Orgy in the Computer: Multi-Parent Reproduction in Genetic Algorithms

A.E. Eiben[1] gusz@cs.ruu.nl
C.H.M. van Kemenade[2] kemenade@cwi.nl
J.N. Kok[1] joost@cs.ruu.nl

[1] Department of Computer Science
Utrecht University
P.O. Box 80089, 3508 TB Utrecht,
The Netherlands

[2] Department of Software Technology
CWI,
P.O. Box 94079, 1090 GB Amsterdam,
The Netherlands

Abstract. In this paper we investigate the phenomenon of multi-parent reproduction, i.e. we study recombination mechanisms where an arbitrary $n > 1$ number of parents participate in creating children. In particular, we discuss *scanning crossover* that generalizes the standard uniform crossover and *diagonal crossover* that generalizes 1-point crossover, and study the effects of different number of parents on the GA behavior. We conduct experiments on tough function optimization problems and observe that by multi-parent operators the performance of GAs can be enhanced significantly. We also give a theoretical foundation by showing how these operators work on distributions.

1 Introduction

Natural and artificial recombination mechanisms (applied in Evolutionary Computation) are rather different. However, they all agree on the number of parents: it is either 1 or 2. As for natural reproduction, the absence of multi-parent reproduction can be understood if we consider the practical difficulties. For instance, one could think of the problems of getting more individuals "in the mood" at the same time and the same place, or those of having sophisticated female mechanisms to keep sperm alive until the necessary number of mating acts and conceptions are performed. In artificial evolutionary systems the restriction on the number of parents is less obvious. In fact, we can expect advantageous changes in the behavior of the classical uniform crossover, as well as of the classical 1-point crossover if we generalize them to $n > 1$ parents. Explanation of such positive expectations is given after defining the n-ary operators in section 2. The main goals of this paper are:

- present two multi-parent recombination mechanisms: scanning crossover and diagonal crossover;
- study the effect of using more than 2 parents on the behavior of the GA for both mechanisms;
- compare scanning crossover to diagonal crossover.

The paper [2] just shows empirical results regarding scanning crossover combined with a generational genetic algorithm using fitness proportional selection. The structure of the paper is the following. After the Introduction we give a description of (different versions of) the scanning crossover mechanism and the diagonal crossover. In Section 3 our test suit consisting of difficult numerical optimization problems and the parameters of the applied GA are presented, furthermore the performance measures are discussed. Thereafter the results of the experiments are displayed in Section 4. In Section 5 we study the expected value and the variation of fitness of a chromosome from generation to generation by tracing expected values of Walsh products. Finally, in Section 6, we draw our conclusions and give an explanation of the results.

2 Multi-parent recombination mechanisms

2.1 Scanning crossover

The simplest form of the scanning crossover mechanism studied in [2] works by taking n parent strings and creating one child through investigating the j-th ($j = 1, ..., k$, where k is the chromosome length) gene of the parents and chosing one of them to be the j-th gene of the child. Notice that the way the choice is made about the gene to be inherited is not specified. This allows different versions of gene scanning, distinguished by different choice mechanisms. Possible problem independent choice mechanisms are for instance uniform random choice, voting or random choice biased by the fitness of the parents. Figure 1 illustrates occurrence based (voting) scanning for bit-pattern representation, where the allele with the highest number of occurences should be inserted in the child.

Parent 1: 1111010111000110
Parent 2: 1100010101000010
Parent 3: 0011101010101011
Parent 4: 0101010101100100
Child : 1111010111000010

Fig. 1. Occurence based scanning crossover on bit patterns

The different choice mechanisms amount to a different level of bias in the genetic operator. In the meanwhile they all have in common that child construction is based on a larger (i.e. $n > 2$) sample of the search space than in classical

GAs, and that a promising gene is chosen for each position of the child. The definition of 'promising' can be problem independent (as in the above examples), but can also be based on on some problem specific heuristics. Therefore, scanning crossover is very well suited for being enriched with heuristics, moreover the presence and the influence of the incorporated heuristics is explicit. Let us note that the scanning crossover operator can also be adjusted for order based representation very easily as it is illustrated in [2].

