{i+1} \leftarrow p_2).$$

Application of conditional independence assumption to Equation (3) gives

$$P(x_{i+1} : x_1, \ldots, x_i; \pi) = \frac{\prod_{j=1}^{i} P(x_j \leftarrow \pi(x_j) | x_{i+1} \leftarrow p_1)}{\prod_{j=1}^{i} P(x_j \leftarrow \pi(x_j) | x_{i+1} \leftarrow p_1) + \prod_{j=1}^{i} P(x_j \leftarrow \pi(x_j) | x_{i+1} \leftarrow p_2)}$$

For the purpose of easy evaluation this expression can be further simplified.

Combining positive and negative evidence: As considered earlier, parents p_1 and p_2 differ in genes $\{x_1, \ldots x_j\}$. In an offspring of p_1 and p_2, some of the genes in $\{x_1, x_2, \ldots, x_i\}$ are inherited from parent p_1, and others from p_2. Let $S_{i,1}$ be the set of genes inherited from parent p_1 and $S_{i,2}$ be the set of genes inherited from parent p_2, then the above equation can be written as

$$P(x_{i+1} : x_1, \ldots, x_i; \pi) = \frac{h_1}{h_1 + h_2}$$

where

$$\begin{aligned} h_1 &= \prod_{j \in S_{i,1}} P(x_j \leftarrow \pi(x_j) | x_{i+1} \leftarrow p_1) \prod_{j \in S_{i,2}} P(x_j \leftarrow \pi(x_j) | x_{i+1} \leftarrow p_1) \\ &= \prod_{j \in S_{i,1}} L(x_j : x_{i+1}) \prod_{j \in S_{i,2}} (1 - L(x_j : x_{i+1})) \end{aligned}$$

and

$$\begin{aligned} h_2 &= \prod_{j \in S_{i,1}} P(x_j \leftarrow \pi(x_j) | x_{i+1} \leftarrow p_2) \prod_{j \in S_{i,2}} P(x_j \leftarrow \pi(x_j) | (x_{i+1} \leftarrow p_2) \\ &= \prod_{j \in S_{i,1}} (1 - L(x_j : x_{i+1})) \prod_{j \in S_{i,2}} L(x_j : x_{i+1}) \end{aligned}$$

Thus, approximation based on the independence assumption suggests that a joint linkage probability such as $L(x_{i+1} : y_1, y_2, \ldots y_j)$ can be estimated based on the pairwise linkage probabilities $L(x_{i+1} : y_1)$, $L(x_{i+1} : y_2), \ldots, L(x_{i+1} : y_j)$. The amount of space taken up by these pairwise linkage probabilities is $O(\text{number of genes per chromosome})^2$, which is reasonable for most problems. For problems amenable to a hierarchical decomposition, efficient sparse matrix representations can be used to reduce space requirements considerably. Problem-specific information can also be easily stated in terms of pairwise linkage probabilities; a local property that examines two components.

The method described above employs pairwise independence to approximate the general linkage probabilities, and is called the LinX crossover operator.

3.1 Adaptive LinX

Few problems are understood well enough that the precise linkage probabilities are known *a priori*. Indeed, the main reason for "tinkering" with several operators is ignorance of relationships between different genes. In such cases, the hardest problem becomes that

of learning the linkage probabilities on the fly, during the application of the evolutionary algorithm to the problem. The neural networks literature provides one useful paradigm for such adaptation: Hebb's rule states that the simultaneous (synchronous) excitation of two neurons results in a strengthening of the connections between them, while asynchronous activation for two neurons will result in a weakening of the connections. The linkage probabilities are analogous to "connection strengths" (weights attached to edges between nodes) in neural networks. In the context of learning pairwise linkage probabilities, Hebb's rule may be adapted as follows: The fitness of the offspring resulting from a crossover should be used to judge the efficacy of the linkage probabilities used for that crossover step, leading to a small change in the same. We use the following adaptation of Hebb's rule to estimate pairwise linkage probabilities.

Let X be an $n \times n$ matrix whose entries ($X[j,k]$) are initially randomly assigned to lie in the interval $(-1, 1)$ and changes according to the Hebb rule. The magnitude of amount of change in $X[j,k]$ is directly proportional to $(f(\text{offspring}) - \overline{f})$. A change in $X[j,k]$ is made whenever both genes in the pair (j,k) are inherited from parent p_1 or p_2; a positive change if $f(\text{offspring}) > \overline{f}$, negative otherwise. Over the course of many generations, $|X[j,k]|$ can grow to be arbitrarily large. To obtain a value $\in [0,1]$, interpretable as a probability, we use the linear transformation (subtract $\min_{\ell,m}(X[\ell,m])$ and divide by $(\max_{\ell,m}(X[\ell,m]) - \min_{\ell,m}(X[\ell,m]))$. Finally, since the proposed changes in $X[j,k]$ are meant to reflect changes in the joint probability

$$P[(x_j \leftarrow p_1 \& x_k \leftarrow p_1)]$$

which, under our assumptions, cannot be larger than $1/2$ at all times, the joint probability matrix is obtained as J where

$$J[j,k] = \frac{0.5(X[j,k] - \min_{\ell,m}(X[\ell,m])}{(\max_{\ell,m}(X[\ell,m]) - \min_{\ell,m}(X[\ell,m])}.$$

Associated conditional probabilities

$$\begin{aligned} P(x_j \leftarrow p_1 | x_k \leftarrow p_1) &= \frac{P[x_j \leftarrow p_1 \& x_k \leftarrow p_1]}{P(x_j \leftarrow p_1)} \\ &= 2J[j,k] \end{aligned}$$

will lie between 0 and 1.

