# From recombination of genes to the estimation of distributions I. Binary parameters

H. Mühlenbein[1] and G. Paaß[1]

GMD – Forschungszentrum Informationstechnik, 53754 Sankt Augustin , Germany

**Abstract.** The Breeder Genetic Algorithm (BGA) is based on the equation for the response to selection. In order to use this equation for prediction, the variance of the fitness of the population has to be estimated. For the usual sexual recombination the computation can be difficult. In this paper we shortly state the problem and investigate several modifications of sexual recombination. The first method is gene pool recombination, which leads to marginal distribution algorithms. In the last part of the paper we discuss more sophisticated methods, based on estimating the distribution of promising points.

## 1 Introduction

The Breeder Genetic Algorithm (BGA) is based on the classical science of livestock breeding. The central part of the theory is the equation for the response to selection

$$R(t) = b(t) \cdot I \cdot \sigma(t) \tag{1}$$

Here $R$ denotes the response, which is defined as the difference between the mean fitness of the population at generation $t+1$ and $t$, $b(t)$ is the *realized heritability*, $I$ is the *selection intensity* and $\sigma$ is the standard deviation of the fitness [12]. If $b(t)$ and $\sigma(t)$ can be estimated, the equation can be used for predicting the mean fitness of the population. In livestock breeding many methods have been developed to estimate the heritability [12], estimating the variance is still an open question [15].

But in evolutionary computation we have more freedom. We can design new recombination operators which have no counterpart in nature and use the above equation to evaluate the operators. In [13] we have made a further step away from the biological example and investigated *gene pool recombination* GPR. With GPR the genes of all selected parents are used to create offspring. The microscopic view of recombining two chromosomes in a Mendelian manner is abandoned.

In this paper we shortly show why the analysis of sexual recombination is so difficult, then we investigate two algorithms which use univariate marginal distribution of selected points to generate new points. In the last section the problem of estimating distributions is shortly discussed. The conditional distribution algorithm is outlined and applied to optimization problems known to be difficult for genetic algorithms.

## 2 Analysis of uniform crossover for two loci

The difficulty of analyzing Mendelian sexual recombination will be shown with a simple example, namely two loci and proportionate selection. In this case there are four possible genotypes: $(0,0),(0,1),(1,0)$, and $(1,1)$ which we index by $j=(0,1,2,3)$. We denote their fitness values $f_0, f_1, f_2$, and $f_3$ respectively. Let $q_j(t)$ be the frequency of genotype $j$ at generation $t$. For simplicity we restrict the analysis to *uniform crossover* [18]. It is an example of *two parent recombination (TPR).*

**Theorem 1.** *For proportionate selection and uniform crossover the gene frequencies obey the following difference equation*

$$q_j(t+1) = \frac{f_j}{\bar{f}(t)} q_j(t) + \frac{1}{2}\epsilon_j \frac{D_s(t)}{\bar{f}(t)^2} \quad j=0,1,2,3, \tag{2}$$

*where* $\boldsymbol{\epsilon} = (-1,1,1,-1)$; $\bar{f}(t) = \sum_{j=0}^{3} f_j q_j(t)$ *is the average fitness of the population; and* $D_s(t)$ *is defined as*

$$D_s(t) = f_0 f_3 q_0(t) q_3(t) - f_1 f_2 q_1(t) q_2(t). \tag{3}$$

**Proof:** For proportionate selection the gene frequencies $q_j^s$ after selection are given by

$$q_j^s(t) = \frac{f_j}{\bar{f}(t)} q_j(t)$$

Now we pair randomly between the selected parents and count how often genotype $j$ arises after uniform crossover. We take $j=0$ as an example. We easily obtain

$$q_0(t+1) = q_0^s(t)\,(q_0^s(t) + q_1^s(t) + q_2^s(t) + 1/2 q_3^s(t)) + 1/2 q_1^s(t) q_2^s(t)$$

Using that $q_0^s(t) + q_1^s(t) + q_2^s(t) + q_3^s(t) = 1$ we obtain the conjecture for $j=0$. The remaining equations are obtained in the same manner. □

Equations (2) are identical to the ones known for diploid chromosomes in population genetics [6], despite the fact that the underlying genetic recombination is different. This shows that uniform crossover can be thought of as Mendelian recombination for haploid organisms. Note that $D_s(t) = 0$ if $q_0(t)q_3(t) = q_1(t)q_2(t)$ and $f_0 f_3 = f_1 f_2$. The first condition is called *linkage equilibrium* in population genetics.

