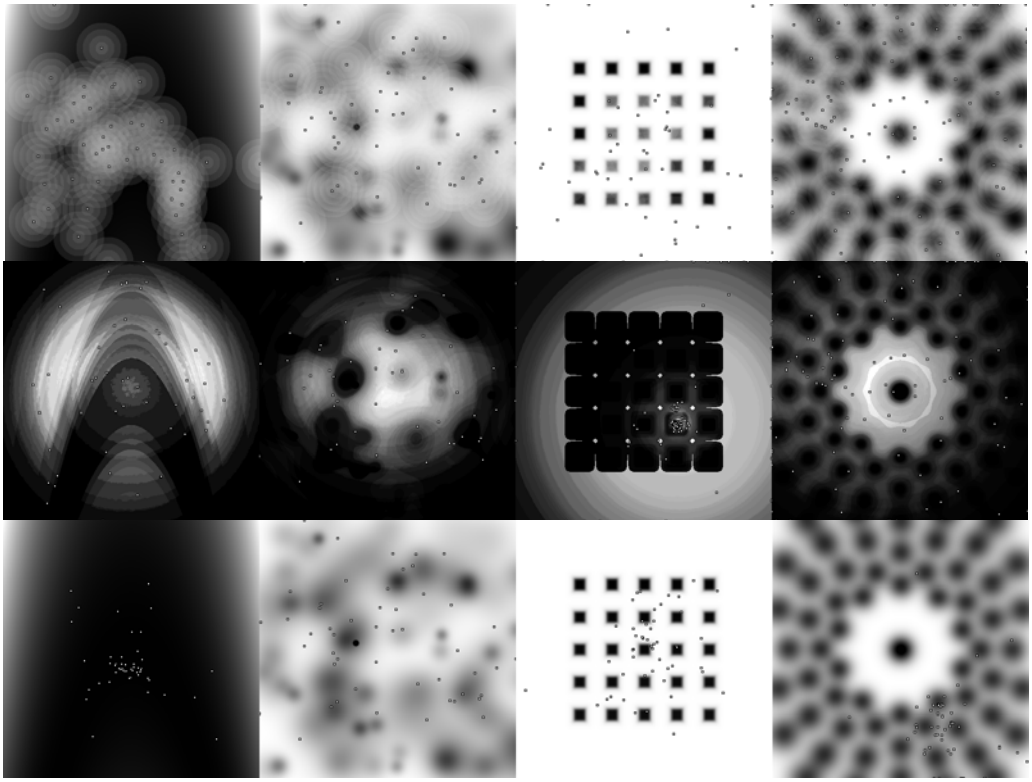
Incorporating frequency dependent selection and sexual selection in genetic algorithms

Paul Snijders
S1214225
July 23, 2005



Internal advisor: Dr. B de Boer (Artificial Intelligence, RUG)
External advisors: Prof. Dr. F. Weissing (Theoretical Biology, RUG)
Dr. S. van Doorn (Theoretical Biology, RUG)
Dr. E. D. de Jong (Information and Computing Sciences, UU)

Artificial Intelligence
University of Groningen

Abstract

In this project, I investigated whether the inclusion of frequency dependent selection in genetic algorithms may overcome the problem of “standard” genetic algorithms that often get stuck at local fitness peaks and do not find the global optimum. I have implemented and tested four methods to maintain diversity in the population of a genetic algorithm by frequency dependent selection. With frequency dependent selection individuals with a rare trait are in the advantage of individuals without the rare trait. When the rare trait becomes common the advantage disappears. Frequency dependent selection is incorporated in the genetic algorithm by selecting individuals that are different from the average population more often than individuals that are less different and have the same fitness value. I tested 4 different methods, a bonus method based on frequency dependent selection in nature, the sharing method as proposed by Goldberg [12], double-objective diversity maintenance as used in genetic programming by de Jong [16], and the fitness uniform selector as proposed by Hutter [15]. These methods were tested on a NK landscape and a series of two-dimensional problems.

Clear winner is the double-objective diversity maintenance method [16] for promoting diversity. This method outperformed the other methods. The bonus method and the sharing method [11] did also perform better than the baseline genetic algorithm, but not as much as the double-objective method. The FUSS method [15] performed worse than the baseline genetic algorithm.

Besides frequency dependent selection I had a look at two other principles from biology, sexual selection and a selective arena. Sexual selection in a genetic algorithm, using different selection pressure and mutation rates for the genders did perform a little better than the baseline genetic algorithm.

By the selective arena in a genetic algorithm the idea of making a lot of offspring and using the ones that seemed best was used. This method was not efficient. The principle of ‘low investment’ and ‘quick and dirty testing’ did not seem to be useful in a genetic algorithm and with the NK landscape problem.

Contents

1. Introduction

This report investigates whether and how methods inspired by biological evolution boost the performance of a genetic algorithm.
The study of genetic algorithms is a sub discipline in the field of evolutionary computing, a rapidly growing area of artificial intelligence. If the performance of genetic algorithms can be improved by using alternative implementations inspired by recent biological insights then this would be of theoretical and practical interest for the area of artificial intelligence. On the other hand, for the evolutionary biologist it is useful not only to study how evolution works in nature but also to look at how evolution works in a different setting, where different constraints apply. Especially artificial evolutionary computing methods that outperform methods found in natural evolution ask for a number of interesting biological questions. What mechanism is responsible for the increased performance of the artificial algorithm, why is it not implemented in nature and why has it not evolved?

Genetic algorithms

Genetic algorithms are optimization algorithms that incorporate the process of evolution. Problems are solved by mimicking an evolutionary process involving selection, mutation, and recombination. Genetic algorithms utilize a population of individuals, each representing a solution to the problem that has to be solved. By means of an optimization function (henceforth called the fitness function) the quality of each solution is expressed in terms of a fitness value, which is then assigned to the corresponding individual. During reproduction, the algorithm rates individuals according to their fitness, such that fitter individuals are more likely to reproduce and, hence, to transmit their problem solving strategy to the next generation. As a result of this selective process the algorithms tend to converge towards a fitness maximum. Genetic algorithms are able to solve a large diversity of problems, but they usually need a higher computation time than more specialized algorithms.
The main problem of generic problem solvers like genetic algorithms is that they often get trapped in a local extreme. A genetic algorithm is usually slightly less vulnerable to this problem because it searches more solutions in parallel (a whole population of solutions) instead of just one solution. In fact, when it does get stuck in a local extreme this is often caused by a loss of genetic diversity. The genetic diversity can be increased by increasing the mutation rate. However, the majority of the mutations is maladaptive, so it is not useful to increase the mutation rate beyond a certain level. In nature frequency-dependent selection (rare individuals have an inherent advantage) and sexual selection are two forces that can maintain genetic diversity in a population. This report is about frequency-dependent selection and sexual selection.

Frequency dependent selection

In the field of conservation biology, it is often stressed that maintaining genetic diversity is essential to ensure the long-term survival of a species. Species face an ever-changing environment. This can be climatic changes, introduction of novel species, diseases or all kinds of other changes. To be able to cope with these changes species must be able to evolve or they become extinct. Populations with a low genetic diversity are not able to evolve fast enough and will not be able to respond to an environment change [9]. One of the main processes in biology that result in maintaining diversity is negative frequency-dependent selection (FDS). Negative frequency-dependent selection occurs when

individuals with some rare trait enjoy a selective advantage relative to individuals without this trait. However, if this trait becomes common it loses its advantage. Negative frequency-dependent selection can result in complex patterns of evolutionary dynamics, including the adaptive emergence of genetic polymorphism [8].
One famous example of frequency-dependent selection is the beak sizes and shapes of Darwin's finches. Different beak sizes and shapes are efficient for eating different kind of seeds. A finch with a rare beak size and shape can have access to different seeds than its competitors. In this way, the finch avoids food competition so it can eat more than other finches and is likely to have more offspring. In this way the Darwin finches have evolved into several species with different beak sizes and shapes [13]. One other example is that predators have a preference for common prey animals. The result is that rare prey animals have an evolutionary advantage above common prey. Sexual selection (i.e., selection not imposed by the natural environment but by the mating tactics and preferences of potential partners) can also lead to frequency-dependent selection when females exert a mating preference for certain rare traits in males.
Would this frequency-dependent selection be efficient in a genetic algorithm? Genetic algorithms do not have to face an ever-changing environment. However, genetic algorithms should be able to evolve easily, so maintaining genetic diversity would seem like a profitable option. This diversity should enable a genetic algorithm to get out of a local optimum and to find a new, better optimum instead. Besides maintaining diversity frequency-dependent selection can lead to complex patterns of evolutionary dynamics which will result in the exploration of a far larger part of the fitness landscape in the genetic algorithm than in a standard genetic algorithm.

In genetic algorithms several methods have been proposed to maintain diversity in a population. A few methods, like the method of islands [20] attempt to maintain diversity by barriers between individuals in the population to prevent mating between groups of individuals. These methods will not be discussed in this report. Here the focus will be on selective methods to maintain diversity (i.e., based on the selection of certain individuals in a population).
I have found three selection schemas that try to promote diversity by giving an advantage to rare individuals. I will mention them here and discuss then later in depth. The first method is sharing, proposed by Goldberg [12]. This method divides each individual's fitness by the number of individuals that are similar to that individual. In this way common individuals will get a lower fitness and rare individuals will get a higher fitness. The second method promotes diversity by the use of a double-objective diversity maintenance genetic algorithm that maximizes the fitness and the diversity in the population as proposed by de Jong [16]. The third method is the fitness uniform selection scheme (FUSS) as proposed by Hutter [15]. This method selects individuals uniformly on their fitness.
Beside these three selection schemes I will propose an own selection schema, bonus selection. In this selection schema each individual that is different will get a fitness bonus. This method looks like the sharing selection schema of Goldberg but is biologically more plausible.

Sexual selection

In nature, the most successful, complex and numerous species are sexually-reproducing animals and flowering plants [21, 27]. The cost of reproducing sexually is however quite high. Therefore there must a reason why in nature it is efficient to reproduce sexually. One possible factor can be the difference in selection pressure and mutation rates between the two genders [1, 24]. To test if this hypothesis can work in a genetic algorithm sexual selection is incorporated in the genetic algorithm by allowing the selection pressures and the mutation rates to be different for the two sexes.
A complete other theory based on the same principle of strong selection without promising species survival is the theory of a selective arena. In some organisms more fertilized zygotes are produced than there are reared to hatching or birth. In the very first days of the existence of the fertilized zygotes there is a strong and relatively indiscriminating phase of selection [25]. This selective arena is mimicked in a genetic algorithm by implementing an additional, early and relatively indiscriminate phase of selection: instead of a single offspring that automatically joins the next generation as an adult, a pair of parents gets several offspring, and a first round of 'quick and dirty' selection will determine which one of these offspring will become an adult in the next generation.

Experimental setup

In this report I will compare the performance of a genetic algorithm with frequency-dependent selection, sexual selection, and a selective arena with a basic genetic algorithm. For this comparison I will test the algorithm on two different problems. The main question will be whether the proposed methods will boost the performance of a genetic algorithm or not.