The adaptive linkage crossover algorithm described in Figure 3 is derived from this heuristic.

4 LinX and Adaptive LinX Algorithms

The canonical genetic algorithm implemented here has the following characteristics:

- Chromosomes are randomly initialized.
- Roulette wheel selection methodology is used to control the mating process.
- The best 10% of the existing population is merged with the best 90% of the generated population.
- Necessary adjustments on chromosomes are applied when required by the problem constraints (e.g., in graph bipartitioning problems, there must be equally many alleles of each kind).
- The whole population, except the best individual, is reinitialized if $|\text{average fitness} - \text{best fitness}| < \epsilon$, a small threshold.

A high-level description of LinX crossover is given in Figure 2. The adaptation steps required by Adaptive LinX are shown in Figure 3.

5 Results

In this section, we address several general questions with respect to the performance of LinX crossover and ALinX algorithm. Specifically, we address the following questions.

- If there are definite known linkages between genes, will LinX outperform other general purpose crossover operators?
- Will ALinX be able to perform competitively when compared to other crossover operators?

Three benchmark problems are used to address the above questions. These benchmark problems belong to two major categories: the deceptive-problem category, and the unknown linkage-structure category.

LinX Crossover:

- Copy all common genes from parents to the offspring. For the remaining genes of the offspring, use the following steps.
- For two randomly chosen genes, select one allele from parent p_1 and the other from p_2.
- Find the remaining (unallocated) offspring genes iteratively as follows:
  - Let $S_1 = \{y_1, \ldots, y_j\}$ be the offspring's genes inherited so far from parent p_1.
  - Let $S_2 = \{z_1, \ldots, z_k\}$ be the offspring's genes inherited so far from parent p_2.
  - Randomly select i, a gene whose value has not yet been determined. Calculate "RelativeLinkage"

$$RL_i = \frac{h_1}{h_1 + h_2},$$

$$h_1 = \prod_{j \in S_1} L(x_j : x_{i+1}) \prod_{j \in S_2} (1 - L(x_j : x_{i+1}))$$

and

$$h_2 = \prod_{j \in S_1} (1 - L(x_j : x_{i+1})) \prod_{j \in S_2} L(x_j : x_{i+1})$$

Generate a random number $0 \leq r \leq 1$ from the uniform distribution and assign

$$o[i] = \begin{cases} p_1[i] \text{ if } r \leq RL_i \\ p_2[i] \text{ otherwise.} \end{cases}$$

- Adjust the generated offspring, if required by the problem constraints.

Figure 2: A high-level description of LinX crossover

5.1 GA Deceptive Problems

The fundamental schema theorem states that the genetic algorithm works by giving the most highly fit schemata an exponentially growing presence in the population, thereby less fit schemata will diminish quickly. This reasoning directs the attention of GA-researchers to a special group of problems known as "GA-deceptive" problems (Bethke, 1980; Goldberg, 1989b,c; Forrest and Mitchell, 1993), where the most competitive schemata are much

ALinX Linkage Adaptation Procedure:

Initial step: Initialize all entries L$[j,k]$'s randomly with positive real numbers between 0 and 1 (via $X[j,k] \in [-1,1]$, as explained in the text).

Reproduction step: Do the following for all offspring generated in this iteration and all $j \neq k$.

- Let $\bar{f}$ be the average fitness of the current population.
- Let o be an offspring of p_1 and p_2 produced in the current generation using LinX.
- Let $p_1[j]$ denote the jth allele of p_1 and $p_2[k]$ denotes the kth allele of p_2.
- Let $g(o) =$ fitness$(o) - \bar{f}$.
- For each j,k, where parents p_1 and p_2 differ in the jth and kth positions and both alleles of the offspring come from the same parent, (i.e., $p_1[j] \neq p_2[j], p_1[k] \neq p_2[k]$ and, $o[j] = p_i[j]$ and $o[k] = p_i[k]$ for $p_i \in \{p_1, p_2\}$) adapt the linkage matrix as described below.

$$\begin{aligned} \triangle X[j,k] &= \eta \times g(o) \\ \mathrm{L}[j,k] &= \frac{(X[j,k] - \min_{\ell,m}(X[\ell,m])}{(\max_{\ell,m}(X[\ell,m]) - \min_{\ell,m}(X[\ell,m])} \end{aligned}$$

Figure 3: A high-level description of Linkage matrix adaptation.

different than the optimal one. A problem with a relatively larger number of local optima than global optima may then mislead the GA towards the wrong attractor. Goldberg, Deb & Horn (1992), Goldberg and Richardson (1987), Goldberg, Korb, & Deb (1989), and others have proposed different problems of this kind. Two of the problems proposed by Goldberg and his colleagues are the *order-3 30-bit deceptive problem* (Goldberg, Korb, & Deb, 1989) and the *order-6 multi-modal bipolar problem* (Goldberg, Deb & Horn, 1992). The following subsections define each problem and corresponding results.

