Gene pool recombination, genetic algorithm, and the onemax function[1]

Heinz Mühlenbein[2] and Uday K. Chakraborty[3]

[2] GMD Forschungszentrum Informationstechnik, SET.AS, Schloß Birlinghoven, Sankt Augustin, Germany
[3] Department of Computer Science and Engineering, Jadavpur University, Calcutta 700 032, India

In this paper we present an analysis of *gene pool recombination* in genetic algorithms in the context of the *onemax* function. We have developed a Markov chain framework for computing the probability of convergence, and have shown how the analysis can be used to estimate the *critical population size*. The Markov model is used to investigate drift in the multiple-loci case. Additionally, we have estimated the minimum population size needed for optimality, and recurrence relations describing the growth of the advantageous allele in the infinite-population case have been derived. Simulation results are presented.

Keywords: Genetic algorithm, gene pool recombination, onemax function, Markov chain, convergence

1. Introduction

Genetic algorithms (GAs) are a special kind of evolutionary algorithm developed by Holland (1975). These algorithms are a class of stochastic, adaptive, general-purpose search heuristics based on concepts borrowed from the principles of natural selection and population genetics. These algorithms use computational models of evolutionary processes as basic elements in the design and analysis of computer-based problem-solving systems. Genetic algorithms usually outperform classical gradient search techniques and various forms of random search and heuristics on more difficult, complex search spaces. Numerous variants of the original genetic algorithm have been applied to a wide variety of problems in science, engineering, economics, business and other fields (see, for example, Belew and Booker, 1991; Forrest, 1993; Eshelman, 1995; Saha and Chakraborty, 1997). For some recent advances in theoretical research on genetic algorithms, see Chakraborty et al. (1996), Chakraborty (1995a, 1995b), Chakraborty and Dastidar (1993), Chakraborty and Mühlenbein (1997), Mühlenbein and Schlierkamp–Voosen (1993, 1994), Mühlenbein (1991, 1992), Whitley (1993), Whitley and Vose (1995).

Genetic algorithms are radically different from the traditional search and optimization algorithms. They operate on the principle of a random yet intelligent search. These algorithms do not presuppose any special property (e.g., continuity, differentiability, etc.) of the search space. They maintain a population of individuals that evolve according to the rules of a number of artificial genetic operators, such as *selection*, *recombination* (also called *crossover*), and *mutation*. The individuals are selected (to create a mating pool) in proportion to some monotone function of their relative fitness. They are then crossed and mutated, to create the next generation of (possibly better) individuals.

A new type of recombination operator, called *gene pool recombination*, was introduced in Mühlenbein and Voigt (1996). It was shown in that paper that gene pool recombination possesses the property of keeping the population binomially distributed. In the present paper we build on that work, by analyzing the behavior of gene pool recombination when it is applied on the linear bit-counting function (also called ONEMAX in the literature (e.g., Ackley, 1987; Syswerda, 1989)). Specifically, we de-

[1] An earlier version of this paper appeared in Mühlenbein and Chakraborty (1997).

velop a Markov chain model for computing the probability of convergence to the optimum, and demonstrate how the analysis can be used to arrive at an estimate of the critical population size necessary for convergence.

The remainder of this paper is organised as follows. Section 2 provides a brief introduction to genetic algorithms. Section 3 presents a Markov model and the simulation results. In that section we use the Markov model to estimate the *critical population size* and to analyze genetic drift in gene pool recombination. The minimum population size needed for optimality is also estimated in that section. Recurrence relations describing the growth of the advantageous allele in the population are established in Section 4. Section 3 deals with finite population sizes while Section 4 considers the infinite-population case. We present the conclusions in Section 5.

2. The genetic algorithm

The outline of the simple, "canonical" genetic algorithm is given below:

```
$t = 0$;
initialize $population(t)$;
evaluate structures in $population(t)$;
while predetermined termination condition not satisfied
{
    $t = t + 1$;
    select $population(t)$ from
        $population(t-1)$;
    apply recombination and mutation to structures in $population(t)$;
    evaluate structures in $population(t)$;
}
```

Genetic algorithms start with a population of randomly (or heuristically) generated candidate points in the search space. Each candidate solution is coded (following some predetermined encoding scheme) to represent some underlying parameter set. (Binary coding is one of the most popular encoding strategies.) The algorithm operates in a number of iterations, in an attempt to improve upon the trial solutions. In each iteration (an iteration is called a "generation" in the GA parlance), several probabilistic operators are applied to the trial solutions with a view to creating (possibly) better solutions. The algorithm terminates when either an optimal / near-optimal solution has been found or a specified number of generations have been completed. It is to be noted that the genetic algorithm is a "weak" method, with no guarantee of finding the optimum solution in a particular run.

The transition from one generation to the next has been depicted in Figure 1 where the three phases — selection, single-point recombination, and mutation — have been shown separately. The example used in Figure 1 shows a population size of six. Note that in this example the selection mechanism has resulted in no copies of AAAAA and FFFFF being present in the mating pool. This example assumes that the random pairing of individuals in the mating pool has led to the three pairs (1,3), (2,5), and (4,6) being crossed.

The following subsections explain the working of the three operators in greater detail.

2.1. Selection

Although there exist a number of selection algorithms in the literature (for an analysis of the effects of various selection schemes, see Chakraborty et al., 1996), the basic principle of selection is the same, and simply stated, this principle is to allocate, in the next generation, more copies to fit individuals and fewer copies to the poor ones. In the present paper we consider proportional selection (also called fitness-proportionate selection) (Goldberg, 1989). In proportional selection an individual is selected with a probability $f_i / \sum_{i=1}^{N} f_i$, where f_i is the fitness of the individual, and N is the population size (the total number of individuals in the population).

2.2. Recombination

The recombination (crossover) operator creates offspring by combining segments (parts) from parents. The idea is that useful portions from good parents can be combined to form a (hopefully) better child. A wide variety of recombination operators have been proposed in the