Structure of the report

This report is structured as follow: In section 2 the basic version of the algorithm, which is used as the standard for comparison, is explained. Section 3 explains the test problems. Section 4 is about frequency-dependent selection. In section 5 the results of the frequency dependent selection methods on the test problems are described. Section 6 describes the sexual selection methods and how the selective arena works. The discussion in section 7 summarizes the observations and puts these in the larger context of differences between evolution in a genetic algorithm and evolution in nature.

2. Genetic algorithms

Genetic algorithms were proposed in the 1970s by John Holland [14]. In the beginning the work on genetic algorithm was mainly theoretical. When the computing power of computers increased, genetic algorithms could be used for optimizing problems in a large range of disciplines. One of the pioneers in the field was David Goldberg who proposed a basic genetic algorithm called the simple genetic algorithm [11].
In this report I will compare some genetic algorithms with a small adjustment to incorporate frequency dependent selection, sexual selection, and a selective arena with a baseline genetic algorithm. In this section I will discuss the baseline genetic algorithm [5].

The baseline genetic algorithm

The baseline genetic algorithm utilizes a population. In this population each individual has a genotype that is constructed from a bit string (string of zeros and ones). This bit string is the encoding of a solution of the problem. A fitness function can rate this solution. The population is initialized with random individuals. Then evolution starts. New generations are made. Each new generation is filled with offspring from parents in the old generation. Some of the offspring are mutated.

The outline of the simple genetic algorithm is:

- Initialize a random population with population size N.
- For G generation do:
  - Calculate the fitness of each individual.
  - Copy the fittest individual unchanged to the new generation (Elitism)
  - Fill the new generation with offspring by sexual reproduction.
  - For each new offspring, mutate it with a probability of μ.
- End loop.
- Return the fittest individual as the best solution.

For the creation of an offspring two parents are chosen by two times a best-of-four tournament selection (the fittest individual is selected from four randomly chosen individuals). I have chosen for a best-of-four selection because in biological populations it is unrealistic that mates are chosen from the complete population as with a roulette wheel selector. I have tried several other selectors but there didn't seem to be a great difference between the selectors.

The offspring is created from two parents by a variance of an n-point crossover. This n-point crossover works as follows: The first bit of a random parent is copied to the offspring. The next bit is then copied from the same parent as the preceding bit with a probability of 1 - χ or copied from the other parent with a probability of χ. In this way the bit string of an offspring consists of stretches of bit strings from both parents (see figure 1). This method seems more biologically plausible than the standard n-point crossover [5] and can easily be tuned by one parameter.

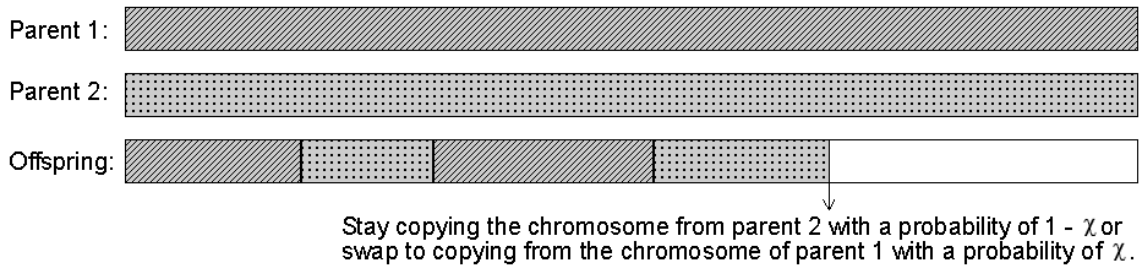


Figure 1. The crossover method.

Each offspring is mutated with a probability of μ. One random bit of the bit string of an offspring will be chosen and will be swapped from 0 to 1 or from 1 to 0. I have chosen to mutate each offspring with a certain probability and not mutate each locus (one bit in the bit string) with a certain probability as in nature because then the optimal mutation rate can be more constant between problems with a different genome length.

3. The test problems

For testing the algorithm I use two fitness landscapes, a two-dimensional problem space with several optima and the NK-landscape problem. The two-dimensional problem is very easy to visualize. It is even possible to see how the algorithm is working by printing an individual on the landscape as a little dot. The NK-landscape is a more complicated problem where the solution is not so easy to find.

3.1 The two-dimensional problem

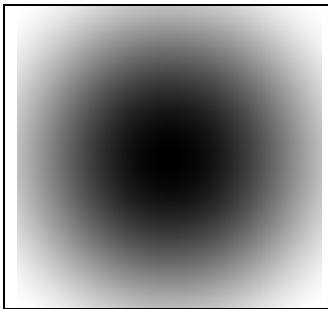
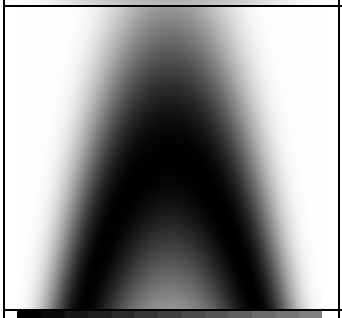
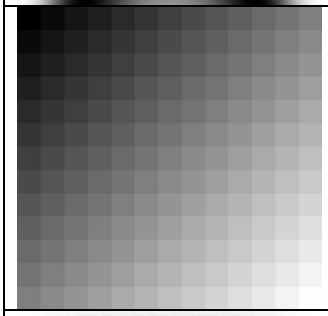
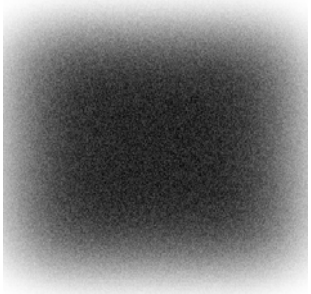
The two dimensional problem is a fitness landscape with a size of 400 by 400. The functions used for creating the landscapes are continuous but I use a discrete search space so an easy lookup table with fitness values can be created. The size of 400 by 400 is arbitrary. For the landscapes I use the 5 test functions from De Jong [17] and 2 sets of self developed fitness landscapes.

Problems

De Jong presented in his work 5 test functions for testing the performance of various optimization algorithms [17, 30]. He chose each function because each of them represents one of the main difficulties that occur in optimization problems.

I will explain here the 5 test functions from de Jong. I converted the functions to the two dimensional domain for simplicity. I rescaled the graphs of the landscapes for clarity, black represents high fitness and white represents low fitness.

The domains here are the domains called in [30]. For my experiments the fitness landscapes will be resized to the domain with a size of 400 by 400.

	Sphere is smooth, symmetric and has only one clear peak. Any optimization algorithm should perform well on this problem. $$f_1(x,y) = x^2 + y^2$$ $$-3.12 \le x, y \le 3.12$$
	Rosenbrock has a narrow ridge around a parabola. If an algorithm is not able to discover good directions this is a hard problem. $$f_2(x,y) = 100(y - x^2)^2 + (x-1)^2$$ $$-3.12 \le x, y \le 3.12$$
	Step is the problem of locally flat surfaces. Flat surfaces are hard for a lot of optimization algorithms because they do not give any local information in which the algorithm should search. $$f_3(x,y) = \lvert x \rvert + \lvert y \rvert$$ $$-3.12 \le x, y \le 3.12$$
	Quartic is a unimodal function padded with noise. Algorithms with problems on noisy data will perform poor on this problem. $$f_4(x,y) = x^4 + 2y^4 + gauss(0,1)$$ $$-1.28 \le x, y \le 1.28$$
	Foxholes is a function with many local optima and fitness gaps that should be overcome. $$f_5(x,y) = \frac{1}{0.002 + \sum_{i=1}^{5}\sum_{j=1}^{5}\left[(x-a_i)^6 + (y-a_i)^6 + j + 1\right]}$$ $$a_i = (-32,-16,0,16,32)$$ $$-63.336 \le x, y \le 63.336$$

Beside these functions from De Jong I have developed two sets of two dimensional fitness landscapes. De Jong functions are hard for most standard optimization algorithms but are all quite easy for a genetic algorithm (without considering computation time). Therefore I created two set of new problems, a set of random landscapes and a set of regular landscapes.
The random landscapes consist of the sum 100 independent randomly generated normal distributions with a random location, width and height (volume). Besides these randomly generated normal distributions there is one narrow and high normal distribution at a random place in the landscape. This is the global optimum of the problem which should be found by the genetic algorithm. Because there is no clue at a place in the landscape where the global optimum can lie we can call this a 'needle in a haystack' problem. A genetic algorithm is not the most efficient way to find the solution for this problem. A random search would be more efficient. I use this landscape to see whether the genetic algorithm is able to search actively through the problem space without concentrating its search around one single optimum.

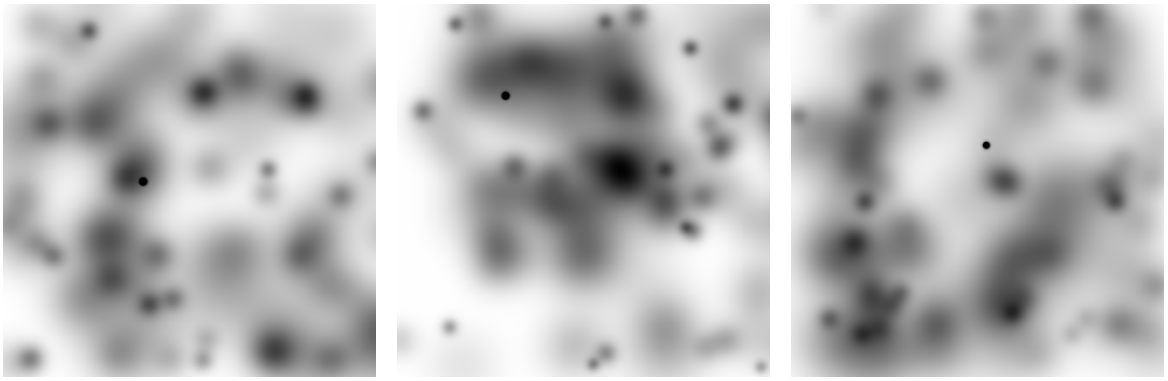


Figure 2. Three fitness landscapes from the series of random two dimensional problem.

The regular landscape is a designed fitness landscape where there is a 'fitness gap' which should be passed by the genetic algorithm. So besides searching actively in the problem space the algorithm should be able to cover this fitness gap. This regular fitness landscapes range from easy (small gap and centered global optimum) to difficult (big gap, global optimum at a side). This is even worse then the 'needle in a haystack problem', it is a 'hiding place with decoys problem'.

Figure 3. Three regular fitness landscapes with a gap between the global optimum and local optima.

Testing

The performance of a genetic algorithm on this problem is measured by counting how many times the global maximum is found by at least one individual in 25 trials after 200 generations on 5 De Jong landscapes, 10 random and 10 regular fitness landscapes.

Representation

An individual in the population, which represents a solution for this problem, can be constructed in various ways, each with its own advantages and disadvantages. I have chosen for a representation in which big mutation or recombination jumps are not possible in the two dimensional landscape so a gap in the picture of the fitness landscape is a gap for the problem as well.

In my implementation each individual is characterized by a bit string with a length of 800. The x location of the individual in the problem space is the sum of the first 400 bits. The y location of the individual in the problem space is the sum of the second 400 bits.

