Optimal Interaction of Mutation and Crossover in the Breeder Genetic Algorithm

Heinz Mühlenbein Dirk Schlierkamp-Voosen
muehlen@gmd.de dirk@gmd.de

German National Research Center for Computer Science (GMD)
P.O. 1316, D-5205 Sankt Augustin 1, Germany

Abstract

The dynamic behavior of mutation and crossover is investigated with the Breeder Genetic Algorithm. The main emphasis is on binary functions. The genetic operators are compared near their optimal performance. It is shown that mutation is most efficient in small populations. Crossover critically depends on the size of the population. Mutation is the more robust search operator. But the BGA combines the two operators in such a way that the performance is better than that of a single operator. For the DECEPTION function it is shown that increasing the size of the population above a certain number decreases the quality of the solutions obtained.

1 Introduction

In this paper mutation, recombination/crossover and their interaction are analyzed. The goal is to demonstrate why and when the *Breeder Genetic Algorithm BGA* works efficiently. The theory of the BGA and applications in continuous domains have been described in [MSV93]. Discrete fitness functions have been investigated in [Müh93].

The problem of mutation vs. crossover has been discussed many times. We will first show why a comparison is difficult. The performance of the genetic operators depends on many parameters. Changing the parameters will change the result of the comparison. Our approach is unique, because we will compare the genetic operators working with optimal parameters. With these parameters the genetic operator solves the problem with optimal efficiency i.e. the least number of function evaluations. We are intersted in finding the most efficient genetic algorithm.

The theory presented in [MSV93] can be used to predict the behavior of the BGA and to estimate the optimal parameters of the genetic operators. The most important parameters are the size of the population, the selection schedule and the fitness landscape. Selection has been investigated in [MSV93], so we will here concentrate on the size of the population and the fitness landscape.

The relative importance of each operator for an efficient search is still highly disputed. The search strategies of both operators are very different. The mutation operator is depends on a good mutation rate. The crossover operator uses a more global search based on restricted chance. The bias is given implicitly by the population. Crossover shuffles substrings contained in the population. The substrings of the optimum have to be contained in the initial population, otherwise crossover is not able to locate the optimum. Therefore crossover critically depends on the the size of the population.

The outline of the paper is as follows. In section 2 previous investigations on the mutation vs. crossover problem are reviewed. Then the test suite of fitness functions is introduced. In section 4 mutation in large populations is investigated theoretically. Numerical results of a virtual race of populations using mutation, crossover and both are reported in section 5.

2 Mutation vs. Crossover

Understanding mutation and crossover as well as their relative importance has long been of interest both in population genetics and in genetic algorithms. In order to compare the results of the different fields, the method of evaluation has to be looked at.

In population genetics mutation and recombination/crossover are evaluated as follows. Mutation or recombination are modelled as selectively neutral genes. Then the distribution of these genes at equilibrium is measured. The gene which is more often represented is considered to be more succcessful. This measure only makes sense near an equilibrium point of the population. Then the genetic operator which is more successful near equilibrium will win the competition.

The performance measure in genetic algorithms should be different. It should be based on the efficiency of the search. The efficiency is defined as the number of function evaluations needed to obtain a certain quality of the solution. In genetic algorithms we should look for a combination of mutation and crossover which searches the fitness landscape more efficiently than each operator alone.

These two evaluation methods are different. If for instance, mutation performs better than crossover far away from the optimum and crossover is more efficient near the optimum, then crossover will win in population genetics. But a genetic algorithm using mutation more heavily at the beginning of the search and then changing to crossover out-performs any algorithm using only a single operator.

With this clarification in mind we will discuss recent investigations of mutation vs. crossover in genetic algorithms and in population genetics. In [SCED89] Schaffer et al. tried to find an optimal combination of mutation rate and crossover rate for a test suite. They analyzed 6 population sizes, 10 crossover rates and 7 mutation rates. The performance measure was average *online* performance. This investigation is in the spirit of this paper. But the authors did not test whether the optimal parameters were contained in their set. Nevertheless, some of their results can be understood by our investigation.

Recent studies have been made by Schaffer et al. [SE91] and Fogel et al [FA90]. In the new investigation Schaffer et al. used five discrete fitness functions of size 100 and a fixed population size of 512. The mutation rate was set to 0.0005. Compared to their earlier study, they made a much more detailed study of mutation and crossover. But unfortunately the parameter setting is far from being optimal. We have shown elsewhere [Müh91], [Müh93] that a mutation rate proportional to the size of the problem is a good first choice. This means that Schaffer et al. should have used a mutation rate of 0.01 in their experiments. Moreover the size of the population should be much smaller. In biased against mutation. Nevertheless, Schaffer observed that sometimes crossover has a detrimental impact on the search.

