GEATbx Introduction

Evolutionary Algorithms:
Overview, Methods and Operators

Hartmut Pohlheim

Documentation for:
Genetic and Evolutionary Algorithm Toolbox for use with Matlab
version: toolbox 3.3
documentation 3.3 (November 2001)

WWW: http://www.geatbx.com/
Email: support@geatbx.com
## Contents

1. **Introduction** ..................................................................................................... 1

2. **Overview** ........................................................................................................... 3
   - 2.1 Selection ........................................................................................................... 5
   - 2.2 Recombination ................................................................................................. 5
   - 2.3 Mutation .......................................................................................................... 6
   - 2.4 Reinsertion ...................................................................................................... 6
   - 2.5 Population models - parallel implementation of evolutionary algorithms ......... 6

3. **Selection** ............................................................................................................ 9
   - 3.1 Rank-based fitness assignment ........................................................................ 9
   - 3.2 Roulette wheel selection .................................................................................. 12
   - 3.3 Stochastic universal sampling .......................................................................... 13
   - 3.4 Local selection ................................................................................................ 13
   - 3.5 Truncation selection ........................................................................................ 15
   - 3.6 Tournament selection ...................................................................................... 16
   - 3.7 Comparison of selection schemes .................................................................... 17

4. **Recombination** ................................................................................................. 21
   - 4.1 All representations - Discrete recombination ............................................... 21
   - 4.2 Real valued recombination ............................................................................. 22
     - 4.2.1 Intermediate recombination ....................................................................... 22
     - 4.2.2 Line recombination .................................................................................... 23
     - 4.2.3 Extended line recombination ................................................................. 24
   - 4.3 Binary valued recombination (crossover) ..................................................... 25
     - 4.3.1 Single-point / double point / multi-point crossover .................................. 25
     - 4.3.2 Uniform crossover .................................................................................... 27
     - 4.3.3 Shuffle crossover ...................................................................................... 28
     - 4.3.4 Crossover with reduced surrogate .......................................................... 28

5. **Mutation** ........................................................................................................... 29
   - 5.1 Real valued mutation ...................................................................................... 29
   - 5.2 Binary mutation ................................................................................................ 31
5.3 Real valued mutation with adaptation of step-sizes ..................................................... 31

6 Reinsertion .................................................................................................................. 35
   6.1 Global reinsertion .................................................................................................... 35
   6.2 Local reinsertion .................................................................................................... 36

7 Population models - Parallel implementations .................................................. 37
   7.1 Global model - worker/farmer ............................................................................ 37
   7.2 Local model - Diffusion model .......................................................................... 38
   7.3 Regional model - Migration .............................................................................. 39

8 Application of different strategies .............................................................. 43
   8.1 Different strategies for each subpopulation ...................................................... 43
   8.2 Competition between subpopulations ............................................................ 43

9 Reference ........................................................................................................... 45
   9.1 Evolutionary Algorithms ................................................................................... 45
   9.2 Population models and parallel EA ................................................................. 56
   9.3 Combinatorial optimization .............................................................................. 57
   9.4 Visualization ...................................................................................................... 59
   9.5 Polyploidy and Evolutionary Algorithms ....................................................... 60
   9.6 Biology, Genetics and Population genetics ..................................................... 61

List of Figures ...................................................................................................... 63

List of Tables ...................................................................................................... 64
1 Introduction

Fig. 1-1: Problem solution using evolutionary algorithms

Different main schools of evolutionary algorithms have evolved during the last 30 years: genetic algorithms, mainly developed in the USA by J. H. Holland [Hol75]; evolutionary strategies, developed in Germany by I. Rechenberg [Rec73] and H.-P. Schwefel [Sch81] and evolutionary programming [FOW66]. Each of these constitutes a different approach, however, they are inspired by the same principles of natural evolution. A good introductory survey can be found in [Fdb94a].

This document describes algorithms of evolutionary algorithms. In Chapter 2 a short overview of the structure and basic algorithms of evolutionary algorithms is given. Chapter 3 describes selection. In Chapter 4 the different recombination algorithms are presented. Chapter 5 explains mutation and Chapter 6 reinsertion. Chapter 7 covers parallel implementations of evolutionary algorithms especially the regional population model employing migration in detail. The application of multiple/different strategies during an optimization including competition between subpopulations is discussed in Chapter 8. Chapter 9 lists all the used references and a large number of other publications from the field of Evolutionary Algorithms.
2 Overview

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution. Evolutionary algorithms operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

Evolutionary algorithms model natural processes, such as selection, recombination, mutation, migration, locality and neighborhood. Figure 2-1 shows the structure of a simple evolutionary algorithm. Evolutionary algorithms work on populations of individuals instead of single solutions. In this way the search is performed in a parallel manner.

Fig. 2-1: Structure of a single population evolutionary algorithm

At the beginning of the computation a number of individuals (the population) are randomly initialized. The objective function is then evaluated for these individuals. The first/initial generation is produced.

If the optimization criteria are not met the creation of a new generation starts. Individuals are selected according to their fitness for the production of offspring. Parents are recombined to produce offspring. All offspring will be mutated with a certain probability. The fitness of the offspring is then computed. The offspring are inserted into the population replacing the parents, producing a new generation. This cycle is performed until the optimization criteria are reached.

Such a single population evolutionary algorithm is powerful and performs well on a wide variety of problems. However, better results can be obtained by introducing multiple subpopulations. Every subpopulation evolves over a few generations isolated (like the single population
evolutionary algorithm) before one or more individuals are exchanged between the subpopulation. The multi-population evolutionary algorithm models the evolution of a species in a way more similar to nature than the single population evolutionary algorithm. Figure 2-2 shows the structure of such an extended multi-population evolutionary algorithm.

Fig. 2-2: Structure of an extended multipopulation evolutionary algorithm

From the above discussion, it can be seen that evolutionary algorithms differ substantially from more traditional search and optimization methods. The most significant differences are:

- Evolutionary algorithms search a population of points in parallel, not just a single point.
- Evolutionary algorithms do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.
- Evolutionary algorithms use probabilistic transition rules, not deterministic ones.
- Evolutionary algorithms are generally more straightforward to apply, because no restrictions for the definition of the objective function exist.
- Evolutionary algorithms can provide a number of potential solutions to a given problem. The final choice is left to the user. (Thus, in cases where the particular problem does not have one individual solution, for example a family of pareto-optimal solutions, as in the case of multi-objective optimization and scheduling problems, then the evolutionary algorithm is potentially useful for identifying these alternative solutions simultaneously.)

The following sections list some methods and operators of the main parts of Evolutionary Algorithms. A thorough explanation of the operators will be given in the following chapters.


2.1 Selection

Selection determines, which individuals are chosen for mating (recombination) and how many offspring each selected individual produces. The first step is fitness assignment by:

- proportional fitness assignment or
- rank-based fitness assignment, see Section 3.1

The actual selection is performed in the next step. Parents are selected according to their fitness by means of one of the following algorithms:

- roulette-wheel selection, see Section 3.2,
- stochastic universal sampling, see Section 3.3,
- local selection, see Section 3.4,
- truncation selection, see Section 3.5 or
- tournament selection, see Section 3.6.

For more info see Chapter 3.