Crossovers will result in an offspring with a position somewhere in between the two parents. This is represented graphically in figure 4. This figure shows, for two parents with given coordinates, the location of hundreds of offspring.

Single mutation, like described in the baseline genetic algorithm would result in only a very small change in the solution of an individual. Therefore we defined the mutation for this problem in another way. Mutations, occurring with a probability of μ are defined as a jump in the landscape of a maximum length of λ. A graphical representation of this mutation jump can be seen in figure 5.

The implementation of this mutation jump is done by taking a random length of the jump with a maximum of λ and a random direction. Now the new position of the individual is known. To encode this position in the bit string a certain number of bits in the bit string is flipped (only from 1 to 0 or 0 to 1 for the x or y coordinate) at a random stretch of the bit string so the new location of the individual is reached resulting in the mutation jump as can be seen in figure 5.

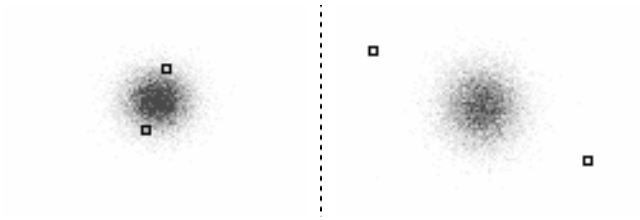


Figure 4. Two pictures of two parents and the location where the offspring is likely to be.

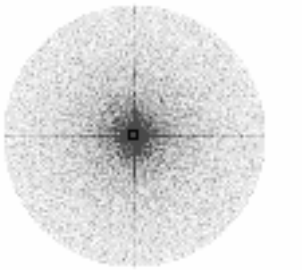


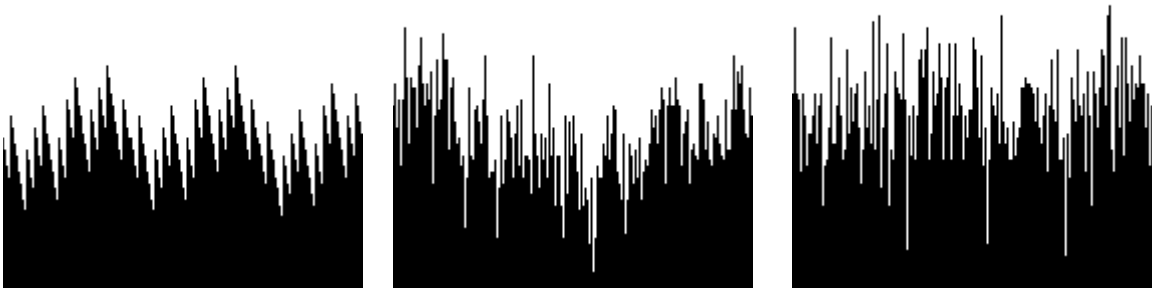
Figure 5. The location where an individual is likely to be after a mutation with λ is 50.

I also considered a representation where each bit of an individual is used as in the binary number system [5]. This would however result in mutation jumps that could easily overcome the fitness gaps of the random and regular landscapes. In other words, the fitness gaps would not be fitness gaps anymore.

Representing the individuals as two integer numbers representing the x and y coordinate was also an option [5]. I have chosen not to use this option because I could not use my n-point crossover method with this representation. Nevertheless, my current representation acts very much like this representation.

3.2 NK landscapes

NK landscapes were proposed by Kauffman [18] as a representation of a rugged fitness landscape. The shape of an NK landscape is completely determined by 2 variables, N, the length of the genome and K, the number of other bits that forms useful combinations with a bit. In this setting K = 0 is a completely regular landscape with only one maximum. K = N – 1 on the other hand, corresponds to a fully random fitness landscape (figure 6). The performance of the genetic algorithm is measured by taking the fitness of the fittest individual after G generations.



N = 8, K = 0 N = 8, K = 4 N = 8, K = 7

Figure 6. A representation of the NK fitness landscapes in 2 dimensions. The bit strings are represented in the number of with they are the binary expansion. In this way all the points that are at a horizontal distance of a power 2 (1, 2, 4, 8 …) of each other are neighbors in the bitwise representation. So this makes the K = 0 landscape a landscape with only one maximum.

The NK landscape problem is coded by a bit string with a length of N. As an example we take a bit string with a length of 8 (N = 8), say 10110100. The fitness of a solution is calculated as follows: All bits in the bit string have a certain fitness. The fitness of the complete bit string is the sum of the fitness of its bits.

$$f = \sum_{k=1}^{N} b_k$$

Here f is the fitness of the individual and b_k is the fitness of the bit number k. N is the number of bits in the bit string.

The fitness of a bit in the string is determined by K other bits in the string that interact with this bit (the “useful combination”). Which bit interacts with which other bits is determined by the pattern of connectivity (see figure 8). When we use a pattern of connectivity in which each bit interacts with the K bits at its right hand side (nearest neighbor method) and we have a K of 1, then the first bit interacts with the bit at its right hand side (0 in this case, what will result in a useful combination of 10). For each bit, there is a fitness table. In this fitness table for useful combination there is a certain fitness assigned. We do a lookup for the bit in the fitness table where we find the fitness of this bit. For each problem, fitness tables are assigned at random in a uniform distribution (see figure 7).

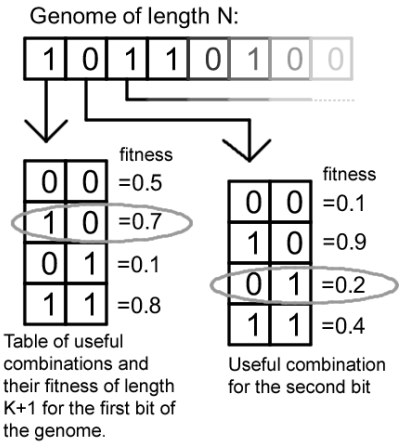


Figure 7. The NK landscape with a genome and two fitness tables for the first and second bit of the genome. Here K = 1 and every bit interact with the bit at his right side.

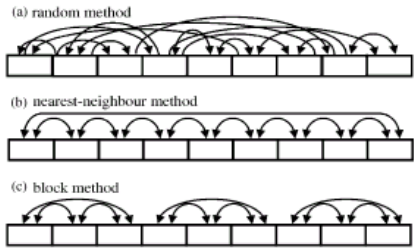


Figure 8. Possible patterns of connectivity of the NK landscape. (Figure from [29])

An NK fitness landscape can have different patterns of connectivity (see figure 8). The way in which these patterns are assigned makes little difference to the statistics of the NK landscape but there is a great difference in analytical tractability [29].
If we want to know the global maximal and minimal solution of the problem (in order to compare the performance of various genetic algorithm implementations) we have to search through all the possible solutions. When we use the nearest neighbor pattern of connectivity and we have a landscape with N = 20 we have search through 2^{20} solutions, which can be done in a few seconds on a current computer. However, if we have an N of 60 we have to search through 2^{60} solutions which will take ages. Therefore I use the block method with a block size of 20. Now we can split up a problem into different problems with an N of 20: we can analyze a problem with an N of 60 by searching 3 x 2^{20} solutions which still can be done in a few seconds. When the minimum and maximum solutions of a problem are calculated I rescale the results of the problem to a domain between 0 and 100.

4. Frequency-dependent selection

In the baseline genetic algorithm the “reproductive success” of a solution only depends on its fitness, where the fitness is determined by one of the landscapes described in section 3. Here selection is frequency independent, so the success of a selection does not depend on the frequency distribution of others in the population. Selection is (negatively) frequency dependent if rare or deviant solutions have a selective advantage with respect to more common solutions. I implement frequency-dependent selection in four different ways. In the bonus method individuals will receive a fitness bonus if they are average different from the population as a whole. The sharing method is similar, but this time rare individuals will not receive a bonus but common individuals will receive a fitness penalty by dividing its fitness by the number of similar individuals. With double-objective diversity maintenance, fitness is not modified, but next to fitness the distance of a solution to other solutions is a second selection criterion. With the FUSS method the selection is based on a different kind of selector which selects individuals uniformly on their fitness.

For the first three methods we need a way to qualify similar and deviate solutions from a set of other solutions. For this we measure the “distance” between two individuals. We could take the genotypic distance, which is the sum of the differences in the bit strings representing the individuals. We can also take the phenotypic distance. For this we must calculate the distance between two individuals. I prefer using this phenotypic distance because in nature selection is on the phenotype. For the two dimensional problem the phenotypic distance between two individuals *i* and *j* is given by the Euclidian distance between their location in phenotype space:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Figure 9. Example of difference in the two dimensional landscape. (Euclidian distance)

For the NK landscape problem I take the genotypic distance, which is defined by the number of positions at which the bit strings of two individuals I and j differ from each other. Formally it can be defined as:

$$d_{ij} = \sum_{k=1}^{n} \left| g_{ik} - g_{jk} \right|$$

Here g_{ik} is bit *k* of individual *i*.

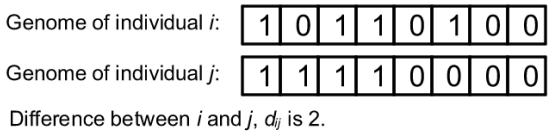


Figure 10. Example of difference in the NK landscape. (Hamming distance)

The average distance of individual *i* to the population as a whole is defined by is the average of all the difference between this individual and all other individuals:

$$\bar{d}_i = \frac{1}{n} \sum_{j=1}^{n} d_{ij}$$

Now we can calculate the distance between two individuals we can calculate the diversity in the population. The diversity in the population is the sum of all the differences between all the individuals in the population.
Figure 11-14 shows the effect of frequency-dependent selection on the fitness landscape that is used for the selection processes. A black and white square in these figures is an individual; white is low selection probability and black is high selection probability. For each figure, the fitness is rescaled to the best visible domain. The figure top-left is the normal fitness landscape. The other figures are representations of selection pressures as is used by the algorithms using frequency dependent selection. The landscapes are dependent of the individuals in the landscapes so they will change each generation.
These figures are generated by placing a hypothetical individual in the landscape and calculating the probability that the hypothetical individual is chosen for reproduction considering the other individuals. The selection pressure around the individuals in the landscapes using frequency dependent selection is clearly lower than in the landscape without frequency-dependent selection. Between figure 12 and 13 the difference between the sharing and bonus frequency dependent selection method is visible. The right group of individuals consists of 40 individuals and the left group of 10 individuals. The sharing method looks whether there are similar individuals nearby one spot. The bonus method looks more if there are different individuals somewhere else. Beside this the effect of the sharing method on the selection pressures is much stronger. In figure 14 the dominance of hypothetical individuals is calculated and in this way the selection pressures of the double-objective diversity maintenance are generated.

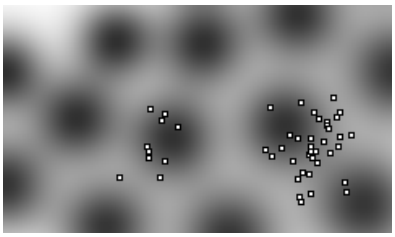


Figure 11. Fitness landscape with two groups of individuals without frequency dependent selection.