5.1.1 Goldberg's Order-Three Problem

Goldberg, Korb, and Deb (1989) defined a 30-bit concatenation of an order-three deceptive problems that is likely to be difficult for a standard crossover operator to solve. This deceptive function depends on linkages across different genes. Linkages are expressed in terms of a subfunction. The stronger the linkage, greater is the fitness of the subfunction. Two versions of it, "Easy_{30}" and "Hard_{30}" (Whitley, 1991) are defined below. Easy_{30} is a 30-bit function, where each strongly related 3-tuple of bits (of the subfunction) are tightly and minimally distributed across the chromosome, i.e., they reside adjacent to each other. In Hard_{30}, the 3 bits of each subfunction are located far away from each other. Separated by other bits, each 3-tuple of bits are located at $i, i+10$, and $i+20$ for $i = 1, 2, \ldots, 10$. We used the following subfunction in our experiments; minor variations of the function led to similar results:

$$\begin{aligned}
\omega_3(1,1,1) &= 1.0, \quad \omega_3(0,0,0) = 0.9 \\
\omega_3(1,1,0) &= \omega_3(1,0,1) = \omega_3(0,1,1) = 0.3 \\
\omega_3(0,1,0) &= \omega_3(0,0,1) = \omega_3(1,0,0) = 0.6
\end{aligned}$$

Using this subfunction, we define

$$\text{Easy}_{3n} = \sum_{i=0}^{n-1} \omega_3(x_{3i+1}, x_{3i+2}, x_{3i+3})$$

and

$$\text{Hard}_{3n} = \sum_{i=1}^{n} \omega_3(x_i, x_{i+n}, x_{i+2n})$$

A standard genetic algorithm should work fine on the Easy_{30} problem using 2PTX or 1PTX. For UX, both problems are equally difficult. On the other hand, it has been shown

(Goldberg, Korb, & Deb, 1989) that they consistently fail to converge to the correct optima for the Hard_{30} version. We apply LinX and ALinX on both problems, and compare with 2PTX, 1PTX and UX. We fixed every parameter to be the same for all of the operators. We execute each of the algorithms for three different sets of parameters as shown in Table 1.

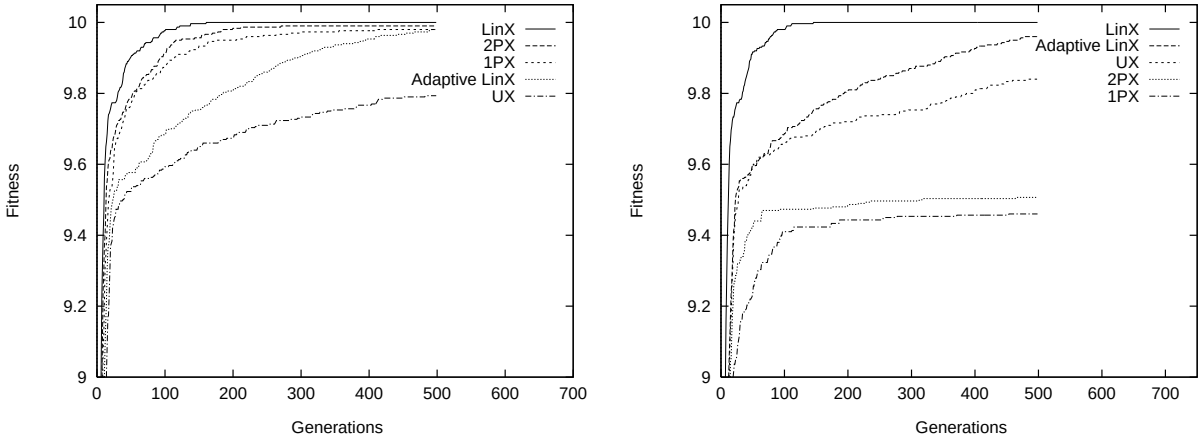
Figures 4, 5 and 6 graph best fitness values against generation number for Easy_{30}, and Hard_{30}. As expected, LinX outperforms all crossover operators for Hard_{30} problem, while competing favorably with other operators in the Easy_{30} case. For Easy_{30}, note that GAs using all operators converge to the correct answer.

Problem version	Pop. size	No. of Gen.	Fitness				
			LinX	ALinX	2PX	1PX	UX
Easy_{30}	50	500	10.00	9.98	9.99	9.98	9.79
	200	200	10.00	9.97	10.00	10.00	9.89
	100	1000	10.00	10.00	10.00	9.99	9.93
Hard_{30}	50	500	10.00	9.96	9.50	9.46	9.84
	200	200	10.00	9.98	9.65	9.55	9.88
	100	1000	10.00	10.00	9.62	9.50	9.90

Table 1: Average of the best solutions over 30 iterations obtained using different crossover operators. Global optimal fitness = 10.0.

