Linkage Crossover For Genetic Algorithms

Ayed A. Salman **Kishan Mehrotra** **Chilukuri K. Mohan**
2-175 Center for Science and Technology
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY 13244-4100
(ayed/kishan/mohan@top.cis.syr.edu)
315-443-2811/2322

Abstract

A new "linkage crossover" operator based on probabilistic methodology is proposed. The genetic algorithm using this operator is shown to be particularly successful for deceptive problems with linkages that are not easily captured by linear relationships. Further, adaptive Hebbian mechanisms are shown to be successful in learning the appropriate linkage relationships when faced with new problems. Results demonstrate the efficacy of this approach, compared with traditional general purpose crossover operators.

1 INTRODUCTION

Several researchers (e.g., [5, 8]) have proposed to incorporate problem-specific knowledge in genetic algorithms; we show that this is accomplished successfully by the use of linkage information in the crossover operator itself. This overcomes the representation bias implicit in the use of bit-string (and other linear) representations with point crossover operators, while not succumbing to the randomness of the "no-linkage" approach of uniform crossover. Problems with no inherent linear linkage structure then become amenable to solution using this method. Furthermore, we show that an adaptive algorithm succeeds in learning the nature of the linkages in new problems.

Other researchers have approached the mismatch between the traditional GA and problem-specific linkage information from different viewpoints. The first class of algorithms divides the optimization process into two phases, one phase for linkage estimation and the other for solution construction [4, 7]. The second class tries to execute the learning and the construction in parallel [6]. Finally, the third class uses a probabilistic approach to estimate these dependencies (linkages) [10, 9, 1].

A genetic algorithm will move rapidly towards optimal solutions of a given problem if it is allowed to exploit known dependencies among genes (species-specific linkages between traits). In this paper, we use *linkage* to refer to the conditional probability of inheriting a gene from a parent, given that other genes are also inherited from that parent. The stronger the linkage between genes, the less should they be separated by crossover. For efficiency reasons, we use only the *first-order linkages* between genes, representing the conditional probability of inheriting gene$_i$ from one parent, given that gene$_j$ is inherited from that parent, for $i \neq j$.

In Section 2, we present a probabilistic methodology that incorporates known problem structural dependencies, "linkages", into the design of genetic algorithm operators. Section 3 describes the *linkage crossover* algorithm, derived from this methodology. Section 4 describes an adaptive linkage learning mechanism for those problems where the linkage structure is not known *a priori*. Results are given in Section 5. Section 6 concludes the paper.

2 PROBABILISTIC FRAMEWORK

This section describes our overall framework, establishing the relation between crossover and probabilistic inference. In our notation, $i \leftarrow p_j$ denotes the event in which 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 prior 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. In examining which genes are inherited from which parent, we restrict attention to those genes where the parents differ. Also, let $\{x_1, x_2, \ldots, x_n\}$ denote 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 relationships between sets of alleles. This dependence can be reformulated in terms of problem-specific conditional probabilities. Specifically, the crossover operator should address the question that if the ith gene is inherited from parent p_1, 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.

Traditional genetic operators can be viewed as special cases of this approach. For instance, 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:

$$P(i \leftarrow p_1 \mid (i+a) \leftarrow p_1 \ \& \ (i-b) \leftarrow p_1) = 1,$$

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. 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" $L(x_{i+1} : \ x_1, \ldots, x_i)$ denotes

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

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 other 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 four positions* ($k = 4$). *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 $P(A|C)$ along with the priors. For specific problems, a dependency structure may be available, enabling calculations of such quantities. This is the approach we have taken.

A first step toward using linkage information would be to develop a crossover operator that makes use of *pairwise linkage*, $L(x_i : x_j)$ (defined as $P(x_i \leftarrow p_1 | x_j \leftarrow p_1)$). 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 $P(x_i \leftarrow p_1 | x_j \leftarrow p_1) = L(x_i : x_j)$ We assume that conditional symmetry prevails, i.e.,

$$\begin{aligned} P(x_i \leftarrow p_1 | x_j \leftarrow p_2) &= P(x_i \leftarrow p_2 | x_j \leftarrow p_1) \\ &= 1 - L(x_i : x_j). \end{aligned}$$