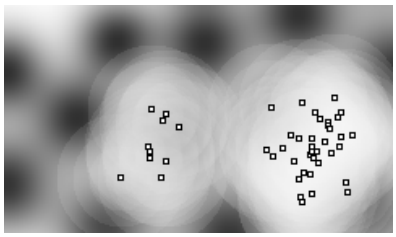


Figure 12. Representation of selection pressures derived from interaction between the fitness landscape and frequency dependency using the sharing method.

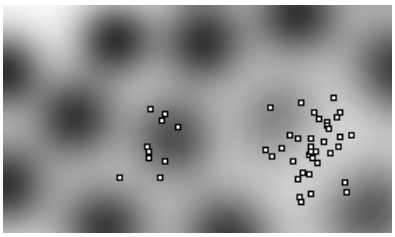


Figure 13. Representation of selection pressures derived from interaction between the fitness landscape and frequency dependency using the bonus method.

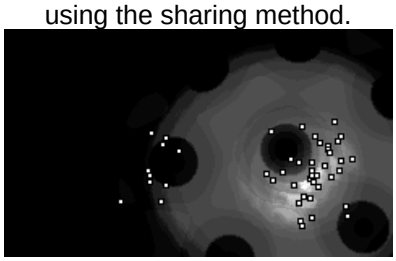


Figure 14. Representation of selection pressures derived from interaction between the fitness landscape and frequency dependency as is used by the double-objective diversity maintenance method. Here the groups of individuals are a bit different for a nicer picture.

4.1 Sharing method

The sharing method is based on a method developed by Goldberg [12] in the context of niching. Niching methods are methods to find more than one good solution by a genetic algorithm. I made some small changes to Goldberg's method.
With sharing, each individual *i* has to "share" its fitness with similar individuals in the population. Individual *i*'s fitness f_i is divided by a "penalty" term $1+\beta \bar{s}_i$, where $\bar{s}_i$ denotes the average similarity of *i* to the rest of the population. Hence selection is determined by "modified fitness" f_i':

$$f_i' = \frac{f_i}{1+\beta \bar{s}_i}$$

The modified fitness, f_i' called the frequency dependent fitness is dependent on the state of the population and hence frequency dependent. To determine the average similarity we first define the similarity between two individuals *i* and *j*, s_{ij}:

$$s_{ij} = \max\{1 - {d_{ij}}/{\sigma} \quad ,0\}$$

Here *d* is the distance between the two individuals *i* and *j*. There is a threshold σ from where individuals are considered completely different. This similarity between two individuals is scaled between 0 and 1.
The average similarity $\bar{s}$ of individual *i* is equal to 1 if there are no similar individuals and is equal to 1 if the individual is completely similar to all the other individuals:

$$\bar{s}_i = \frac{1}{n-1} \sum_{\substack{j=1 \\ j \neq i}}^{n} s_{ij}$$

Here *n* is the number of individuals in the population.

In short, the frequency dependent fitness, f_i' will be equal to the fitness if the individual is completely dissimilar to all the other individuals. If all the individuals are equal, f_i' will be equal to the fitness of an individual divided by 1 plus the scaling factor β (figure 15).

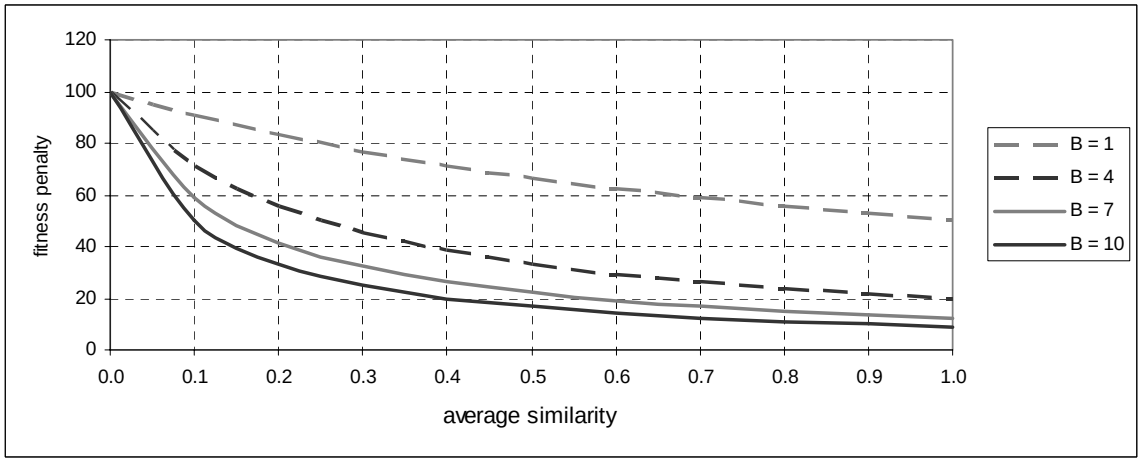


Figure 15. Fitness penalty, $\frac{1}{1+\beta \bar{s}}$ relative to the average similarity using different β calculated by the sharing method.

4.2 Bonus method

The bonus method is similar to the sharing method but biologically more plausible. Instead of sharing the fitness between individuals, the individuals that are rare receive a fitness bonus.
With the bonus method, each individual will receive a fitness bonus if it is not equal to other individuals. Its frequency dependant fitness can be calculated by:

$$f_i' = f_i(1 + \beta \bar{d}_i)$$

Here β is again the scaling factor and $\bar{d}$ is the average distance between the individual and all the others.

$$\bar{d}_i = \frac{1}{n}\sum_{j=1}^{n} d'_{ij}$$

Here *d'* is the distance between two individuals scaled to a domain between zero and one.

$$d'_{ij} = \min\{{}^{d_{ij}}\!/_{\sigma}, 1\}$$

Again, here *d* is the distance between the two individuals and σ is the threshold from where individuals are completely different.
Summarizing, if $\bar{d}_i = 0$ an individual is identical to all the other individuals then the frequency dependent fitness is equal to the prior fitness of an individual. On the other hand, if the individual is more different from all the other individuals than the threshold, then that individual will receive a frequency dependent fitness of the original fitness multiplied by 1 plus the scaling factor β (figure 16).

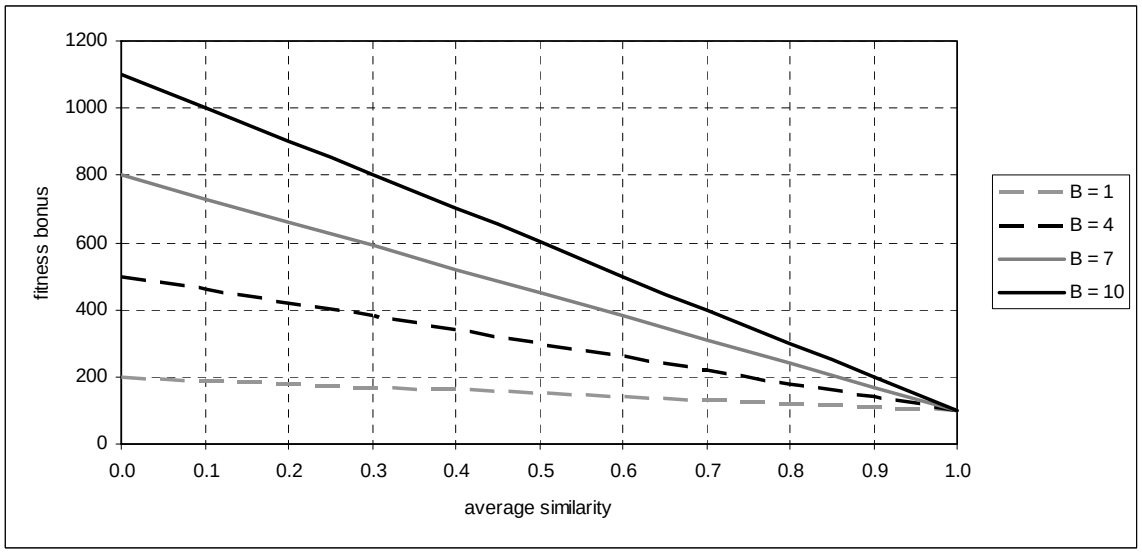


Figure 16. Fitness bonus $1 + \beta \bar{d}_i$ as a function of the average difference using different β calculated by the bonus method. The *average difference* is equal to 1 – *average similarity*.

4.3 Double-objective diversity maintenance (DODM)

Promoting diversity by considering diversity as an objective using a double-objective genetic algorithm is a promising way to maintain diversity. This idea was first proposed by Edwin de Jong and others [16]. They compare a basic genetic program with a multi-objective genetic program with fitness, diversity and program size as objectives. Their multi-objective genetic program outperforms the basic genetic program on a 3, 4 and 5-parity test problems.
Instead of modifying the fitness function as is being done by the sharing and the bonus method, the double-objective diversity maintenance method judges the fitness and the distance of an individual as independent entities. This method is a special implementation of multi-objective methods. In multi-objective genetic algorithms the basic idea of optimizing is to search for multiple solutions which satisfy the different objectives to different degrees. A key concept in multi-objective optimization is dominance. Individual A dominates B if A is equal or better then B on each objective and if A is better then B on at least 1 objective. So A dominates B if:

$$\forall i \in [1..n] : A_i \geq B_i \wedge \exists i : A_i > B_i$$

Individuals who are not dominated by any other individual are called *Pareto optimal* solutions.
With double-objective diversity maintenance the objectives are (1) fitness and (2) distance (as explained as in the beginning of section 4). This idea is incorporated in the baseline genetic algorithm by calculating by how many individuals an individual is dominated using fitness and distance as objectives. The best of four tournament selector selects the individual that is dominated by the fewest other individuals out of four randomly chosen individuals.

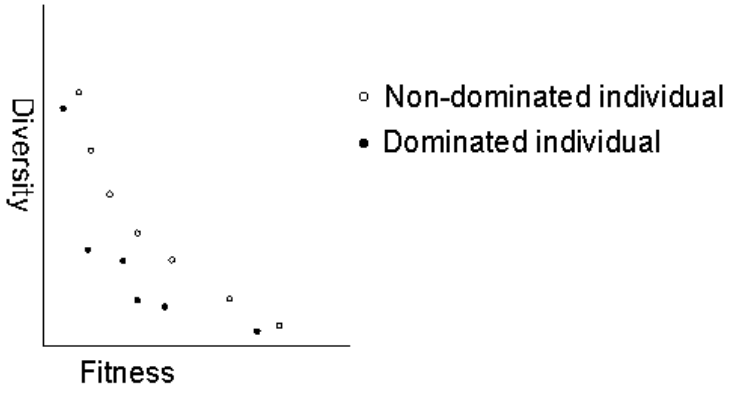


Figure 17. A graph of individuals in a population and their objectives. The individuals where there are no other individuals that have a better fitness or have a greater distance from the rest of the population are the non-dominated individuals.