2.2 Recombination

Recombination produces new individuals in combining the information contained in the parents (parents - mating population). Depending on the representation of the variables of the individuals the following algorithms can be applied:

- All presentation:
  - discrete recombination, see Subsection 4.1 (known from recombination of real valued variables), corresponds with uniform crossover, see Subsection 4.3.2 (known from recombination of binary valued variables),
- Real valued recombination, see Section 4.2:
  - intermediate recombination, see Subsection 4.2.1,
  - line recombination, see Subsection 4.2.2,
  - extended line recombination, see Subsection 4.2.3,
- Binary valued recombination, see Section 4.3:
  - single-point / double-point /multi-point crossover, see Subsection 4.3.1,
  - uniform crossover, see Subsection 4.3.2,
  - shuffle crossover, see Subsection 4.3.3,
  - crossover with reduced surrogate, see Subsection 4.3.4.

For more info see Chapter 4.

---

1 For the recombination of binary valued variables the name 'crossover' is established. This has mainly historical reasons. Genetic algorithms mostly used binary variables and the name 'crossover'. Both notions (recombination and crossover) are equivalent in the area of Evolutionary Algorithms. For consistency, throughout this study the notion 'recombination' will be used (except when referring to specially named methods or operators).
2.3 Mutation

After recombination every offspring undergoes mutation. Offspring variables are mutated by small perturbations (size of the mutation step), with low probability. The representation of the variables determines the used algorithm. Two operators are explained:

- mutation operator for real valued variables, see Section 5.1,
- mutation for binary valued variables, see Section 5.2.

For more info see Chapter 5.

2.4 Reinsertion

After producing offspring they must be inserted into the population. This is especially important, if less offspring are produced than the size of the original population. Another case is, when not all offspring are to be used at each generation or if more offspring are generated than needed. By a reinsertion scheme is determined which individuals should be inserted into the new population and which individuals of the population will be replaced by offspring.

The used selection algorithm determines the reinsertion scheme:

- global reinsertion for all population based selection algorithm (roulette-wheel selection, stochastic universal sampling, truncation selection),
- local reinsertion for local selection.

For more info see Chapter 6.

2.5 Population models - parallel implementation of evolutionary algorithms

The extended management of populations (population models) allows the definition of extensions of Evolutionary Algorithms. These extensions can contribute to an increased performance of Evolutionary Algorithms.

The following extensions can be distinguished:

- global model, see Section 7.1,
- local model (diffusion model, neighborhood model, fine grained model), see Section 7.2,
- regional model (migration model, island model, coarse grained model), see Section 7.3.

For more info see Chapter 7.

2.6 Application of multiple/different strategies and competition between subpopulations

Based on the regional population model the application of multiple different strategies at the same time is possible. This is done by applying different operators and parameters for each subpopulation. For an efficient distribution of resources during an optimization competing subpopulations are used.
• application of multiple strategies, see Section 8.1.
• competition between subpopulations, see Section 8.2.

These extensions of the regional population model contribute to an increased performance of Evolutionary Algorithms, especially for large and complex real-world applications.

For more info see Chapter 8.
3 Selection

In selection the offspring producing individuals are chosen. The first step is fitness assignment. Each individual in the selection pool receives a reproduction probability depending on the own objective value and the objective value of all other individuals in the selection pool. This fitness is used for the actual selection step afterwards.

Throughout this section some terms are used for comparing the different selection schemes. The definition of this terms follows [Bak87] and [BT95].

selective pressure
- probability of the best individual being selected compared to the average probability of selection of all individuals

bias:
- absolute difference between an individual's normalized fitness and its expected probability of reproduction

spread
- range of possible values for the number of offspring of an individual

loss of diversity
- proportion of individuals of a population that is not selected during the selection phase

selection intensity
- expected average fitness value of the population after applying a selection method to the normalized Gaussian distribution

selection variance
- expected variance of the fitness distribution of the population after applying a selection method to the normalized Gaussian distribution

3.1 Rank-based fitness assignment

In rank-based fitness assignment, the population is sorted according to the objective values. The fitness assigned to each individual depends only on its position in the individuals rank and not on the actual objective value.

Rank-based fitness assignment overcomes the scaling problems of the proportional fitness assignment. (Stagnation in the case where the selective pressure is too small or premature convergence where selection has caused the search to narrow down too quickly.) The reproductive range is limited, so that no individuals generate an excessive number of offspring. Ranking introduces a uniform scaling across the population and provides a simple and effective way of controlling selective pressure.

Rank-based fitness assignment behaves in a more robust manner than proportional fitness assignment and, thus, is the method of choice. [BH91], [Why89].
Consider $Nind$ the number of individuals in the population, $Pos$ the position of an individual in this population (least fit individual has $Pos=1$, the fittest individual $Pos=Nind$) and $SP$ the selective pressure. The fitness value for an individual is calculated as:

**Linear ranking:**

$$
\text{Fitness}(Pos) = 2 - SP + 2 \cdot (SP - 1) \cdot \frac{(Pos - 1)}{(Nind - 1)}
$$

(3-1)

Linear ranking allows values of selective pressure in $[1.0, 2.0]$.

A new method for ranking using a non-linear distribution is introduced. The use of non-linear ranking permits higher selective pressures than the linear ranking method.

**Non-linear ranking:**

$$
\text{Fitness}(Pos) = \frac{Nind \cdot X^{Pos-1}}{\sum_{i=1}^{Nind} X^{i-1}}
$$

(3-2)

$X$ is computed as the root of the polynomial:

$$
0 = (SP - 1) \cdot X^{Nind-1} + SP \cdot X^{Nind-2} + \ldots + SP \cdot X + SP
$$

(3-3)

Non-linear ranking allows values of selective pressure in $[1, Nind - 2]$.

Figure 3-1 compares linear and non-linear ranking graphically.

The probability of each individual being selected for mating depends on its fitness normalized by the total fitness of the population.

Table 3-1 contains the fitness values of the individuals for various values of the selective pressure assuming a population of 11 individuals and a minimization problem.
Table 3-1: Dependency of fitness value from selective pressure

<table>
<thead>
<tr>
<th>objective value</th>
<th>fitness value (parameter: selective pressure)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>linear ranking</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td>1</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>1.8</td>
</tr>
<tr>
<td>4</td>
<td>1.6</td>
</tr>
<tr>
<td>7</td>
<td>1.4</td>
</tr>
<tr>
<td>8</td>
<td>1.2</td>
</tr>
<tr>
<td>9</td>
<td>1.0</td>
</tr>
<tr>
<td>10</td>
<td>0.8</td>
</tr>
<tr>
<td>15</td>
<td>0.6</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
</tr>
<tr>
<td>30</td>
<td>0.2</td>
</tr>
<tr>
<td>95</td>
<td>0.0</td>
</tr>
</tbody>
</table>

In [BT95] an analysis of linear ranking selection can be found.

Fig. 3-2: Properties of linear ranking

Selection intensity

\[
SelInt_{LinRank}(SP) = (SP - 1) \cdot \frac{1}{\sqrt{\pi}}
\]  

(3-4)

Loss of diversity

\[
LossDiv_{LinRank}(SP) = \frac{(SP - 1)}{4}
\]  

(3-5)
Selection variance

\[ SelVar_{LinRank}(SP) = 1 - \frac{(SP - 1)^2}{\pi} = 1 - SelInt_{LinRank}(SP) \]  

(3-6)

### 3.2 Roulette wheel selection

The simplest selection scheme is roulette-wheel selection, also called stochastic sampling with replacement [Bak87]. This is a stochastic algorithm and involves the following technique:

The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness. A random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals is obtained (called mating population). This technique is analogous to a roulette wheel with each slice proportional in size to the fitness, see figure 3-3.