In this paper we restrict ourselves to uniform scanning crossover, where the allele that is inserted in the child is chosen randomly by giving an equal chance to each parent to deliver its allele. Recall that the classical uniform crossover is a very disruptive operator. Applying an n-ary version can reduce the level of disruptivity by using a bigger sample of the search space and by creating only one child.

2.2 Diagonal crossover

Traditional crossover creates two children from two parents by splicing the parents along the single crossover point and exchanging the 'tails'. The basic idea behind diagonal crossover is to generalize this mechanism to an n-ary $(n-1)$-point crossover. Diagonal crossover selects $(n-1)$ crossover points resulting in n chromosome segments in each of the n parents and composes n children by taking the pieces from the parents 'along the diagonals'. For instance, the first child is composed by taking $substring_1$ from $parent_1$, $substring_2$ from $parent_2$, etc., while the second child would have $substring_1$ from $parent_2$, $substring_2$ from $parent_3$, etc. Figure 2 illuminates this idea.

$1a$	$1b$	$1c$	$parent-1$
$2a$	$2b$	$2c$	$parent-2$
$3a$	$3b$	$3c$	$parent-3$

$1a$	$2b$	$3c$	$child-1$
$2a$	$3b$	$1c$	$child-2$
$3a$	$1b$	$2c$	$child-3$

Fig. 2. Diagonal crossover with three parents

Notice that for $n = 2$ diagonal crossover coincides with the traditional 1-point crossover. The reason to expect that the use of more parents in diagonal crossover leads to improved GA performance is basically that the search becomes more more explorative, without hindering exploitation. The more explorative character is the result of having more crossover points and thus a higher level of disruptiveness, and the fact that using more parents there is more 'consensus' needed to focus the search to a certain region.

When considering the higher number of crossover-points in diagonal crossover the question obviously arises whether the same results could be achieved with the classical n-point crossover, which uses two parents. Therefore we decided to test this operator too, in order to see if the higher number of parents contributes to a better performance.

Genetic algorithms typically just use one gender, so do the multi-parent operators.

3 Test functions and setup of the experiments

We have decided to test multi-parent crossovers on tough optimization problems. We have chosen a test suit consisting of the second de DeJong function F2, the Ackley, the Griewangk, the Michalewicz, the Rastrigin and the Schwefel functions. All these functions, except for the F2 function, have a large number of local optima, which make those functions difficult to optimize. The defining formulas of these functions can be found in [4], [6], [9] and [11]. For each function we applied binary representation; the most important properties of the test functions and their representation are summarized in Table 1. Let us remark that the Michalewicz function is to be maximized, while all the others are to be minimized.

	F2	ACKL	GRIE	MICH	RAST	SCHE
dimension	2	30	10	2	20	10
chrom. length	22	600	200	33	400	210
global opt.	0	0	0	38.8503	0	0

Table 1. Properties of the test functions

In all of the experiments we used the same GA-setup, which is exhibited in Table 2.

When monitoring the performance we maintained different measures, namely efficiency (speed), and success rate (percentage of cases when an optimum was found). We measured speed by the total number of function evaluations (averaged over all runs). Let us note that the results on uniform scanning crossover deviate from those presented in [2] because there a generational GA with fitness proportional selection was used. Further tests, however, indicated that using a steady state GA with ranked selection yields better results.

4 Experimental results

We will review the results grouped around the two performance measures discused in the previous section: success rate and efficiency. Since 6 test functions

nr. of parents	2-10 or 2-15
GA type	steady state
selection mechanism	ranked bias 1.2
xover rate	0.7
mut. rate	1/chrom. length
pool size	200
max. nr. of func. eval.	70.000
termination cond.	optimum hit or population converged
averages over	100 runs

Table 2. GA parameters

and 2 performance measures would result in 12 Figures we only display some of them here. The complete date files can be obtained from the authors on request.