4.4 Fitness uniform selection

The fitness uniform selection scheme (FUSS) as proposed by Hutter [15] selects individuals by picking randomly a fitness value from a uniform distribution and then searching an individual from the population that best matches that fitness value. In this way, individual with different fitness, and so different genotypes are chosen so the diversity in maintained. Selection uniformly on fitness value means that each individual that has a rare fitness value will be chosen more frequently than individuals with a common fitness value. Individuals with a high fitness values are usually less frequent than individuals with a normal fitness value so the fitter individuals are selected more often than the unfit ones.
The fitness uniform selection was compared with the tournament selector by Legg and others [19]. They conclude that FUSS performs better on a deceptive two dimensional problem but does not perform well at most other problems. While the total genetic diversity is very strong, the diversity among the fittest individuals is very low. In this way the genetic algorithm has to exploit a small number of individuals too much to find a good solution. To overcome this problem I have tested another version of FUSS where I select 2 individuals by FUSS of which I select one by a basic tournament selector. In this way the individuals with a low fitness are chosen less frequently and there will be more individuals with a high fitness. I will call this second method FUSS2.
One of the advantages of FUSS compared with the method mentioned above is that FUSS does not need a measurement of difference for exploiting a sort of frequency dependent selection.

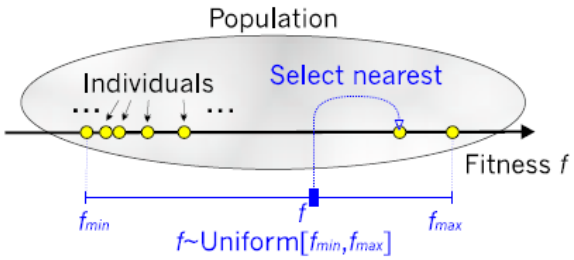


Figure 18. The selector schema used by the FUSS method. From the fitness domain a uniform random value is chosen. Then the individual that has a fitness that is nearest to this random value is selected. Figure from [19].

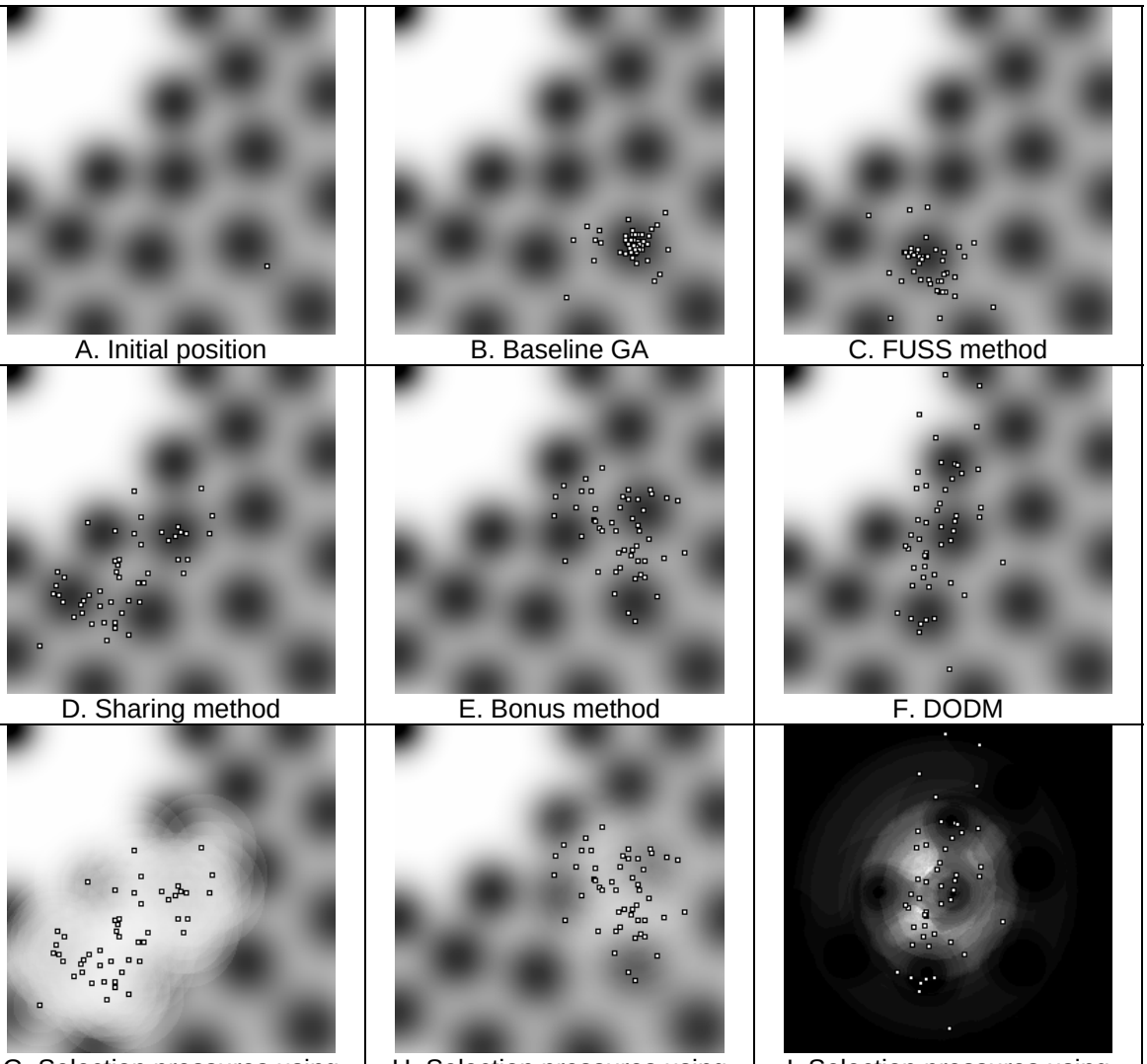
5. Results

5.1 Two-dimensional problems

Testing the performance of a genetic algorithm with respect to a two-dimensional problem quantitatively is tricky because the performance is in a given case closely dependent on the parameters of the genetic algorithm. In this way you can determine beforehand whether it is likely that the algorithm will find the global optimum or not, by choosing suitable parameters. The main challenge in the two dimensional problem is whether the population will shift from a lower fitness peak to a higher fitness peak. Such a peak shift can happen when the new fitness peak is in reach of one mutation step from an offspring which can be created a crossover from two individuals in the population.
We can analyze three factors which have influence on the peak shifting:

- Method by which an offspring is created.
- Size of the possible mutation step.
- Position of the current individuals in the population.

An offspring will find a position in the fitness landscape that lies between the x and y coordinates of its parents because of the crossover operation (as in figure x at the modeling framework section). New optima are only found in this way if the location of the new optimum is in between two individuals who are selected to be the parents.
After a mutation an individual can find a position at a maximum distance of λ away from the original individual. In theory each position can be reached by a mutation if λ is big enough but with very small probability. I will only use relatively small λ because using a big λ in a big search space will result in a nearly complete random search which will usually not result in finding any good solutions.
The position of current individuals in the population is the most interesting when looking at frequency-dependent selection. Without frequency-dependent selection all the individuals will cluster around the highest fitness peak which is already found. When we use frequency-dependent selection individuals will spread over a larger area of the landscape and will be able to find a better optimum more easily.
Figure 19 shows how the population is distributed over the landscape after 40 generations beginning from one spot (figure 19A).
You can see that with the sharing frequency dependent selection the individuals are most distributed over the landscape. Without any frequency dependent selection the individuals are least distributed over the landscape.
In the absence of frequency dependency (figure 19B) the individuals are still highly clustered around a single fitness peak. Applying the FUSS method (figure 19C) leads to a population that is a bit more spread out, but not to a major extend. In contrast, the other methods (figure 19D-F) will lead to a population that is spread out around the fitness landscape. Panels 19 G – I shows a representation of selection pressures derived from interaction between the fitness landscape and frequency dependency methods as it is using the current population.



A. Initial position | B. Baseline GA | C. FUSS method

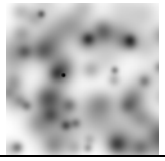
D. Sharing method | E. Bonus method | F. DODM

G. Selection pressures using the sharing method | H. Selection pressures using the bonus method | I. Selection pressures using DODM

Figure 19, Right bottom part of a regular landscape. Black is high fitness. White is low fitness. The panels B, C, D, E, and F show the normal fitness landscape with the individuals of the genetic algorithm printed in the fitness landscape after 40 generations using $\lambda = 40$ starting at a fixed position (as is shown in figure A). The three bottom panels show a representation of selection pressures derived from interaction between the fitness landscape and frequency dependency methods.

I tested each genetic algorithm 100 times on the 5 De Jong landscapes. If the global optimum was found within 200 generations, I rewarded the algorithm with a point. The sum of all the points received was the performance of the algorithm. So if the algorithm had a score of 100 it found the global maximum every time and if it had a score of 0 it did not find the global optimum once. The parameters were chosen as follows: mutation rate $\mu = 0.50$, crossover parameter $\chi = 0.025$, maximum size of a mutation $\lambda = 50$ and a population size of 50. I tested this 25 times. The initial population started each time in the corner opposite to the global optimum. On the sphere and step function each algorithm found the global optimum every time so I will not discuss them further.
I tested the regular and random landscapes as described in section 3.1 in the same way as the de Jong landscapes. For these parameters the algorithms found the solution only

at 4 of 10 random landscapes and never found the solution in the regular landscapes. Therefore only the four random landscapes where the optimum was found some times are discussed. I will discuss the average of the results of the four random landscapes because the different performances between these landscapes are not interesting.
The results are: (mean +/- standard error)

Times global optimum found with landscape:	FUSS	FUSS2	Baseline	Bonus	Sharing	DODM
Rosenbrock 	4.88 +/- 0.35	14.76 +/- 2.49	48.68 +/- 0.97	80.32 +/- 0.79	91.20 +/- 0.47	99.92 +/- 0.05
Quartic 	95.64 +/- 0.37	68.20 +/- 8.99	100 +/- 0	100 +/- 0	100 +/- 0	100 +/- 0
Foxholes 	87.20 +/- 0.61	57.20 +/- 4.34	36.84 +/- 0.73	63.4 +/- 0.94	78.72 +/- 0.83	81.52 +/- 0.73
Random 	20.19 +/- 0.32	8.69 +/- 0.71	3.66 +/- 0.16	10.82 +/- 0.32	47.11 +/- 0.31	72.00 +/- 0.38

Using the Tukey-Kramer test for multiple comparisons [31] we can conclude that at a significance level of $\alpha = 0.05$, all methods perform differently with respect to the Rosenbrock function. We get the following order:

DODM > Sharing > Bonus > Baseline > FUSS2 > FUSS

When applied to the Quartic function, most of the methods perform equally well. Only the FUSS2 method performs significant worse.

DODM, Sharing, Bonus, Baseline, FUSS > FUSS2

For the Foxholes function the FUSS, DODM and Sharing algorithm performs best, followed by the Bonus, FUSS2 and baseline algorithm performs significantly worse.

FUSS, DODM, Sharing > Bonus, FUSS2 > Baseline

When applied to the Random landscapes, DODM performs significantly best and the methods perform worse.

DODM > Sharing > FUSS > Bonus, FUSS2, Baseline

In all cases, double-objective diversity maintenance belonged to the best performing methods. Only in a single case (the Foxholes problem) another method (FUSS) performed better on average, but this difference was not statistically significant.