We assume that the problem description specifies $L(x_i : x_j)$, for each x_i, x_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. For any two events A and B, Bayes' rule gives

$$\begin{aligned} P(A|B) &= \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\overline{A})P(\overline{A})} \\ &= \frac{P(B|A)}{P(B|A) + o(A)P(B|\overline{A})} \qquad (1) \end{aligned}$$

where $o(A) = P(\overline{A})/P(A)$ represents the odds (of the prior probabilities of occurrence) of A. Applying Equation (1) to the problem of interest gives: $P(x_{i+1} : x_1, \ldots, x_i; \pi) = P(x_{i+1} \leftarrow p_1 | \cap_{j=1}^{i} x_j \leftarrow \pi(x_j))$

$$= \frac{P(\cap_{j=1}^{i} x_j \leftarrow \pi(x_j)|(x_{i+1} \leftarrow p_1)}{P(\cap_{j=1}^{i} x_j \leftarrow \pi(x_j)|(x_{i+1} \leftarrow p_1) + o(x_{i+1} \leftarrow p_1)\alpha} \qquad (2)$$

where $\alpha = P(\cap_{j=1}^{i} x_j \leftarrow \pi(x_j)|(x_{i+1} \leftarrow p_2))$.

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, because $P(x_{i+1} \leftarrow p_1) = P(x_{i+1} \leftarrow p_2) = 0.5$; there is no *a priori* preference that the allele in the $i+1$th position of the offspring should come from a specific parent. For the rest of this paper, we assume that 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

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

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 (2) gives

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

$$\frac{\prod_{j=1}^{i} P(x_j \leftarrow \pi(x_j)|x_{i+1} \leftarrow p_1)}{\prod_{j=1}^{i} P(x_j \leftarrow \pi(x_j)|x_{i+1} \leftarrow p_1) + \beta}$$

where $\beta = \prod_{j=1}^{i} P(x_j \leftarrow \pi(x_j)|x_{i+1} \leftarrow p_2)$. This expression can be further simplified for ease of computation.

LinX crossover: Let 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 are inherited from parent p_1, and others from p_2. In LinX, the x_{i+1}^{th} gene of the offspring is choosen from p_1 with probability

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

where
$S_{i,1}$ is the set of genes inherited from parent p_1,
$S_{i,2}$ the set of genes inherited from parent p_2,

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

and

$$h_2 = \prod_{j \in S_{i,1}} P(x_j \leftarrow \pi(x_j)|x_{i+1} \leftarrow p_2)$$

$$\times \prod_{j \in S_{i,2}} P(x_j \leftarrow \pi(x_j)) | (x_{i+1} \leftarrow p_2)$$

$$= \prod_{j \in S_{i,1}} (1 - L(x_j : x_{i+1})) \prod_{j \in S_{i,2}} L(x_j : x_{i+1})$$

This approximation based on the independence assumption suggests that a joint linkage probability such as $L(x_{i+1} : x_1, x_2, \ldots x_j)$ can be estimated based on the pairwise linkage probabilities $L(x_{i+1} : x_1)$, $L(x_{i+1} : x_2), \ldots, L(x_{i+1} : x_j)$. 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.

4 LINKAGE ADAPTATION

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 the 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. Linkage probabilities are analogous to "connection strengths" (weights attached to edges between nodes) in neural networks.

This idea is exploited in the 'ALinX' algorithm, by adapting linkage values during the execution of the GA. In our previous work [11], the pairwise linkage probabilities were adapted using the Average Population Fitness (APF) method. APF uses the fitness of the offspring resulting from a crossover to judge the efficacy of the linkage probabilities used for that crossover step. An improvement over that method is the "Genes Perturbation Fitness Adaptation" (GPF) described in Figure 1. The main idea is that we should examine the possible consequences of inheriting a gene from the other parent, rather than rely on the fitness of unrelated individuals in the population and their offspring. The resulting improvement for one problem is graphed in Figure 2. For the rest of the paper, all results for AlinX are obtained using the GPF method unless otherwise specified.