4.1 Success rates

Let us first consider the rate of success of the different operators, which is the most important measure from a strict optimization point of view. Table 3 shows the optimal versions of the genetic operators and the corresponding success rate for each test function; within brackets we displayed the success rate of the 2-parent versions.

test	Scanning Xover		Diagonal Xover		N-point Xover	
function	#par	succ.	#par	succ.	#Xover points	succ.
F2	7	.91 (.73)	11	.88 (.38)	11	.84
Mic	10	.72 (.57)	15	.76 (.34)	15	.6
Schw	2	.015 (.015)	15	.24 (.00)	10	.1
Grie	10	.48 (.22)	14	.32 (.04)	10	.15
Ras	5	.10 (.00)	13	.28 (.00)	15	.06
Ackl	8	.90 (.84)	15	.89 (.00)	10	.24

Table 3. Optimal nr. of parents and corresponding success rates. Within brackets the results for 2 parents. (For n-point crossover the number of parents is always 2.)

It appears immediately that the optimal number of parents is always higher than 2, with one exception. Also, the gains achieved by using more than two parents are substantial, especially for the diagonal crossover. The figures within brackets show an interesting phenomenon too. Namely, on all tests functions the

standard uniform crossover performs much better than 1-point crossover (diagonal crossover for two parents). At first sight one would expect uniform crossover to be too disruptive, as a single individual represents a vector of integers, so there is often a strong correlation between successive bits. Uniform crossover seems to disrespect these boundaries and turns out to be less sensitive for premature convergence. Looking at the results of diagonal crossover and n-point (2 parent) crossover we can see that the better performance of diagonal crossover is not only the consequence of applying more crossover points, but the higher number of parents contributes considerably.

As for the effect of different number of parents on success rates we observed that diagonal crossover yielded better results when the number of parents increased on all of the test functions. To illustrate this effect we show the success rates on the Griewangk and the Schwefel functions in Figure 3.

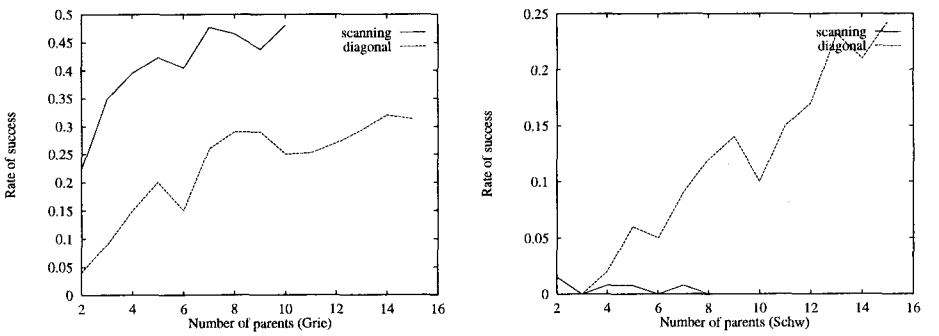

Fig. 3. Success rates on the Griewangk and the Schwefel functions

The success rates grow with the number of parents on these functions except for the scanning crossover on the Schwefel. The scanning crossover fails completely on the Schwefel function. This is a result of the large distance between the best and the second best optimum in this case. On the two-dimensional Michalewicz function and the F2 function the success rates of scanning crossover do not show that more or less monotonous growth that can be seen for diagonal crossover. Recall from Table 3, however, that the best performance was mostly obtained with more than two parents.

4.2 Efficiency

As for efficiency an increased performance means that it takes fewer function evaluations to reach a (sub)optimum if the number of parents increases. We observed this effect on the Ackley, the Griewangk and the Rastrigin functions for the diagonal crossover as well as for scanning crossover, see Figure 4.