Table 3-2 shows the selection probability for 11 individuals, linear ranking and selective pressure of 2 together with the fitness value. Individual 1 is the most fit individual and occupies the largest interval, whereas individual 10 as the second least fit individual has the smallest interval on the line (see figure 3-3). Individual 11, the least fit interval, has a fitness value of 0 and get no chance for reproduction.

<table>
<thead>
<tr>
<th>Number of individual</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>fitness value</td>
<td>2.0</td>
<td>1.8</td>
<td>1.6</td>
<td>1.4</td>
<td>1.2</td>
<td>1.0</td>
<td>0.8</td>
<td>0.6</td>
<td>0.4</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>selection probability</td>
<td>0.18</td>
<td>0.16</td>
<td>0.15</td>
<td>0.13</td>
<td>0.11</td>
<td>0.09</td>
<td>0.07</td>
<td>0.06</td>
<td>0.03</td>
<td>0.02</td>
<td>0.0</td>
</tr>
</tbody>
</table>

For selecting the mating population the appropriate number of uniformly distributed random numbers (uniformly distributed between 0.0 and 1.0) is independently generated.

Sample of 6 random numbers:

0.81, 0.32, 0.96, 0.01, 0.65, 0.42.

Figure 3-3 shows the selection process of the individuals for the example in table 3-2 together with the above sample trials.

Fig. 3-3: Roulette-wheel selection
After selection the mating population consists of the individuals:

1, 2, 3, 5, 6, 9.

The roulette-wheel selection algorithm provides a zero bias but does not guarantee minimum spread.

### 3.3 Stochastic universal sampling

Stochastic universal sampling [Bak87] provides zero bias and minimum spread. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here equally spaced pointers are placed over the line as many as there are individuals to be selected. Consider $N_{\text{Pointer}}$ the number of individuals to be selected, then the distance between the pointers are $1/N_{\text{Pointer}}$ and the position of the first pointer is given by a randomly generated number in the range $[0, 1/N_{\text{Pointer}}]$.

For 6 individuals to be selected, the distance between the pointers is $1/6=0.167$. Figure 3-4 shows the selection for the above example.

![Stochastic universal sampling](image)

Fig. 3-4: Stochastic universal sampling

After selection the mating population consists of the individuals:

1, 2, 3, 4, 6, 8.

Stochastic universal sampling ensures a selection of offspring which is closer to what is deserved then roulette wheel selection.

### 3.4 Local selection

In local selection every individual resides inside a constrained environment called the local neighborhood. (In the other selection methods the whole population or subpopulation is the selection pool or neighborhood.) Individuals interact only with individuals inside this region. The neighborhood is defined by the structure in which the population is distributed. The neighborhood can be seen as the group of potential mating partners.

The first step is the selection of the first half of the mating population uniform at random (or using one of the other mentioned selection algorithms, for example, stochastic universal sampling or truncation selection). Now a local neighborhood is defined for every selected indi-
individual. Inside this neighborhood the mating partner is selected (best, fitness proportional, or uniform at random).

The structure of the neighborhood can be:
- linear
  - full ring, half ring (see Figure 3-5)
- two-dimensional
  - full cross, half cross (see Figure 3-6, left)
  - full star, half star (see Figure 3-6, right)
- three-dimensional and more complex with any combination of the above structures.

Local selection is part of the local population model, see Section 7.2.

Fig. 3-5: Linear neighborhood: full and half ring

The distance between possible neighbors together with the structure determines the size of the neighborhood. Table 3-3 gives examples for the size of the neighborhood for the given structures and different distance values.

Fig. 3-6: Two-dimensional neighbourhood; left: full and half cross, right: full and half star

Between individuals of a population an ‘isolation by distance’ exists. The smaller the neighborhood, the bigger the isolation distance. However, because of overlapping neighborhoods, propagation of new variants takes place. This assures the exchange of information between all individuals.
The size of the neighborhood determines the speed of propagation of information between the individuals of a population, thus deciding between rapid propagation or maintenance of a high diversity/variability in the population. A higher variability is often desired, thus preventing problems such as premature convergence to a local minimum. Similar results were drawn from simulations in [VBS91]. Local selection in a small neighborhood performed better than local selection in a bigger neighborhood. Nevertheless, the interconnection of the whole population must still be provided. Two-dimensional neighborhood with structure half star using a distance of 1 is recommended for local selection. However, if the population is bigger (>100 individuals) a greater distance and/or another two-dimensional neighborhood should be used.

### Tab. 3-3: Number of neighbors for local selection

<table>
<thead>
<tr>
<th>structure of selection</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>full ring</td>
<td>2</td>
</tr>
<tr>
<td>half ring</td>
<td>1</td>
</tr>
<tr>
<td>full cross</td>
<td>4</td>
</tr>
<tr>
<td>half cross</td>
<td>2</td>
</tr>
<tr>
<td>full star</td>
<td>8</td>
</tr>
<tr>
<td>half star</td>
<td>3</td>
</tr>
</tbody>
</table>

### 3.5 Truncation selection

Compared to the previous selection methods modeling natural selection truncation selection is an artificial selection method. It is used by breeders for large populations/mass selection.

In truncation selection individuals are sorted according to their fitness. Only the best individuals are selected for parents. These selected parents produce uniform at random offspring. The parameter for truncation selection is the truncation threshold $Trunc$. $Trunc$ indicates the proportion of the population to be selected as parents and takes values ranging from 50%-10%. Individuals below the truncation threshold do not produce offspring. The term selection intensity is often used in truncation selection. Table 3-4 shows the relation between both.

### Tab. 3-4: Relation between truncation threshold and selection intensity

<table>
<thead>
<tr>
<th>truncation threshold</th>
<th>1%</th>
<th>10%</th>
<th>20%</th>
<th>40%</th>
<th>50%</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>selection intensity</td>
<td>2.66</td>
<td>1.76</td>
<td>1.2</td>
<td>0.97</td>
<td>0.8</td>
<td>0.34</td>
</tr>
</tbody>
</table>

In [BT95] an analysis of truncation selection can be found. The same results have been derived in a different way in [CK70] as well.

Selection intensity

$$SelInt_{Truncation}(Trunc) = \frac{1}{Trunc} \cdot \frac{1}{\sqrt{2 \cdot \pi}} \cdot e^{-\frac{f^2}{2}}$$  (3-7)
Loss of diversity

\[ \text{LossDiv}_{\text{Truncation}}(\text{Trunc}) = 1 - \text{Trunc} \quad (3-8) \]

Selection variance

\[ \text{SelVar}_{\text{Truncation}}(\text{Trunc}) = 1 - \text{SelInt}_{\text{Truncation}}(\text{Trunc}) \cdot (\text{SelInt}_{\text{Truncation}}(\text{Trunc}) - f_c) \quad (3-9) \]

Fig. 3-7: Properties of truncation selection

3.6 Tournament selection

In tournament selection [GD91], a number \( \text{Tour} \) of individuals is chosen randomly from the population and the best individual from this group is selected as parent. This process is repeated as often as individuals must be chosen. These selected parents produce uniform at random offspring. The parameter for tournament selection is the tournament size \( \text{Tour} \). \( \text{Tour} \) takes values ranging from 2 to \( \text{Nind} \) (number of individuals in population). Table 3-5 and figure 3-8 show the relation between tournament size and selection intensity [BT95].

<table>
<thead>
<tr>
<th>tournament size</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>selection intensity</td>
<td>0</td>
<td>0.56</td>
<td>0.85</td>
<td>1.15</td>
<td>1.53</td>
<td>2.04</td>
</tr>
</tbody>
</table>

In [BT95], an analysis of tournament selection can be found.