This system of four nonlinear difference equations has not yet been solved analytically (see the discussion in [15]), but it is possible to derive an exact expression for the realized heritability.

**Theorem 2.** *The realized heritability* $b(t)$ *for uniform crossover is given by*

$$b(t) = 1 - \frac{1}{2}(f_0 - f_1 - f_2 + f_3)\,\frac{D(t)}{\bar{f}(t)V(t)}. \tag{4}$$

**Proof:** *By summation we obtain*

$$R(t) = \bar{f}(t+1) - \bar{f}(t) = \frac{V(t)}{\bar{f}(t)} - \frac{1}{2}(f_0 + f_3 - f_1 - f_2)\frac{D(t)}{\bar{f}(t)^2}, \tag{5}$$

*where* $V(t) = \sigma^2(t)$ *denotes the variance of the population. Using* $S(t) = V(t)/\bar{f}(t)$ *we obtain equation (4).* □

Uniform crossover in genetic algorithms, which models Mendelian recombination, leads to very difficult systems of difference equations. The genetic population moves away from linkage equilibrium. This makes an analysis of the algorithm almost impossible. But in genetic algorithms we may use recombination schemes which lead to simpler equations.

Simpler equation are obtained if the population is in linkage equilibrium. Without proof we note that linkage equilibrium is identical to the gene frequencies being in Robbins proportions [16]. Here the probability of a genotype $p(\mathbf{x})$ is given by

$$p(\boldsymbol{x}) = \prod_{i=1}^{n} p_i(x_i), \tag{6}$$

where $p_i(x_i)$ are the univariate marginal frequencies.

The assumption of linkage equilibrium is not as severe as one might think. We have numerically confirmed the conjecture that without selection, the gene frequencies of a population using uniform crossover will converge to linkage equilibrium. This means that linkage equilibrium can be considered to be the limiting distribution of any genetic recombination scheme applied without selection.

## 3 Univariate Marginal Distributions

There exist a simple recombination scheme that maintains the population in linkage equilibrium; we have called it *gene pool recombination (GPR)* [13]. In GPR, for each locus the two alleles to be recombined are chosen *independently* from the gene pool defined by the selected parent population. The biologically inspired idea of restricting the recombination to the alleles of two parents for each offspring is abandoned.

**Definition:** *In gene pool recombination the two "parent" alleles of an offspring are randomly chosen for each locus with replacement from the gene pool given by the parent population* **selected** *before. Then the offspring allele is computed using any of the standard recombination schemes for TPR.*

For a discussion of gene pool recombination and its analysis see [13]. Gene pool recombination leads to simple difference equations for the marginal frequencies $p_i(x_i)$ [13]. We generalize this idea and define a conceptual algorithm which uses univariate marginal frequencies directly.

Let $\mathbf{x} = (x_1, \ldots, x_n)$, $x_i \in \{0, 1\}$, $f(\mathbf{x})$ be its fitness and $q(\mathbf{x})$ its frequency. Then the univariate marginal frequencies can be computed from

$$p_i(x_i) = \sum_{\mathbf{x}|x_i} q(\mathbf{x}) \tag{7}$$

where the sum is taken over all $\mathbf{x}$ with $x_i$ held fixed. The conceptual Univariate Marginal Distribution Algorithm (UMDA) is defined as follows.

**UMDA**

- **STEP0:** Set $t = 1$. Generate $N \gg 0$ points randomly.
- **STEP1:** Select $M \leq N$ points according to a selection schedule. Compute the marginal frequencies $r_{t;i}(x_i)$ of the selected set.
- **STEP2:** Generate $N$ new points according to the distribution $q_{t+1}(\mathbf{x}) = \prod_{i=1}^{n} r_{t;i}(x_i)$. Set $t = t + 1$.
- **STEP3:** If not terminated, go to STEP1

**Theorem 3.** *For proportionate selection the marginal frequencies of UMDA obey the difference equation*

$$p_{t+1;i}(x_i) = p_{t;i}(x_i)\frac{\bar{f}_{t;i}(x_i)}{\bar{f}(t)} \tag{8}$$

*where*

$$\bar{f}_{t;i}(x_i) = \sum_{\mathbf{x}|x_i} f(\mathbf{x}) \prod_{\substack{j=1 \\ j\neq i}}^{n} p_{t;j}(x_j)$$