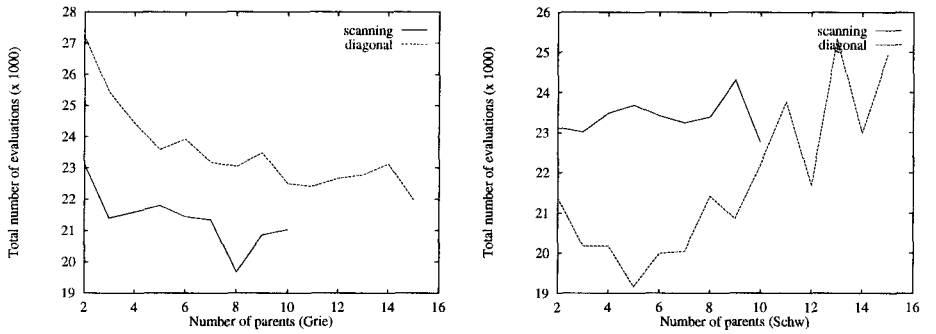

Fig. 4. Efficiency on the Griewangk and the Schwefel functions

Notice, however, the remarkable behavior of diagonal crossover on the Schwefel function: from $n = 5$ the number of function evaluations is growing as the number of parents is increased. This phenomenon also occured on the Rastrigin function and the Michalewicz function.

5 Distributions

In this section we consider the effect of uniform scanning crossover and diagonal crossover on distributions. This gives a theoretical basis for the comparison of these operators. This section is based on the paper [5], and we generalize here the results of that paper in that we give also formulae for uniform scanning crossover and diagonal crossover (and not only for uniform crossover and 1-point crossover).

A population will be regarded as a probability distribution over chromosomes. A distribution assigns to each chromosome x a probability $P(x)$: the probability that x occurs. Here we do not estimate the size of the population needed in order that the population evolves with but a minor deviation from the corresponding distribution.

Let f be the fitness function. We are interested in tracing the expected value of the fitness of an individual $E[f] = \sum_x f(x)P(x)$ from generation to generation and also its variance.

Instead of following the probabilities, we follow the expected values of Walsh products. The expected values of Walsh products are equivalent to probabilities in the sense that if we know the probabilities, then we can compute the expected values of the Walsh products and the other way around. Following the expected values of Walsh products is much more efficient than following probabilities in terms of computational effort, and the formulae are nicer and mathematically more tractable.

First we introduce some notation. We can interpret a chromosome x in three different ways:

1. as a bit string of length n, that is $x = x_1 \cdots x_n$, with $x_i \in \{0,1\}$ $i = 1, \ldots, n$,
2. as subset $\{i : x_i = 1\}$ of $\{1, \ldots, n\}$, or
3. as integer $\sum_{i=1}^{n} x_i \cdot 2^{i-1}$.

As an example, the bit string 10110 is equivalent to the set $\{1,3,4\}$ and to the integer 13. The main advantage of using different representations is that the formulae become simpler.

Given a bit string x and a set i, the Walsh product $R_{x,i}$ is either 1 or -1 and is computed as follows. Construct x' from x by replacing each 0 by -1. Then take the product of all those elements of x' whose index is in i. For example if $x = 10110$ and $i = \{1,2,5\}$, then $x' = 1\ -1\ 1\ 1\ -1$ and $R_{x,i} = 1 \cdot -1 \cdot -1 = 1$. Walsh products are used in the literature on genetic algorithms, for example to construct deceptive functions [3].