Selection intensity

\[ \text{SelInt}_{\text{Turnier}}(\text{Tour}) \approx \sqrt{2 \cdot \ln(\text{Tour}) - \ln(\sqrt{4.14 \cdot \ln(\text{Tour})})} \quad (3-10) \]
3.7 Comparison of selection schemes

Loss of diversity

\[ \text{LossDiv}_{\text{Turnier}}(\text{Tour}) = \text{Tour}^{-1} - \text{Tour}^{-1} \]  
(3-11)

(About 50% of the population are lost at tournament size \( \text{Tour}=5 \)).

Selection variance

\[ \text{SelVar}_{\text{Turnier}}(\text{Tour}) \approx \frac{0.918}{\ln(1.186 + 1.328 \cdot \text{Tour})} \]  
(3-12)

Fig. 3-8: Properties of tournament selection

3.7 Comparison of selection schemes

As shown in the above sections the selection methods behave similarly assuming similar selection intensity. Figure 3-9 shows the relation between selection intensity and the appropriate parameters of the selection methods (selective pressure, truncation threshold and tournament size). It should be stated that with tournament selection only discrete values can be assigned and linear ranking selection allows only a smaller range for the selection intensity.
However, the behavior of the selection methods is different. Thus, the selection methods will be compared on the parameters loss of diversity (figure 3-10) and selections variance (figure 3-11) on the selection intensity.

Truncation selection leads to a much higher loss of diversity for the same selection intensity compared to ranking and tournament selection. Truncation selection is more likely to replace less fit individuals with fitter offspring, because all individuals below a certain fitness threshold do not have a probability to be selected. Ranking and tournament selection seem to behave similarly. However, ranking selection works in an area where tournament selection does not work because of the discrete character of tournament selection.
For the same selection intensity truncation selection leads to a much smaller selection variance than ranking or tournament selection. As can be seen clearly ranking selection behaves similar to tournament selection. However, again ranking selection works in an area where tournament selection does not work because of the discrete character of tournament selection. In [BT95] was proven that the fitness distribution for ranking and tournament selection for SP=2 and Tour=2 (SelInt=1/sqrt(pi)) is identical.
4 Recombination

Recombination produces new individuals in combining the information contained in two or more parents (parents - mating population). This is done by combining the variable values of the parents. Depending on the representation of the variables different methods must be used.

Section 4.1 describes the discrete recombination. This method can be applied to all variable representations. Section 4.2 explains methods for real valued variables. Methods for binary valued variables are described in Section 4.3.

The methods for binary valued variables constitute special cases of the discrete recombination. These methods can all be applied to integer valued and real valued variables as well.

4.1 All representations - Discrete recombination

Discrete recombination [MSV93a] performs an exchange of variable values between the individuals. For each position the parent who contributes its variable to the offspring is chosen randomly with equal probability.

\[ \text{Var}_i^O = \text{Var}_i^P \cdot a_i + \text{Var}_i^P \cdot (1-a_i), i \in \{1,2,\ldots,N\text{var}\}, \]
\[ a_i \in \{0,1\} \text{uniform at random, } a_i \text{ for each } i \text{ new defined} \] (4-1)

Discrete recombination generates corners of the hypercube defined by the parents. Figure 4-1 shows the geometric effect of discrete recombination.

Fig. 4.1: Possible positions of the offspring after discrete recombination

Consider the following two individuals with 3 variables each (3 dimensions), which will also be used to illustrate the other types of recombination for real valued variables:

| individual 1 | 12 | 25 | 5 |
| individual 2 | 123 | 4 | 34 |
For each variable the parent who contributes its variable to the offspring is chosen randomly with equal probability:

<table>
<thead>
<tr>
<th>sample 1</th>
<th>sample 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

After recombination the new individuals are created:

<table>
<thead>
<tr>
<th>offspring 1</th>
<th>offspring 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Discrete recombination can be used with any kind of variables (binary, integer, real or symbols).

### 4.2 Real valued recombination

The recombination methods in this section can be applied for the recombination of individuals with real valued variables.

#### 4.2.1 Intermediate recombination

Intermediate recombination [MSV93a] is a method only applicable to real variables (and not binary variables). Here the variable values of the offspring are chosen somewhere around and between the variable values of the parents.

Offspring are produced according to the rule:

\[
    V_{\text{ar}}^i = V_{\text{ar}}^n_i \cdot a_i + V_{\text{ar}}^p_i \cdot (1 - a_i) \quad i \in (1, 2, \ldots, N_{\text{var}}),
\]

\[
    a_i \in [-d, 1 + d] \quad \text{uniform at random, } d = 0.25, a_i \text{ for each } i \text{ new}
\]

where \(a\) is a scaling factor chosen uniformly at random over an interval \([-d, 1+d]\) for each variable anew.

The value of the parameter \(d\) defines the size of the area for possible offspring. A value of \(d = 0\) defines the area for offspring the same size as the area spanned by the parents. This method is called (standard) intermediate recombination. Because most variables of the offspring are not generated on the border of the possible area, the area for the variables shrinks over the generations. This shrinkage occurs just by using (standard) intermediate recombination. This effect can be prevented by using a larger value for \(d\). A value of \(d = 0.25\) ensures (statistically), that the variable area of the offspring is the same as the variable area spanned by the variables of the parents. See figure 4-2 for a picture of the area of the variable range of the offspring defined by the variables of the parents.

### Fig. 4-2: Area for variable value of offspring compared to parents in intermediate recombination

![Area for variable value of offspring compared to parents in intermediate recombination](image)
Consider the following two individuals with 3 variables each:

individual 1  12  25  5
individual 2  123  4  34

The chosen \( a \) for this example are:

sample 1  0.5  1.1  -0.1
sample 2  0.1  0.8  0.5

The new individuals are calculated as:

offspring 1  67.5  1.9  2.1
offspring 2  23.1  8.2  19.5

Intermediate recombination is capable of producing any point within a hypercube slightly larger than that defined by the parents. Figure 4-3 shows the possible area of offspring after intermediate recombination.

![Possible area of the offspring after intermediate recombination](image)

4.2.2 Line recombination

Line recombination is similar to intermediate recombination, except that only one value of \( a \) for all variables is used. The same \( a \) is used for all variables:

\[
Var_i^o = Var_i^p \cdot a_i + Var_i^p \cdot (1 - a_i) \quad i \in (1, 2, \ldots, Nvar),
\]

\[
a_i \in [-d, 1+d] \text{ uniform at random, } d=0.25, a_i \text{ for all } i \text{ identical}
\] (4-3)

For the value of \( d \) the statements given for intermediate recombination are applicable.

Consider the following two individuals with 3 variables each:

individual 1  12  25  5
individual 2  123  4  34

The chosen Alpha for this example are:

sample 1  0.5
sample 2  0.1

The new individuals are calculated as:

offspring 1  67.5  14.5  19.5
offspring 2  23.1  22.9  7.9
Line recombination can generate any point on the line defined by the parents. Figure 4-4 shows the possible positions of the offspring after line recombination.

### 4.2.3 Extended line recombination

Extended line recombination [Müh94] generates offspring on a line defined by the variable values of the parents. However, extended line recombination is not restricted to the line between the parents and a small area outside. The parents just define the line where possible offspring may be created. The size of the area for possible offspring is defined by the domain of the variables.