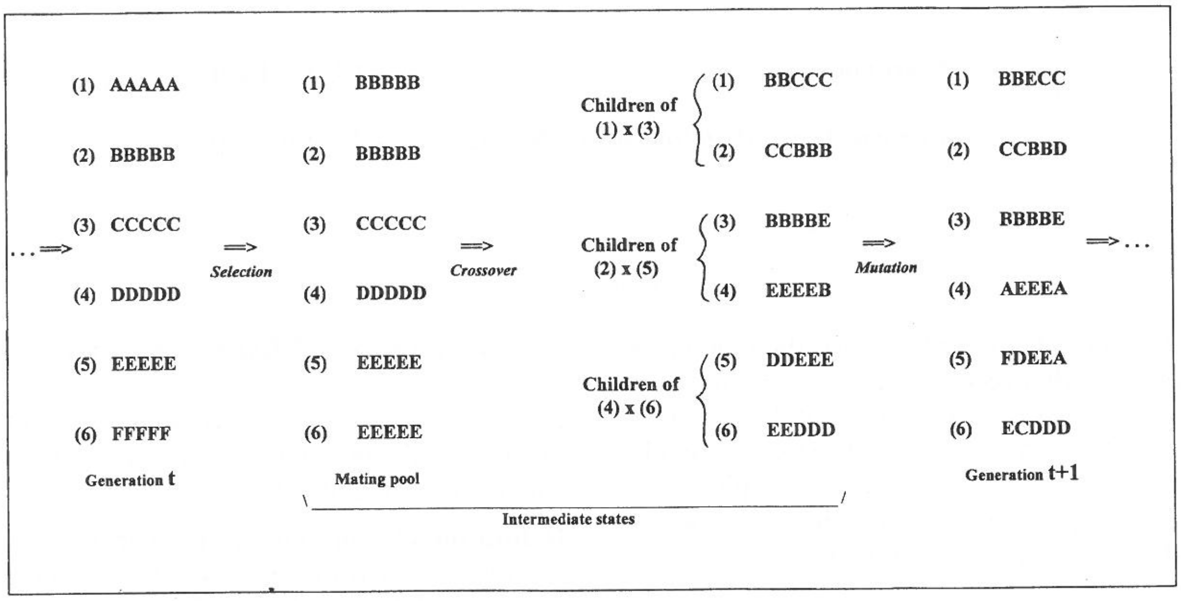

Fig. 1. The simple genetic algorithm. One generation has been broken down into three phases: selection, crossover, and mutuation.

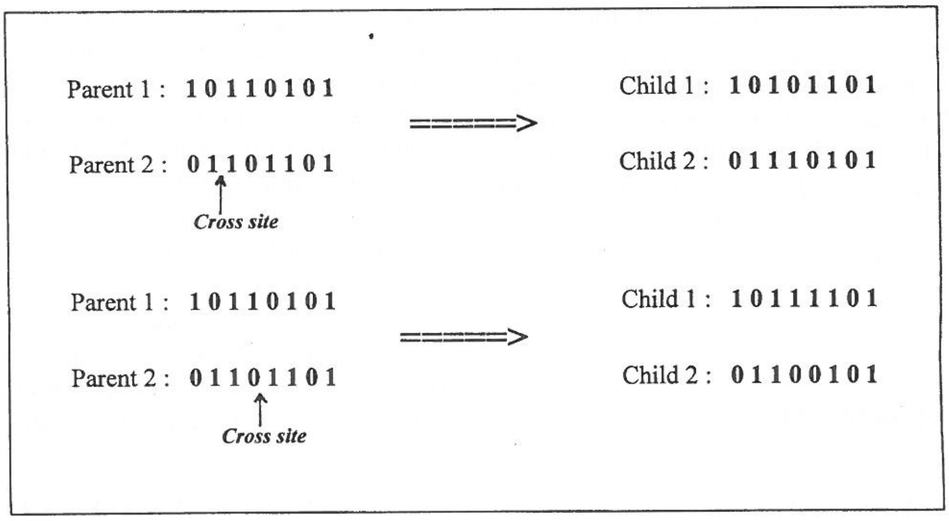

Fig. 2. The single-point crossover operator. Two cases, corresponding two different cut points, have been shown.

literature. One of the most frequently used operators is the single-point crossover. This operator creates two children from two parents by choosing a cut-point uniformly randomly along the length of the individual and then swapping segments between the parents (see Figure 2). Important examples of other types of crossover include two-point (Cavicchio, 1970), multi-point (De Jong, 1975), modified crossover (Davis, 1985), partially matched crossover (PMX) (Goldberg and Lingle, 1985), order crossover (OX) (Oliver et al., 1987), cycle crossover (CX) (Oliver et al., 1987), shuffle crossover (Caruana et al., 1989), uniform crossover (Syswerda, 1989), and order-based, position-based and edge-recombination crossover (Syswerda, 1991). Instead of producing two children from two parents, some GA-practitioners prefer to create only one child from two parents.

In gene pool recombination (Mühlenbein and Voigt, 1996), if p_i is the relative frequency (that is, proportion) of a particular allele at locus i ($i = 1, 2, \ldots, n$, where n is the number of loci)

Before mutation	:	**1 1 0 1 1 1 0 0**
After mutation at the third and the fifth bits :		**1 1 1 1 0 1 0 0**

Fig. 3. The mutuation operator.

in the pool of selected parents, then the probability that after recombination a new individual k will have that same allele at locus i is p_i, $k = 1, 2, \ldots, N$ (N = population size). In other words, gene pool recombination implements a Bernoulli process at each locus, with mean p_i and variance $p_i(1 - p_i)$ at locus i.

2.3. Mutation

The mutation operator changes an allele with a pre-determined low probability. In a binary string, it complements a 1 to a 0 or vice versa. This operator is applied to all bits in the string. Figure 3 illustrates this operator. Mutation is necessary to (re)introduce diversity in the population, that is, it can prevent the genetic algorithm from getting stuck at a local optimum. (In the present paper we do not consider mutation.)

3. Finite population size

The Wright–Fisher model (Fisher, 1930; Wright, 1931; Ewens, 1979; Naglyaki, 1992) in mathematical genetics is one of the most fundamental models for studying the behavior of gene frequencies in finite populations. That model considers the diallelic, single-locus case. In this section we develop a Markov chain model for studying gene frequencies in a diallelic, multi-locus genetic algorithm with proportionate selection and gene pool recombination. We also show how the Markov model can be used for analyzing drift in small populations. We consider genetic algorithms with discrete, non-overlapping generations where the offspring population replaces the parent population at the end of each generation.

3.1. Multiple loci: A Markov chain model

In this section we set up a Markov chain model to compute the probability that convergence occurs by a given generation.

Definition 1: Throughout this paper, by *convergence* we will mean the situation where the entire population is filled with copies of the optimal solution.

Mühlenbein and Schlierkamp–Voosen (1993, 1994), Thierens and Goldberg (1994), and Miller and Goldberg (1996) have used this definition of convergence. For the onemax function, the optimal solution is the all-1 string, and thus convergence represents the case when the population contains no zeros. (The reader should note that for a selection-recombination genetic algorithm there is no guarantee that the optimal solution would be obtained in *every* run of the GA.) Convergence as defined here is not to be confused with the situation where the population is filled with copies (clones) of some string that is not the optimum (this situation has been referred to as "premature convergence" in the literature).

Our use of this interpretation of convergence should not be construed to mean that computational efforts in a GA must necessarily be directed to achieve convergence. In real problems it is sufficient if the GA is able to produce a single copy of the optimal solution.

For notational convenience we consider three loci (but this method is perfectly general and is applicable to any problem size). Let m0, m1, ..., m7 represent, respectively, the fitness values of the $2^3 = 8$ genotypes 000, 001, ..., 111. We consider a linear bit-counting function for which the fitness of a string is obtained by multiplying the number of 1s in the string by s, where s is the "selection coefficient". Then m0 = 0, m1 = s, m2 = s, and so on. Clearly, for $s = 1$, this function reduces to the familiar ONEMAX problem.

As pointed out by Miller and Goldberg (1996), the onemax problem is very popular in the GA-literature because of the following factors: (1) The problem is non-epistatic, (2) The alleles are uniformly scaled, (3) The proportion of the advantageous allele in the initial population can be easily determined, (4) The mean and variance of the population fitness can be expressed as functions of the proportion of the advantageous allele, (5) The crossover operator neither creates nor destroys building blocks.

