GEATbx Introduction

Evolutionary Algorithms:

Overview, Methods and Operators

version 3.7 (November 2005)

Hartmut Pohlheim

Documentation for:

GEATbx version 3.7

(Genetic and Evolutionary Algorithm Toolbox for use with Matlab)

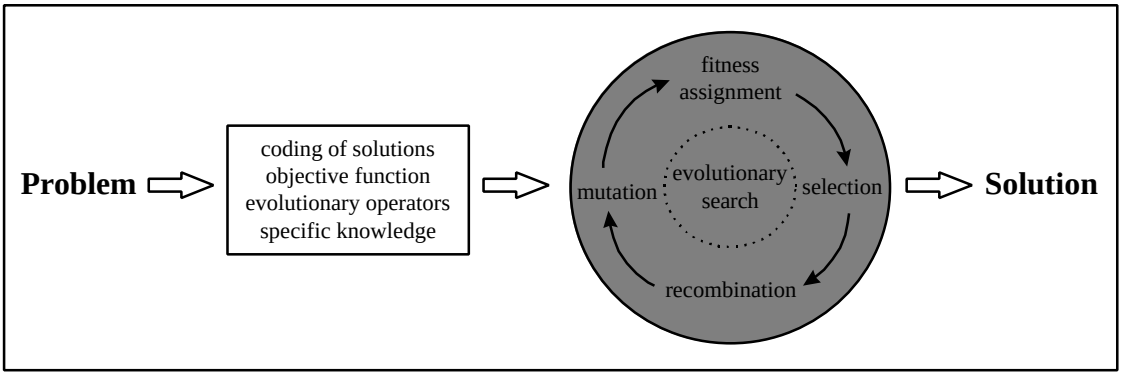
WWW: http://www.geatbx.com/

Email: support@geatbx.com

Contents

1 Introduction

Fig. 1-1: Problem solution using evolutionary algorithms



Different main schools of evolutionary algorithms have evolved during the last 40 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, p.3 a short overview of the structure and basic algorithms of evolutionary algorithms is given. Chapter 3, p.9 describes selection. In Chapter 4, p.25 the different recombination algorithms are presented. Chapter 5, p.33 explains mutation and Chapter 6, p.39 reinsertion.

Chapter 7, p.41 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, p.47.

Chapter 9, p.59 explains how complete optimization algorithms can be created from the different evolutionary operators. The respective options are discussed in detail. Each of the presented optimization algorithms represents an evolutionary algorithm.

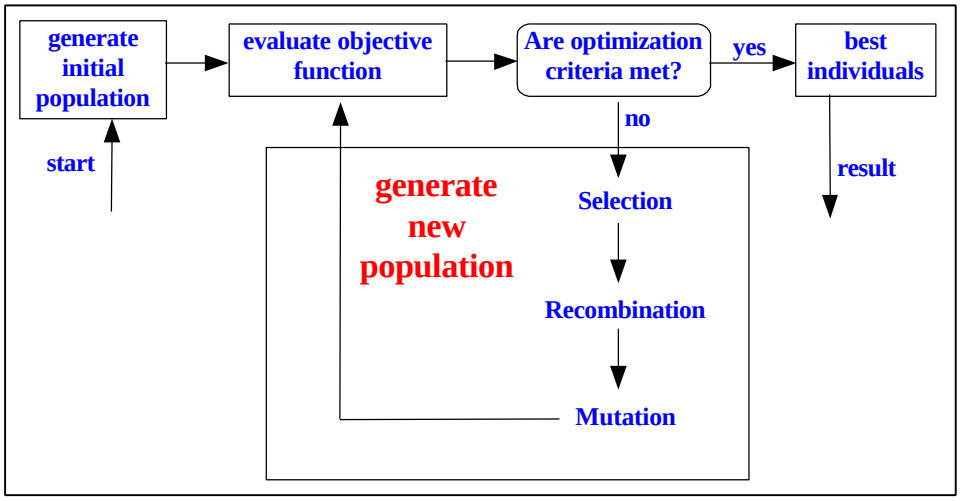
Chapter 10, p.71 lists all the used references and a large number of other publications from the field of Evolutionary Algorithms.

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 an d 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



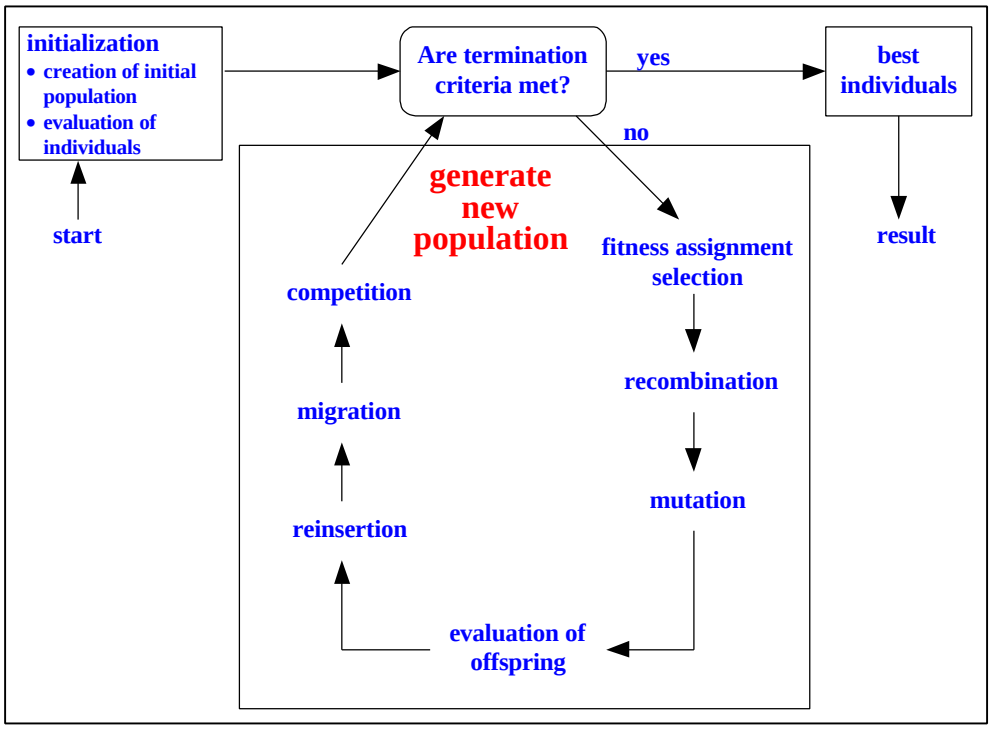
At the beginning of the computation a number of individuals (the population) are randomly initialized. The objective function is then evaluated for these individuals. The first/initial generation is produced.

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, p.9,
- multi-objective ranking, see Section 3.2, p.12.