Formally, define the matrix of Walsh products R as follows $(x, i = 0, \ldots, 2^n - 1)$

$$R_{x,i} = \prod_{k \in i} (2x_k - 1).$$

These matrices can be constructed recursively as follows:

$$\mathcal{R}_0 = (1)$$

$$\mathcal{R}_{n+1} = \begin{pmatrix} \mathcal{R}_n & -\mathcal{R}_n \\ \mathcal{R}_n & \mathcal{R}_n \end{pmatrix}$$

Given a distribution, the expected values of Walsh products are (for $i = 0, \ldots, 2^n - 1$)

$$ER_i = \sum_x R_{x,i} P(x)$$

and, from the expected values of the Walsh products we can get the distribution back:

$$P(x) = \frac{1}{2^n} \sum_i R_{x,i} ER_i$$

The expected value of the fitness function $E[f]$ can be computed from the expected values of the Walsh products as follows. Define

$$r_i = \frac{1}{2^n} \sum_x R_{x,i} f(x)$$

Then

$$E[f] = \sum_i r_i ER_i$$

This is based on the fact that the fitness function can be written as a weighted sum over the Walsh products:

$$f(x) = \sum_i r_i R_{x,i}$$

We next give formulae how the expected values of Walsh products change under the genetic operators. In these formulae the primed values denote the values after applying the genetic operator.

Mutation (mutation rate p_m): Let $||i||$ denote the number of elements in the set i. Then

$$ER'_i = (1 - 2p_m)^{||i||} ER_i$$

Note, that if we applied only mutation to a distribution, then all ER_i would go to zero. This is in agreement with the intuition that we obtain a distribution in which every string has equal probability.

Uniform scanning crossover (k parents, crossover rate p_c): Let $\mathcal{S}_i = \{\langle i_1, \ldots, i_k\rangle : i_1 \cup \cdots \cup i_k = i \wedge \forall \alpha, \beta : \alpha \neq \beta \Rightarrow i_\alpha \cap i_\beta = \emptyset\}$. Then

$$ER'_i = (1 - p_c)ER_i + p_c \left(\frac{1}{k}\right)^{||i||} \sum_{\langle i_1,\ldots,i_k\rangle \in \mathcal{S}_i} \prod_{j=1}^{k} ER_{i_j}$$

In order to find the expected value of Walsh product i after uniform scanning crossover, we have to take a sum over subsets of i. These subsets need to be disjoint, and their union should be i. It is of interest to note that it does not matter if we first do mutation and then uniform scanning crossover or the other way around.

Diagonal crossover (k parents, crossover rate p_c): Define the set $\mathcal{S}$ of crossover-points by $\mathcal{S} = \{\langle i_0, \ldots, i_k\rangle : i_0 = 1 < i_1 < \cdots < i_k = n + 1\}$. Then

$$ER'_i = (1 - p_c)ER_i + p_c \frac{1}{||\mathcal{S}||} \sum_{\langle i_0,\ldots,i_k\rangle \in \mathcal{S}} \prod_{j=1}^{k} ER_{i \cap \{i_{j-1},\ldots,i_j - 1\}}$$

Note that the number of elements in the set $\mathcal{S}$ is significantly less than in the set $\mathcal{S}_i$ in the definition of the uniform scanning crossover. Hence, for diagonal crossover the expected values of the Walsh products are easier to compute than for the uniform scanning crossover.

Proportional selection: Let $xor(i, j)$ be the xor-operator on bit string. For example, $xor(1010, 1001) = 0011$. Then

$$ER'_i = \frac{\sum_j r_{xor(i,j)} ER_j}{\sum_j r_j ER_j}$$

Now we can use the expected values of Walsh products to trace the expected value of the fitness. Using the above transformation formulae, the value $E[f]$ can be traced from generation to generation. We took the length of the bit string $n = 10$, and the fitness function an 1-dimensional inverted rastrigin function with an optimal fitness of 100. We start from an initial distribution in which every string has equal probability. We took $p_c = 1$ and $p_m = 0$. Then in Figure 5 the expected values of the fitness of a chromosome for the uniform scanning