The results of this paper are valid only for the onemax function. It should be noted that the onemax problem is *not* representative of the complex fitness landscapes of hard, real-world optimization problems. As observed by Mühlenbein and Schlierkamp–Voosen (1994), the onemax domain in GAs serves to play the role of the "ideal gas" in thermodynamics: even though no ideal gas exists in reality, the ideal gas theory provides valuable insights into the overall behavior of gases.

Let us index the genotypes with $i = 0, 1, \ldots, 7$. Let p_i denote the frequency of 1s at the ith locus. Then the present model allows us to study the progress of the genetic algorithm as it moves from a gene frequency vector $(p_1(t), p_2(t), p_3(t))$ at generation t to another vector $(p_1(t+1), p_2(t+1), p_3(t+1))$ at generation $t+1$. At convergence, $p_1 = p_2 = p_3 = 1$.

From the principle of fitness-proportionate selection, we know that in a single trial, the probability with which a genotype i is selected into the mating pool is given by

$$prob_{i,select} = \frac{m_i x_i(t)}{\sum f(t)} \qquad (1)$$

where m_i is the fitness of genotype i, $x_i(t)$ denotes the number of copies of genotype i, and $\sum f$ is the sum of the fitnesses of all individuals in the population at generation t.

It follows that the probability that a single trial picks a 1 at the leftmost locus (locus 1) is given by

$$p1_{select} = \sum_{k=4}^{7} prob_{k,select} \qquad (2)$$

For a binomially distributed population, we have:

$$\begin{aligned}
x_0(t) &= (1-p1(t))(1-p2(t))(1-p3(t))N \\
x_1(t) &= (1-p1(t))(1-p2(t))p3(t)N \\
x_2(t) &= (1-p1(t))p2(t)(1-p3(t))N \\
x_3(t) &= (1-p1(t))p2(t)p3(t)N \\
x_4(t) &= p1(t)(1-p2(t))(1-p3(t))N \\
x_5(t) &= p1(t)(1-p2(t))p3(t)N \\
x_6(t) &= p1(t)p2(t)(1-p3(t))N \\
x_7(t) &= p1(t)p2(t)p3(t)N
\end{aligned}$$

where N is the population size and $pk(t)$ is the gene frequency at the kth locus at generation t.

Noting that the population average fitness at generation t is given by $s(p1(t)+p2(t)+p3(t))$, we have from the above equations:

$$\begin{aligned}
p1_{select} = &\frac{p1(t)}{s(p1(t)+p2(t)+p3(t))}(m4(1-p2(t))\times \\
&\times(1-p3(t))+m5(1-p2(t))p3(t) \\
&+m6p2(t)(1-p3(t))+m7p2(t)p3(t)) \qquad (3)\\
p2_{select} = &\frac{p2(t)}{s(p1(t)+p2(t)+p3(t))}(m2(1-p1(t))\times \\
&\times(1-p3(t))+m3(1-p1(t))p3(t) \\
&+m6p1(t)(1-p3(t))+m7p1(t)p3(t)) \qquad (4)\\
p3_{select} = &\frac{p3(t)}{s(p1(t)+p2(t)+p3(t))}(m1(1-p1(t))\times \\
&\times(1-p2(t))+m3(1-p1(t))p2(t) \\
&+m5p1(t)(1-p2(t))+m7p1(t)p2(t)) \qquad (5)
\end{aligned}$$

The probability that there will be exactly j $(0 \leq j \leq N)$ 1s at the kth locus after selection is given by

$$p_{k,j} = \binom{N}{j} pk_{select}^{j}(1 - pk_{select})^{N-j}$$

where $pk_{select}, k = 1, 2, 3$, are given by equations 3, 4 and 5.

Thus the probability that *after selection* there will be i 1s at the first locus and j 1s at the second locus and k 1s at the third locus is

$$p_{1,i}p_{2,j}p_{3,k}. \qquad (6)$$

The gene frequencies at the next generation are obtained by noting that the gene pool recombination mechanism implements three independent sequences of Bernoulli trials at the three

Population Size	Ultimate Probability of Convergence
4	0.5332
6	0.6407
8	0.7600
10	0.8460

Table 1. Ultimate convergence probabilities for 3 loci (initial frequency = 0.5 for all loci)

loci in the selected parent pool. Thus, given i, j, k 1s at the three loci before recombination, the probability that there will be i' 1s at the first locus, j' at the second and k' at the third *after recombination* is given by

$$\binom{N}{i'}\left(\frac{i}{N}\right)^{i'}\left(1-\frac{i}{N}\right)^{N-i'}\times$$
$$\binom{N}{j'}\left(\frac{j}{N}\right)^{j'}\left(1-\frac{j}{N}\right)^{N-j'}\times$$
$$\binom{N}{k'}\left(\frac{k}{N}\right)^{k'}\left(1-\frac{k}{N}\right)^{N-k'}. \qquad (7)$$

The transition probability matrix of our Markov chain is thus given by the product of two matrices — the selection matrix (given by equation 6) and the recombination matrix (given by equation 7). It is interesting to observe from equations 3–7 that because of the nature of the bit-counting function, the transition probability values are independent of s.

Definition 2: The *ultimate convergence probability* is defined as the convergence probability as the number of generations tends to infinity.

In Table 1 and Table 2 we present some representative results of our Markov chain calculations. Most figures in this paper have been rounded to four decimal places. In Tables 1 and 2, the last column represents the saturation value of the convergence probability — once this value is reached, the convergence probability does not increase any further, regardless of the number of generations.

Definition 3: We define the *critical population size* as the minimum population size needed to obtain convergence with a specified probability.

If, in the definition of the critical population size, we choose the acceptable value of the probability of convergence to be 80% (say), then Table 1 shows that for three loci, the ONEMAX function requires a critical population size of 10.

It is sometimes necessary to be able to analyze bit-counting functions where the fitness values are defined as follows:

$$fitness=\begin{cases}1, & \text{for the all-0 string}\\ 1+s\times\text{number of} & \\ \quad\text{1s in the string,} & \text{otherwise.}\end{cases}$$

For such a fitness function, the average fitness of the population at generation t is $1+s(p1(t)+p2(t)+p3(t))$, and equations 3, 4 and 5 are changed to

$$p1_{select}=\frac{p1(t)}{1+s(p1(t)+p2(t)+p3(t))}(m4(1-p2(t))\times$$
$$\times(1-p3(t))+m5(1-p2(t))p3(t)$$
$$+m6p2(t)(1-p3(t))+m7p2(t)p3(t))$$

Popsize	Initial Frequency at each Locus	Ultimate Prob. of Convergence
4	0.5000	0.8042
5	0.4000	0.8286
6	0.5000	0.9094
7	0.2857	0.8917
8	0.5000	0.9608
10	0.5000	0.9837
10	0.2000	0.9538