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 3.3, p.15,
- stochastic universal sampling, see Section 3.4, p.16,
- local selection, see Section 3.5, p.16,
- truncation selection, see Section 3.6, p.18 or
- tournament selection, see Section 3.7, p.20.

For more info see Chapter 3, p.9.

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, p.25, (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, p.26:
  - 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, p.29:
  - 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 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).

For more info see Chapter 4, p.25.

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, p.33,
- mutation for binary valued variables, see Section 5.2, p.35.

For more info see Chapter 5, p.33.

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, p.39.

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, p.41,
- local model (diffusion model, neighborhood model, fine grained model), see Section 7.2, p.42,
- regional model (migration model, island model, coarse grained model), see Section 7.3, p.43.

For more info see Chapter 7, p.41.

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 8.3, p.54,
- competition between subpopulations, see Section 8.2, p.50.

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, p.47.

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 these terms follow [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]

3.1.1 Linear ranking

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:

$$Fitneß(Pos) = 2 - SP + 2 \cdot (SP - 1) \cdot \frac{(Pos - 1)}{(Nind - 1)} \qquad (3\text{-}1)$$

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

3.1.2 Non-linear ranking

A new method for ranking using a non-linear distribution was introduced in [Poh95]. The use of non-linear ranking permits higher selective pressures than the linear ranking method.

Non-linear ranking:

$$Fitneß(Pos) = \frac{Nind \cdot X^{Pos-1}}{\sum_{i=1}^{Nind} X^{i-1}} \qquad (3\text{-}2)$$

X is computed as the root of the polynomial:

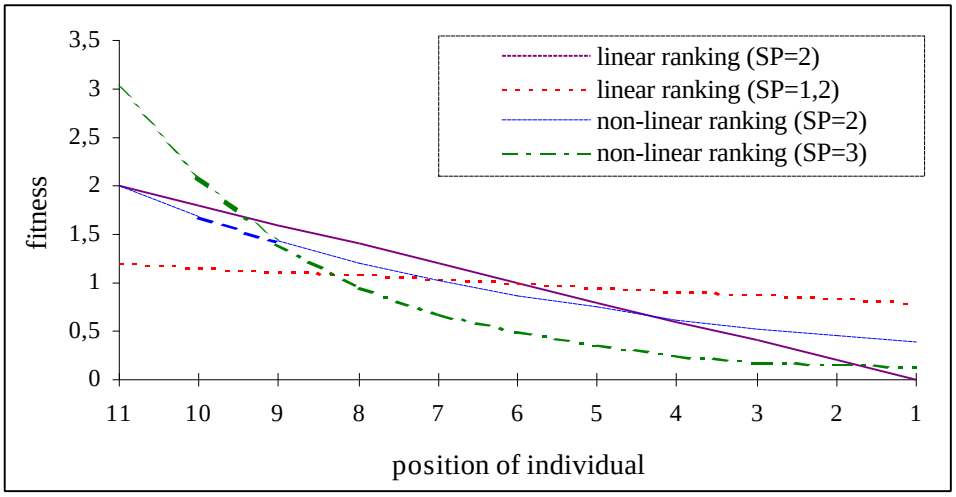
$$0 = (SP - Nind) \cdot X^{Nind-1} + SP \cdot X^{Nind-2} + \ldots + SP \cdot X + SP \qquad (3\text{-}3)$$

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

3.1.3 Comparison of linear and non-linear ranking

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

Fig. 3-1: Fitness assignment for linear and non-linear ranking



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.

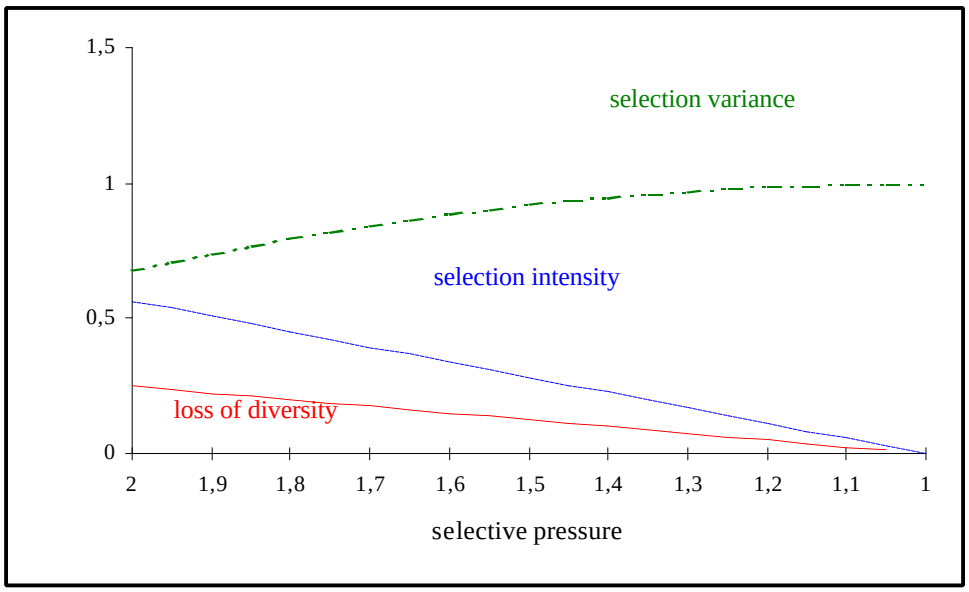
Table 3-1: Dependency of fitness value from selective pressure

	fitness value (parameter: selective pressure)				
	linear ranking		no ranking	non-linear ranking	
objective value	2.0	1.1	1.0	3.0	2.0
1	2.0	1.10	1,0	3.00	2.00
3	1.8	1.08	1,0	2.21	1.69
4	1.6	1.06	1,0	1.62	1.43
7	1.4	1.04	1,0	1.99	1.21
8	1.2	1.02	1,0	0.88	1.03
9	1.0	1.00	1,0	0.65	0.87
10	0.8	0.98	1,0	0.48	0.74
15	0.6	0.96	1,0	0.35	0.62
20	0.4	0.94	1,0	0.26	0.53
30	0.2	0.92	1,0	0.19	0.45
95	0.0	0.90	1,0	0.14	0.38

3.1.4 Analysis of linear ranking

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

Fig. 3-2: Properties of linear ranking



Selection intensity

$$SelInt_{LinRank}(SP) = (SP - 1) \cdot \frac{1}{\sqrt{p}} \qquad (3\text{-}4)$$

Loss of diversity

$$LossDiv_{LinRank}(SP) = \frac{(SP-1)}{4} \tag{3-5}$$

Selection variance

$$SelVar_{LinRank}(SP) = 1 - \frac{(SP-1)^2}{p} = 1 - SelInt_{LinRank}(SP)^2 \tag{3-6}$$

3.2 Multi-objective Ranking

Where proportional and rank-based fitness assignment is concerned it is assumed that individuals display only one objective function value. In many real world problems, however, there are several criteria which have to be considered in order to evaluate the quality of an individual. Only on the basis of the comparison of these several criteria (thus multi-objective) can a decision be made as to the superiority of one individual over another. Then, as in single-objective problems, an order of individuals within the population can be established from these reciprocal comparisons – multi-objective ranking. After this order has been established the single-objective ranking methods from the subsection 3.1, p.9 can be used to convert the order of the individuals to corresponding fitness values.