**Proof:** For proportionate selection the frequency of the selected points $\tilde{q}_t(\mathbf{x})$ is given by

$$\tilde{q}_t(\mathbf{x}) = \frac{f(\mathbf{x})}{\bar{f}^t} q_t(\mathbf{x})$$

From

$$r_{t;i}(x_i) = p_{t+1;i}(x_i) = \sum_{\mathbf{x}|x_i} \tilde{q}_t(\mathbf{x})$$

equation (8) follows. □

The difference equation (8) can also be written in the form

$$p_{t+1;i}(x_i) = p_{t;i}(x_i) + p_{t;i}(x_i)\frac{F_{t;i}(x_i)}{\bar{f}(t)} \tag{9}$$

where

$$F_{t;i}(x_i) = \bar{f}_{t;i}(x_i) - \bar{f}(t) \tag{10}$$

The term $F_{t;i}(x_i)$ was already introduced in [1] ( there it is denoted $f_{(i)}(x_i)$). The terms $F_{t;i}(x_i)$ minimize the weighted quadratic error

$$\sum_{\mathbf{x}} q_t(\mathbf{x}) \left( f(\mathbf{x}) - \bar{f}(t) - \sum_{i=1}^{n} \alpha_i(x_i) \right)^2$$

The terms are used to define the *additive genetic variance* $V_A$, called $V_1$ in [1].

$$V_A(t) = \sum_{x_i} p_{t;i}(x_i) \left(F_{t;i}(x_i)\right)^2 \tag{11}$$

It is obvious that $p_{t+1;i}(x_i) = p_{t;i}(x_i)$ iff $F_{(i)}(x_i) = 0$ or $p_{t;i}(x_i) = 0$. Therefore we obtain:

**Corollary:** *For proportionate selection the UMDA stays in equilibrium iff $V_A = 0$.*

The response to selection is zero if the additive variance is zero. UMDA only exploits the additive genetic variance. It is interesting to note that this result is also used as a rule of thumb in livestock breeding. Up to now no rigorous mathematical proof is available.

The corollary also implies that UMDA is not a global optimization method for difficult fitness functions. This problem has already been discussed for gene pool recombination in [13].

The following theorem is a correct version of Fisher's Fundamental Theorem of Natural Selection [8], restricted to populations in linkage equilibrium.

**Theorem 4.** *For UMDA with proportionate selection the response to selection is given by*

$$R(t) = \frac{V_A(t)}{\bar{f}(t)} + \sum_x \Delta q(\boldsymbol{x}) \left( f(\boldsymbol{x}) - \bar{f}(t) - \sum_{i=1}^{n} F_{t;i}(x_i) \right) \tag{12}$$

*where $\Delta q(\boldsymbol{x}) = q_{t+1}(\boldsymbol{x}) - q_t(\boldsymbol{x})$ is the difference of the frequencies of genotype $\boldsymbol{x}$.*

**Proof:**

$$\begin{aligned}
\sum_x \Delta q(\boldsymbol{x}) \sum_{i=1}^{n} F_{t;i}(x_i) &= \sum_{i=1}^{n} F_{t;i}(x_i) \sum_{x|x_i} \Delta q(\boldsymbol{x}) \\
&= \sum_{i=1}^{n} F_{t;i}(x_i) \Delta p_i(x_i) \\
&= \sum_{i=1}^{n} p_{t;i} F_{t;i}^2(x_i) / \bar{f}(t)
\end{aligned}$$

Here $\Delta p_i(x_i) = p_{t+1;i}(x_i) - p_{t;i}(x_i)$. Equation (9) was used for the last step. By using $\sum_x \Delta q(\boldsymbol{x}) f(\boldsymbol{x}) = R(t)$ and $\sum_x \Delta q(\boldsymbol{x}) = 0$, the conjecture is obtained. □

**Corollary:** *The realized heritability of any algorithm based on univariate marginal distributions can be estimated by*

$$b(t) \approx \frac{V_A(t)}{V(t)}. \tag{13}$$

**Proof:** We have

$$R(t) \approx \frac{V_A(t)}{\bar{f}(t)} = \frac{V_A(t)}{V(t)} \frac{V(t)}{\bar{f}(t)} = \frac{V_A(t)}{V(t)} S(t). \quad \square$$

$V_A(t)/V(t)$ is called *heritability in the narrow sense* in livestock breeding [7], abbreviated by $h_n^2$. Estimating $h_n$ is one of the most difficult parts in the science of livestock breeding.