Fogel used continuous fitness functions of 10 variables and a population size of 100. He concluded that the genetic operators crossover and inversion do not compare favorably with the more simple random mutation scheme.

In population genetics Feldman has recently investigated the recombination problem. He proved the *reduction principle* under strict assumptions. This principle states that a selectively neutral recombination gene, introduced near an equilibrium of a large randomly mating population will succeed, if it reduces recombination. In a simulation study [BF92] Bergman and Feldman investigated the above problem under less severe assumptions. They conjectured that if the fitness function is very jagged, low recombination has a strong advantage. Furthermore they observed that the evolution of recombination depends on the initial gene frequencies at the loci, the shape and the strength of the selection.

This short survey shows how involved the mutation vs. crossover problem is. The relative strength of the operators compared to the other depends on many circumstances. There will be no simple answer claimimg that one is better than the other.

3 The test functions

The outcome of a comparison of mutation and crossover depends on the fitness landscape. Therefore a carefully chosen set of test functions is necessary. We will use test functions which we have theoretically analyzed in [Müh93]. They are similar to the test functions used by Schaffer [SE91]. The test suite consists of

ONEMAX
PLATEAU(k,l)
SYMBASIN(k,l)
DECEPTION(k,l)

The fitness of ONEMAX is given by the number of 1's in the string. For the PLATEAU function k bits have to be flipped in order that the fitness increases by k. The DECEPTION function has been defined by Goldberg [GDK90]. The fitness of DECEPTION(k,l) is given by the sum of l deceptive functions of size k. A deceptive function and a smoothed version of order $k = 3$ is defined in the following table

bit	DECEP	SYMBA	bit	DECEP	SYMBA
111	30	30	100	14	14
101	0	26	010	22	22
110	0	22	001	26	26
011	0	14	000	28	28

A DECEPTION function has 2^l local maxima. Neighboring maxima are k bits apart. Their fitness value differs by two. The basin of attraction of the global optimum is of size k^l, the basin of attraction of the smallest optimum is of size $(2^k - 1)^l$. The DECEPTION function is called deceptive because the search is mislead to the wrong maxima $(0, 0, \ldots, 0)$. The global optimum is particularly isolated.

The SYMBASIN(k,l) function is like a deceptive function, but the basins of attraction of the two peaks are equal. In the simulations we used the values given in the above table for SYMBA.

We have analyzed the probability of success of the mutation operator for single individuals in [Müh91], [Müh93]. In this section we will analyze mutation in large populations. The analysis is based on the *response to selection* equation of population genetics [Fal81], [Bul80]. This equation was also used in [Müh93] for the analysis of selection and recombination.

$$R(t+1) = b_t * S(t) \tag{1}$$

$S(t)$ is called the selection differential. It is defined as the difference between the mean value of the selected parents and the mean of the population at generation t. The response $R(t+1)$ is the difference between the mean of the population at generation $t+1$ and generation t. b_t is called the *heritability* in quantitative genetics. It is the regression coefficient of the offspring generation to the selected parents. b_t can be estimated from the ratio

$$b_t \approx \frac{prob(off.better)}{prob(off.worse)}$$

$prob(off.better)$ means that the offspring is better than the parent(s). The importance of the above ratio has been independently discovered by Schaffer et al [SE91]. They termed it *safety ratio* for the operator.

The ratio can be used in connection with the response equation to compare different genetic operators. The higher b_t, the better the operator. But unfortunately b_t depends on the fitness function and on the population at generation t. We will therefore restrict our mutation analysis to the ONEMAX function. Schaffer correctly observed that in this case $b_t = 1$ for crossover and $b_t \longrightarrow 0$ for mutation. We will make this observation more precise. The analysis is similar to the one used in [Müh91], [Müh93]. We assume a small mutation rate $m \ll 1$. Furthermore suppose that the selected parents have i bits wrong. Let

$$b(i) = \frac{prob(off.better|i)}{prob(off.worse|i)}$$

$b(i)$ denotes the heritability if the parent has i bits wrong.