Multi-objective fitness assignment (and with it *multi-objective optimization*) is concerned with the simultaneous minimization of *NObj* criteria f_r, with $r = 1, ..., NObj$. The values f_r are determined by the objective function, which in turn is dependent on the variables of the individuals (the decision variables).

A straightforward example should serve as the motivation for the following considerations. When objects are produced, the production costs should be kept low and the objects should be produced quickly. Various solutions can be developed during production planning which may differ regarding the number and type of the machines employed, as well as regarding the number of workers. The criteria production costs f_1 and production time f_2, both of which are determined by the objective function, serve as evaluation criteria for each solution.

3.2.1 PARETO-ranking

The superiority of one solution over the other can be decided by comparing 2 solutions. It can be carried out both generally and for as many criteria as desired, following the schema in equation 3-7.

PARETO-ranking resp. PARETO-dominance:

$$\begin{aligned} &\forall i \in \{1,\ldots,NObj\},\ f_i^{Lös_1} \le f_i^{Lös_2} \quad and \quad \exists i \in \{1,\ldots,NObj\},\ f_i^{Lös_1} < f_i^{Lös_2} \\ &\Rightarrow solution_1 \ \mathrm{p}< \ solution_2 \quad (\mathrm{p}<: \ partially\ less\ than\text{-}\ plt) \end{aligned} \tag{3-7}$$

If solution$_1$ is p< (partially less than) solution$_2$, it follows that solution$_1$ dominates solution$_2$. In the example used here this means: if costs and time are less for solution$_1$ than for solution$_2$, it follows that solution$_1$ is superior to solution$_2$. It would even be sufficient if one of the two values was equal for both solutions (equal costs) and only the other value was lower (less time).

If, however, none of the solutions dominates the other both solutions are to be regarded as equivalent with respect to the PARETO-order. The same rank is assigned to individuals which do not dominate each other.

The rank of an individual within the population ($rank_i$) depends on the number of individuals ($NumInd_{dominated}$) dominating this individual [Fon95]:

$$Rang_i = 1 + NumInd_{dominated} \tag{3-8}$$

All solutions that are found during optimization and are not dominated by a different solution constitute the PARETO-optimal solutions (*PARETO-optimal set*) of this problem (PARETO-optimality). These solutions are assigned a rank value of 1. In the case of each PARETO-optimal solution it is not possible to improve one of the criteria without one or several of the other criteria deteriorating.

3.2.2 Goal attainment or method of inequalities

When using plain PARETO-ranking in equation 3-7 all PARETO-optimal solutions are equivalent. It is possible, however, to make a further differentiation for many practical problems. As above, the example of object production shall be used to illustrate this. In an extreme case a solution could produce nothing at all. In this case, the costs would equal zero and the production time would be infinite. No other solution could produce the objects at lower costs. Thus, as this solution cannot be dominated, it would belong to the PARETO-optimal solutions if PARETO-dominance was applied exclusively. In a second extreme case a solution could produce objects in a very short time. This would result in very high expenditure, and therefore very high costs. It should be obvious that both cases are not desirable although they belong to the non-dominated solutions which are not dominated.

Goals for the individual criteria can be introduced in order to preclude PARETO-optimal set solutions, which lie outside of the results desired by the user. A solution is only acceptable when the goals for the individual criteria have been reached. This procedure is also termed *method of inequalities* , MOI, or *goal programming*. The individual goals are predefined as inequalities.

In the production example used here, this could mean that an upper limit for costs and production time is determined. The result is not acceptable until both goals are simultaneously reached or fallen short of by means of a solution.

When the goals are included in the multi-objective ranking the comparison of two solutions becomes somewhat more complex. The following assumptions are made:

$$F^{Lös_1} = \left[f_1^{Lös_1}, f_2^{Lös_1}, \ldots, f_{NObj}^{Lös_1}\right], \quad F^{Lös_2} = \left[f_1^{Lös_2}, f_2^{Lös_2}, \ldots, f_{NObj}^{Lös_2}\right]$$
$$Goals = \left[Goal_1, Goal_2, \ldots, Goal_{NObj}\right] \tag{3-9}$$

In the definitions the operator *partially less* p< from equation 3-7 is used for each of the following definitions. It is possible to distinguish 3 different cases.

1. Solution$_1$ does not fulfill any goals:

$$F^{Lös_1} > Goals \quad \wedge \quad solution_1 \text{ p} < solution_2$$
$$\Rightarrow \quad solution_1 \text{ preferred} \tag{3-10}$$

2. Solution$_1$ fulfills all goals:

$$F^{Lös_1} \le Goals \quad \wedge \quad \left(solution_1 \text{ p} < solution_2 \quad \vee \quad \sim\left(F^{Lös_2} \le Goals\right)\right) \quad \Rightarrow \quad solution_1 \text{ preferred} \tag{3-11}$$

3. Solution$_1$ fulfills some goals:

$$\text{Statement}: \quad F_M = \left[f_1, f_2, \ldots, f_m\right], \quad F_K = \left[f_{m+1}, f_{m+2}, \ldots, f_{NObj}\right]$$
$$\text{Assumption}: F_M^{Lös_1} > Goals_M \quad \wedge \quad F_K^{Lös_1} \le Goals_M$$
$$\left(F_M^{Lös_1} \text{ p} < F_M^{Lös_2} \quad \vee \quad F_M^{Lös_1} = F_M^{Lös_2}\right) \quad \wedge \quad \left(F_K^{Lös_1} \text{ p} < F_K^{Lös_2} \quad \vee \quad \left(F_K^{Lös_2} \le Goals_K\right)\right)$$
$$\Rightarrow solution_1 \text{ preferred} \tag{3-12}$$

PARETO-ranking using a vector of goals for the individual criteria, allows for an improved generation of the sequence of a number of solutions compared to plain PARETO-ranking.

Multi-objective optimization is carried out in order to find a number of non-dominated solutions, the PARETO-optimal set. A normal evolutionary algorithm, however, converges at a single solution. This process is termed a genetic drift. For this reason, methods must be built in, which achieve the maintenance or expansion of population diversity (prevention of *premature convergence*).

3.2.3 Sharing

On the one hand, the genetic drift can be counteracted by the application of *fitness sharing*). On the other hand, the algorithm has the effect that a greater part of the pareto-optimal solution is ascertained. The basic principle is that individuals in a particular niche must share the resources available amongst themselves. In this way, the more individuals are near an individual, the lower its fitness becomes.