Table 2. Ultimate convergence probabilities for 2 loci

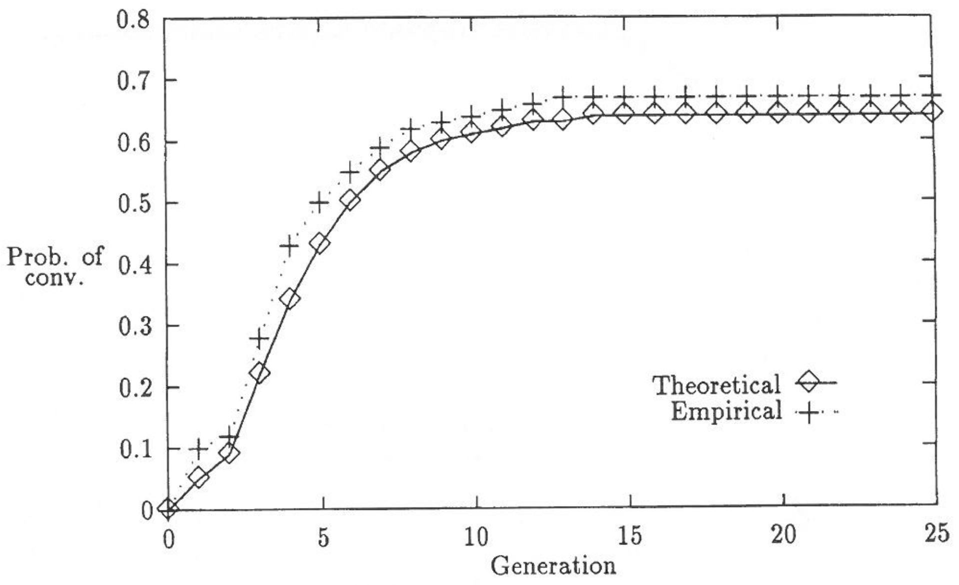

Fig. 4. Comparison of theoretical and experimental convergence probabilities (population size = 6, number of loci = 3, initial frequency of 1s at each locus = 0.5)

$$p2_{select} = \frac{p2(t)}{1+s(p1(t)+p2(t)+p3(t))}(m2(1-p1(t))\times \times(1-p3(t))+m3(1-p1(t))p3(t) +m6p1(t)(1-p3(t))+m7p1(t)p3(t))$$

$$p3_{select} = \frac{p3(t)}{1+s(p1(t)+p2(t)+p3(t))}(m1(1-p1(t))\times \times(1-p2(t))+m3(1-p1(t))p2(t) +m5p1(t)(1-p2(t))+m7p1(t)p2(t))$$

with $m0 = 1, m1 = 1+s, m2 = 1+s, m3 = 1+2s, m4 = 1+s, m5 = 1+2s, m6 = 1+2s$ and $m7 = 1+3s$.

3.2. Comparison with simulation

To obtain empirical support for our model, we conducted experiments by running the genetic algorithm (with proportionate selection and gene pool recombination) on the ONEMAX problem. For each experiment, a set of values for (a) population size, (b) number of loci, and (c) the initial gene frequency was chosen, and using these values, 500 independent runs (with as many different seeds for the pseudorandom number generator) were taken. All of the 500 runs in any single experiment were started with the same initial population, the initial population having been generated with the specified initial probability of 1s. The generation number at which convergence occurred in each of these 500 runs was noted, and the convergence probability at a given generation was computed from the relation

$$prob_{convergence}(t) = \frac{s(t)}{500},$$

where $s(t)$ denotes the total number of runs (out of 500) in which convergence occurred at or before generation t.

The experimentally obtained convergence probabilities agreed with the values computed by the Markov model. In Figures 4 and 5 we compare two representative theoretical-experimental pairs of plots.

In the following subsection we show how the Markov chain model can be used for analyzing drift (Crow and Kimura, 1970) in small-population genetic algorithms that use gene pool recombination.

3.3. Multiple loci: Drift in gene pool recombination

Drift at a single locus was analyzed by Goldberg and Segrest (1987). Drift in a multi-locus genetic algorithm has been studied in Asoh and Mühlenbein (1994), but the uniqueness of the present approach is that unlike that paper, it does not treat the multi-locus case as a simple generalization of the single-locus analysis, and more

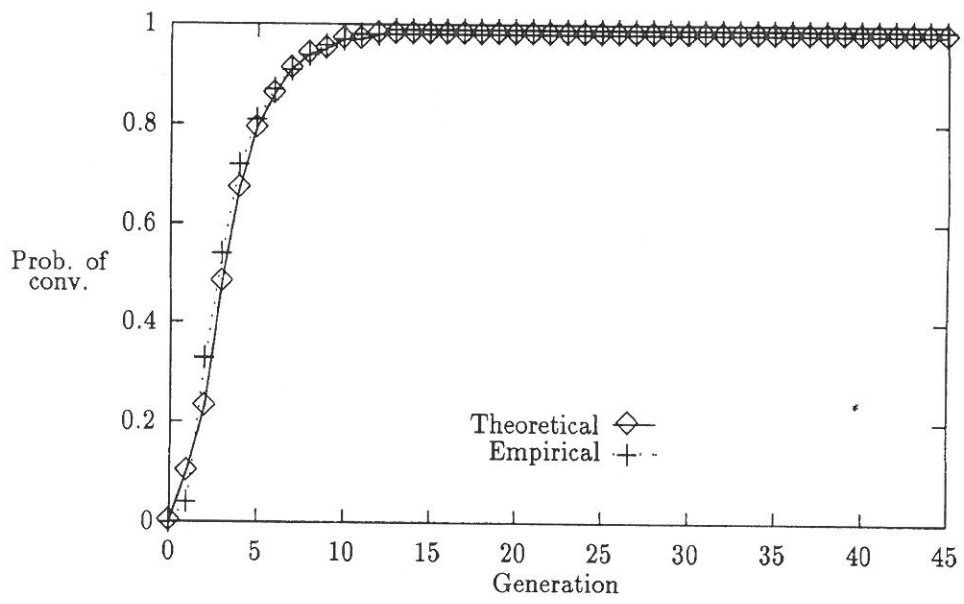

Fig. 5. Theoretical and experimental convergence probabilities (population size = 10, number of loci = 2, initial frequency of 1s at each locus = 0.5)

importantly, it allows us to treat drift as a special case of selection (where the selection pressure is zero).

Gene pool recombination is a novel recombination scheme where the offspring allele at a particular locus is obtained by uniformly randomly choosing one allele from all alleles at that locus in the selected pool. This recombination method ensures that the population always remains binomially distributed. In this recombination algorithm, the process of generation of the offspring population from the selected parent population can be thought to be a collection of n independent sampling processes with replacement, where n denotes the function size (= number of loci).