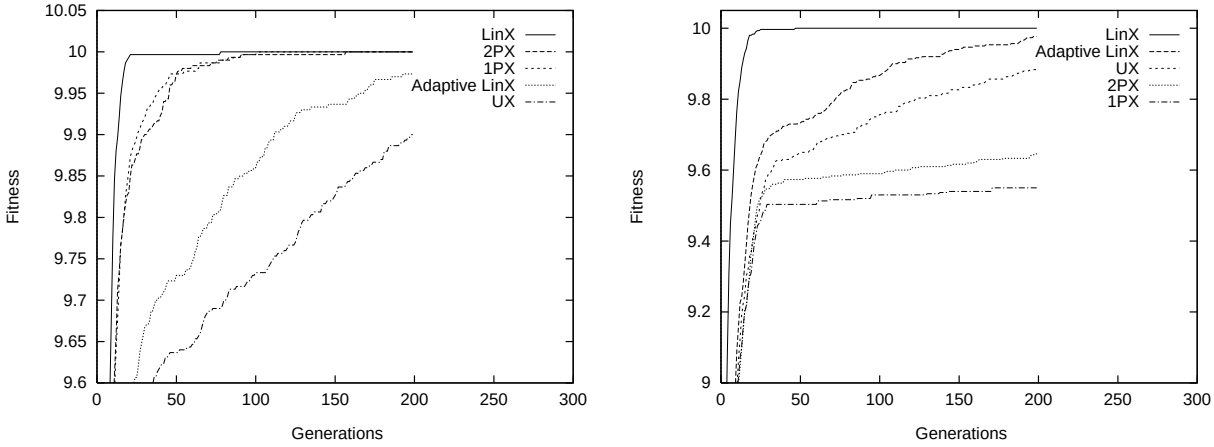
5.1.2 Bipolar Deceptive Problem

For this deceptive problem, the solution string is constructed by concatenation of many *order-6 bipolar functions* i.e., discrete functions with two global optima for complementary bit-strings. Points near the global optima are the lowest in fitness, and the function tends to take larger values as we move towards bit-strings with as many zeroes as 1s, where local optima are located. Concatenation of five such 6-bit functions yields a problem with 5 million suboptima and 32 global optima (Goldberg, Deb & Horn, 1992), making this problem a challenging one. By varying the coupling between each function's bits, we can instantiate different versions of this problem. An "easy order-6 bipolar" problem is defined to have tightly coupled bits close together; i.e., the distance between the first and last bits of the 6-bits is exactly 5; a "hard order-6 bipolar" problem is such that the six bits of each



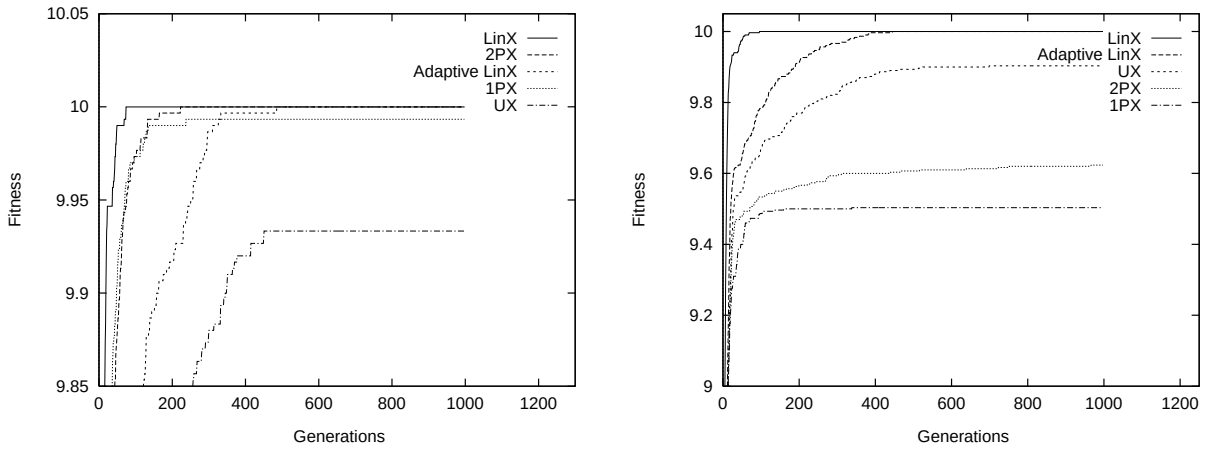
(a) (b)

Figure 4: Fitness of the best solution (average of 30 iterations) versus number of generation. Population size = 50, maximum number of generations = 500. (a) Easy_{30} (b) Hard_{30}



(a) (b)

Figure 5: Fitness of the best solution (average of 30 iterations) versus number of generation. Population size = 200, maximum number of generations = 200. (a) Easy_{30} (b) Hard_{30}



(a) (b)

Figure 6: Fitness of the best solution (average of 30 iterations) versus number of generation. Population size = 100, maximum number of generations = 1000. (a) Easy_{30} (b) Hard_{30}

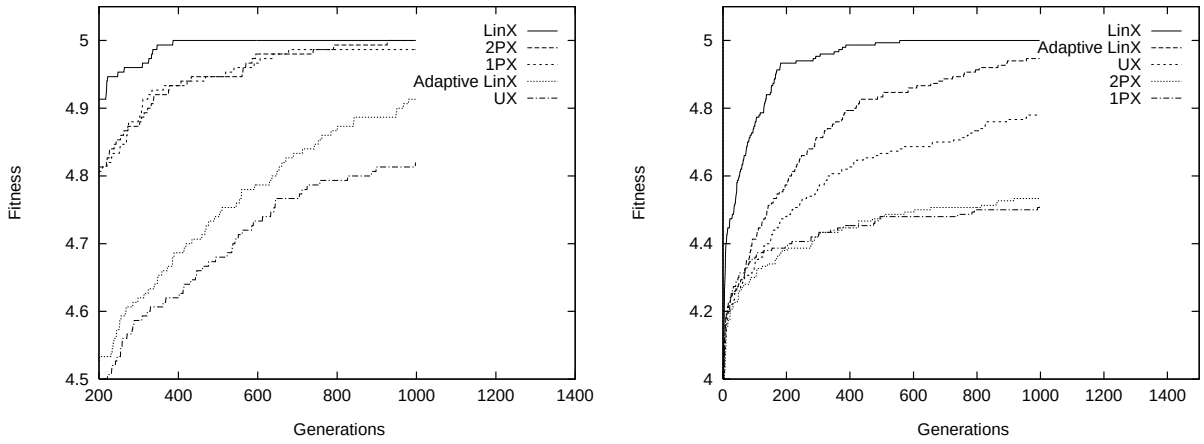
constituent function are physically separated by greater distances.