During application the size of the niches and the allocation of resources within each niche must be established. Methods for fitness sharing are suggested in [HN93], [SD94] and [Fon95],amongst others.

3.2.4 Further information on multi-objective optimization

In this subsection it was only possible to provide a short introduction to the multi-objective fitness assignment within the context of evolutionary algorithms. The relevant literature should be referred to for more specific questions regarding individual procedures: [Hor97], [Fon95], [ZT98], [HN93], [SD94], [Vel99]. A list of a large amount of literature on multi-objective optimization with evolutionary algorithms can be accessed in [Coe99]. There is a great deal of literature on multi-objective optimization not specific to the evolutionary algorithm. [Mie99] is recommended as a good starting point.

3.2.5 Weighted sum – aggregation or scalarization of multiple objectives

When considering different methods and component parts used for multi-objective optimization one should not forget classic methods for the integration of several criteria (*scalarization method*, also called aggregation of objectives).

The weighted sum is the most well-known method. Here each criterion is assigned a weighting value. By means of the linear combination of all the weighted criteria a combined objective function value F_{ws} is achieved:

$$F_{WS} = \sum_{r=1}^{NObj} W_r \cdot f_r \qquad (3\text{-}13)$$

The weighted sum is especially used in cases when the varying significance of the individual criteria is known or can be estimated. As this is frequently the case where practical application is concerned, the weighted sum is often applied.

If a multi-objective problem is solved by means of single-objective optimization, only a point solution is obtained. The advantage of obtaining several solutions of equal value relating to a target vector is lost. At this stage the user must decide whether the simple use of the weighted sum or the approximation of the PARETO-optimal solutions is more important for the solution of the problem.

3.3 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 3-2: Selection probability and fitness value

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

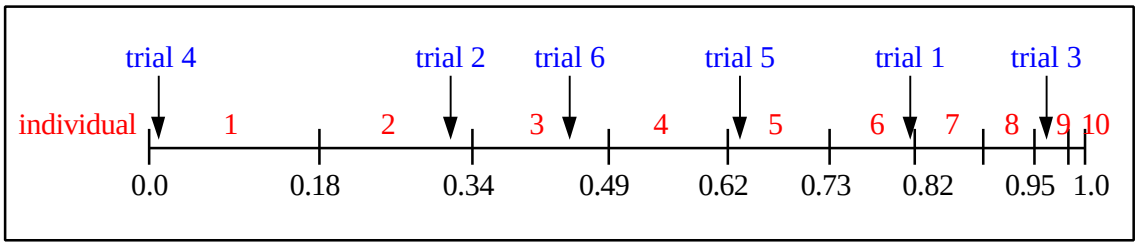
For selecting the mating population the appropriate number of uniformly distributed random numbers (uniform distributed between 0.0 and 1.0) is independently generated.

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.

3.4 Stochastic universal sampling

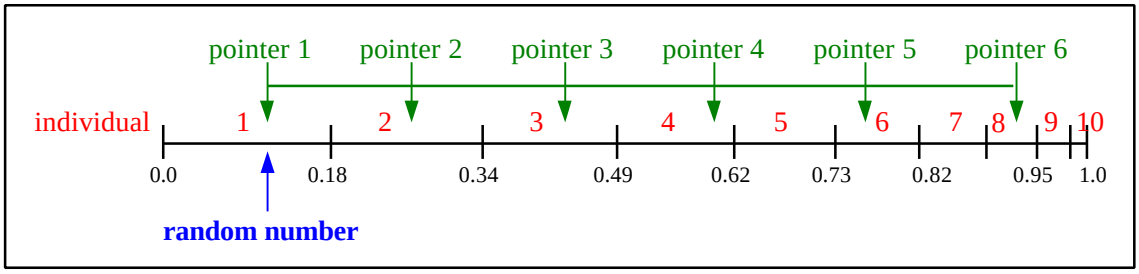
Stochastic universal sampling [Bak87] provides zero bias and minimum spread. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here equally spaced pointers are placed over the line as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

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.

sample of 1 random number in the range [0, 0.167]:

```
0.1.
```

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.

3.5 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.

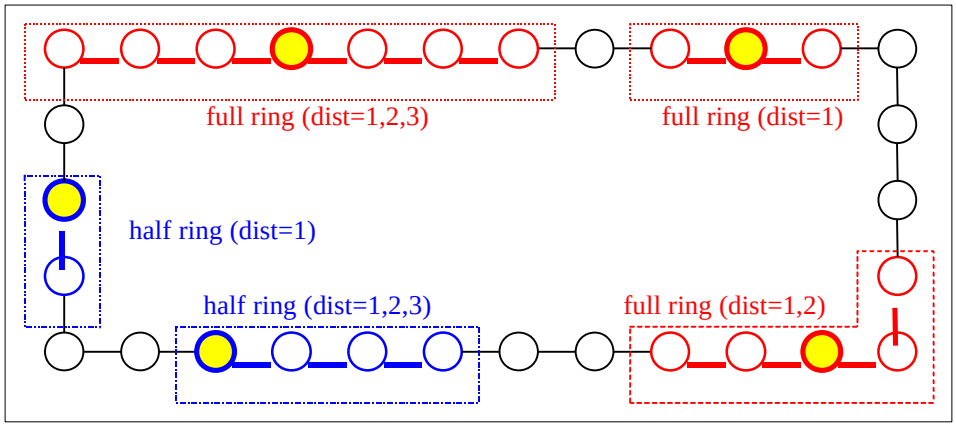
Local selection is part of the local population model, see Section 7.2.

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 individual. Inside this neighborhood the mating partner is selected (best, fitness proportional, or uniform at random).

The structure of the neighborhood can be:

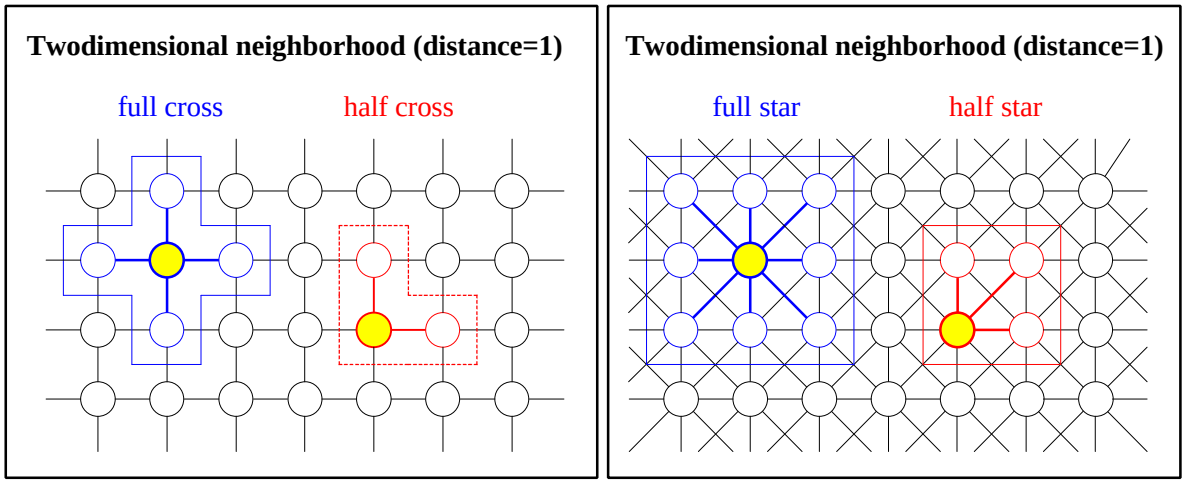
- linear
  - full ring, half ring (see Figure 3-5)
- two-dimensional
  - full cross, half cross (see Figure 3-6, left)
  - full star, half star (see Figure 3-6, right)
- three-dimensional and more complex with any combination of the above structures.