## 4 A simple UMDA implementation

In order to implement UMDA, estimates for the marginal distributions are necessary. Especially the computation of $F_{t;i}(x_i)$ in Equation 9 is fairly computing intensive. Independently of the theory presented in this paper a simple algorithm has been already proposed in [2]. In this algorithm the univariate marginal frequencies are updated according to

$$p_{t+1;i}(x_i) = p_{t;i}(x_i) + \lambda(r_{t;i}(x_i) - p_{t;i}(x_i)) \tag{14}$$

where $r_{t;i}(x_i)$ are the marginal frequencies of the selected points and $\lambda$ is a control parameter. The resulting algorithm we call the simple univariate marginal distribution algorithm (SUMDA).

**SUMDA**

- **STEP0:** Set $t = 1$. Set $p_{1,i}(x_i)$.
- **STEP1:** Generate $N$ new points according to the distribution $q_{t+1}(\mathbf{x}) = \prod_i^n p_{t;i}(x_i)$.
- **STEP2:** Select $M \leq N$ points according to a selection schedule. Compute the marginal frequencies $r_{t;i}(x_i)$ of the selected set.
- **STEP3:** Update the marginal frequencies according to equation (14). Set $t = t + 1$.
- **STEP4:** If not terminated, go to STEP1

Note that $\lambda$ influences the speed of convergence. The smaller $\lambda$, the less the convergence speed. Before we show some computational results, we qualitatively analyze the algorithm. We simplify the notation $p_{t;i}(x_i) \equiv p(t)$ and $r_{t;i}(x_i) \equiv r(t)$. We start with the simplest case.

**Theorem 5.** *Assume that* $r(t) \equiv c$ *with* $0 \leq c \leq 1$. *Then*

$$p(t+1) = p(1)(1-\lambda)^t - c(1-\lambda)^t + c \quad t = 0, 1, \ldots \tag{15}$$

The proof is straightforward and will be omitted. We obviously have

$$lim_{t \to \infty} p(t) = c$$

In a real algorithm $r(t)$ will oscillate. A qualitative analysis of the SUMDA algorithm has been first made in [11].

**Theorem 6.** *If the difference Equation (14) can be approximated by the differential equation*

$$\frac{dp(t)}{dt} = \lambda \left( r(t) - p(t) \right), \tag{16}$$

*the solution is given by*

$$p(t) = p(1)e^{-\lambda t} + \lambda e^{-\lambda t} \int_0^t r(\tau) e^{\lambda \tau} d\tau \tag{17}$$

If $r(t) \equiv c$ one obtains an approximation of equation (15). But in real simulations one observes that SUMDA often consists of two phases. In the first phase ($0 \leq t \leq t_1$) $r(t)$ more or less randomly oscillates about a mean $< r(t) >_0^{t_1}$. If $p(t)$ gets more focused, then $r(t)$ changes accordingly.

In table 1 we give numerical results for the linear function $ONEMAX$. Note how $\lambda$ influences the convergence speed. Because the size of the population, $N$, is very large, the speed of convergence is almost independent of the size of the problem $n$. For difficult multi modal fitness functions the success of SUMDA depends on the parameter $\lambda$ and $N$. We have to omit this discussion here. But it should be obvious that SUMDA suffers from the problem, all algorithms using marginal distributions only have: they are not able to handle higher order gene interactions.

| | $n = 30$ | | $n = 30$ | | $n = 60$ | | $n = 90$ | |
|---|---|---|---|---|---|---|---|---|
| t | $\bar{p}$ | $std(p)$ | $\bar{p}$ | $std(p)$ | $\bar{p}$ | $std(p)$ | $\bar{p}$ | $std(p)$ |
| 10 | 0.726 | 0.049 | 0.952 | 0.024 | 0.887 | 0.086 | 0.834 | 0.122 |
| 20 | 0.893 | 0.025 | 0.997 | 0.001 | 0.993 | 0.005 | 0.985 | 0.014 |
| 30 | 0.963 | 0.009 | 1.000 | 0.000 | 1.000 | 0.001 | 0.999 | 0.001 |

**Table 1.** SUMDA: $N = 1024; \lambda = 0.1, n = 30; \lambda = 0.25$ else

## 5 Conditional distributions

Gene pool recombination with two parent mating and uniform crossover as well as the two marginal distribution algorithms UMDA and SUMDA exploit the additive genetic variance mainly. The suitability of these algorithms for solving optimization problems with strongly interacting genes at different loci seems limited.