The Markov chain model developed in Section 3.1 can be used for analyzing drift by modifying only the selection matrix (the recombination matrix remains unaltered). We begin by noting that in the case of random drift there is no systematic selection pressure in favor of any genotype and equation 1 becomes

$$prob_{i,select} = \frac{x_i(t)}{N} \tag{8}$$

Equation 2 holds for drift also. Using equations 8 and 2, and substituting gene-frequency expressions for genotype frequencies, we get

$$\begin{aligned} p1_{select} &= p1(t) \\ p2_{select} &= p2(t) \\ p3_{select} &= p3(t). \end{aligned}$$

It follows that the probability that after uniform random selection there will be v 1s at the uth locus (for $u = 1, 2, \cdots, n$) is given by $p_{u,v}$ where $p_{u,v}$ depends only on the gene frequency at the uth locus, p_u:

$$p_{u,v} = \binom{N}{v} p_u^v (1 - p_u)^{N-v}.$$

Thus the probability that after uniform random (i.e., unbiased) selection there will be v_1 1s at the first locus,$\cdots$, v_n 1s at the nth locus is given by

$$p_{1,v_1} \cdot p_{2,v_2} \cdots p_{n,v_n}. \tag{9}$$

As before, the transition probability matrix for drift is obtained by multiplying the selection matrix (given by equation 9) and the recombination matrix (given by equation 7).

The probability of fixation (to any absorbing state) by generation t is obtained from the t-step transition probability matrix by summing all the probabilities corresponding to the absorbing states. As an example, for a two-loci case, the probability of fixation by generation t

n	N=2	N=4	N=8	N=16
2	2.6667	6.3153	13.7165	28.7069
4	3.5048	8.3474	18.1047	37.7954

Table 3. Expected fixation time with deterministic selection and gene pool recombination (initial frequency = 0.5 at each locus)

n	N=2	N=4	N=8	N=16
2	1.6000	3.4041	7.1084	14.6035
4	1.9929	4.4241	9.3023	19.1477

Table 4. Expected fixation time with uniform random selection and gene pool recombination (initial frequency = 0.5 at each locus)

is given by

$$
\begin{aligned}
&p_{fixation}(t) = \\
&prob_t((p_{1,0}, p_{2,0}) \to (0,0)) + prob_t((p_{1,0}, p_{2,0}) \\
&\quad \to (0,1)) + prob_t((p_{1,0}, p_{2,0}) \to (1,0)) \\
&\qquad + prob_t((p_{1,0}, p_{2,0}) \to (1,1))
\end{aligned}
$$

where $p_{k,0}$ denotes the gene-frequency at the kth locus at generation 0 and the right arrow ($\to$) indicates a transition from the left hand side gene frequency vector to the right hand side vector.

It is easy to see that $p_{fixation}(t)$ defines a cumulative distribution function. The mean (expected) fixation time is then given by

$$\sum_{k=0}^{\infty} kq(k)$$

where

$$q(k) = \begin{cases} 0, & k = 0, \\ p_{fixation}(k) - p_{fixation}(k-1), & k \geq 1. \end{cases}$$

Tables 3 and 4 show the mean fixation times for some representative cases. The values in Table 3 have been obtained for the case when the whole of the population at a particular generation is deterministically chosen to be parents. In other words, in this case all members at a generation go into the selected parent pool. This is how drift was treated by Mühlenbein and Voigt (1996) whose results agree with the data in Table 3. Interestingly, the fixation times in Table 3 agree with the results of Asoh and Mühlenbein (1994), too. In general, some sort of selection is applied to create the parent pool, and gene pool recombination is then applied on the pool of selected parents. In Table 4 we show the fixation times after taking into account uniform random selection and gene pool recombination. It is interesting to note from the data in Tables 3 and 4 that uniform random selection followed by gene pool recombination is about two times faster than deterministic selection and gene pool recombination.

3.4. Multiple loci: Minimum population size for optimality

In this section we continue our analysis of gene pool recombination on the simple ONEMAX function. We show by simple probabilistic analysis that it is possible to arrive at a lower bound for the population size needed to find the optimum.

We consider the ONEMAX problem and a selection-recombination genetic algorithm. No mutation is applied. To compute the minimum population size needed in this case, we proceed by noting the fact that in the absence of mutation, the optimal string (all 1s) can never be achieved if, in the initial population, all the bits at a locus are zeroes. Thus, for the optimal individual to be ever achieved, every locus should contain at least one 1. Now because of stochastic effects in small populations, having just a single occurrence of the advantageous allele at a locus may not be sufficient to guarantee the presence of 1 at that locus over a number of generations. It is therefore reasonable to require the presence of a certain number, say k (≥ 1), of 1s at each locus in the initial generation. It follows, then, that the probability that convergence to the optimum will not occur is given by the probability that the initial population contains at least one

No. of Loci	k	p	Required Minimum Popsize
32	8	0.5	53
32	8	0.25	121
32	8	0.125	256
64	16	0.5	79
64	16	0.25	178
64	16	0.125	373
100	1	0.5	27
100	3	0.5	36
100	5	0.5	44
100	10	0.5	61
128	32	0.5	125
128	32	0.25	277
128	32	0.125	577
200	1	0.5	28
200	2	0.5	33
256	64	0.5	210
256	64	0.25	455
256	64	0.125	942

Table 5. Minimum population size for specified n and k values

locus with less that k 1s. We compute this probability next.

Let the initial population be generated with probability p for the advantageous allele (1). We then have the same p for each locus. Now the probability that a particular locus (column) in the initial population has i 1s (we are not concerned at this stage with exactly which positions are 1s; the knowledge of the total number of 1s is sufficient for the present purpose) is given by

$$\binom{N}{i} p^i(1-p)^{N-i}.$$

Therefore the probability that a locus has 0 or 1 or 2 or ... or $k-1$ 1s is

$$\sum_{j=0}^{k-1} \binom{N}{j} p^j(1-p)^{N-j} = p_A \text{ (say)}.$$

We note that out of a total of n loci, one or two or ... or n loci may have less than k 1s. Thus the probability that *at least* one locus has less than k 1s is given by

$$\sum_{r=1}^{n} \binom{n}{r} p_A^r(1-p_A)^{n-r}$$
$$= \sum_{r=0}^{n} \binom{n}{r} p_A^r(1-p_A)^{n-r} - (1-p_A)^n$$
$$= 1-(1-p_A)^n.$$

The minimum population size can be found by setting

$$1-(1-p_A)^n \leq \epsilon, \qquad (10)$$

where ϵ is a predetermined small quantity, and solving for N. Unfortunately, it is difficult to solve the above inequality analytically. We wrote a program that uses equation (10) to find out the minimum population size by iteration. Table 5 presents the results for some sample cases. For all entries in this table, $\epsilon = 0.0000001$.

3.5. Single locus: Markov chain analysis

In this section we consider a single-locus genetic algorithm where the fitness of a 0 is 1 and that of a 1 is $1+s$, s being a predetermined selection coefficient. We will apply the Markov chain approach to obtain the critical population size, the ultimate probability of convergence, and the mean time to fixation (when both absorbing states — the "all 0" state and the "all 1" state — are considered).

Let N denote the population size and let there be i 1s at any generation. Then in proportionate selection, the probability that in a single trial a 1 would be picked is given by

$$p_{select} = \frac{i(1+s)}{i(1+s)+N-i}.$$

The probability that given i 1s in the present generation, the next generation would have j 1s is

$$p_{ij} = \binom{N}{j} p_{select}^{j}(1-p_{select})^{N-j}.$$

The above equation defines the transition probabilities of the Markov chain, and it is now straightforward to compute the ultimate probability of convergence and the mean fixation time. In Tables 6 - 8 we present some numerical results.