ALinX2: GPF Adaptation Algorithm:
Initial step: Initialize each entry $\mathrm{L}[j,k] \in [0,1]$ (and $X[j,k] \in [-1,1]$) randomly.
Reproduction step:

- Let O_i be the i^{th} offspring of this generation;
- For each pair of genes j,k such that $O_i[j] = p_1[j] \neq p_2[j]$ and $O_i[k] = p_1[k] \neq p_2[k]$, where p_1 is the parent of O_i from which both genes are inherited and p_2 is the other parent, do:

  Let f_0 be the fitness of O_i;
  Let f_1 be the fitness of the individual obtained from O_i by changing gene j to be inherited from p_2;
  Let f_2 be the fitness of the individual obtained from O_i by changing gene k to be inherited from p_2;
  The linkage matrix is then modified as follows:

$$\begin{aligned} \triangle X[j,k] &= \eta \times (f_0 - (f_1 + f_2)/2) \\ \mathrm{L}[j,k] &= \frac{(X[j,k] - \min_\ell(X[j,\ell])}{(\max_\ell(X[j,\ell]) - \min_\ell(X[j,\ell])} \end{aligned}$$

Figure 1: GPF adaptation algorithm

5 RESULTS

In this section, we address the following questions with respect to the performance of LinX crossover and ALinX algorithm:

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

The canonical genetic algorithm used in our experiments has the following characteristics:

- Chromosomes are randomly initialized.
- Roulette wheel selection methodology is used to control the mating process.

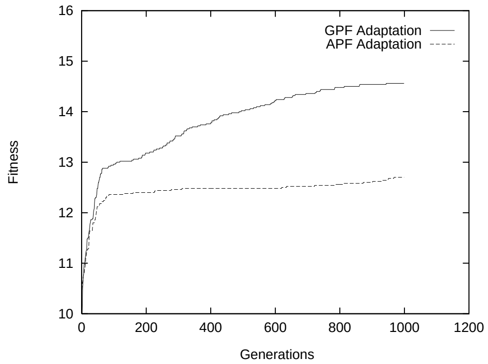


Figure 2: Comparison between new (GPF) and old (APF) linkage adaptation algorithms for the 90-bit hard bipolar problem; population size 100, averages over 10 trials.

- The best 10% of the existing population is merged with the best 90% of the generated population.
- A repair mechanism is forced upon infeasible solutions among the population imposed by the problem constraints (e.g., in graph bipartitioning problems, there must be equally many alleles of each kind).
- Mutation rate is $1/N$, where N is the chromosome size.

Experiments were carried out on several benchmark problems, of which some were deceptive and some had unknown linkage structure. In the results reported below, we use the abbreviations in Table 1.

Best	Best fitness, averaged over all trials
O.N.	No. of trials in which global optimum is reached
A.I.	Average number of iterations required
Var.	Variance of best solutions over trials
Skew.	Skewness of the best solutions over trials
Pop.	Problem's population size
$\mathbf{E}_{xxx}$	Easy version of the problem
$\mathbf{H}_{xxx}$	Hard version of the problem

Table 1: Abbreviations used in the paper

5.1 Order-3 Deceptive Problems

Goldberg, Korb, and Deb [4] defined problems obtained by concatenating multiple instances of an order-3 deceptive problem; fitness of an individual is the sum of the fitnesses of the order-3 components. Results for these problems are shown in Table 2. In the '*Hard*' versions of these problems, the three bits of each order-3 combination are scattered over the string; ALinX was able to learn the problem linkage structure along the run and outperform other operators. In the '*Easy*' versions, the bits of each combination are adjacent; performance of ALinX was then very similar to that of 2PTX and 1PTX. In all cases, LinX outperformed all crossover operators.

5.2 Bipolar Deceptive Problems

Goldberg, Deb and Horn [2] defined problems obtained by concatenating multiple instances of a 6-bit 'bipolar' (with two global optima) deceptive problem. Once again, these could be 'easy' or 'hard'. For such a 30-bit problem, for instance, there are 32 global optima and 5 million local optima. As shown in Table 3, LinX outperforms all other crossover operators for both Easy and Hard versions of the problems, and the global solution is obtained in a very short time. 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 a factor of 5 in some cases, due to 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.

5.3 Graph Bipartitioning

Nodes in a graph must be partitioned into two bins, maximizing the sum of edge costs between nodes belonging to the same bin. Here, the appropriate linkage values are not known; indeed applying Linx with linkage entries proportional to edge weights yielded results almost identical to those obtained with Uniform Crossover. However, Table 4 shows that ALinX succeeds in learning the appropriate linkage matrix, and outperforms other operators.

5.4 Mühlenbein's Problem

Mühlenbein [10] instantiates a GA challenging problem with a structure very similar to the order-3 and the bipolar. The problem building block constitutes of 5 bit subfunction which has one optimal and 4 local optima. Concatenating these subfunctions yields an exponential number of local optima. Both LinX and AlinX demonstrate statistical superiority over other operators as shown in Table 5.