5.2 The NK landscape problem

I used the NK landscape with an N of 60 and a K of 10. I looked at the best fitness after 2000 generation with a population of 100 individuals. The mutation rate μ is optimized for each problem. The crossover parameter χ is 0.025. The solutions from the problem are scaled in a way that 100 is the best possible solution and 0 is the worst possible solution. The algorithm is tested 500 times. The results are:

	FUSS	FUSS2	Baseline	Sharing	Bonus	DODM
Optimal mutation rate μ.	2.2	2.3	1.9	1.2	0.9	0.3
Maximal fitness +/- standard error	91.49 +/- 0.10	91.82 +/- 0.09	93.83 +/- 0.08	95.03 +/- 0.08	95.04 +/- 0.09	96.16 +/- 0.06
Average fitness in the population	64.45	72.19	74.23	81.45	84.23	72.47

Using the Tukey-Kramer test for multiple comparisons [31] we can conclude that at a significance level of α = 0.05, the double-objective method for diversity maintenance performs best thereafter the Bonus and Sharing method perform best, then the Baseline algorithm and finally the FUSS methods performs the worst of all. We get the following order:

DODM > Bonus, Sharing > Baseline > FUSS2 > FUSS

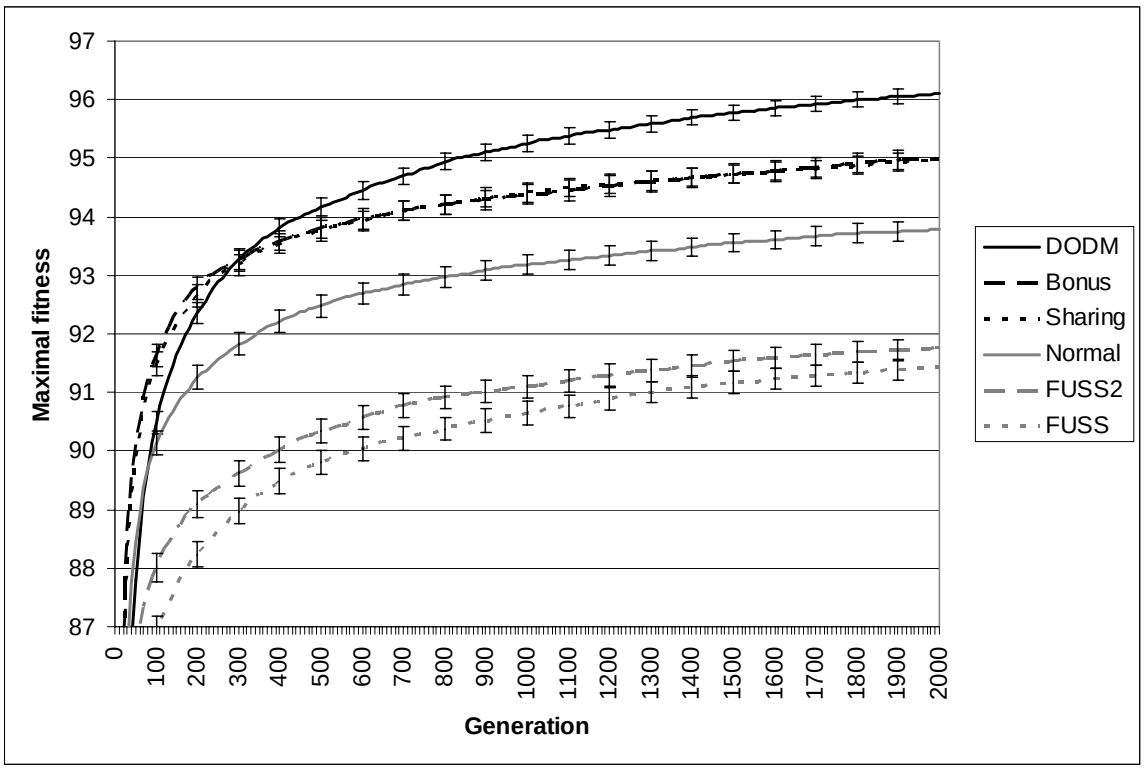


Figure 20. Performance of various genetic algorithms with respect to the NK-problem (N=60 and K=10) as a function of the number of generations with a 95% confidence interval.

Figure 22 and 23 show the maximal fitness and diversity in the population relative to the mutation rate. These figures are based on the average results of an NK landscape with an N of 60 and a K of 10 with a population of 100 individuals after 2000 generations.
The mutation rate is one of the parameter which is of most influence to the algorithm and which shows the difference between the algorithms a good way (see figure 21). If we increase the mutation rate, we can see that the best fitness found by the algorithm without frequency dependent selection increases slowly while the best fitness of the algorithms with the sharing and bonus method of frequency dependent selection is quite constant with low mutation rates. If the mutation rates become quite high we see that the algorithms with bonus and sharing frequency dependent selection find about the same best fitness as the normal genetic algorithm. The best fitness found by the double-objective algorithm starts really high, but declines only when the mutation rates increases.
Figure 22 shows that the diversity in the various genetic algorithms averaged over all the generations. In the algorithm without frequency dependent the diversity is mainly caused by mutations. The algorithms using the sharing and bonus method have an amount of diversity in the population that is maintained by the frequency dependent selection. The diversity in the double-objective algorithm is always high.

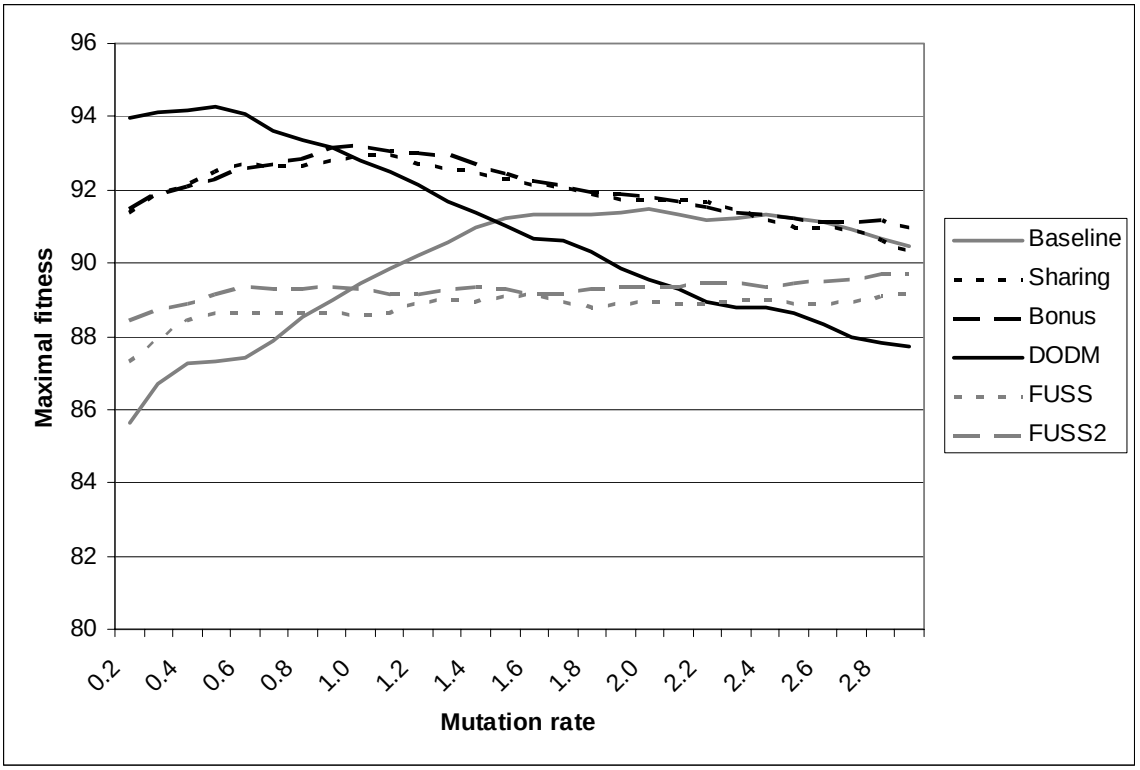


Figure 21. Effects of the mutation rate μ on the maximal fitness achieved after 2000 generations.

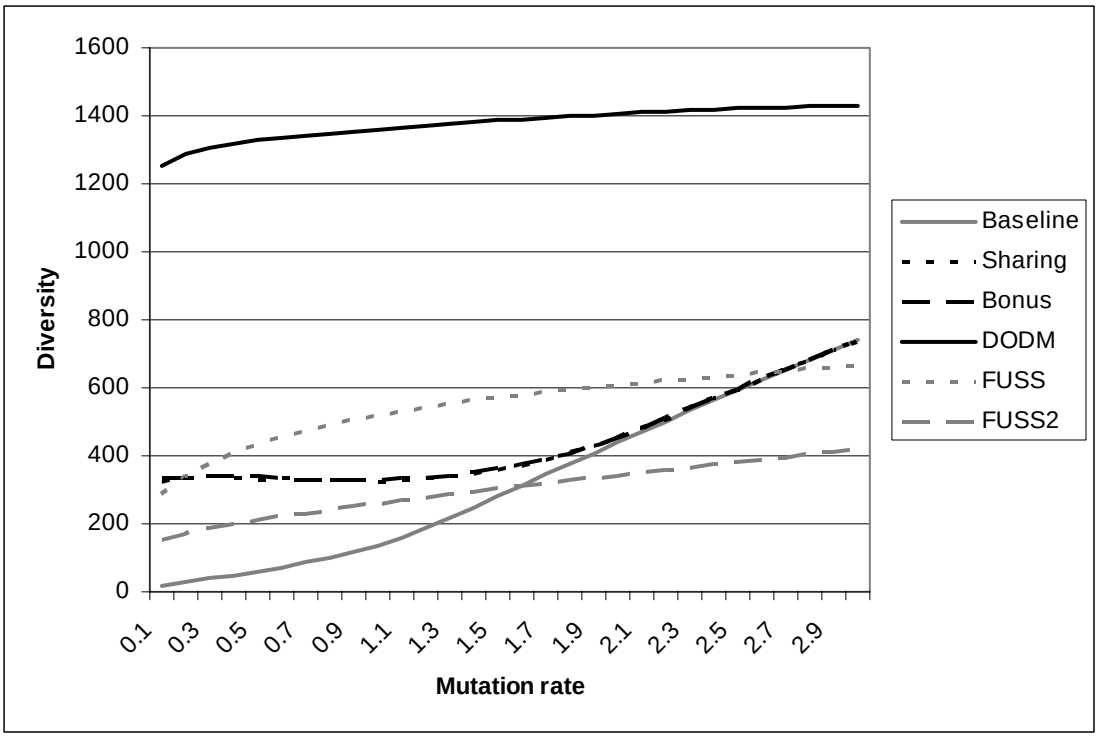


Figure 22. The diversity in the population of the algorithms compared with the mutation rate.