Experiments were carried out attempting to solve both easy and hard-order-6-bipolar problems with 1PTX, 2PTX, and UX along with the LinX and ALinX crossover operators. In order to compare the performance of these crossover operators, all other parameters of GA were taken to be the same in all experiments.

Problem Version	No. of bits	Pop. size	No. of Gen.	Fitness					
				optimal	LinX	ALinX	2PX	1PX	UX
Easy	30	30	1000	5.0	5.00	4.91	5.00	4.99	4.82
	90	100	5000	15.0	15.00	14.19	14.81	14.21	13.00
	120	200	5000	20.0	20.00	18.53	19.91	19.71	17.08
Hard	30	30	1000	5.0	5.00	4.94	4.53	4.51	4.78
	90	100	5000	15.0	14.90	14.09	12.36	12.41	13.04
	120	200	5000	20.0	20.00	18.00	16.46	16.34	17.00

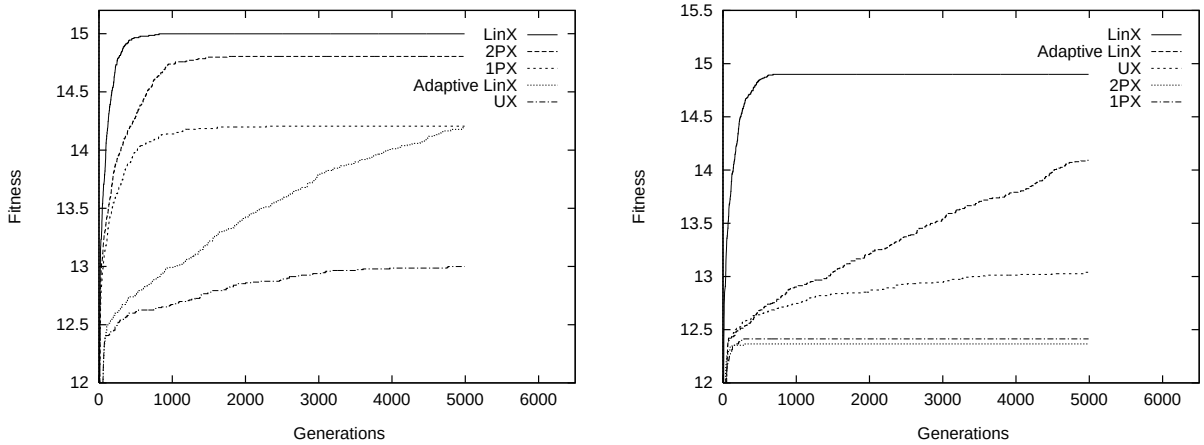
Table 2: Bipolar order-6 problems: Average over 30 trials of the fitness of the best solutions and the global optima.

Table 2 shows the average of the best solution fitness found at the end of the execution. Graphs 7, 8, and 9 show that LinX outperforms all other crossover operators for both versions of the problem, and the global solution is obtained in a very short time. 1PTX and 2PTX find the optimal solution most of the time for the easy versions of the



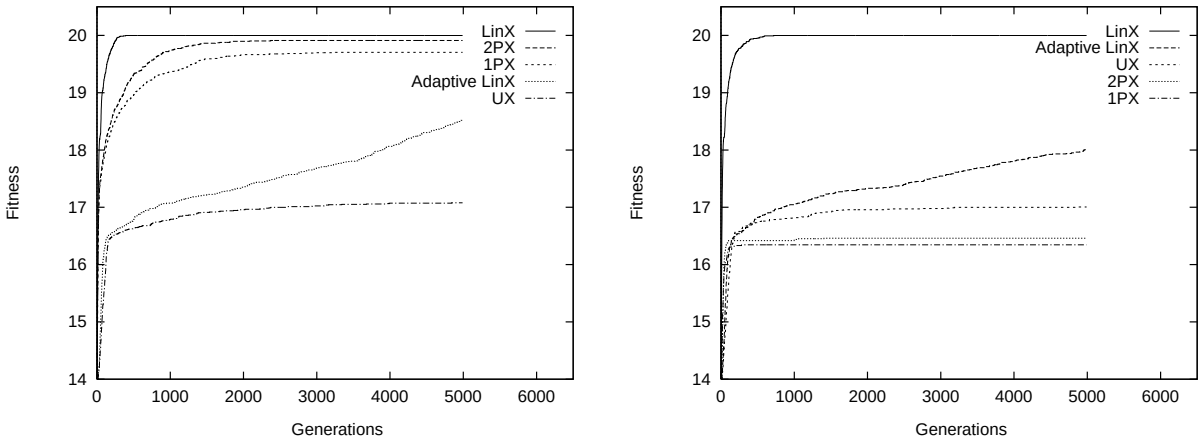
(a) (b)

Figure 7: Fitness of the best solution (average over 30 trials) versus number of generation for population size = 30, maximum number of generations = 1000. (a) Easy bipolar with 30 bits, (b) Hard bipolar with 30 bits.