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 neighborhood; 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

structure of selection	distance	
	1	2
full ring	2	4
half ring	1	2
full cross	4	8 (12)
half cross	2	4 (5)
full star	8	24
half star	3	8

3.6 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

truncation threshold	1%	10%	20%	40%	50%	80%
selection intensity	2.66	1.76	1.2	0.97	0.8	0.34

3.6.1 Analysis of truncation selection

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

$$SelInt_{Truncation}(Trunc) = \frac{1}{Trunc} \cdot \frac{1}{\sqrt{2 \cdot p}} \cdot e^{-\frac{f_c^2}{2}} \quad (3\text{-}14)$$

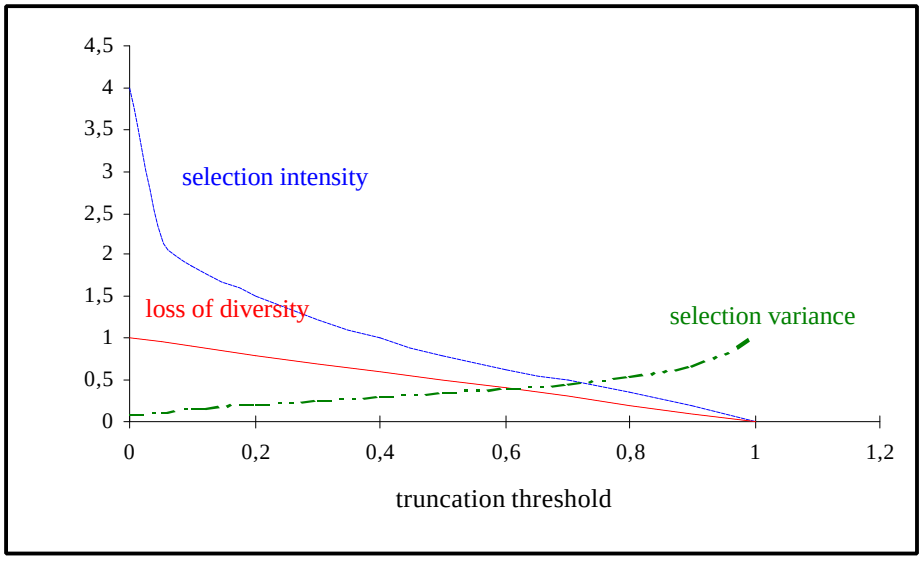
Loss of diversity

$$LossDiv_{Truncation}(Trunc) = 1 - Trunc \quad (3\text{-}15)$$

Selection variance

$$SelVar_{Truncation}(Trunc) = 1 - SelInt_{Truncation}(Trunc) \cdot (SelInt_{Truncation}(Trunc) - f_c) \quad (3\text{-}16)$$

Fig. 3-7: Properties of truncation selection



3.7 Tournament selection

In tournament selection [GD91] a number *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 *Tour*. *Tour* takes values ranging from 2 to *Nind* (number of individuals in population). Table 3-5 and figure 3-8 show the relation between tournament size and selection intensity [BT95].

Tab. 3-5: Relation between tournament size and selection intensity

tournament size	1	2	3	5	10	30
selection intensity	0	0.56	0.85	1.15	1.53	2.04

3.7.1 Analysis of tournament selection

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

Selection intensity

$$SelInt_{Turnier}(Tour) \approx \sqrt{2 \cdot \left(\ln(Tour) - \ln\left(\sqrt{4.14 \cdot \ln(Tour)}\right)\right)} \quad (3\text{-}17)$$

Loss of diversity

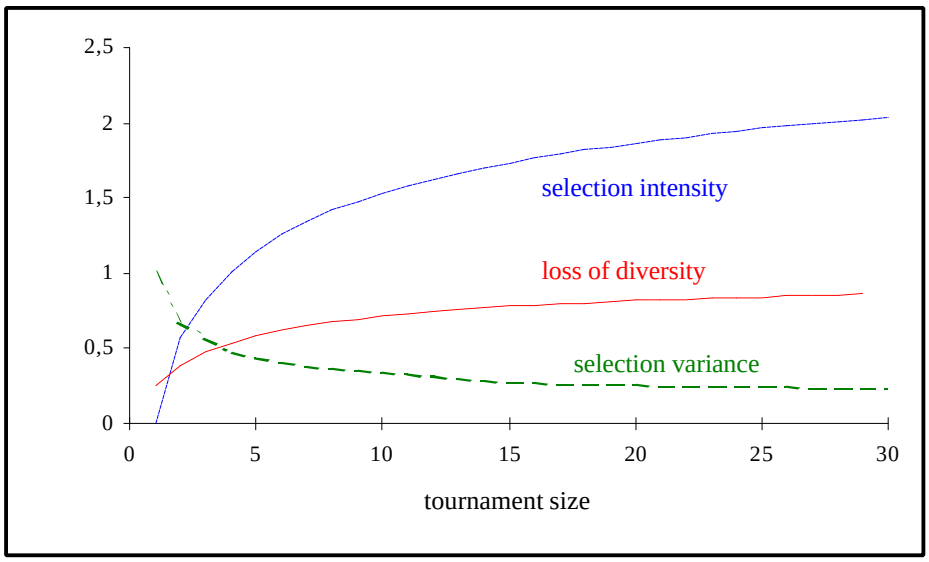
$$LossDiv_{Turnier}(Tour) = Tour^{\frac{-1}{Tour-1}} - Tour^{\frac{-Tour}{Tour-1}} \quad (3\text{-}18)$$

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

Selection variance

$$SelVar_{Turnier}(Tour) \approx \frac{0.918}{\ln(1.186 + 1.328 \cdot Tour)} \quad (3\text{-}19)$$