6. Sexual selection

6.1 Theory behind sexual selection

The twofold cost of sex is a principle from the evolutionary biology that states that reproducing asexually has a twofold fitness advantage above reproducing sexually [26]. Raising the offspring in most of the sexual organisms is mainly done by the females. The males do not contribute in the raising of the offspring. The offspring of the females consists only for the half of the genetic information of the female. If the female could reproduce asexually its offspring would completely exist of the genes of the female. In this way reproducing asexually has a twofold fitness advantage. Because most successful, complex and numerous species are sexually-reproducing animals and flowering plants there must be advantages of reproducing sexually that are at least equally strong as the costs.
Agrawal and Siller [1, 24] state that one of the advantages of reproducing sexually is that in the male population there can be a greater variance in the fitness caused by a greater mutational load than in the female population. Because the parental care of the males is very low the selection pressure of the male population can be much stronger than on the female population.
Sexual selection in a genetic algorithm is described by Todd and Miller as a potential source of boosting the performance of a genetic algorithm [21, 27]. They describe sexual selection as a very promising method to improve the performance of a genetic algorithm without any experiments proving their theory. Sánchez-Velazco and Bullinaria [22, 23] have done some experiments with sexual selection where some results are positive but their methods are not biologically plausible and seems to be specific for one problem.
In this section the theory of Agrawal and Siller incorporated in a genetic algorithm is tested. Thereafter a different biological theory based on the principle of strong selection without promising species survival is tested.

6.2 Sexual selection in a genetic algorithm

To incorporate sexual selection into the genetic algorithm the population is divided into a male and female population were the selection pressure and mutation pressure will be different for both genders. For this I make a few changes to the outline of the baseline genetic algorithm. The population is divided into two subpopulations. One population is male and the other population is female. For reproduction, the female is chosen by a best of 2 tournament selection and the male is chosen by a best of 6 tournament selection. They will get 2 offspring, one male and the other female. The male population will have a mutation probability of $\mu \times (2 - \tau)$ and the female population will have a mutation probability of $\mu \times \tau$. As we take a τ that is smaller then 1, we have a stronger selection on the male population and a stronger mutational load on the male population. In this way, females can select males with a profitable mutation and can reject males with a deleterious mutation.
The outline of this genetic algorithm with sexual selection is:

- Initialize a random population with population size $N_{population}$ where half of the individuals are female and the other half are male.
- For G generation do:
  - Calculate the fitness of each individual.

  - Copy the fittest individual to the female population of the new generation
  - Choose a female by a best of 2 tournament selection and a male by best of 6 tournament selection. Create from this male and female two offspring, one male and one female for the new generation by sexual reproduction.
  - Mutate each male offspring with a probability of μ x (2 - τ).
  - Mutate each female offspring with a probability of μ x τ.
- End loop.
- Return the fittest individual as the best solution.

6.3 Results of sexual selection on the NK landscape.

I tested the sexual selection genetic algorithm 500 times on a NK landscapes with an N of 60 and a K of 10. For the genetic algorithm a μ of 1.90, a τ of 0.50 and a population of 100 individuals is used. Figure 23 shows that the algorithm with sexual selection performs significantly a bit better.

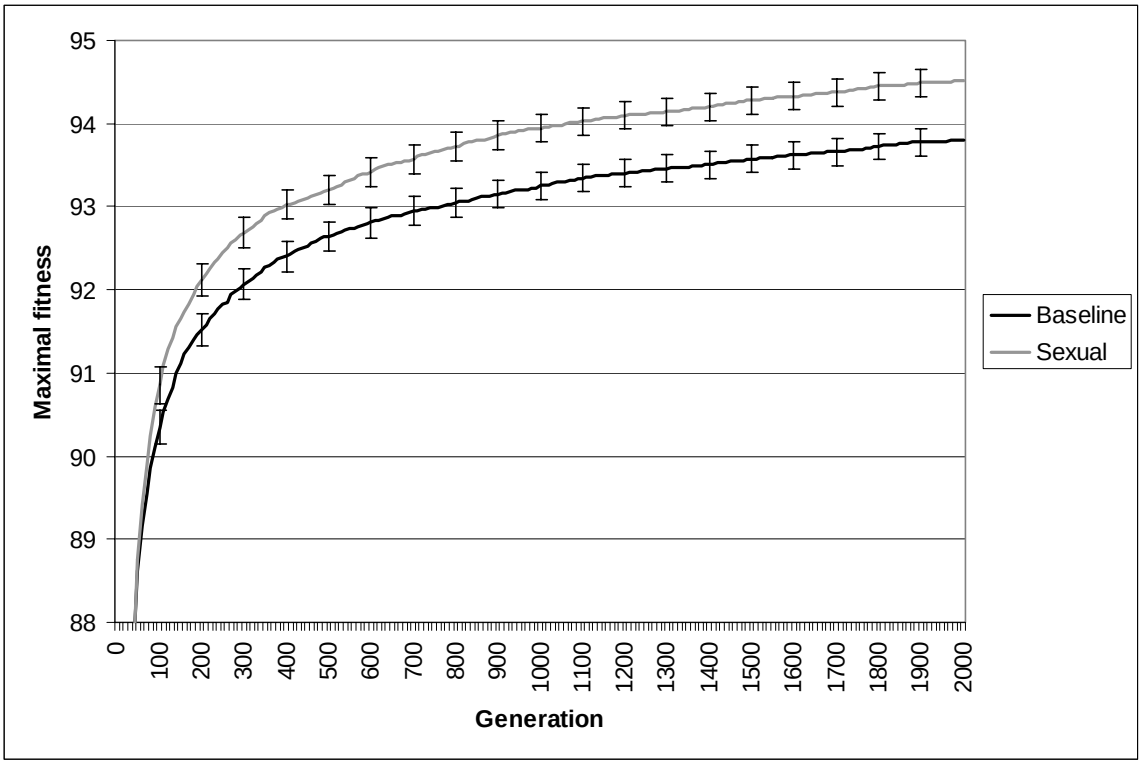


Figure 23. Performance of the baseline and the sexual genetic algorithms with respect to the NK-problem (N=60 and K=10) as a function of the number of generations with a 95% confidence interval.

6.4 Selective arena

A different theory based on the same principle of strong selection is the theory of a selective arena [25]. This theory explains why in nature a lot of species have a very strong selection in the first days of the existence of a fertilized egg. The house mice for example will have a lot of fertilized eggs in the uterus of which only a small fraction will survive the very first days. This first selection is a very quick and dirty selection. The investment is low and errors can be made. I try to mimic this process in a genetic algorithm. When two individuals are chosen to create an offspring, 5 offspring are made. These 5 offspring are evaluated by a fitness function that evaluates only a small part of the whole individual. The individual which scores best on this fitness function is the winner and will be placed in the next generation.
I tested this selective arena on 100 NK landscapes with an N of 100 and a K of 10. The quick and dirty fitness function looks at the fitness of a randomly chosen string of 20 connected bits. I use a mutation rate μ of 1.90 and a population of 50 individuals. Figure 24 shows that the algorithm with the selective arena performs a little better then the baseline algorithm, but not significantly.

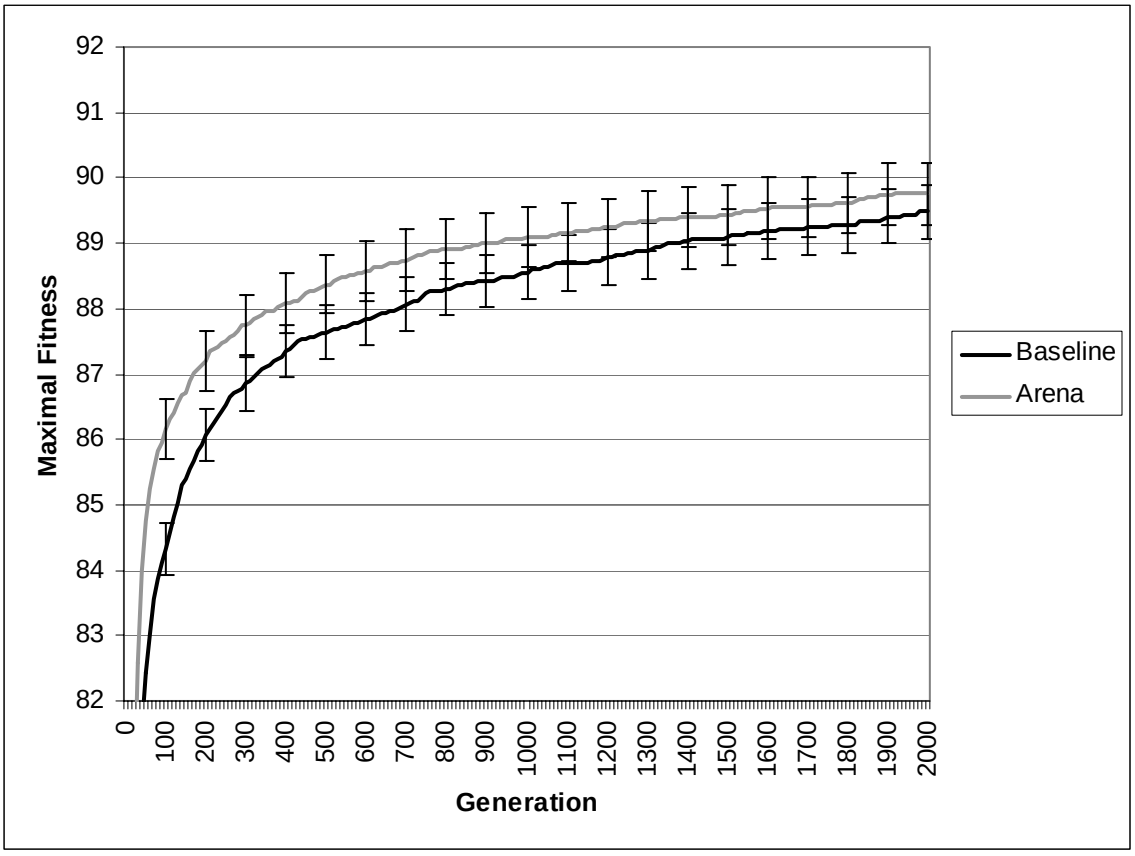


Figure 24. Maximal fitness and generation averages of 100 runs with 95% confidence intervals of the baseline algorithm and a genetic algorithm with selective arena tested on a NK problem with N = 100 and K = 10.

7. Discussion

7.1 Differences between biological evolution and genetic algorithms

As I tried to incorporate methods found in nature to increase the performance in a genetic algorithm I found that a lot of the methods are not successful in genetic algorithms as they would be in nature, especially for the sexual selection and the selective arena. This is the consequence of some major differences between evolution in nature and evolution in a genetic algorithm. I will here mention some of the differences and some of their consequences.

A big difference between evolution in nature and in genetic algorithms is that in genetic algorithms the best solution found so far can be copied unchanged to the new generation. This elitism operation is very important for the performance of a genetic algorithm. Because of the elitism operator the best fitness in a population can only increase over time. This elitism operator implies a relation between the fitness and the mutation rate. In nature mutations are independent of the fitness of an organism. In genetic algorithms using the elitism operator, the evolutionary principle of Muller's ratchet (in a small and asexual population, the number of deleterious mutations can only accumulate [26]) does not apply.

In a population in a genetic algorithm there is no chance of extinction. One of the results from this fact is that the selection pressure can be much higher in a genetic algorithm than in nature. Therefore the methods in which nature tries to achieve a higher selection pressure without compromising species survival are not efficient in a genetic algorithm.