4. Infinite population size: Proportion of the advantageous allele

4.1. An analytical model

We consider a linear bit-counting function for which the fitness of a string is given by the number of 1s in the string times s, where s is the selection coefficient. Let us assume that the proportion of 1s is the same for all loci, and let this proportion at generation t be denoted by $p(t)$. Then the proportion of 1s in the population at generation t is also $p(t)$. For a sufficiently large population size, we have the following theorem (for proportionate selection):

THEOREM 1: Let n be the number of loci. Then

$$p(t+1) = p(t) + \frac{1-p(t)}{n}. \qquad (11)$$

Proof: It is easy to see that the average fitness of the population, $f_{avg}(t)$, is given by $nsp(t)$. If we concentrate on a particular locus, we see that there are 2^{n-1} genotypes with a 1 at that particular locus, and out of this total of 2^{n-1} genotypes, $\binom{n-1}{k}$ have exactly k 1s ($k = 0, 1, \ldots, n-1$). Proceeding as in Section 3, we have for proportionate selection

$$\begin{aligned} p(t+1) &= \frac{1}{nsp(t)}p(t)\sum_{k=0}^{n-1}\binom{n-1}{k}(k+1)\times \\ &\quad \times sp^k(1-p)^{n-1-k} \\ &= \frac{1}{n}(\sum_{k=0}^{n-1}\binom{n-1}{k}p^k(1-p)^{n-1-k}k+1) \\ &= \frac{1}{n}((n-1)p+1) \end{aligned}$$

Q.E.D.

It is interesting to note that equation 11 is independent of s. Equation 11 is a linear difference equation and can be solved analytically. The solution is given by

$$p(t) = 1 - \left(1-\frac{1}{n}\right)^t (1-p(0)). \qquad (12)$$

Next we consider the case when the fitness of a string with k 1s is $1+ks$. Letting n denote the number of loci, we have the following theorem:

Popsize	s	Mean Fix. Time	Ultimate Prob. of Conv.
16	0.5	7.5374	0.9987
16	0.25	11.8694	0.9734
16	0.125	16.6570	0.8687
16	0.0625	19.4179	0.7253
16	0.03125	20.3889	0.6207
32	10.0	2.1920	1.0000
32	1.0	5.8155	1.0000
32	0.5	8.9711	1.0000
32	0.25	14.4805	0.9993
32	0.125	23.6204	0.9778
32	0.0625	33.9414	0.8747
32	0.03125	39.8874	0.7281

Table 6. Ultimate convergence probabilities and mean fixation times for specified values of the selection coefficient (initial frequency of $1 = 0.5$)

s	Critical Population Size
1.0	4
0.5	6
0.25	10
0.125	20
0.0625	38
0.03125	72

Table 7. Critical population size corresponding to ultimate probability of convergence into the all-1 state = 0.9 (initial frequency of 1s = 0.5)

s	Critical Population Size
1.0	6
0.5	8
0.25	14
0.125	26
0.0625	50
0.03125	96

Table 8. Critical population size corresponding to ultimate probability of convergence into the all-1 state = 0.95 (initial frequency of 1s = 0.5)

THEOREM 2:

$$p(t+1) = \frac{p(t)(1+s+snp(t)-sp(t))}{1+nsp(t)}. \quad (13)$$

Proof: The proof is similar to that of Theorem 1. Noting that the average fitness of the population at generation t is $1+nsp(t)$, we get

$$\begin{aligned} p(t+1) &= \frac{1}{1+nsp(t)} p(t) \sum_{k=0}^{n-1} \binom{n-1}{k} (1 \\ &\quad +(k+1)s)p^k(1-p)^{n-1-k} \\ &= \frac{p}{1+nsp}\Big(\sum_{k=0}^{n-1}\binom{n-1}{k}p^k \times \\ &\quad \times (1-p)^{n-1-k}(k+1)s+1\Big) \\ &= \frac{p}{1+nsp}\Big(1+s\Big(\sum_{k=0}^{n-1}\binom{n-1}{k}\times \\ &\quad \times p^k(1-p)^{n-1-k}k+1\Big)\Big) \\ &= \frac{p}{1+nsp}(1+s((n-1)p+1)) \end{aligned}$$

Q.E.D.

Unfortunately, equation 13 is a non-linear difference equation and is extremely difficult to solve analytically (other than by iteration).

4.2. Empirical results

In Figure 6 we have compared the theoretical model (equation 12) with the experimental results. Figure 7 compares the theoretical $p(t)$ values (obtained from equation 13 by iteration) and simulation results. A single-locus case for equation 13 has been shown in Figure 8. Each of the simulation curves in Figures 6-8 represents the average of 100 independent runs. It is interesting to note that for small population sizes, the $p(t)$ curve levels off to a less-than-unity value. The envelope (i.e., the curve corresponding to equation 12 or 13) is not achieved until the population size is sufficiently large. These findings are similar in spirit to the observations made in Mühlenbein and Schlierkamp–Voosen (1994, Section 3). We capture the insight obtained from these experimental results in the following empirical law:

Empirical law: During the early generations, the rate of growth of the proportion of the advantageous allele is more or less independent of the population size. The population size does, however, determine the quality of the ultimate solution. For population sizes greater than a certain threshold, the speed with which convergence occurs does not depend on the popu-

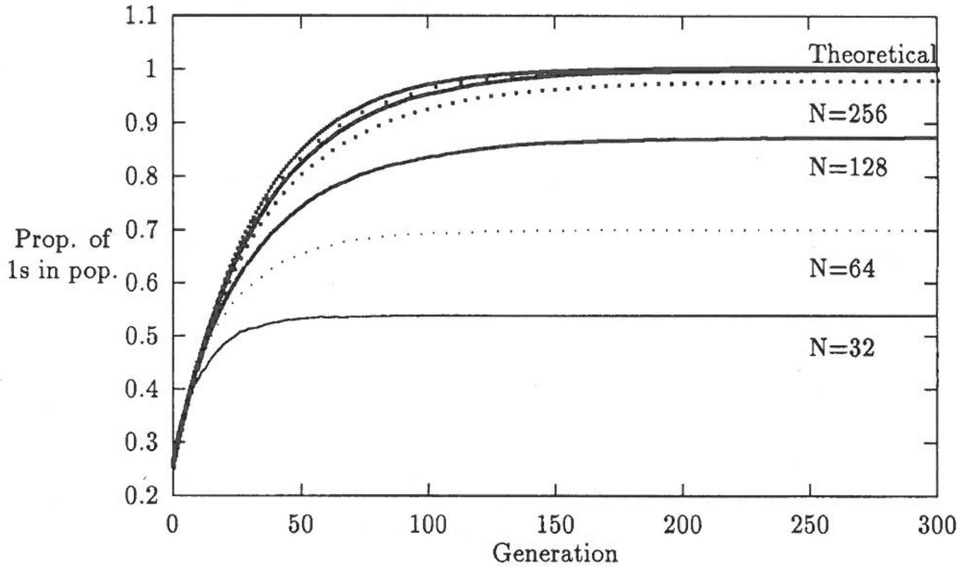

Fig. 6. Proportion of the advantageous allele in the population under proportionate selection and gene pool recombination (number of loci = 32, initial frequecy at each locus = 0.25)