Inside this possible area the offspring are not uniform at random distributed. The probability of creating offspring near the parents is high. Only with low probability offspring are created far away from the parents. If the fitness of the parents is available, then offspring are more often created in the direction from the worse to the better parent (directed extended line recombination).

Offspring are produced according to the following rule:

\[
Var_{i}^{O} = Var_{i}^{H} + s_{i} \cdot r_{i} \cdot a_{i}, \quad \frac{Var_{i}^{H} - Var_{i}^{P}}{Var_{i}^{H} - Var_{i}^{P}} i \in (1,2,\ldots,N\text{var}),
\]

\[
a_{i} = 2^{-k \cdot \mu}, \quad \mu \text{ mutation precision}, u \in [0,1] \text{uniform at random},
\]

\[
a_{i} \text{ for all } i \text{ identical},
\]

\[
r_{i} = r \cdot \text{domain}: \text{range of recombination steps},
\]

\[
s_{i} \in \{-1,1\}, \quad \text{uniform at random: undirected recombination},
\]

\[+1 \text{ with probability} > 0.5: \text{directed recombination}, \quad (4-4)\]

The creation of offspring uses features similar to the mutation operator for real valued variables (see Section 5.1). The parameter \( a \) defines the relative step-size, the parameter \( r \) the maximum step-size and the parameter \( s \) the direction of the recombination step.

Figure 4-5 tries to visualize the effect of extended line recombination.

The parameter \( k \) determines the precision used for the creation of recombination steps. A larger \( k \) produces more smaller steps. For all values of \( k \) the maximum value for \( a \) is \( a = 1 \).
(u = 0). The minimum value of a depends on k and is a = 2^k (u = 1). Typical values for the precision parameter k are in the area from 4 to 20.

Fig. 4-5: Possible positions of the offspring after extended line recombination according to the positions of the parents and the definition area of the variables

A robust value for the parameter r (range of recombination step) is 10% of the domain of the variable. However, according to the defined domain of the variables or for special cases this parameter can be adjusted. By selecting a smaller value for r the creation of offspring may be constrained to a smaller area around the parents.

If the parameter s (search direction) is set to -1 or +1 with equal probability an undirected recombination takes place. If the probability of s = +1 is higher than 0.5, a directed recombination takes place (offspring are created in the direction from the worse to the better parent - the first parent must be the better parent).

Extended line recombination is only applicable to real variables (and not binary or integer variables).

4.3 Binary valued recombination (crossover)

This section describes recombination methods for individuals with binary variables. Commonly, these methods are called 'crossover'. Thus, the notion 'crossover' will be used to name the methods.

During the recombination of binary variables only parts of the individuals are exchanged between the individuals. Depending on the number of parts, the individuals are divided before the exchange of variables (the number of cross points). The number of cross points distinguish the methods.

4.3.1 Single-point / double point / multi-point crossover

In single-point crossover one crossover position k ∈ [1, 2, ..., Nvar-1], Nvar: number of variables of an individual, is selected uniformly at random and the variables exchanged between the
individuals about this point, then two new offspring are produced. Figure 4-6 illustrates this process.

Consider the following two individuals with 11 binary variables each:

\[
\begin{align*}
\text{individual 1} & : 0 1 1 1 0 0 1 1 0 1 0 \\
\text{individual 2} & : 1 0 1 0 1 1 0 0 1 0 1
\end{align*}
\]

The chosen crossover position is:

\[
\text{crossover position} & : 5
\]

After crossover the new individuals are created:

\[
\begin{align*}
\text{offspring 1} & : 0 1 1 1 0 | 1 0 0 1 0 1 \\
\text{offspring 2} & : 1 0 1 0 1 | 0 1 1 0 1 0
\end{align*}
\]

Fig. 4-6: Single-point crossover

In double-point crossover two crossover positions are selected uniformly at random and the variables exchanged between the individuals between these points. Then two new offspring are produced.

Single-point and double-point crossover are special cases of the general method multi-point crossover.

For multi-point crossover, \(m\) crossover positions \(k_i \in [1,2,...,N\text{var}-1]\), \(i=1:m\), \(N\text{var}\): number of variables of an individual, are chosen at random with no duplicates and sorted into ascending order. Then, the variables between successive crossover points are exchanged between the two parents to produce two new offspring. The section between the first variable and the first crossover point is not exchanged between individuals. Figure 4-7 illustrates this process.

Consider the following two individuals with 11 binary variables each:

\[
\begin{align*}
\text{individual 1} & : 0 1 1 1 0 0 1 1 0 1 0 \\
\text{individual 2} & : 1 0 1 0 1 1 0 0 1 0 1
\end{align*}
\]

The chosen crossover positions are:

\[
\text{cross pos. (m=3)} & : 2 \quad 6 \quad 10
\]

After crossover the new individuals are created:

\[
\begin{align*}
\text{offspring 1} & : 0 1 | 1 0 1 1 | 0 1 1 1 | 1 \\
\text{offspring 2} & : 1 0 | 1 1 0 0 | 0 0 1 0 | 0
\end{align*}
\]
4.3 Binary valued recombination (crossover)

The idea behind multi-point, and indeed many of the variations on the crossover operator, is that parts of the chromosome representation that contribute most to the performance of a particular individual may not necessarily be contained in adjacent substrings \cite{Boo87}. Further, the disruptive nature of multi-point crossover appears to encourage the exploration of the search space, rather than favouring the convergence to highly fit individuals early in the search, thus making the search more robust \cite{SDJ91a}.

4.3.2 Uniform crossover

Single and multi-point crossover define cross points as places between loci where an individual can be split. Uniform crossover \cite{Sys89} generalizes this scheme to make every locus a potential crossover point. A crossover mask, the same length as the individual structure is created at random and the parity of the bits in the mask indicate which parent will supply the offspring with which bits. This method is identical to discrete recombination, see Section 4.1.

Consider the following two individuals with 11 binary variables each:

\begin{align*}
\text{individual 1} & \quad 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \\
\text{individual 2} & \quad 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \\
\end{align*}

For each variable the parent who contributes its variable to the offspring is chosen randomly with equal probability. Here, the offspring 1 is produced by taking the bit from parent 1 if the corresponding mask bit is 1 or the bit from parent 2 if the corresponding mask bit is 0. Offspring 2 is created using the inverse of the mask, usually.

\begin{align*}
\text{sample 1} & \quad 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \\
\text{sample 2} & \quad 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \\
\end{align*}

After crossover the new individuals are created:

\begin{align*}
\text{offspring 1} & \quad 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \\
\text{offspring 2} & \quad 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\
\end{align*}

Uniform crossover, like multi-point crossover, has been claimed to reduce the bias associated with the length of the binary representation used and the particular coding for a given parameter set. This helps to overcome the bias in single-point crossover towards short substrings without requiring precise understanding of the significance of the individual bits in the individuals representation. \cite{SDJ91a} demonstrated how uniform crossover may be parameterized by applying a probability to the swapping of bits. This extra parameter can be used to control the amount of disruption during recombination without introducing a bias towards the length of the representation used.
4.3.3 Shuffle crossover

Shuffle crossover [CES89] is related to uniform crossover. A single crossover position (as in single-point crossover) is selected. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled in reverse. This removes positional bias as the variables are randomly reassigned each time crossover is performed.

4.3.4 Crossover with reduced surrogate

The reduced surrogate operator [Boo87] constrains crossover to always produce new individuals wherever possible. This is implemented by restricting the location of crossover points such that crossover points only occur where gene values differ.
5 Mutation

By mutation individuals are randomly altered. These variations (mutation steps) are mostly small. They will be applied to the variables of the individuals with a low probability (mutation probability or mutation rate). Normally, offspring are mutated after being created by recombination.