An extension of univariate marginal distribution algorithms are multivariate ones. Unfortunately it is difficult to generate the probabilty $p(\boldsymbol{x})$ of genotype $\boldsymbol{x}$ from multivariate marginal distributions. We demonstrate the problem with an example. For $n = 4$ loci for instance, we may use $p(\boldsymbol{x}) = p(x_1, x_2)\, p(x_3, x_4)$. But then four of the six bivariate distribution are left out. There are methods to solve

this problem by using a system of equations as constraints, but the number of multivariate distributions scales exponentially. Therefore it seems easier to use *conditional distributions* $p(x_i|x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$ to reconstruct interactions between the variables. We use the notation

$$\boldsymbol{x}_{-i} := (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n).$$

Then $p(x_i|\boldsymbol{x}_{-i})$ is the probability of $x_i$ given $\boldsymbol{x}_{-i}$. Besag [3] has proven that the $n$ different conditional distributions $p(x_i|\boldsymbol{x}_{-i})$, $i=1,\ldots,n$, completely determine the joint distribution $p(\boldsymbol{x})$. In our algorithm we will use conditional distributions $p(x_1,\ldots,x_m|x_{m+1},\ldots,x_n)$. In order to keep the description simple we will start with an algorithm using $p(x_i|\boldsymbol{x}_{-i})$ only.

There are a number of methods in statistics that estimate conditional distributions. We selected regression tree methods because they are reasonably accurate and computationally cheap. Algorithms for classification and regression trees date back to Sonquist and Morgan [17] and have been improved by Breiman et al.[4], see also [5] [14].

It turns out that the estimation of conditional distributions is very difficult. We are not able to describe our conditional distribution algorithm CDA here. It uses tree regression to estimate conditional distributions, and the Metropolis-Hastings algorithm to correct the estimates. In order to speed up the computation we use a cluster algorithm to compute the conditional distributions for correlated variables. Furthermore we scale the probability distribution $p(\boldsymbol{x})$ in order to concentrate its mass near the optima of $f(\boldsymbol{x})$.

In the next Section we give first numerical results.

## 6 Numerical results

Deceptive problems have been introduced by Goldberg [9] as a challenge to genetic algorithm. For these functions genetic algorithms will converge to suboptimal points. To overcome this problem, a radically different type of genetic algorithm called a *messy genetic algorithm* has been devised and tested by Goldberg and his coworkers. In a messy genetic algorithm the interaction of genes is tested with substrings in a primordial phase. Of crucial importance is the initialization. The interested reader should consult [10] for a recent update of the messy genetic algorithm applied to deceptive problems.

Our conditional distribution algorithm CDA tries to determine the important interactions more globally with well-known statistical techniques. The implementation of CDA is much more difficult than a genetic algorithm or an algorithm using univariate marginal distributions only.

For our numerical experiments we used the following deceptive functions.

$$f_3(d,\boldsymbol{x}) = \sum_{i=0}^{[n/3]-1} g_3(d,x_{3i+1},x_{3i+2},x_{3i+3}) \tag{18}$$

$$\tag{19}$$

where

$$g_3(d, x_1, x_2, x_3) = \begin{cases} 1-d, & \sum x_i = 0; \\ 1-2\,d, & \sum x_i = 1; \\ 0, & \sum x_i = 2; \\ 1, & \sum x_i = 3. \end{cases} \tag{20}$$

(21)

The function $f_5$ is defined similarly to $f_3$. The function $f_{(5,3)}$ has clusters of three and five interacting variables. By using an exponential transformation the marginal distributions of the clusters are independent from each other.