Theorem 1 *Let the selected parents have i bits wrong on the average. Then*

$$b(i) \approx \frac{i}{n-i}(1-m)^{n-2i} \tag{2}$$

Proof The probability that an offspring is better than the parent is approximately given by the product of the probabilities of changing at least one of the wrong bits and not changing a correct bit.

$$prob(i|off.better) = (1-m)^{n-i}(1-(1-m)^i)$$

Similarly we obtain

$$prob(i|off.worse) = (1-m)^i(1-(1-m)^{n-i})$$

$$b(i) = (1-m)^{n-2i} \frac{1-(1-m)^i}{1-(1-m)^{n-i}}$$

Taylor series expansion gives for $n \gg 1$ equation (2).
$\blacksquare$

Remark 1 *For $i < n/2$ we have $b(i) > 1$, for $i = 1$ we have $b(i) = 1$ and for $i > n/2$ we have $b(i) < 1$. Mutation is successful far away from the optimum.*

In order to compare mutation with crossover we have to convert $b(i)$ into the population dependent b_t. This can be done with the help of the marginal probability $p(t)$ that allele 1 is at a locus. Then

$$n(1-p(t)) \approx i$$

Therefore we obtain

$$b_t \approx \frac{1-p(t)}{p(t)}(1-m)^{2p(t)-n}$$

For $m = 1/n$ we obtain approximately

$$b_t \approx \frac{1-p(t)}{p(t)} e^{1-2p(t)}$$

We are now able to qualitatively compare populations using mutation and crossover. The response to selection is given by

$$R^M(t+1) = b_t * S(t)$$

$$R^C(t+1) = S(t)$$

These equations quantify what we said earlier about mutation and crossover. If the size of the populations is equal and the selection schedule is identical, then crossover is more efficient for $p(0) \geq 0.5$. This comparison is biased against mutation. It is easily seen from the above eauations, that mutation in large populations is more effective with strict selection. We will try to make a detailed comparison with the help of simulations in the next section.

5 Numerical results

All simulations have been done with the breeder genetic algorithm BGA. In order to keep the number of parameters for the experiments small, several parameters were fixed. The mutation rate was set to the value $1/n$ where n denotes the size of the problem. We have shown in [Müh91],[Müh93], that this is a good first choice. It is optimal for the ONEMAX and the PLATEAU function. The parents were selected with a truncation threshold of 50% or 35% best individuals. Within the selected parents random mating was done.

We will compare populations using mutation only, using crossover only and using both combined. In the following tables the average number of generations is reported which are

OP	N	Fitness Values						sd	FE
		48	56	61	62	63	64		
M	2	41	94	156	183	226	309	82	618
M	64	18	40	65	80	102	143	56	9161
C*	64	7	11	15	15	17	19	1.1	1210
C	128	5	9	12	12	13	15	10.8	1898
M&C	4	23	51	81	96	115	152	47	608
M&C	64	7	13	17	19	20	22	2.1	2102

Table 1: ONEMAX(64); average over 100 runs; C* found optimum in 84 runs only

needed in order that the best individual is above a predefined fitness value. With these values it is possible to imagine a type of race between the populations using the different operators. Table 1 shows the results for ONEMAX of size 64. FE denotes the number of function evaluations necessary to reach the optimum. SD is the standard deviation of the number of generations. The initial population was randomly generated with a probability $p(0) = 0.5$ that there is a 1 at a locus.

The simulations confirm the theory. Mutation in small populations is a very effective search. But the variance in the number of generations needed to reach the optimum is very high. Furthermore, the success of mutation decreases the nearer the population comes to the optimum. Crossover is more predictable. The progress of the population is constant. But crossover needs a certain population size to converge to the optimum. A larger population decreases the efficiency of a population using mutation. The most efficient search is done by the BGA using both mutation and crossover with a population size of $N = 4$.

In table 2 the initial population was generated with $p(0) = 1/8$.

OP	N	Fitness Values					sd	FE
		24	32	62	63	64		
M	2	14	24	192	237	307	85	615
M	64	8	16	96	117	161	72	10388
C*	256	6	9	24	25	27	0.9	6790
C	320	6	9	24	25	26	0.9	8369
M&C	4	11	19	114	136	180	52	725
M&C	64	5	8	29	31	34	3	2207

Table 2: ONEMAX(64); initial population $p(0) = 1/8$; C* found optimum in 84 runs only

The population with mutation needs about eight generations for an increase of the fitness from 24 to 32. This is independent of the popsize. Then the success decreases. In this experiment, mutation in small populations is much more efficient than a crossover population. But the combined search is also performing good.

In table 3 the results are given for the PLATEAU function. The efficiency of the small population with mutation only is slightly larger than for ONEMAX. But the efficiency of the large population is better than for ONEMAX. The best efficiency gives the BGA with mutation and crossover and a popsize of $N = 4$.