For the definition of the mutation steps and the mutation rate two approaches exist:
- Both parameters are constant during a whole evolutionary run. Examples are methods for the mutation of real variables, see Section 5.1 and mutation of binary variables, see Section 5.2.
- One or both parameters are adapted according to previous mutations. Examples are the methods for the adaptation of mutation step-sizes known from the area of evolutionary strategies, see Section 5.3.

5.1 Real valued mutation

Mutation of real variables means, that randomly created values are added to the variables with a low probability. Thus, the probability of mutating a variable (mutation rate) and the size of the changes for each mutated variable (mutation step) must be defined.

The probability of mutating a variable is inversely proportional to the number of variables (dimensions). The more dimensions one individual has, the smaller is the mutation probability. Different papers reported results for the optimal mutation rate. [MSV93a] writes, that a mutation rate of $1/n$ ($n$: number of variables of an individual) produced good results for a wide variety of test functions. That means, that per mutation only one variable per individual is changed/mutated. Thus, the mutation rate is independent of the size of the population.

Similar results are reported in [Bäc93] and [Bäc96] for a binary valued representation. For unimodal functions a mutation rate of $1/n$ was the best choice. An increase in the mutation rate at the beginning connected with a decrease in the mutation rate to $1/n$ at the end gave only an insignificant acceleration of the search.

The given recommendations for the mutation rate are only correct for separable functions. However, most real world functions are not fully separable. For these functions no recommendations for the mutation rate can be given. As long as nothing else is known, a mutation rate of $1/n$ is suggested as well.

The size of the mutation step is usually difficult to choose. The optimal step-size depends on the problem considered and may even vary during the optimization process. It is known, that small steps (small mutation steps) are often successful, especially when the individual is already well adapted. However, larger changes (large mutation steps) can, when successful, produce good results much quicker. Thus, a good mutation operator should often produce small step-sizes with a high probability and large step-sizes with a low probability.
In [MSV93a] and [Müh94] such an operator is proposed (mutation operator of the Breeder Genetic Algorithm):

\[
Var_{i}^{Mut} = Var_{i} + s_{i} \cdot r_{i} \cdot a_{i}, \quad i \in \{1,2,\ldots,n\} \quad \text{uniform at random},
\]

\[
s_{i} \in \{-1, +1\} \quad \text{uniform at random}
\]

\[
r_{i} = r \cdot \text{domain}, \quad r: \text{mutation range} \quad (\text{standard: } 10%),
\]

\[
a_{i} = 2^{-u \cdot k}, \quad u \in [0,1] \quad \text{uniform at random}, \quad k: \text{mutation precision},
\]

This mutation algorithm is able to generate most points in the hyper-cube defined by the variables of the individual and range of the mutation (the range of mutation is given by the value of the parameter \(r\) and the domain of the variables). Most mutated individuals will be generated near the individual before mutation. Only some mutated individuals will be far away from the not mutated individual. That means, the probability of small step-sizes is greater than that of bigger steps. Figure 5-1 tries to give an impression of the mutation results of this mutation operator.

Fig. 5-1: Effect of mutation of real variables in two dimensions

The parameter \(k\) (mutation precision) defines indirectly the minimal step-size possible and the distribution of mutation steps inside the mutation range. The smallest relative mutation step-size is \(2^{-k}\), the largest \(2^{k} = 1\). Thus, the mutation steps are created inside the area \([r, \ r \cdot 2^{k}]\) (\(r:\) mutation range). With a mutation precision of \(k = 16\), the smallest mutation step possible is \(r^{2^{-16}}\). Thus, when the variables of an individual are so close to the optimum, a further improvement is not possible.\(^2\)

Typical values for the parameters of the mutation operator from equation 5-1 are:

- mutation precision \(k\): \(k \in \{4,5,\ldots,20\}\)
- mutation range \(r\): \(r \in [0.1,10^{-6}]\)

---

\(^2\) This can be circumvented by decreasing the mutation range (restart of the evolutionary run or use of multiple strategies).
By changing these parameters very different search strategies can be defined.

## 5.2 Binary mutation

For binary valued individuals mutation means the flipping of variable values, because every variable has only two states. Thus, the size of the mutation step is always 1. For every individual the variable value to change is chosen (mostly uniform at random). Table 5-1 shows an example of a binary mutation for an individual with 11 variables, where variable 4 is mutated.

**Tab. 5-1: Individual before and after binary mutation**

<table>
<thead>
<tr>
<th>before mutation</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>after mutation</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Assuming that the above individual decodes a real number in the bounds [1, 10], the effect of the mutation depends on the actual coding. Table 5-2 shows the different numbers of the individual before and after mutation for binary/gray and arithmetic/logarithmic coding.

**Tab. 5-2: Result of the binary mutation**

<table>
<thead>
<tr>
<th>scaling</th>
<th>linear</th>
<th>logarithmic</th>
</tr>
</thead>
<tbody>
<tr>
<td>coding</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gray</td>
<td></td>
<td></td>
</tr>
<tr>
<td>binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gray</td>
<td></td>
<td></td>
</tr>
<tr>
<td>before mutation</td>
<td>5.0537</td>
<td>4.2887</td>
</tr>
<tr>
<td>after mutation</td>
<td>4.4910</td>
<td>3.3346</td>
</tr>
</tbody>
</table>

However, there is no longer a reason to decode real variables into binary variables. Powerful mutation operators for real variables are available, see the operator in Section 5.1. The advantages of these operators were shown in some publications (for instance [Mic94] and [Dav91]).

## 5.3 Real valued mutation with adaptation of step-sizes

For the mutation of real variables exists the possibility to learn the direction and step-size of successful mutations by adapting these values. These methods are a part of evolutionary strategies ([Sch81] and [Rec94]) and evolutionary programming ([Fdb95]).

Extensions of these methods or new developments were published recently:

- Adaptation of $n$ (number of variables) step-sizes ([OGH93], [OGH94], ES-algorithm with derandomized mutative step-size control using accumulated information),
- Adaptation of \( n \) step-sizes and one direction (derandomized adaptation of \( n \) individual step-sizes and one direction - A II).
- Adaptation of \( n \) step-sizes and \( n \) directions (derandomized adaptation of the generating set - A I).

For storing the additional mutation step-sizes and directions additional variables are added to every individual. The number of these additional variables depends on the number of variables \( n \) and the method. Each step-size corresponds to one additional variable, one direction to \( n \) additional variables. To store \( n \) directions \( n^2 \) additional variables would be needed.

In addition, for the adaptation of \( n \) step-sizes \( n \) generations with the calculation of multiple individuals each are needed. With \( n \) step-sizes and one direction (A II) this adaptation takes \( 2n \) generations, for \( n \) directions (A I) \( n^2 \) generations.

When looking at the additional storage space required and the time needed for adaptation it can be derived, that only the first two methods are useful for practical application. Only these methods achieve an adaptation with acceptable expenditure. The adaptation of \( n \) directions (A I) is currently only applicable to small problems.

The algorithms for these mutation operators will not be described at this stage. Instead, the interested reader will be directed towards the publications mentioned. An example implementation is contained in [GEATbx]. Some comments important for the practical use of these operators will be given in the following paragraphs.

The mutation operators with step-size adaptation need a different setup for the evolutionary algorithm parameters compared to the other algorithms. The adapting operators employ a small population. Each of these individuals produces a large number of offspring. Only the best of the offspring are reinserted into the population. All parents will be replaced. The selection pressure is 1, because all individuals produce the same number of offspring. No recombination takes place.