(a) (b)

Figure 8: Fitness of the best solution (average over 30 trials) versus number of generation for population size = 100, maximum number of generations = 5000. (a) Easy bipolar with 90 bits, (b) Hard bipolar with 90 bits.



(a) (b)

Figure 9: Fitness of the best solution (average over 30 trials) versus number of generation for population size = 200, maximum number of generations = 5000. (a) Easy bipolar with 120 bits, (b) Hard bipolar with 120 bits.

problem, but fail to do so for the hard versions. For easy as well as hard versions, ALinX manages to discover the approximate linkages between genes. This makes it possible for AlinX to slowly advance towards the optimal solution and get much better results than 1PTX, 2PTX, and UX. Each generation of LinX and ALinX takes more time than those for the other three crossover operators, occasionally by as much as factors of 2.5 and 5 respectively. The extra time is needed for the manipulation of the linkage matrix and computation of probability values. Although the time to evaluate the next population is higher for the ALinX operator, the best fitness obtained improves with each generation. On the other hand 1PTX, 2PTX, and UX rapidly converge to local optima, and performance does not improve with additional generations. Time required to obtain a good quality result is much less for the new algorithms than using traditional crossover operators.

5.2 Graph Partitioning

The graph bipartitioning problem has been attempted by several researchers using the GA approach (see Maini, Mehrotra, Mohan & Ranka, 1994b). The optimization criterion of this problem is to minimize the total cost of communication between the two partitions. As in the previous section, the purpose of the simulations is to compare the performance of LinX and ALinX with other crossover operators. In this example, the linkage behavior is not predetermined, it has to be learned. Several data sets were considered. For ALinX,

the linkage matrix was initialized randomly, whereas for LinX, the linkage matrix was constructed using the following heuristic:

> The larger the cost of communication between two nodes, the greater the need to put them into the same bin.

To minimize the cost of communication across the two bins we attempt to maximize the sum of edge costs between nodes belong to the same bin. A linkage matrix that reflects this property will lead the algorithm in the right direction. To implement this heuristic we make the linkage entries proportional to the communication costs between the nodes; the higher the communication cost, the higher is the linkage, and *vice versa*. Table 3 shows the results (minimum cost of communication) for each operator on three graphs containing 45, 80, and 100 nodes with random edge weights between 0 and 1, respectively. Results are obtained by taking averages of the best solutions over 10 trials. Figure 10 shows how the GA performance improves with the number of generations for various crossover operators. Results show that ALinX outperforms the other operators, including LinX.

In the graph bipartitioning problem, it is difficult to predetermine the right linkage matrix from the graph. Thus, it is not surprising that ALinX learns a linkage matrix that outperforms the LinX operator whose linkage matrix was based on a reasonable heuristic.

No. of Nodes	Pop. size	Generations	Communication cost of best Solution			
			ALinX	LinX	2PTX	UX
45	100	1000	222.68	222.80	222.74	222.73
75	90	10000	620.06	651.32	632.69	650.39
80	100	4000	714.24	748.05	731.14	747.16
100	200	6000	1154.81	1189.90	1180.49	1189.08

Table 3: Graph Partitioning Problem, average over 10 trials

6 Evaluating the Adaptation Process

This section examines some of the important issues relating to the linkage matrix adaptation process. In particular, we consider how to evaluate the linkage matrix adaptation process.

Naturally, the first significant issue relates to performance – what is the quality of the solutions obtained using the adaptively modified linkage matrix? The average and best