In table 4 the results are shown for the DECEPTION(3,10) function.

OP	N	Fitness Values					SD	FE
		288	291	294	297	300		
M	4	27	42	64	95	184	107	737
M	64	5	8	13	19	31	9	2064
C*	64	3	4	6	7	9	1	569
C	128	3	4	5	6	8	1	1004
M&C	4	22	32.5	49	73	134	63	539
M&C	64	10	10	10	10	12	2	793

Table 3: PLATEAU(3,10); C* found optimum in 78 runs only

OP	N	Fitness Values					SD	FE
		283	291	294	297	300		
M	4	419	3520	4721	6632	9797	4160	39192
M	16	117	550	677	827	1241	595	19871
M	64	35	202	266	375	573	246	36714
C*	32	11						
M&C	4	597	3480	4760	6550	9750	3127	38245
M&C	16	150	535	625	775	1000	389	16004
M&C*	64	1170						

Table 4: DECEPTION(3,10); * stagnated far from optimum

We observe a new behavior. Mutation clearly outperforms uniform crossover. But note that a popsize of $N = 16$ is twice as efficient than a popsize of $N = 4$. The performance decreases till $N = 1$. Mutation is most efficient with a popsize between 12 and 24. Crossover does not come near to the optimum. Furthermore, increasing the size of the population from 32 to 4000 gives worse result. This crossover behavior can also be observed in the BGA with mutation and crossover. The BGA does not find the optimum if run with popsizes greater than 50. This is a very unpleasant effect. There exist only a small range of popsizes where the BGA will find the optimum. This problem vanishes if we use 1-point crossover instead of uniform crossover. But then the results depend on the bit positions of the deceptive function. For the *ugly* deceptive function [Müh91] 1-point crossover performs worse than uniform crossover. Therefore we will not discuss these experiments here.

The absolute performance of the BGA is impressive compared to other algorithms. The previous best results for uniform crossover have been achieved by the CHC algorithm of Eshelman [Esh91]. The CHC algorithm needed 20960 function evaluations to find the optimum. The BGA needs only 16000 function evaluations. The efficiency can be increased if steepest hillclimbing is used. We have shown in [Müh91] that mutation in connection with steepest ascent hillclimbing needs only 7900 function evaluations.

The results for SYMBASIN are different. In table 5 the results are given. For mutation this function is only a little easier than the DECEPTION function. Good results are achieved with a popsize between 8 and 64. The SYMBASIN function is a lot more easier for uniform crossover. Therefore the BGA with mutation and crossover performs best. Increasing the popsize decreases the number of generations needed to find the optimum.

In the last table we will show that the above results are also valid for continuous functions. In table 6 results for Rastrigin's function [MSB91] are shown.

OP	N	Fitness Values					SD	FE
		283	291	294	297	300		
M	4	41	1092	2150	3585	7404	4200	29621
M	16	24	125	205	391	765	530	12250
M	64	18	46	68	106	221	136	14172
C*	512	6	16	18	19	20		
C	2048	4	14	15	17	18	0.2	36741
M&C	4	33	1642	2987	5537	9105	1183	36421
M&C	16	15	95	186	331	615	418	9840
M&C	64	12	33	53	90	161	158	10307

Table 5: SYMBASIN(3,10);C*: only 50% reached the optimum

OP	N	Fitness Values				SD	FE
		1.0	.1	.01	.001		
M	4	594	636	691	801	40	3205
M	64	139	176	225	286	9	18316
M&C	4	531	599	634	720	38	2881
M&C	64	50	66	91	123	3	7932

Table 6: Rastrigin's function ($n = 10$)

The results are similar to the results of the ONEMAX function. The reason of this behavior has been described in [MSV93]. A BGA using mutation and recombination with a popsize of $N = 4$ performs most efficiently.

6 Competition

In this section the mutation vs. crossover problem will be investigated in the spirit of population dynamics. This research is the foundation of the migration analysis in the Distributed Breeder Genetic Algorithm. We have shown in the previous section that sometimes a large population is detrimental for the genetic search. In this case a comparison between a panmictic population and a distributed population of the same size will favour the distributed population.

Here we will simulate a competition between populations using different genetic operators. The total size of all populations is fixed. We will describe our competition algorithm for only two populations. After each generation the population with better average fitness increases and the other population decreases. Occasionally individuals are exchanged between the populations. Figure 1 shows the development of the population sizes of two populations. One is using mutation only, the other crossover only. The initial population was generated randomly with $p(0) = 1/64$. This picture confirms the theory. First the population using mutation only grows, then the crossover population takes over. Note that we have a minimal popsize of four for each population, so it cannot get extinguished.