Good values for the mentioned parameters are:
- 1 (1-3) individuals per population or subpopulation,
- 5 (3-10) offspring per individual => generation gap = 5,
- the best offspring replace parents => reinsertion rate = 1,
- no selection pressure => \( SP = 1 \),
- no recombination.

When these mutation operators were used one problem had to be solved: the initial size of the individual step-sizes. The original publications just give a value of 1. This value is only suitable for a limited number of artificial test functions and when the domain of all variables is equal. For practical use the individual initial step-sizes must be defined depending on the domain of each variable. Further, a problem-specific scaling of the initial step-sizes should be possible. To achieve this the parameter mutation range \( r \) can be used, similar to the real valued mutation operator.

Typical values for the mutation range of the adapting mutation operators are:

\[
\text{mutation range } r : r \in [10^{-3},10^{-7}]
\] (5-3)

The mutation range determines the initialization of the step-sizes at the beginning of a run only. During the following step-size adaptation the step-sizes are not constrained.
A larger value for the mutation range produces larger initial mutation steps. The offspring are created far away from the parents. Thus, a rough search is performed at the beginning of a run. A small value for the mutation range determines a detailed search at the beginning. Between both extremes the best way to solve the problem at hand must be selected. If the search is too rough, no adaptation takes place. If the initial step sites are too small, the search takes extraordinarily long and/or the search gets stuck in the next small local minimum.

The adapting mutation operators should be especially powerful for the solution of problems with correlated variables. By the adaptation of step-sizes and directions the correlations between variables can be learned. Some problems (for instance the ROSENBROCK function - contains a small and curve shaped valley) can be solved very effectively by adapting mutation operators.

The use of the adapting mutation operators is very difficult (or useless), when the objective function contains many minima (extrema) or is noisy.
6 Reinsertion

Once the offspring have been produced by selection, recombination and mutation of individuals from the old population, the fitness of the offspring may be determined. If less offspring are produced than the size of the original population then to maintain the size of the original population, the offspring have to be reinserted into the old population. Similarly, if not all offspring are to be used at each generation or if more offspring are generated than the size of the old population then a reinsertion scheme must be used to determine which individuals are to exist in the new population.

The used selection method determines the reinsertion scheme: local reinsertion for local selection and global reinsertion for all other selection methods.

6.1 Global reinsertion

Different schemes of global reinsertion exist:

- produce as many offspring as parents and replace all parents by the offspring (pure reinsertion).
- produce less offspring than parents and replace parents uniformly at random (uniform reinsertion).
- produce less offspring than parents and replace the worst parents (elitist reinsertion).
- produce more offspring than needed for reinsertion and reinsert only the best offspring (fitness-based reinsertion).

Pure Reinsertion is the simplest reinsertion scheme. Every individual lives one generation only. This scheme is used in the simple genetic algorithm. However, it is very likely, that very good individuals are replaced without producing better offspring and thus, good information is lost.

The elitist combined with fitness-based reinsertion prevents this losing of information and is the recommended method. At each generation, a given number of the least fit parents is replaced by the same number of the most fit offspring (see figure 6.1). The fitness-based reinsertion scheme implements a truncation selection between offspring before inserting them into the population (i.e. before they can participate in the reproduction process). On the other hand, the best individuals can live for many generations. However, with every generation some new individuals are inserted. It is not checked whether the parents are replaced by better or worse offspring.
Because parents may be replaced by offspring with a lower fitness, the average fitness of the population can decrease. However, if the inserted offspring are extremely bad, they will be replaced with new offspring in the next generation.

### 6.2 Local reinsertion

In local selection individuals are selected in a bounded neighborhood. (see Section 3.4). The reinsertion of offspring takes place in exactly the same neighborhood. Thus, the locality of the information is preserved.

The used neighborhood structures are the same as in local selection. The parent of an individual is the first selected parent in this neighborhood.

For the selection of parents to be replaced and for selection of offspring to reinsert the following schemes are possible:

- insert every offspring and replace individuals in neighborhood uniformly at random,
- insert every offspring and replace weakest individuals in neighborhood,
- insert offspring fitter than weakest individual in neighborhood and replace weakest individuals in neighborhood,
- insert offspring fitter than weakest individual in neighborhood and replace parent,
- insert offspring fitter than weakest individual in neighborhood and replace individuals in neighborhood uniformly at random,
- insert offspring fitter than parent and replace parent.
7 Population models - Parallel implementations

The population models may be distinguished from each other by looking at the range of the selection strategies of the parents and the definition of the selection pool. Three population models can be defined:

- **global model, see Section 7.1:**
  In the global model the selection takes place inside the whole population. That means, any two or more individuals may be selected together for the production of offspring. No restrictions exist.
- **local model, see Section 7.2:**
  The local model constrains the selection of parents to a local neighborhood.
- **regional model, see Section 7.3:**
  The regional model constrains the selection of parents to parts of the population isolated from each other, called subpopulation. Inside the subpopulation the selection is unrestricted (similar to the global model).

Figure 7-1 presents the corresponding selection pool.

Fig. 7-1: Classification of population models by range of selection (selection pool)

<table>
<thead>
<tr>
<th>Globales Modell</th>
<th>Regionales Modell</th>
<th>Lokales Modell</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Globales Modell" /></td>
<td><img src="image2" alt="Regionales Modell" /></td>
<td><img src="image3" alt="Lokales Modell" /></td>
</tr>
</tbody>
</table>

### 7.1 Global model - worker/farmer

The global model does not divide the population. Instead, the global model employs the inherent parallelism of evolutionary algorithms (population of individuals). The global model corresponds to the classical evolutionary algorithm.

The calculations where the whole population is needed - fitness assignment and selection - are performed by the master. All remaining calculations which are performed for one or two individuals each can be distributed to a number of slaves. The slaves perform recombination, mu-
tation and the evaluation of the objective function separately. This is known as synchronous master-slave-structure, see Figure 7.2.

Fig. 7.2: Global population model (master-slave-structure)

![Global population model](image)

The slave calculations can be done in parallel. For most problems the evaluation of the objective function is the most time consuming part. In this case, the whole evolutionary algorithm is calculated by the master and only the objective function evaluation is distributed to the slaves. A nearly linear acceleration of the calculation time may be achieved (as long as the evaluation time of the objective function is higher than the communication time between master and slaves).

The global model is a simple way (and inherent to every evolutionary algorithm) to reduce very long computation times. Additionally, the distribution of objective function evaluation can be employed for any other population model as well.

### 7.2 Local model - Diffusion model

The local model (diffusion model) handles every individual separately and selects the mating partner in a local neighborhood by local selection, see Section 3.4. Thus, a diffusion of information through the population takes place. During the search virtual islands, see figure 7.3 will evolve.
7.3 Regional model - Migration

The regional model (migration model) divides the population into multiple subpopulations. These subpopulations evolve independently of each other for a certain number of generations (isolation time). After the isolation time a number of individuals is distributed between the subpopulation (migration). The number of exchanged individuals (migration rate), the selection method of the individuals for migration and the scheme of migration determines how much genetic diversity can occur in the subpopulation and the exchange of information between subpopulation.

The parallel implementation of the regional model showed not only a speed up in computation time, but it also needed less objective function evaluations when compared to a single population algorithm. So, even for a single processor computer, implementing the parallel algorithm in a serial manner (pseudo-parallel) delivers better results (the algorithm finds the global optimum more often or with less function evaluations).