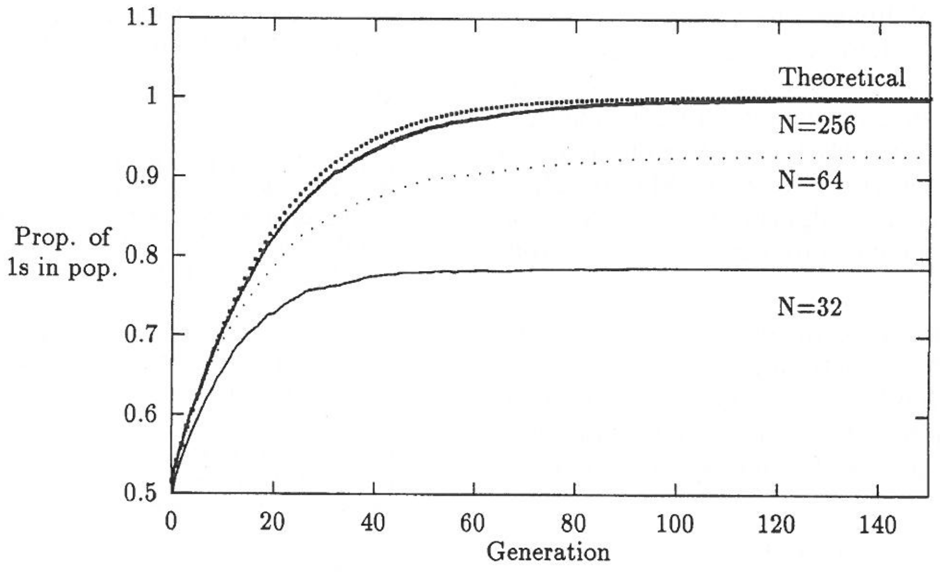

Fig. 7. Proportion of the advantageous allele in the population under proportionate selection and gene pool recombination (number of loci = 16, initial frequecy at each locus = 0.5, $s = 0.5$)

lation size. If the population size is less than the threshold value, convergence will never be achieved.

5. Conclusion

This paper provides a mathematical description of genetic algorithm behavior. A Markov model has been developed, and the model has been used to obtain several quantities of fundamental interest in genetic algorithm theory. By employing a mix of theoretical analysis and experiments, we have been able to study the genetic algorithm behavior. The ultimate probability of convergence, mean fixation time and proportion of the advantageous allele have been obtained, under certain simplifying assumptions. The results of this paper sharpen our insight into how the genetic algorithm really works.

In the present Markov chain approach, we have to deal with $(N+1)^n$ distinct states, where N and n represent population size and number of loci, respectively, and the transition probability

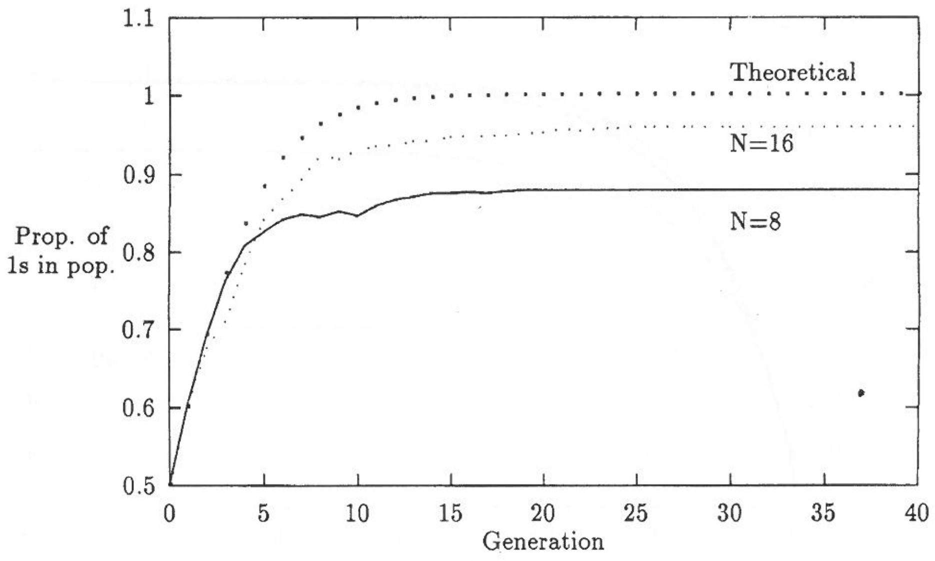

Fig. 8. Proportion of the advantageous allele in the population under proportionate selection and gene pool recombination (number of loci = 1, initial frequecy at each locus = 0.5, $s = 0.5$)

matrices are of dimension $(N+1)^n \times (N+1)^n$. Thus even though the approach is perfectly general for any N and n, for large N and n values the computation time for manipulating the matrices becomes prohibitively large. We note here that most of the previous work on Markov chain analysis of genetic algorithms (e.g., Nix and Vose, 1992) seem to suffer from the same problem.

We believe that a useful approach to unravel the mystery of the genetic algorithm operation is to study the fundamental issues raised in mathematical population genetics, and to interpret their significance with reference to genetic algorithms.

Acknowledgment Thanks to the anonymous referees for their detailed comments. This work was supported in part by the SIFOGA project under the Real-World Computing Partnership. The contribution of the second author was made while he was working at GMD as a guest researcher.

References

[1] D. H. ACKLEY (1987) *A Connectionist Machine for Genetic Hillclimbing*, Kluwer, Boston.

[2] H. ASOH AND H. MÜHLENBEIN (1994) On the mean convergence time of evolutionary algorithms without selection and mutation, in Y. Davidor, H.-P. Schwefel and R. Männer (editors), Parallel Problem Solving from Nature — 3, pp. 88–97, Springer-Verlag, Berlin.

[3] R. BELEW AND L. BOOKER (editors) (1991) Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA.

[4] R. A. CARUANA, L. J. ESHELMAN AND J. D. SCHAFFER (1989) Representation and hidden bias II: eliminating defining length bias in genetic search via shuffle crossover, Proceedings of International Joint Conference on Artificial Intelligence, pp. 750–755.

[5] D. J. CAVICCHIO (1970) Adaptive Search Using Simulated Evolution, Doctoral dissertation, University of Michigan, Ann Arbor.

[6] U. K. CHAKRABORTY AND D. G. DASTIDAR (1993) Using reliability analysis to estimate the number of generations to convergence in genetic algorithms, *Information Processing Letters*, Vol. **46**, No. 4, pp. 199–209.

[7] U. K. CHAKRABORTY (1995a) A simpler derivation of schema hazard in genetic algorithms, *Information Processing Letters*, Vol. **56**, No. 2, pp. 77–78.

[8] U. K. CHAKRABORTY (1995b) A branching process model for genetic algorithms, *Information Processing Letters*, Vol. **56**, No. 5, pp. 281–292.