Figure 2 shows the dynamics of mutation. Here four populations compete with four different mutation rates. The population was initialized with $p(0) = 8/64$. At the beginning the population with the highest mutation rate grows, then the two smallest mutation rates take over. The efficiency of this competition is less than of one individual with a muthe last bits correct. Improvements at the beginning of the search do not substantially increase the overall efficiency. This shows that for discrete domains a variable mutation rate is not so important as it is often believed.

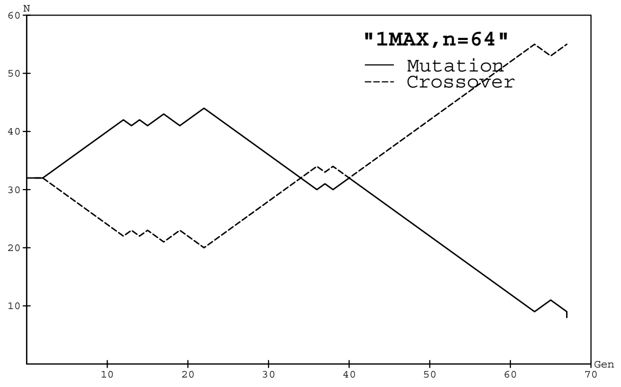

Figure 1: Competition between mutation and crossover

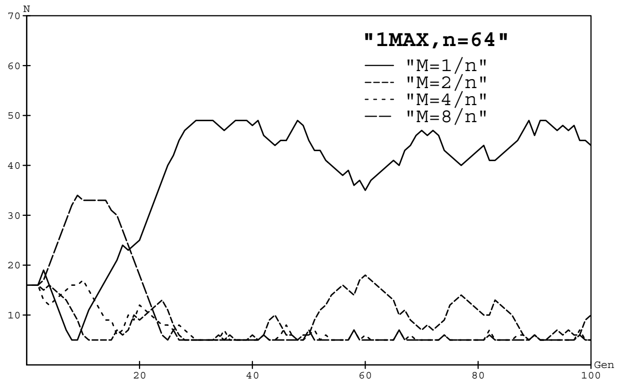

Figure 2: Competition between different mutation rates

7 Conclusion

We have investigated the influence of mutation and crossover on the genetic search. For all the test functions used in this paper a fixed mutation rate of $1/n$ works efficiently. The efficiency of mutation normally decreases with the size of the population. The success of crossover depends on the popsize. It needs a minimal size to converge to the optimum. The combination of both operators in the Breeder Genetic Algorithm leads to a genetic algorithm which performs more robust than with one of the operators alone. A very interesting behavior has been observed for the DECEPTION function. Here good results are only obtained in a small range of popsizes.

In the future we will extend our analysis to still more complex fitness landscapes.

[BF92] A. Bergman and M.W. Feldman. Recombination dynamics and the fitness landscape. *Physica D*, 56:57–67, 1992.

[Bul80] M. G. Bulmer. *"The Mathematical Theory of Quantitative Genetics"*. Clarendon Press, Oxford, 1980.

[Esh91] L.J. Eshelman. The CHC Adaptive Search Algorithm: How to Have Safe Search when Engaging in Nontraditional Genetic Recombination. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 265–283, San Mateo, 1991. Morgan-Kaufman.

[FA90] D.B. Fogel and J.W. Atmar. Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems. *Biol. Cybern*, 63:111–114, 1990.

[Fal81] D. S. Falconer. *Introduction to Quantitative Genetics.* Longman, London, 1981.

[GDK90] D.E. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems*, 4:415–444, 1990.

[MSB91] H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17:619–632, 1991.

[MSV93] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm: Continuous Parameter Optimization. *Evolutionary Computation*, 1, 1993.

[Müh91] H. Mühlenbein. Evolution in time and space - the parallel genetic algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337, San Mateo, 1991. Morgan-Kaufman.

[Müh93] H. Mühlenbein. Evolutionary algorithms: Theory and applications. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, Chichester, 1993. Wiley.

[SCED89] J. D. Schaffer, R. A. Caruana, L. J. EsheOBlman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In H. Schaffer, editor, *3rd Int. Conf. on Genetic Algorithms*, pages 51–60, San Mateo, 1989. Morgan Kaufmann.

[SE91] J.D. Schaffer and L.J. Eshelman. On crossover as an evolutionary viable strategy. In R. K. Belew and L. Booker, editors, *Procedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68, San Mateo, 1991. Morgan Kaufmann.