The selection of the individuals for migration can take place:
- uniformly at random (pick individuals for migration in a random manner),
- fitness-based (select the best individuals for migration).

Many possibilities exist for the structure of the migration of individuals between subpopulation. For example, migration may take place:
- between all subpopulations (complete net topology - unrestricted), see figure 7-4
- in a ring topology, see figure 7-6
- in a neighbourhood topology, see figure 7-7
The most general migration strategy is that of unrestricted migration (complete net topology). Here, individuals may migrate from any subpopulation to another. For each subpopulation, a pool of potential immigrants is constructed from the other subpopulation. The individual migrants are then uniformly at random determined from this pool.

Figure 7-5 gives a detailed description of the unrestricted migration scheme for 4 subpopulations with fitness-based selection. Subpopulation 2, 3 and 4 construct a pool of their best individuals (fitness-based migration). 1 individual is uniformly at random chosen from this pool and replaces the worst individual in subpopulation 1. This cycle is performed for every subpopulation. Thus, it is ensured that no subpopulation will receive individuals from itself.

The most basic migration scheme is the ring topology. Here, individuals are transferred between directionally adjacent subpopulations. For example, individuals from subpopulation 6 migrate only to subpopulation 1 and individuals from subpopulation 1 only migrate to subpopulation 2.
A similar strategy to the ring topology is the neighbourhood migration of figure 7.7. Like the ring topology, migration is made only between the nearest neighbours. However, migration may occur in either direction between subpopulations. For each subpopulation, the possible immigrants are determined, according to the desired selection method, from the adjacent subpopulations and a final selection is made from this pool of individuals (similar to figure 7.5).

Figure 7.7 shows a possible scheme for a 2-D implementation of the neighbourhood topology. Sometimes this structure is called a torus.

With the multipopulation evolutionary algorithm better results were obtained for many functions tested than for a single population algorithm with the same number of individuals. Similar results are reported in [Loh91], [MSB91], [Rud91], [SWM91], [Tan91], [VBS91], and [VSB92].
8 Application of different strategies
tbd.

8.1 Different strategies for each subpopulation
tbd., see paper at GECCO'2001, Late breaking papers

8.2 Competition between subpopulations
tbd., see paper at GECCO'2001, Late breaking papers
9 Reference

The reference list contains all the references used during the creation of this report. To provide a better overview and orientation the entries are sorted according to the main topics covered in this documentation.

Section 9.1 contains papers and books about Evolutionary Algorithms in general. Section 9.2 collects publications about population models and parallel implementations of Evolutionary Algorithms. Section 9.3 presents papers on combinatorial optimization using Evolutionary Algorithms. Section 9.4 contains papers and books on visualization. Section 9.5 contains papers and books on the special topic polyploidy and Section 9.6 on biology and genetics.

This division of the reference entries provides a better overview and gives the chance to scan through the references connected with a particular topic.

9.1 Evolutionary Algorithms


9.1 Evolutionary Algorithms


http://borneo.gmd.de/AS/gmdsp/editorial.html

http://borneo.gmd.de/AS/gmdsp/muehlen.html

ftp://borneo.gmd.de/pub/as/ga/gmd_as ga-93_01.ps

ftp://borneo.gmd.de/pub/as/ga/gmd_as ga-95_03.ps


http://www.geatbx.com/index.html
9.1 Evolutionary Algorithms


http://www.pohlheim.com/publications.html


http://www.pohlheim.com/publications.html


http://www.pohlheim.com/publications.html

http://www.pohlheim.com/publications.html

http://www.pohlheim.com/publications.html


http://www.pohlheim.com/publications.html


http://www.lania.mx/~ccoello/EMOO/veldhuizen99a.ps.gz


ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-95_01.ps


http://www.santafe.edu/sfi/publications/Working-Papers/95-02-010.ps


9.2 Population models and parallel EA


ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-91_01.ps

9.3 Combinatorial optimization


http://www.densis.fee.unicamp.br/~moscato/TSPBIB_home.html


http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html


9.4 Visualization

http://intarch.ac.uk/journal/issue1/beardah_toc.html


http://kmi.open.ac.uk/~trevor/research/publications/PPIG-97.ps.gz


[EF95] s. S.51


[Poh97] s. S.52

[Poh99a] s. S.53

9.5 Polyploidy and Evolutionary Algorithms


9.6 Biology, Genetics and Population genetics


List of Figures

| Fig. 1-1: | Problem solution using evolutionary algorithms .................................................................1 |
| Fig. 2-1: | Structure of a single population evolutionary algorithm .....................................................3 |
| Fig. 2-2: | Structure of an extended multipopulation evolutionary algorithm ........................................4 |
| Fig. 3-1: | Fitness assignment for linear and non-linear ranking ........................................................10 |
| Fig. 3-2: | Properties of linear ranking ................................................................................................11 |
| Fig. 3-3: | Roulette-wheel selection ....................................................................................................12 |
| Fig. 3-4: | Stochastic universal sampling ............................................................................................13 |
| Fig. 3-5: | Linear neighbourhood: full and half ring ............................................................................14 |
| Fig. 3-6: | Two-dimensional neighbourhood; left: full and half cross, right: full and half star ..........14 |
| Fig. 3-7: | Properties of truncation selection ......................................................................................16 |
| Fig. 3-8: | Properties of tournament selection .....................................................................................17 |
| Fig. 3-9: | Dependence of selection parameter on selection intensity ................................................18 |
| Fig. 3-10: | Dependence of loss of diversity on selection intensity.......................................................18 |
| Fig. 3-11: | Dependence of selection variance on selection intensity ....................................................19 |
| Fig. 4-1: | Possible positions of the offspring after discrete recombination .......................................21 |
| Fig. 4-2: | Area for variable value of offspring compared to parents in intermediate recombination ..........22 |
| Fig. 4-3: | Possible area of the offspring after intermediate recombination ........................................23 |
| Fig. 4-4: | Possible positions of the offspring after line recombination .............................................24 |
| Fig. 4-5: | Possible positions of the offspring after extended line recombination according to the positions of the parents and the definition area of the variables ........................................25 |
| Fig. 4-6: | Single-point crossover .......................................................................................................26 |
| Fig. 4-7: | Multi-point crossover .......................................................................................................27 |
| Fig. 5-1: | Effect of mutation of real variables in two dimensions .....................................................30 |
| Fig. 6-1: | Scheme for elitist insertion ................................................................................................36 |
| Fig. 7-1: | Classification of population models by range of selection (selection pool) .........................37 |
| Fig. 7-2: | Global population model (master-slave-structure) ...............................................................38 |
| Fig. 7-3: | Local model (diffusion evolutionary algorithm) ..................................................................39 |
| Fig. 7-4: | Unrestricted migration topology (Complete net topology) ................................................40 |
| Fig. 7-5: | Scheme for migration of individuals between subpopulation ............................................40 |
| Fig. 7-6: | Ring migration topology; left: distance 1, right: distance 1 and 2 .....................................41 |
| Fig. 7-7: | Neighbourhood migration topology (2-D grid) ..................................................................41 |
# List of Tables

| Tab. 3-3: | Number of neighbours for local selection | .................................................................15 |
| Tab. 3-4: | Relation between truncation threshold and selection intensity | .........................................................15 |
| Tab. 3-5: | Relation between tournament size and selection intensity | .......................................................16 |
| Tab. 5-1: | Individual before and after binary mutation | ...............................................................31 |
| Tab. 5-2: | Result of the binary mutation | ..................................................................................31 |