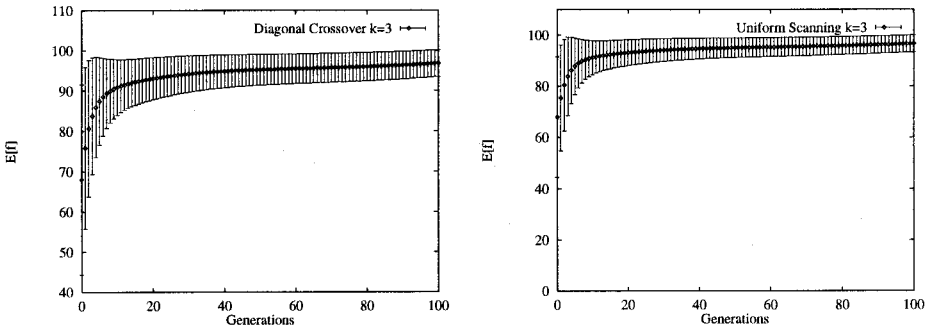

Fig. 5. Expected value of the fitness of a bit string (and its variance) from generation to generation. On the left is diagonal crossover and on the right is uniform scanning crossover.

crossover and the diagonal crossover with three parents are plotted. There is not much difference in the two graphs, and if we look at the underlying data we see that the difference (for each generation) of the expected values of the fitness is never bigger than one.

The second moment is given by $\sum_i ER_i \sum_j r_j r_{xor(i,j)}$, from which the variance can be computed in the standard way.

6 Conclusions and further work

The main conclusion that can be drawn from the experiments is that using more parents does increase GA performance. However, refinements of this statement are necessary due to the different multi-parent recombination mechanisms we tested and the different performance measures we considered.

As for the success rate (the percentage of cases when an optimum is found) we can conclude that both operators, i.e. scanning crossover and diagonal crossover, achieve their best performance when using more than 2 parents, see Table 3. The only exception is scanning crossover on the Schwefel function, which is probably a random effect, since scanning crossover has a very low success rate on this function. We observed a positive correlation between the number of parents and the success rate for diagonal crossover. For scanning crossover this is not always the case. An important conclusion from Table 3 is that the increased success rates of diagonal crossover are not simply the consequence of using more crossover points. The comparison between n-point crossover, which takes two parents, and diagonal crossover shows that the usage of more parents substantially contributes to the better results.

It seems that of all measures we considered more parents have the least effect on the efficiency, i.e. the total number of function evaluations. This is actually not surprising if we realize that using more parents makes it harder for a

super-chromosome to deliver its identical copies in the next generation. In other words, if the number of parents increases, so do the expected takeover times. This implies a more diverse search with a reduced danger of premature convergence. Nevertheless, diagonal crossover shows a roughly decreasing number of evaluations when the number of parents is increasing on three test function (F2, Griewangk and Ackley). This effect comes together with an increasing success rate, thus a 'double profit' is made. On the Michalewicz and the Schwefel function, however, the curves are rather increasing than decreasing. Scanning seems to get faster by using more parents on the Ackley, the Griewangk and the Rastrigin functions, and - just like diagonal crossover - gets visibly slower on the Michalewicz function.

In explaining the above results two notions play an important role: disruptiveness and takeover times. Disruptiveness is a powerful means to prevent premature convergence. Nevertheless, a very disruptive operator might prevent the global optimum from being found. This stresses the importance of a 'right proportion' of disruptiveness. Increasing the number of parents results in an increase of the disruptiveness of an operator, as less schemata will be preserved. So, by tuning the number of parents we can tune the level of disruptiveness by little steps.

The comparison between diagonal crossover and 2-parent n-point crossover provides evidence that we are also dealing with another effect that is not related to disruptiveness. Take-over times might be an important parameter. When recombination takes part between n parents the chance that a complete copy of one of the parents occurs among the offspring becomes smaller as n increases, that is a larger fraction of the population has to be centered around an optimum before the complete population converges to this optimum. Thus, when using more than two parents, the time it takes for a single good individual to take over the complete population will increase.