Xover	Best	O.N.	A.I.	Var.	Skew.
Bits=30 Pop.=50 Max.Gens.=1000 E_{ord3} *t-test*=0.4899					
LinX	9.997	29	344	0.000	-4.942
UX	9.920	13	869	0.009	-1.513
1PTX	9.973	23	691	0.003	-1.684
2PTX	9.983	25	605	0.001	-1.700
AlinX	9.987	26	547	0.001	-2.050
Bits=30 Pop.=50 Max.Gens.=1000 H_{ord3} *t-test*=3.5					
LinX	9.987	29	400	0.005	-4.942
UX	9.880	11	874	0.017	-1.107
1PTX	9.837	3	998	0.012	-1.031
2PTX	9.843	9	911	0.032	-1.747
AlinX	9.977	26	611	0.006	-3.917
Bits=60 Pop.=80 Max.Gens.=4000 E_{ord3}					
LinX	19.993	28	1912	0.001	-3.302
UX	19.603	0	4000	0.038	-0.238
1PTX	19.793	4	3929	0.026	-1.172
2PTX	19.950	19	3034	0.006	-1.493
AlinX	19.857	6	3726	0.011	-0.349
Bits=60 Pop.=80 Max.Gens.=4000 H_{ord3} *t-test*=6.209					
LinX	19.990	27	1970	0.001	-2.534
UX	19.573	1	3992	0.039	0.203
1PTX	19.240	0	4000	0.062	0.167
2PTX	19.350	0	4000	0.045	0.124
AlinX	19.853	10	3636	0.022	-0.866

Table 2: Results for *Order-3* problem; for 30 trials; mutation rate= 1/(number of bits); *t-test* was done between AlinX and the nearest competitive crossover, fitness of the optimal solution equal to (number of bits)/3.

Xover	Best	O.N.	A.I.	Var.	Skew.
Bits=60 Pop.=80 Max.Gens.=2000 E_{bip} *t-test*=0.9272					
LinX	10.000	30	745	0.000	†
UX	9.580	1	1972	0.062	-0.550
1PTX	9.780	8	1833	0.029	-0.153
2PTX	9.900	18	1653	0.021	-1.535
AlinX	9.933	23	1515	0.017	-1.642
Bits=60 Pop.=80 Max.Gens.=2000 H_{bip} *t-test*=7.619					
LinX	10.000	30	706	0.000	†
UX	9.587	4	1939	0.072	-0.218
1PTX	9.320	0	2000	0.062	-0.271
2PTX	9.540	2	1968	0.111	-0.708
AlinX	9.973	26	1439	0.005	-2.050
Bits=90 Pop.=200 Max.Gens.=10000 E_{bip}					
LinX	15.000	30	3744	0.000	†
UX	13.920	0	10000	0.291	-0.483
1PTX	14.787	8	9564	0.036	-1.061
2PTX	14.947	24	6263	0.016	-2.794
AlinX	14.840	18	8121	0.065	-1.631
Bits=90 Pop.=200 Max.Gens.=10000 H_{bip} *t-test*=4.311					
LinX	14.927	29	3741	0.161	-4.942
UX	13.860	0	10000	0.271	-0.557
1PTX	13.847	1	9940	0.258	-0.533
2PTX	14.167	1	9870	0.240	-1.091
AlinX	14.740	11	9063	0.290	-4.351

Table 3: Results for *Bipolar* problem; for 30 trials; mutation rate= 1/(number of bits) ; *t-test* was done between AlinX and the nearest competitive crossover, optimal solution fitness equal to (number of bits)/6, †: solutions obtained are all optimal.

# Nodes	Pop. size	# Gen.	Min. Communication cost		
			ALinX	2PTX	UX
30	50	300	96.4	96.7	96.5
75	90	1000	630.8	630.7	635.3
75	90	2000	626.2	627.9	630.6
75	90	4000	624.6	625.9	627.8
80	100	4000	713.8	716.1	720.1

Table 4: Graph Partitioning Problem, average over 30 trials

5.5 Statistical Analysis

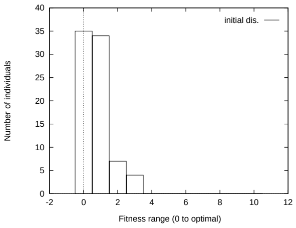
We performed statistical analysis of the fitness distributions resulting from applying GAs with various crossover operators. The variance of the best solutions found by LinX was either zero or very small, indicating that it got to either the optimal or a near-optimal solution in every trial. LinX also required many fewer iterations than the other operators.