Another result of the fact that in a genetic algorithm there is no chance of extinction is that it is not needed to produce offspring of which most are able to reproduce. Therefore a high average fitness in the population is not needed. The only result that counts in a genetic algorithm is the final result of the fittest individual in the population. In biology, when most of the offspring are not able to reproduce the population size can decline and a species can become extinct. In this way, the average fitness in the population is very important. It should be high enough so enough individuals are capable of producing enough offspring that are capable of producing healthy offspring themselves.

This difference results in the fact that in a genetic algorithm there can be much more experimenting with individuals, this experimenting will usually result in individuals with a low fitness, however this does not matter because these individuals do not have to contribute in producing offspring for the new generation. If this experimenting results in an individual with a higher fitness there is a lot of profit for the genetic algorithm. This experimenting with individuals in the genetic algorithm is done by a high mutation rate and the creation of offspring by crossovers which is quite rare in nature.

Another difference is the profitability of mutation. Mutations in biological evolution usually destroy the working of a protein and are in this way very deleterious. Only a very small fraction of the mutations are profitable. The probability that a mutation in a genetic algorithm is profitable is higher than in biological evolution. This difference can be caused by the fact that evolution has already evolved good solutions and the better the solution, the bigger the probability that a mutation is deleterious.

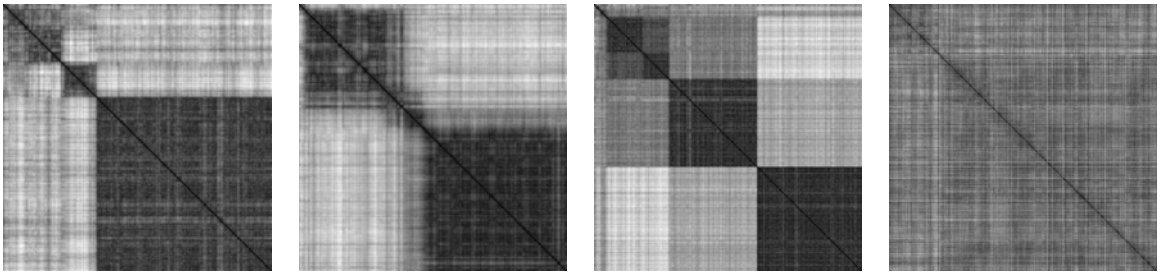
Because of these two differences the mutation rate in genetic algorithms can be much higher than in biological evolution. The probability that a mutation is profitable is higher and a deleterious mutation is not serious because the deleterious mutants can easily be replaced in the new generation by another individual.

The fitness of an individual in nature depends on the ever-changing environment. The fitness function in a normal genetic algorithm is constant. Genetic algorithms using

coevolution use a changing fitness function and do often perform better than normal genetic algorithms.

7.2 Conclusion

From the results we can conclude that the double-objective diversity maintenance method is most effective to maintain diversity and to find the best solution. The other frequency-dependent selection methods and the control algorithm perform worse. The double-objective diversity maintenance algorithm does use a little more computation time than the normal genetic algorithm. The improvements in performance however is big enough to neglect this small increase of the computation time, especially when the evaluation of the fitness function does cost a lot of computing power.
The question to understand the process better is of course: Why does the double-objective diversity maintenance method performs best? In a normal algorithm the population converges to the best solution. When a local fitness peak is reached the algorithm will search in the neighbourhood of this peak for new solutions. The double-objective diversity maintenance genetic algorithm not only searches around the local fitness peak which it has found, but searches also for individuals that are as different from the rest of the population as possible. Because, being different from the rest of the population changes over time, a big part of the search space is covered. Covering a big part of the search space is possible because of the big genetic variation in the algorithm.



Baseline Bonus FUSS DODM

Figure 25. Matrices of differences between the average individual of each generation: The difference between the average individuals of a generation compared to other generations in a single run of a genetic algorithm with 400 generations. White is a lot of difference, black is no difference. In the baseline, bonus and FUSS algorithm differences between generations that are not far apart (near the diagonal) are often small and there are a few jumps in the composition of the generation. After each jump the algorithm must have found a new fitness optimum. At the figure of the DODM algorithm nearly all average individuals between generations are different form each other, where in the other algorithms mainly the differences between more then one generation are very different from each other.

Double-objective diversity maintenance performs not only best of all; it also performs best using the lowest mutation rates. Considering a low mutation rate, finding new solutions with a high fitness is probably caused for a main part by the recombination of individuals. In this way the potential of the genetic differences in the population is used for finding the best solution.
The normal genetic algorithm finds its best solution when there is a high mutation rate. I think that the good solutions found in a normal genetic algorithm are for a great deal constructed by mutation of the best solution and only for a small part by recombination of fit individuals.

The recombinations of double-objective diversity maintenance seem to be far more effective than the mutations of the normal genetic algorithm. The mutations will be less effective at difficult problems where there is a gap between a local fitness optimum and a better fitness optimum which is bigger than one or two mutation steps. When applying the algorithms on easier problems the normal algorithm performs better than the double-objective diversity maintenance algorithm. Most problems for genetic algorithms are difficult and complex problems so for most problems it seems to be more effective to use a double-objective frequency dependent genetic algorithm instead of a normal genetic algorithm.
Finding a fitter individual in a population of fit individuals by mutation is very hard because in a fit population nearly all mutations are maladaptive, like in nature. The probability of creating a fitter individual by recombination of two fit individuals is higher. This recombination is done more often successfully in the double-objective diversity maintenance algorithm than in the normal algorithm.
From our analyses of the main difference between evolution in genetic algorithm and in nature we would expect the sexual selection model not to perform better than the normal genetic algorithm. The experiments show however that the genetic algorithm using the model of sexual selection does perform better than the normal genetic algorithm. Experiments combining sexual selection and a frequency dependent selection method do not show improved performance compared to algorithms using only the frequency dependent selection method. Considering the differences between evolution in nature and in genetic algorithms the main part of the theory behind the efficiency of sexual selection in genetic algorithms can be rejected.

7.3 Further research

I have concluded that on the test problems a genetic algorithm using double-objective diversity maintenance method of frequency dependent selection is much more efficient than the normal genetic algorithm. Now the question is whether this method is also useful if it is used on real problems which have to be solved by a genetic algorithm.
Another question is how double-objective diversity maintenance performs in combination with other improvements on the basic genetic algorithm. Coevolution for example can improve the performance of the genetic algorithm. Coevolution in combination with the double-objective diversity maintenance could perhaps boost the performance of a genetic algorithm even more.

References

[1] Agrawal, A. J. (2001): Sexual selection and the maintenance of sexual reproduction. *Nature* 411, 692-695.

[2] Allenson, R. (1992): Genetic algorithms with gender for multi-function optimisation, Technical report EPCC–SS92–01, Edinburgh Parallel Computing Centre

[3] Andersson, M. (1994): *Sexual Selection*. Princeton University.

[4] Calabretta, R. Ferdinando, A. D. Wagner, G. P. and Parisi, D. (2003): What does it take to evolve behaviorally complex organisms*? Biosystems.* 69:245-62

[5] Coley, D. A. (1999): *An Introduction to Genetic Algorithms for Scientists and Engineers*. Singapore: World Scientific.

[6] Darwin, C. (1859): *The Origin of Species*. 1st Ed. London: John Murray.

[7] Darwin, C. (1871): *The Descent of Man, and Selection in Relation to Sex.* London: John Murray.

[8] Dieckmann, U. Doebeli, M. Metz J. A. J. and Tautz, D. (2004): *Adaptive Speciation*. Cambridge University Press.

[9] Frankham, R. Ballou, J. D. and Briscoe, D. A. (2002): *Introduction to conservation genetics*. Cambridge university press.

[10] Goh, K. S. Lim, A. and Rodrigues, B. (2003): Sexual selection for genetic algorithms, *Artificial Intelligence Review* 19**:** 123–152

[11] Goldberg, D. E. (1989): *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading Massachusetts, Addison-Wesley.

[12] Goldberg, D. E. and Richardson, J. (1987): Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette (Ed.), *Genetic Algorithms and Their Applications: Proc. of the 2nd Int. Conf. on Genetic Algorithms* (41-49). Hillsdale, NJ: Lawrence Erlbaum Assoc.

[13] Grant, P. R. (1986): *Ecology and Evolution of Darwin's Finches*. New Jersey: Princeton University Press.

[14] Holland, J. H. (1975): *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press.

[15] Hutter, M. (2002): Fitness uniform selection to preserve genetic diversity. In X. Yao, editor, *Proceedings of the 2002 Congress on Evolutionary Computation (CEC-2002)*: 783-788.

[16] De Jong, E. D. Watson, R. A. and Pollack, J. B (2001): Reducing bloat and promoting diversity using multi-objective methods. In L. Spector et al., editors*, Proceedings of the Genetic and Evolutionary Computation Conference:* 11-18.

[17] De Jong, K. (1975): *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan.

[18] Kauffman, S.A. (1993): *The Origins of Order. Self-Organization and Selection in Evolution.* Oxford University Press, New York.

[19] Legg, S. Hutter, M. Kumar, A. (2004): Tournament versus fitness uniform selection. *Technical Report IDSIA-04-04.*

[20] Martin, W. N. Lienig, J. and Cohoon, J. P. (2000). Island (migration) models: evolutionary algorithms based on punctuated equilibria. *Evolutionary Computation 2*, chapter 15. Institute of Physics Publishing, Bristol, UK.

[21] Miller, G. F. and Todd, P. M. (1995): The role of mate choice in biocomputation: Sexual selection as a process of search, optimization, and diversification, *Evolution and Biocomputation, Computational Models of Evolution*, 169-204

[22] Sánchez-Velazco, J. and Bullinaria, J.A. (2003): Sexual selection with competitive/co-operative operators for genetic algorithms. *Proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence* 191-196. IASTED/ACTA Press.

[23] Sánchez-Velazco, J. and Bullinaria, J.A. (2003): Gendered selection strategies in genetic algorithms for optimization. In: J.M. Rossiter & T.P. Martin (Eds), *Proceedings of the UK Workshop on Computational Intelligence* 217-223. Bristol, UK: University of Bristol.

[24] Siller, S. (2001): Sexual selection and the maintenance of sex. *Nature* 411, 689-692

[25] Stearns, S. C. (1987): The selection-arena hypothesis. *Experientia Supplementum* 55:237-49

[26] Stearns, S. C. and Hoekstra, R. F. (2000): *Evolution an introduction*. First edition, Oxford university press.

[27] Todd, P. M. and Miller, G. F. (1997): Biodiversity through sexual selection. In: C.G. Langton & K. Shimohara (Eds.): *Artificial Life V*, 289 - 299. Cambr. Mass.:The MIT Press.

[28] Watson, R. A. (2001): Analysis of recombinative algorithms on a non-separable building-block problem. In: Worthy N. M., Spears, W. M. and Kaufmann, M. (Eds), *Foundations of Genetic Algorithms, Volume 6*. Proceedings of FOGA VI, Charlottesville, VA

[29] Welch, J. J. and Waxman, D. (2005): The nk model and population genetics. *Journal of Theoretical Biology*, 234, 329-340.

[30] Yuret, D. (1994): From genetic algorithms to efficient optimization. *A.I. Technical report No. 1569.* Massachusetts institute of technology, Boston.

[31] Zar, J. H. (1999): *Biostatistical Analysis*. Fourth edition, New Jersey: Prentice Hall.