[9] U. K. CHAKRABORTY, K. DEB, AND M. CHAKRABORTY (1996) Analysis of selection algorithms: A Markov chain approach, *Evolutionary Computation*, Vol. **4**, No. 2, pp. 133–167.

[10] U. K. CHAKRABORTY AND H. MÜHLENBEIN (1997) Linkage equilibrium and genetic algorithms, Proc. 4th IEEE International Conference on Evolutionary Computation, Indianapolis, USA, April 1997, pp. 25–29.

[11] J. F. CROW AND M. KIMURA (1970) *An Introduction to Population Genetics Theory*, New York: Harper and Row.

[12] L. DAVIS (1985) Applying adaptive algorithms to epistatic domains, Proceedings of International Joint Conference on Artificial Intelligence, pp. 162–164.

[13] K. A. DE JONG (1975) An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph D Thesis, University of Michigan, Ann Arbor.

[14] L. J. ESHELMAN (editor) (1995) Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA.

[15] W. J. EWENS (1979) *Mathematical Population Genetics*, Springer–Verlag, Berlin.

[16] R. A. FISHER (1930) *The Genetical Theory of Natural Selection*, Clarendon Press, Oxford.

[17] S. FORREST (editor) (1993) Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA.

[18] D. E. GOLDBERG AND R. LINGLE (1985) Alleles, loci, and the traveling salesman problem, in J. J. Grefenstette (editor), Proceedings of an International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, pp. 154–159.

[19] D. E. GOLDBERG AND P. SEGREST (1987) Finite Markov chain analysis of genetic algorithms, in J. J. Grefenstette (editor), Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, pp. 1–8.

[20] D. E. GOLDBERG (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, Reading, MA.

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

[22] B. L. MILLER AND D. E. GOLDBERG (1996) Genetic algorithms, selection schemes, and the varying effects of noise, *Evolutionary Computation* 4(2), pp. 113–131.

[23] H. MÜHLENBEIN (1991) Evolution in time and space — the parallel genetic algorithm, in G. J. E. Rawlins (editor), Foundations of Genetic Algorithms (pp. 316–337), Morgan Kaufmann, San Mateo, CA.

[24] H. MÜHLENBEIN (1992) How genetic algorithms really work: mutation and hill-climbing, in R. Männer and B. Manderick (editors), Parallel Problem Solving from Nature (pp. 15–26), North–Holland, Amsterdam.

[25] H. MÜHLENBEIN AND U. K. CHAKRABORTY (1997) Analysis of the selection-recombination genetic algorithm for the bit-counting function, Proceedings of the 2nd International Conference on Soft Computing (SOCO–97), Nimes, France, September 1997.

[26] H. MÜHLENBEIN AND D. SCHLIERKAMP–VOOSEN (1993) Predictive models for the breeder genetic algorithm, *Evolutionary Computation* 1 (1), pp. 25–49.

[27] H. MÜHLENBEIN AND D. SCHLIERKAMP–VOOSEN (1994) The science of breeding and its application to the breeder genetic algorithm (BGA), *Evolutionary Computation*, 1 (4), pp. 335–360.

[28] H. MÜHLENBEIN AND H. –M. VOIGT (1996) Gene pool recombination in genetic algorithms, in J. P. Kelly and I. H. Osman (editors), Metaheuristics: Theory and Applications, Norwell (Kluwer Academic).

[29] T. NAGLYAKI (1992) *Introduction to Theoretical Population Genetics*, Springer–Verlag, Berlin.

[30] A. NIX AND M. D. VOSE (1992) Modeling genetic algorithms with Markov chains, *Annals of Mathematics and Artificial Intelligence*, 5, 79–88.

[31] I. M. OLIVER, D. J. SMITH AND J. R. C HOLLAND (1987) A study of permutation crossover operators on the traveling salesman problem, in J. J. Grefenstette (editor), Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, pp. 224–230.

[32] D. SAHA AND U. K. CHAKRABORTY (1997) An efficient link enhancement strategy for computer networks using genetic algorithm, *Computer Communications*, Vol. 20, No. 9, pp. 798–803.

[33] G. SYSWERDA (1989) Uniform crossover in genetic algorithms, in J. D. Schaffer (editor), Proceedings of the Third International Conference on Genetic Algorithms, pp. 2–9, Morgan Kaufmann, San Mateo.

[34] G. SYSWERDA (1991) Schedule optimization using genetic algorithms, in L. Davis (editor), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, pp. 332–349.

[35] D. THIERENS AND D. E. GOLDBERG (1994) Convergence models of genetic algorithm selection schemes, in Y. Davidor et al. (eds.), Parallel Problem Solving from Nature — III, Lecture Notes in Computer Science Vol. 866, Berlin: Springer, pp. 119–129.

[36] L. D. WHITLEY (editor) (1993) Foundations of Genetic Algorithms — 2, Morgan Kaufmann, San Mateo, CA.

[37] L. D. WHITLEY AND M. D. VOSE (editors) (1995) Foundations of Genetic Algorithms — 3, Morgan Kaufmann, San Mateo, CA.

[38] S. WRIGHT (1931) Evolution in Mendelian populations, *Genetics* 16, pp. 97–159.

Received: February, 1997
Accepted: October, 1997

Contact address:
Heinz Mühlenbein
GMD Forschungszentrum Informationstechnik
SET.AS, Schloß Birlinghoven
D-53754 Sankt Augustin
Germany
E-mail: muehlenbein@gmd.de

Uday K. Chakraborty
19 Kalibari Road
Santoshpur
Calcutta 700075
India
E-mail: uday@jadav.ernet.in
Phone: +91 33 413 0382

DR. HEINZ MUEHLENBEIN is currently head of the research group Adaptive Systems (http://set.gmd.de/SET/as_f.html) at the GMD, Sankt Augustin, Germany. He has published research work in the areas of computer networks, parallel processing, evolutionary algorithms, neural networks and robotics. He is the European editor of the journal "Evolutionary Computation" and an editor of "Journal of Heuristics".

UDAY KUMAR CHAKRABORTY is a Reader at the Department of Computer Science & Engineering at Jadavpur University, India, where he has been teaching since 1990. He received his PhD from Jadavpur University for his work on theoretical analysis of genetic algorithms. During 1988–90 he worked as a systems engineer at CMC Limited, India. During 1986–88 he worked as a senior research associate at Computer Aided Design Centre, Calcutta, India.

He has held visiting positions at Carnegie-Mellon University, USA (1987) and at GMD, Germany (1995 and 1996). He has authored (or co-authored) 30 papers and a book. He received a UNIDO Fellowship (1987), a Commonwealth Scholarship (1992), and an AICTE Career Award (1996). He has been on the program committees of many conferences including the 7th International Conference on Genetic Algorithms (Michigan, 1997), the 4th IEEE International Conference on Evolutionary Computation (Indianapolis, 1997), the 5th IEEE International Conference on Evolutionary Computation (World Congress on Computational Intelligence) (Anchorage, 1998), and the Fifth International Conference on Parallel Problem Solving from Nature (Amsterdam, 1998).