For the 'hard' versions of problems, ALinX was statistically superior to the others as apparent from *t-test* results comparing ALinX with its nearest competitor. In other cases where no statistical superiority exist, ALinX's results were more skewed than the others, indicating that it pushes towards the optimal solution more successfully than the others. Further, ALinX was successful in learning the underlying linkage structure of these problems.

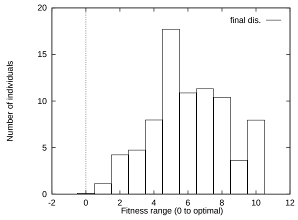
We also examined the initial populations and the populations obtained at the end of the generations (the terminal population). For Mühlenbein's problem, this analysis uncovered interesting behavior. Compared to 1PTX, 2PTX, and UX, ALinX was better at obtaining best solutions (largest average of best solutions and largest number of optimal found) and better at shifting the whole population towards better areas of search space (best population average, less variance, and more skewed). At the terminal stage, ALinX dis-

Xover	Best	O.N.	A.I.	Var.	Skew.
Bits=20 Pop.=30 Max.Gens.=500 *t-test*=0.797					
LinX	15.783	18	263	0.081	-0.796
UX	15.567	13	331	0.202	-0.525
1PTX	15.383	7	414	0.219	-0.280
2PTX	15.450	8	374	0.213	-0.575
Alinx	15.650	13	361	0.123	-0.451
Bits=35 Pop.=60 Max.Gens.=4000 *t-test*=3.2					
LinX	27.667	15	2582	0.144	-0.595
UX	27.300	8	2998	0.355	-0.761
1PTX	27.100	4	3597	0.283	-0.053
2PTX	27.100	2	3737	0.283	-0.885
AlinX	27.700	15	2720	0.114	-0.625
Bits=45 Pop.=70 Max.Gens.=5000 *t-test*=3.257					
LinX	35.617	14	3602	0.167	-0.424
UX	35.067	4	4345	0.375	-0.244
1PTX	34.450	1	4836	1.661	-3.186
2PTX	34.950	1	4885	0.230	-0.036
AlinX	35.500	8	3976	0.155	-0.409

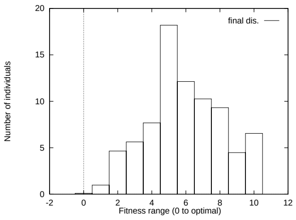
Table 5: Results for hard *Mühlenbein's* problem; for 30 trials; mutation rate= 1/(number of bits) ; *t-test* was done between AlinX and the nearest competitive crossover, optimal solution fitness equal to 4×(number of bits/5).



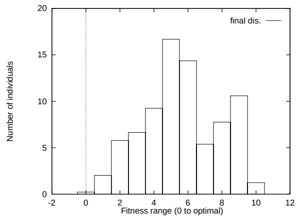
Initial.



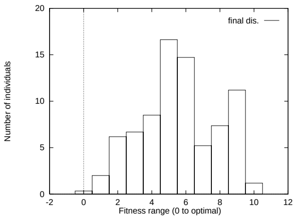
Final:LinX.



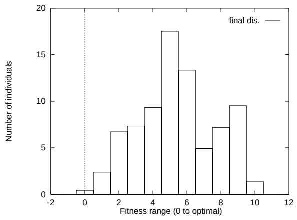
Final:AlinX.



Final:2PTX.



Final:1PTX.



Final:UX.

Figure 3: Number of individuals in the population versus the fitness range of those individuals; *Mühlenbein's* problem, 35 bits, Population size = 80, maximum number of generations = 4000.

tribution was the most symmetrical (near-zero skewness). LinX leads to the maximum number of optimal solutions, followed by ALinX. The remaining three have an approximately equal number of optimal solutions, fewer than observed in ALinX. Interestingly, both LinX and ALinX have fewer deceptive solutions (local optima) than the other operators. These results are shown in Figure 3.

5.6 Convergence

Using problem-specific knowledge may in some cases lead to rapid convergence to local optima, to the detriment of global search. However, we found that LinX and ALinX were surprisingly robust in this respect, as long as simple mutation (with mutation probability reciprocal to chromosome length) was also used. Indeed, performance of the algorithm did not vary when an artificial method to overcome premature convergence (random reinitialization) was introduced into the computer program. LinX overcomes the danger of premature convergence usually associated with algorithms using problem-specific knowledge, because it uses this knowledge in a very loose way (only a global linkage matrix is used to direct the decision of the crossover probabilistically).