Fig. 3-8: Properties of tournament selection



3.8 Comparison of selection schemes

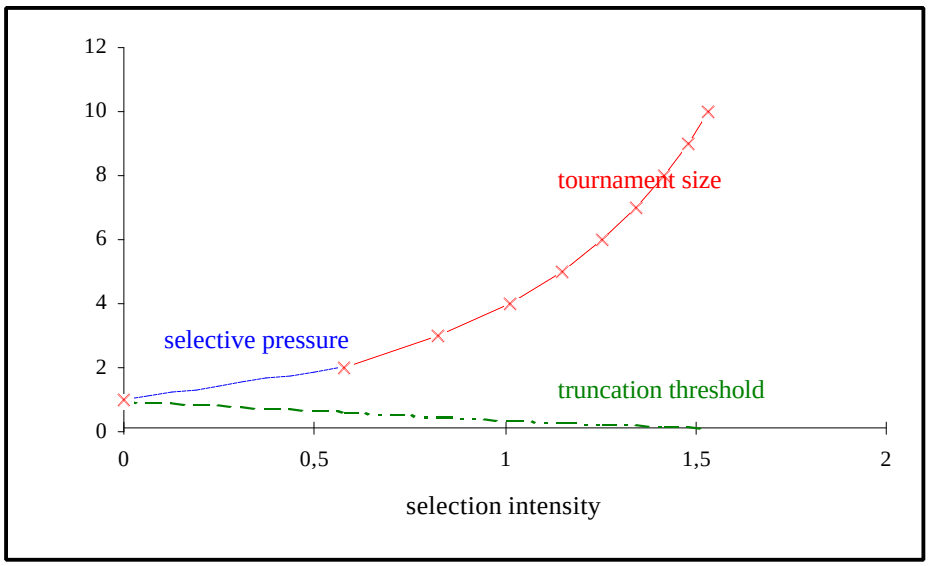
As shown in the previous sections of this chapter the selection methods behave similarly assuming similar selection intensity.

3.8.1 Selection parameter and 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.

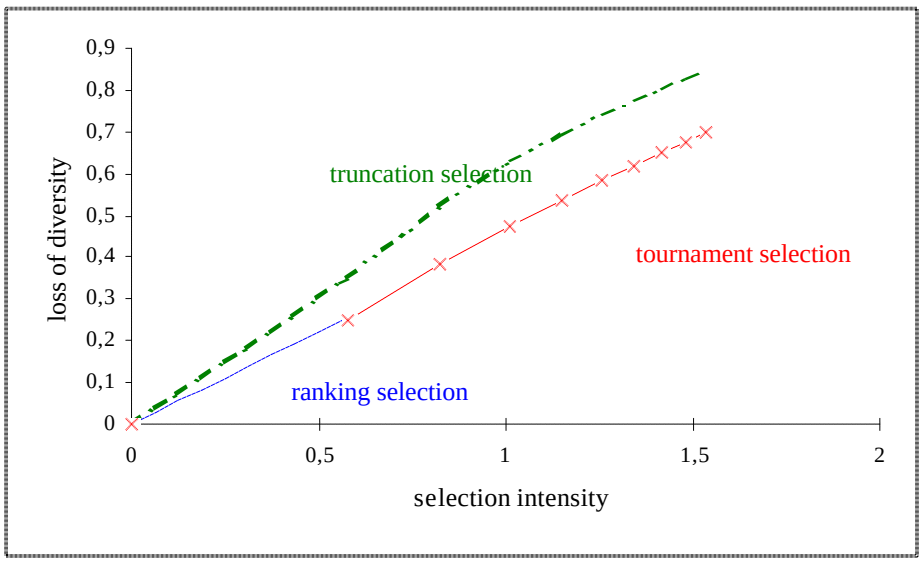
However, the behavior of the selection methods is different. Thus, the selection methods will be compared on the parameters loss of diversity (figure 3-10) and selections variance (figure 3-11) on the selection intensity.

Fig. 3-9: Dependence of selection parameter on selection intensity



3.8.2 Loss of diversity and selection intensity

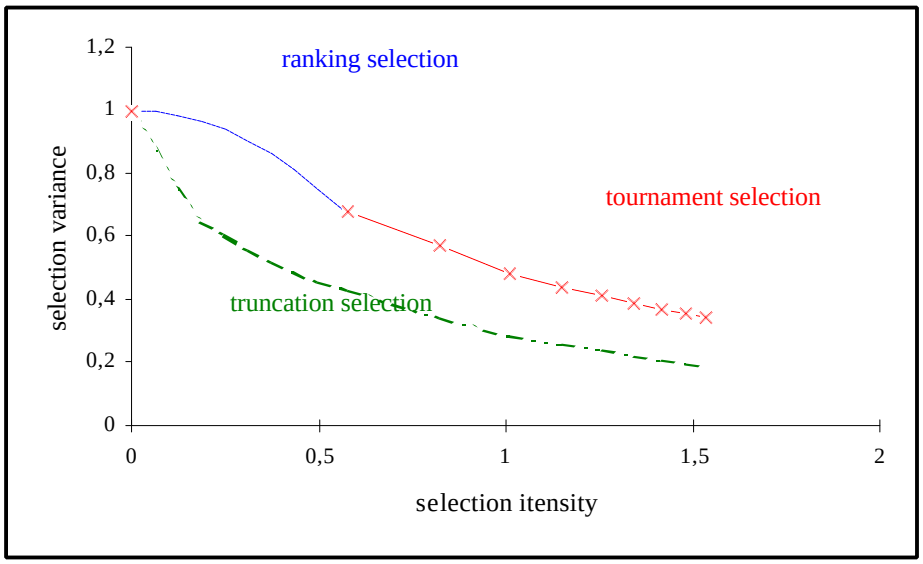
Fig. 3-10: Dependence of loss of diversity on selection intensity



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.

3.8.3 Selection variance and selection intensity

Fig. 3-11: Dependence of selection variance on selection intensity



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.

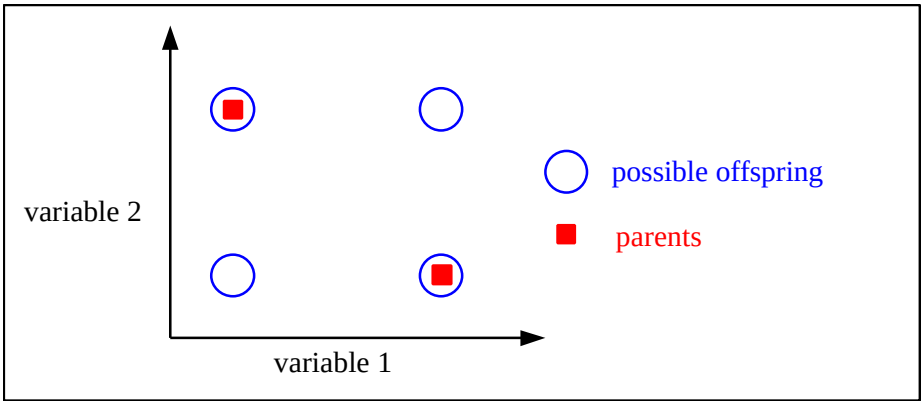
4.1 All representations - Discrete recombination

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.

$$Var_i^O = Var_i^{P_1} \cdot a_i + Var_i^{P_2} \cdot (1 - a_i) i \in (1,2,\ldots,Nvar), \quad a_i \in \{0,1\} \text{ uniform at random, } a_i \text{ for each } i \text{ new defined} \tag{4-1}$$

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.:

```
sample 1          2      2      1
sample 2          1      2      1
```

After recombination the new individuals are created:

```
offspring 1      123     4      5
offspring 2       12     4      5
```

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.