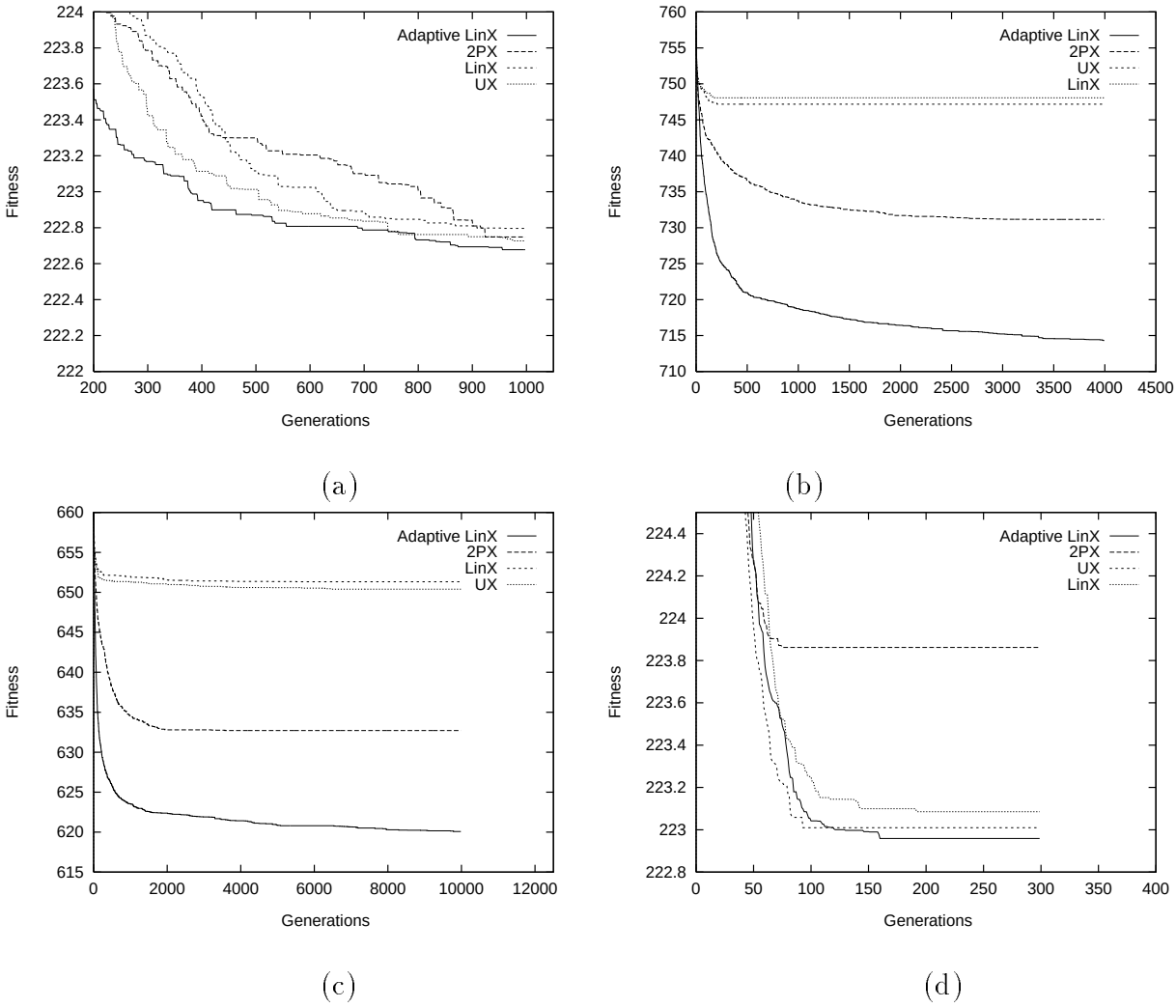


Figure 10: Fitness (average over 30 trials) improvement versus number of generation for the graph bipartitioning problem. (a) Graph size = 45 nodes, population size = 100, maximum number of generations = 1000. (b) Graph size = 80 nodes, population size = 100, maximum number of generations = 4000. (c) Graph size = 75 nodes, population size = 90, maximum number of generations = 1000. (d) Graph size = 45 nodes, population size = 100, number of generations = 300 with no reinitialization

fitness obtained using ALinX can be compared with the corresponding values obtained by using other different crossover operators and algorithms. In the previous section, we used this as the main comparison criterion.

The second issue is: Will the adaptation process stabilize? Will elements of the linkage matrix deviate little in later iterations of the GA? A deviation measure such as

$$D_L = \frac{1}{N}\sqrt{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(\text{NewLinkage}[i][j] - \text{OldLinkage}[i][j])^2}$$

may be used to compare the linkage matrix entries before and after each generation. A small value of D_L indicates that stabilization has occurred.

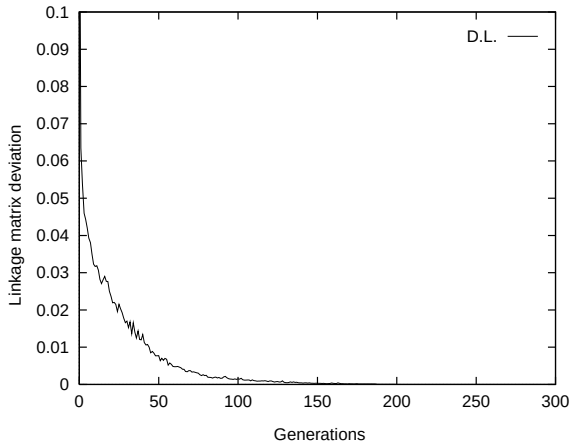


Figure 11: D_L (variation in linkage values) for ALinX for the graph bipartition problem with 45 nodes and population size 100.

Finally, does the adaptive algorithm result in a linkage matrix whose elements are reasonable and easy to interpret? This is easy to verify for those problems where the linkage matrix is well known. An objective measure may be constructed as a function of the actual linkage values and the linkage values obtained by adaptation. In the Easy_{30}, Hard_{30}, and the bipolar problems, we had prior knowledge of the linkage matrices. Our experiments confirm that the ALinX is successful in adapting an initially random matrix to obtain the desired linkage values.

In some problems, no detectable linkages among elements exist. In such cases, an evolutionary algorithm may be successful if it uses linkage-neutral operators (such as uniform crossover). The adaptive algorithm should approach the results of such algorithms. However, since the linkage values are irrelevant to the problem, elements in the matrix may fluctuate randomly in each iteration.

In many multi-constraint optimization problems, linkages exist between different elements, but the best choice of a linkage matrix is not known *a priori*. In these situations, the adaptive algorithm is expected to be more successful than other strategies using traditional crossover operators or a fixed linkage matrix. If some information about the problem is available (e.g., in graph partitioning problems), this may be used to initialize the linkage matrix; starting the adaptive process from such a state would give better solutions faster than from a randomly initialized state. In other words, any available problem-specific information should be utilized by the adaptive algorithm, not ignored.

Some problems are structured so that adjacent elements in the individual are strongly related, so that one-point or two-point crossover operators succeed in finding good quality solutions without much effort. In such cases, the adaptive algorithm is to be judged satisfactory if its results approach that of 1PTX or 2PTX, with high linkage values for adjacent elements. Our experiments (on Goldberg's Easy_{30} and bipolar problems) confirm that ALinX is capable of achieving this result.