Convergence of the linkage adaptation process may also be examined: will elements of the linkage matrix stabilize, remaining almost constant in later iterations of the GA? Figure 4 answers this question in the affirmative, using the deviation measure

$$D_L = \frac{1}{N}\sqrt{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(\mathrm{NewL}[i][j] - \mathrm{OldL}[i][j]\right)^2}$$

to compare the linkage matrix entries before and after each generation.

5.7 Interpretability

Does the adaptive algorithm result in a linkage matrix whose elements are reasonable and easy to interpret? Table 6 shows that ALinX is successful in adapting an initially random matrix to obtain the desired linkage values, for Hard_{30} problems; similar results were obtained with Easy_{30} and the bipolar problems.

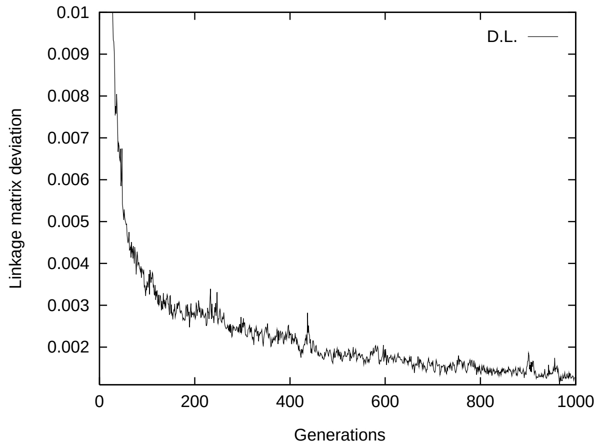


Figure 4: D_L (variation in linkage values) for ALinX for an order$_3$ problem with 60 bits and population size 70.

# bits	Pop. size	Max # Gens.	Average strongly linked	Average weakly linked
21	30	500	0.766	0.166
30	50	1000	0.819	0.146
60	80	4000	0.765	0.096

Table 6: Average linkage values for strongly linked and weakly linked genes after adaptation using ALinX for the hard order$_3$ problem (initialized with random values with mean 0.5).

6 CONCLUSION

The main contribution of this research is to relate 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.

For problems with unknown linkage structures, using adaptive linkage crossover, we were able to identify regions where dependencies should be strong. These regions constitute the building blocks which should be evolved together. Stopping ALinX at some point during the GA run (considered as a parameter for ALinX) and continuing with a local improvement operator may be another fruitful approach, yet to be explored.

References

[1] Baluja, S. and Davies, S., "Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space," Tech.Rep. CMU-CS-97-107, Carnegie Mellon University, Pittsburgh, 1997.

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

[3] Goldberg, D. E., Deb K., and Korb, B., " Messy genetic algorithms revisited: Studies in mixed size and scale," Complex Systems, 4:415-444, 1990.

[4] Goldberg, D.E., Korb, B. and Deb, K., "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems*, 3:493–530, 1988.

[5] Grefenstette, J. J., "Incorporating problem specific knowledge into genetic algorithms," *Genetic Algorithms and Simulated Annealing*, L. Davis (ed.), Morgan Kaufmann Pub., 1987.

[6] Harik, G. R. and Goldberg, D. E., "Learning linkage," *Foundation of Genetic Algorithms – IV*, Morgan Kaufmann, pp. 247–262, 1997.

[7] Kargupta, H., "The gene expression messy genetic algorithm," *Proceedings of IEEE Inter. Conference of Evolutionary Computing (ICEC96)*, pp. 814-819, 1996.

[8] Maini, H. S., Mehrotra, K. G., Mohan, C. K., and Ranka, S., "Knowledge-based nonuniform crossover," *Complex Systems,* 8:257-293, 1994.

[9] Mühlenbein, H., Mahing, T., and Rocdriguez A. O.,"Schemata, Distributions and graphical models in evolutionary optimization," *Submitted for publication,* http://set.gmd.de/AS/ga/publineu.html, 1998.

[10] Pelikan, M. and Mühlenbein, H., "Marginal distribution in evolutionary algorithms," at *http://darwin.chtf.stuba.sk/martin/work.html* (internet-available report), 1998.

[11] Salman, A. A. , Mehrotra, K. G., and Mohan, C. K., "Adaptive linkage crossover," *Proceedings ACM Symposium on Applied Computing (SAC'98),* 1998.