| n | func | N | $\beta$ | $eval$ |
|---|---|---|---|---|
| 50 | $f_3,\ d = 0.1$ | 1000 | (4, 25) | 57000 |
| 50 | $f_5, Goldberg$ | 1000 | (4, 45) | 121000 |
| 50 | $f_5,\ d = 0.1$ | 2000 | (4, 45) | 710000 |
| 50 | $f_{(5,3)},\ d = 0.1$ | 1000 | (4, 45) | 911000 |
| 100 | $f_3,\ d = 0.1$ | 1000 | (5, 35) | 100000 |
| 100 | $f_5,\ d = 0.1$ | 4000 | (5, 45) | 4960000 |
| 200 | $f_3,\ d = 0.1$ | 1000 | (5, 45) | 500000 |

**Table 2.** Numerical results for CDA

Preliminary numerical results are presented in the Table 2. They clearly show that the algorithm is able to solve large deceptive problems. But the algorithm is at this stage more a conservative statistical estimation procedure than an optimization algorithm. It will take some time and lots of numerical experiments to end up with an efficient and reliable optimization algorithm.

Our results are not directly comparable to [10], because there a a deceptive function with a fitness value of 0.58 for the local optimum $\boldsymbol{x} = (0, 0, 0, 0, 0)$ is used. We have used a fitness value of 0.9. Goldberg's deceptive function is substantially easier to solve, because the difference between the global and the local optimum is larger. Our algorithm needs for Goldberg's deceptive function of $n = 50$ about 1/6 function evaluations compared to our deceptive function.

Nevertheless, Goldberg's messy genetic algorithm seems to need substantially less function evaluations than our algorithm. But in our opinion, the messy genetic algorithm uses the cluster size as a priori information! Furthermore, the interacting variables are supposed to be contiguously located. Our algorithm detects all interactions without prior information. This is shown with the function $f_{(5,3)}$. Here clusters of size 3 and size 5 alternate, unknown to the algorithm.

From our statistical experience we believe that it is impossible to detect all important gene interactions by simply manipulating substrings like it is done in the messy genetic algorithm. Whether our conjecture is true the future will show, when experiments with a variety of deceptive functions are made.

## References


1. H. Asoh and H. Mühlenbein. Estimating the heritability by decomposing the genetic variance. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 866, pages 98–107. Springer-Verlag, 1994.
2. S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In *Proc. of the 12th Intern. Conf. on Machine Learning*, Lake Tahoe, 1995.
3. J. Besag. Spatial interaction and the statistical analysis of lattice systems. *J. Royal Statistical Society, Series B*, pages 192–236, 1974.
4. L. Breiman, J.H. Friedman, R. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, California, 1984.
5. W. Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.
6. J. F. Crow and M. Kimura. *An Introduction to Population Genetics Theory*. Harper and Row, New York, 1970.
7. D. S. Falconer. *Introduction to Quantitative Genetics*. Longman, London, 1981.
8. R. A. Fisher. *The Genetical Theory of Natural Selection*. Dover, New York, 1958.
9. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, 1989.
10. D.E. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In S. Forrest, editor, *Proc. of the Fith Int. Conf. on Genetic Algorithms*, pages 56–64, San Mateo, 1993. Morgan-Kaufman.
11. V. Kvasnicka, M. Pelikan, and J. Pospichal. Hill-climbing with learning: An abstraction of genetic algorithm. Technical report, Slovak Technical University, Bratislava, 1995.
12. H. Mühlenbein and D. Schlierkamp-Voosen. The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, 1:335–360, 1994.
13. H. Mühlenbein and H.-M. Voigt. Gene pool recombination in genetic algorithms. In J.P. Kelly and I.H Osman, editors, *Metaheuristics: Theory and Applications*, Norwell, 1996. Kluwer Academic Publisher.
14. K.V.S. Murthy. *On Growing Better Decision Trees from Data*. PhD thesis, The John Hopkins University, Baltimore, Maryland, 1995.
15. T. Naglyaki. *Introduction to Theoretical Population Genetics*. Springer, Berlin, 1992.
16. R.B. Robbins. Some applications of mathematics to breeding problems iii. *Genetics*, 3:375–389, 1918.
17. J.N. Sonquist and J.N. Morgan. *The Detection of Interaction Effects*, volume Monograph 35. Survey Research Center, Institute for Social Research, University of Michigan, 1964.
18. G. Syswerda. Uniform crossover in genetic algorithms. In H. Schaffer, editor, *3rd Int. Conf. on Genetic Algorithms*, pages 2–9, San Mateo, 1989. Morgan Kaufmann.


This article was processed using the LaTeX macro package with LLNCS style