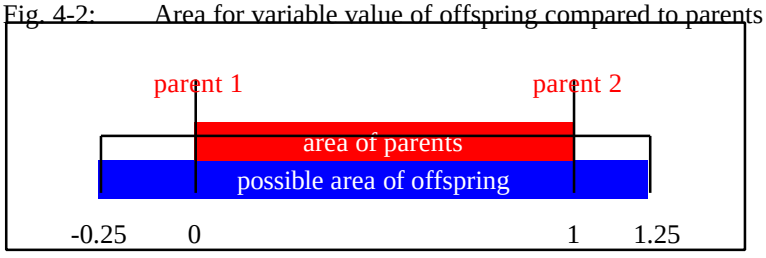
Offspring are produced according to the rule:

$$Var_i^O = Var_i^{P_1} \cdot a_i + Var_i^{P_2} \cdot (1 - a_i) i \in (1,2,\ldots,Nvar), \quad a_i \in [-d, 1+d] \text{ uniform at random, } d=0.25, a_i \text{ for each } i \text{ new} \tag{4-2}$$

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



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:

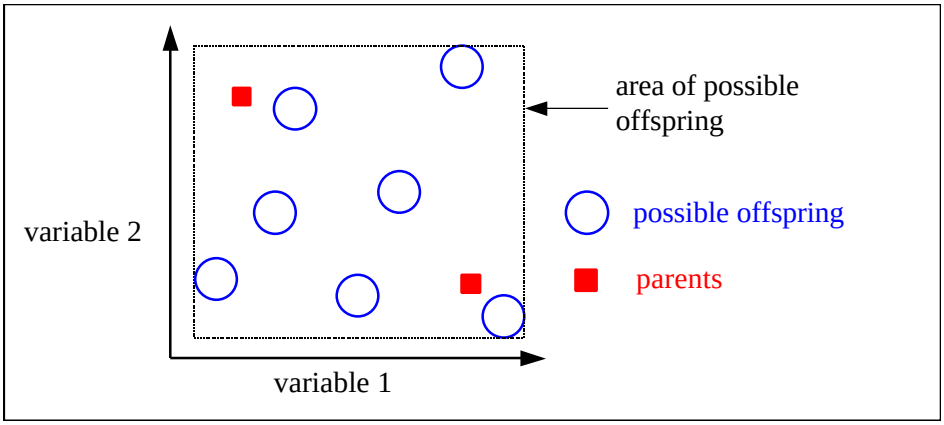
```
sample 1          0.5     1.1    -0.1
sample 2          0.1     0.8     0.5
```

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.

Fig. 4-3: Possible area of the offspring after intermediate recombination



4.2.2 Line recombination

Line recombination [MSV93a] 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:

$$Var_i^O = Var_i^{P_1} \cdot a_i + Var_i^{P_2} \cdot (1 - a_i) i \in (1,2,\ldots,Nvar), \quad a_i \in [-d, 1+d] \text{ uniform at random, } d=0.25, a_i \text{ for all } i \text{ identical} \tag{4-3}$$

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:

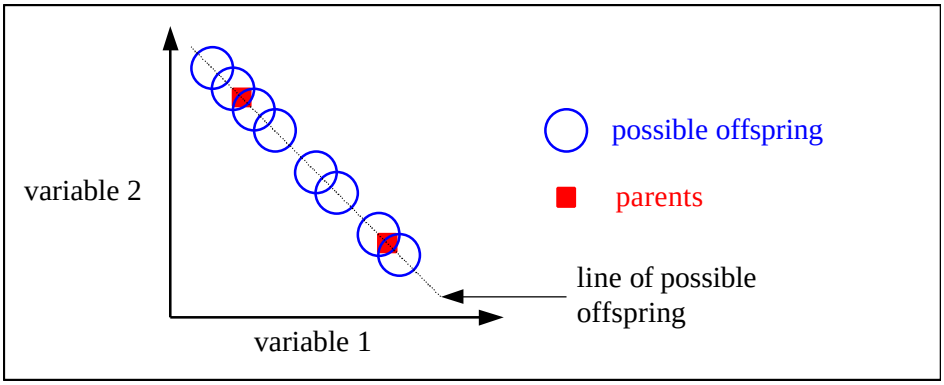
```
sample 1          0.5
sample 2          0.1
```

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.

Fig. 4-4: 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:

$$Var_i^O = Var_i^{P_1} + s_i \cdot r_i \cdot a_i \cdot \frac{Var_i^{P_2} - Var_i^{P_1}}{\left\| Var^{P_1} - Var^{P_2} \right\|} i \in (1,2,\ldots,Nvar),$$

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

$$a_i \text{ for all } i \text{ identical},$$

$$r_i = r \cdot domainr: \text{range of recombination steps},$$

$$s_i \in \{-1,+1\}, \quad \text{uniform at random}: \text{undirected recombination},$$

$$+1 \text{ with probability} > 0.5: \text{directed recombination},$$

(4-4)

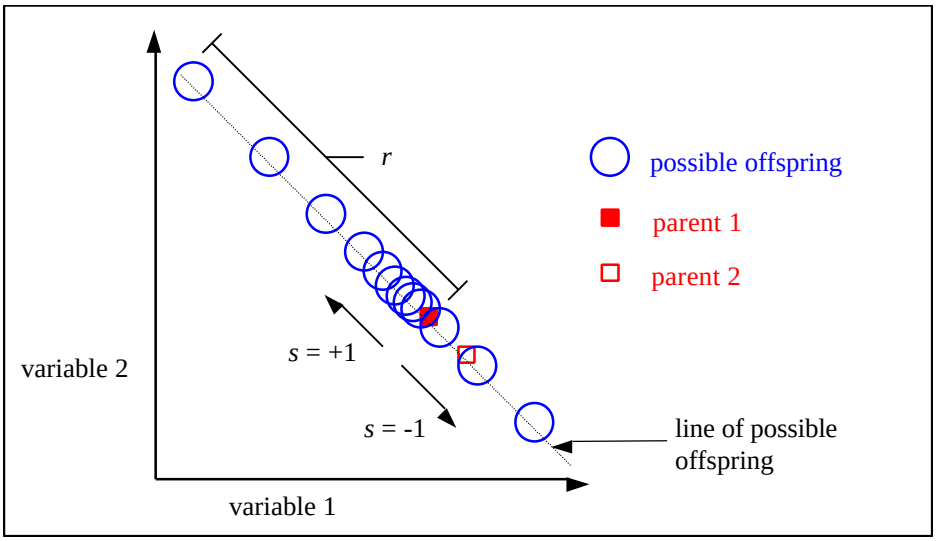
The creation of offspring uses features similar to the mutation operator for real valued variables (see Section 5.1). The parameter *a* defines the relative step-size, the parameter *r* the maximum step-size and the parameter *s* the direction of the recombination step.