For other problems whose linkages are well understood, so that a good linkage matrix can be chosen *a priori*, the adaptive algorithm is to be judged satisfactory if it yields solutions whose quality approaches that of the algorithm using the predetermined linkage matrix. We should also expect that the adaptive algorithm will yield a linkage matrix close to the good linkage matrix known *a priori*. Experiments on Hard_{30} and other deceptive problems confirms that ALinX satisfies this property as well. Data in Table 4 gives some of the values for strongly linked and weakly linked genes.

Pop. size	No. of Gen.	Strongly linked genes		Weakly linked genes	
		Initial	Final	Initial	Final
50	500	0.634	0.552	0.616	0.169
200	200	0.507	0.694	0.463	0.164
100	1000	0.414	0.820	0.438	0.097

Table 4: Average conditional probabilities for strongly linked and weakly linked genes, before and after adaptation using ALinX

7 Conclusion

This paper relates the field of probabilistic inference to the application of crossover operators in genetic algorithms. Probabilistic computations have a long history, and can be used with considerable advantage in GAs. The framework presented in this paper allows explicit formulation of problem-specific linkages and their subsequent use in crossover. A new class of crossover operators is presented, implemented and tested. These operators exploit problem-specific linkages among components in a chromosome. The concept of adapting linkage between genes is shown to be effective and successful, even for problems with only partially known linkage structure. Many practical optimization problems such as load balancing, scheduling, routing, and assignment problems are characterized by a set of constraints that relate some parameters. Such problems can be solved using a GA with the LinX methodology, using the following principles, assuming a direct representation with one gene per problem parameter.

- If a constraint relates parameter i with parameter j, then the linkage probability is high for the corresponding pair of genes.
- Hard constraints correspond to higher linkage probabilities than weak constraints.
- If x_{i+1} depends on $\{x_1, x_2, \ldots, x_i\}$ only through a subset $S \subset \{x_1, x_2, \ldots, x_i\}$, then $P(x_{i+1} : x_1, x_2, \ldots, x_i; \pi) = P(x_{i+1} : S)$.
- If a problem parameter x_{i+1} is deterministically dependent on $\{x_1, x_2, \ldots, x_i\}$, then the appropriately determined value is to be assigned to the offspring gene instead of using linkage probabilities.

References

Atmar, W. (1992). On the rules and nature of simulated evolutionary programming, *Proc. First Conf. Evolutionary Programming,* La Jolla (CA), pp.17–26.

Bethke, A. D. (1980). Genetic algorithms as function optimizers, Ph.D. thesis, University of Michigan, Ann Arbor.

Forrest, S. and Mitchell, M. (1993) What makes a problem hard for genetic algorithm? Some anomalous results and their explanation, *Machine Learning*, vol 13, pp. 285–319.

Goldberg, D.E. (1989a). *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, Reading, MA.

Goldberg, D.E. (1989b). Genetic algorithms and Walsh functions: Part I, A gentle introduction. *Complex Systems*, vol 3, pp.129–152.

Goldberg, D.E. (1989c). Genetic algorithms and Walsh functions: Part II, Deception and its analysis. *Complex Systems*, vol 3, pp.153–171.

Goldberg, D.E. and Richardson J. (1987), Genetic algorithms with sharing for multimodal function optimization, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms,* Erlbaum, pp. 41.

Goldberg, D.E., Korb, B. & Deb, K.(19000). Erlbaum, Messy genetic algorithms: Motivation, analysis, and first results, *Complex Systems*, vol 3, pp. 493–530.

Goldberg, D.E., Deb, K. & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms, *Parallel Problem Solving from Nature*, vol 2, Elsevier Science, pp. 37–46.

Grefenstette, J. J. (1987). Incorporating problem specific knowledge into genetic algorithms, *Genetic Algorithms and Simulated Annealing*, L. Davis and Morgan Kaufmann, eds..

Harik, G. R. & Goldberg, D. E. (1997). Learning Linkage, *Foundation of Genetic Algorithms – IV,* Morgan Kaufmann, pp. 247–262.

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

Kennedy, J. and Eberhardt, R. C. (1995). Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, pp. 1942–1948.

Liepins, G. E. and Vose, M. D. (1991). Deceptiveness and Genetic Algorithm Dynamics, *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 36–50.

Maini, H. S., Mehrotra, K. G., Mohan, C. K., & Ranka, S. (1994a). Knowledge-Based Nonuniform Crossover, *Complex Systems,* Vol.8, pp.257-293.

Maini, H. S., Mehrotra, K. G., Mohan, C. K., & Ranka, S. (1994b). Genetic Algorithms for graph partitioning and incremental graph partitioning. In *Proceedings of Supercomputing'94*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems,* Morgan Kaufmann.

Shafer. G. & Pearl, J. (eds.), (1990). *Readings in Uncertainty Reasoning,* Morgan Kaufmann.

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems, Doctoral Dissertation, U. of Michigan.

Spears, W. M. and De Jong, K. A. (1991). An Analysis of Multi-Point Crossover, *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 301–315.

Syswerda, G. (1989). Uniform crossover in genetic algorithms, *Proc. of the 3rd ICGA*, pp. 2–9.

Whitley, L. D. (1991). Fundamental principles of deception in genetic search, *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 221–241.

Wolpert, D. H. and Macready, W. G. (1995). No free lunch theorems for search, Tech. Rep. SFI-TR-02-010, Santa Fe Institute.