When looking for the best operator for function optimization we cannot appoint a clear winner. As the figures in Table 3 show, scanning crossover wins on F2, the Griewangk and the Ackley functions, while diagonal crossover is better on the Michalewicz, the Schwefel and the Rastrigin functions. However, it should be noted that diagonal crossover is a cheaper operator. Namely, uniform scanning crossover requires the generation of many random numbers, which makes it computationally more expensive. An interesting fact is that the profit of more parents for diagonal crossover is the highest on the problems with long chromosomes cq. high dimensional problems.

Finally, let us make a note to put these results in a broader context. Evolutionary computation consists of three main branches: Genetic Algorithms, Evolution Strategies and Evolutionary Programming. One of the main differences between GAs and the other two branches is the arity of the typically applied operators. Using binary operators, i.e. sexual reproduction, is inherent to GAs, while this is not the case in ES and EP. There are researchers who question the usefulness of sex in Evolutionary Computation, and indeed, in some experimental comparisons ES and EP exhibit better performance then GAs, [1]. Some

recent publications show that GAs can be enhanced by new features, out of the traditional GAs paradigm, such as Lamarckian/Baldwinian effects or by applying a problem decomposition, [9], [11]. In this paper we follow another approach. We do not leave the GA paradigm, but rather boost it, that is we raise the extent of sexuality by using 'orgies', i.e. multi-parent reproduction. The results show that there are very promising possibilities within the GA paradigm.

Currently we are doing further research to obtain a better understanding of the behavior of the multi-parent operators. One of our tools is the theoretical model described in section 5. Furthermore we are trying to identify some guidelines for selecting the operator and choosing the most appropriate number of parents.

Acknowledgements: For our tests we used LibGA, provided by A. Corcoran.

References

1. T. Bäck and H.-P. Schwefel, *An Overview of Evolutionary Algorithms for Parameter Optimization,* Journal of Evolutionary Computation 1(1), 1993, pp. 1-23.
2. A.E. Eiben, P.-E. Raué and Zs. Ruttkay, *Genetic Algorithms with Multi-parent Recombination,* in Proceedings of ICEC/PPSN 3, eds. Y. Davidor, H.-P. Schwefel and R. Männer, LNCS 866, Springer-Verlag, 1994, pp. 78-87.
3. D.E. Goldberg *Genetic Algorithms and Walsh functions: Part I, A gentle introduction Complex Systems* 3:129-152, 1989.
4. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, 1989.
5. J.N. Kok, P. Floréen and M. Rasch *Tracing Expected Fitness Values in Genetic Algorithms using Bit products and Walsh Products,* Technical Report, Utrecht University, 1994.
6. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* Springer-Verlag, 2nd edition, 1994.
7. H. Mühlenbein, *Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization,* in Proceedings of ICGA-89, eds. J. Schaffer, Morgan Kaufmann, 1989, pp. 416-421.
8. K.F. Pál, *Selection Schemes with Spatial Isolation for Genetic Optimization,* in Proceedings of ICEC/PPSN 3, eds. Y. Davidor, H.-P. Schwefel and R. Männer, LNCS 866, Springer-Verlag, 1994, pp. 170-179.
9. M.A. Potter and K.A. De Jong *A Cooperative Coevolutionary Approach to Function Optimization,* in Proceedings of ICEC/PPSN 3, eds. Y. Davidor, H.-P. Schwefel and R. Männer, LNCS 866, Springer-Verlag, 1994, pp. 249-257.
10. G. Seront and H. Bersini, *In Search of a Good Evolution-Optimization Crossover,* in Proceedings of PPSN 2, eds. R. Männer and B. Manderick, North-Holland, 1992, pp. 479-488.
11. D. Whitley, V. Scott Gordon and K. Mathias *Lamarckian Evolution, the Baldwin Effect and Function Optimization,* in Proceedings of ICEC/PPSN 3, eds. Y. Davidor, H.-P. Schwefel and R. Männer, LNCS 866, Springer-Verlag, 1994, pp. 6-15.