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 $k \in [1,2,...,Nvar-1]$, $Nvar$: number of variables of an individual, is selected uniformly at random and the variables exchanged between the indi-

viduals 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:

```
individual 1     0  1  1  1  0  0  1  1  0  1  0
individual 2     1  0  1  0  1  1  0  0  1  0  1
```

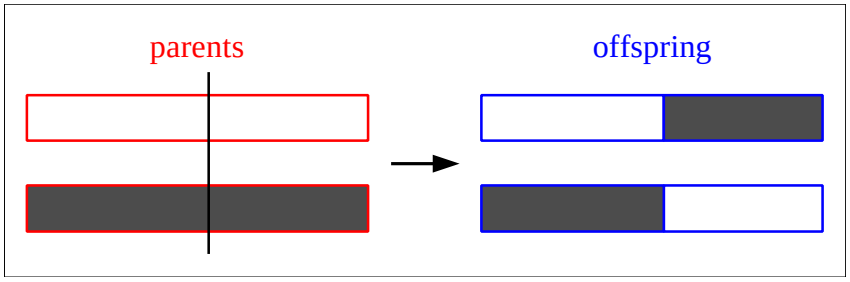
The chosen crossover position is:

```
crossover position            5
```

After crossover the new individuals are created:

```
offspring 1      0  1  1  1  0| 1  0  0  1  0  1
offspring 2      1  0  1  0  1| 0  1  1  0  1  0
```

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 $k_i \in [1,2,...,Nvar-1]$, $i=1:m$, $Nvar$: 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:

```
individual 1     0  1  1  1  0  0  1  1  0  1  0
individual 2     1  0  1  0  1  1  0  0  1  0  1
```

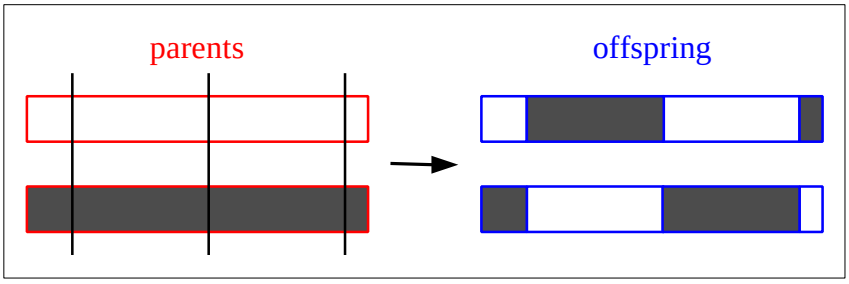
The chosen crossover positions are:

```
cross pos. (m=3)     2           6           10
```

After crossover the new individuals are created:

```
offspring 1      0  1| 1  0  1  1| 0  1  1  1| 1
offspring 2      1  0| 1  1  0  0| 0  0  1  0| 0
```

Fig. 4-7: Multi-point 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 [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 [SDJ91b].

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 [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:

```
individual 1     0  1  1  1  0  0  1  1  0  1  0
individual 2     1  0  1  0  1  1  0  0  1  0  1
```

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.

```
sample 1         0  1  1  0  0  0  1  1  0  1  0
sample 2         1  0  0  1  1  1  0  0  1  0  1
```

After crossover the new individuals are created:

```
offspring 1      1  1  1  0  1  1  1  1  1  1  1
offspring 2      0  0  1  1  0  0  0  0  0  0  0
```

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. [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.

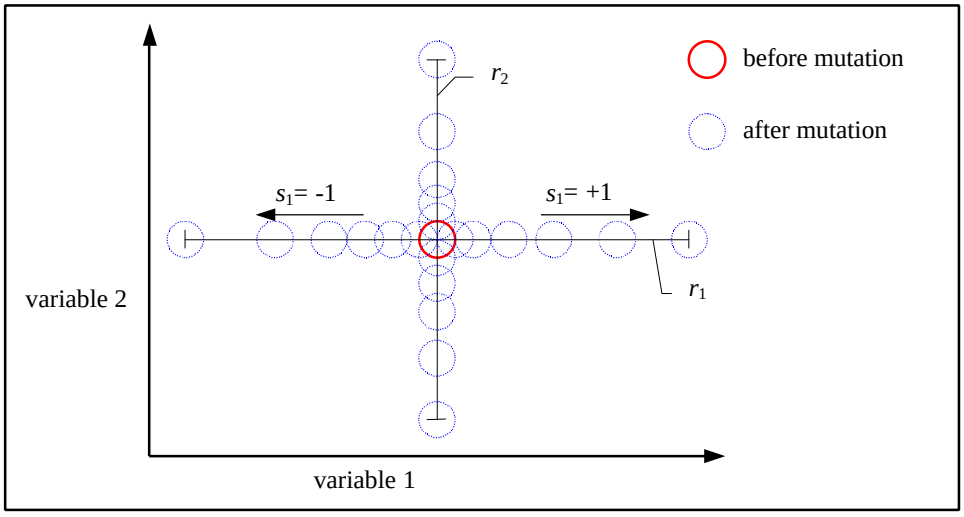
The size of the mutation step is usually difficult to choose. The optimal step-size depends on the problem considered and may even vary during the optimization process. It is known, that small steps (small mutation steps) are often successful, especially when the individual is already well adapted. However, 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*):

$$\begin{aligned} Var_i^{Mut} &= Var_i + s_i \cdot r_i \cdot a_i \; i \in \{1,2,\ldots,n\} \text{ uniform at random,} \\ & s_i \in \{-1,+1\} \text{ uniform at random} \\ & r_i = r \cdot domain_i,\ r: \text{mutation range (standard}: 10\%), \\ & a_i = 2^{-u \cdot k},\ u \in [0,1] \text{ uniform at random, } k: \text{mutation precision,} \end{aligned} \tag{5-1}$$

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 $2^0 = 1$. 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 $r \cdot 2^{-16}$. Thus, when the variables of an individual are so close to the optimum, a further improvement is not possible. This can be circumvented by decreasing the mutation range (restart of the evolutionary run or use of multiple strategies)

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

$$\begin{aligned} &\text{mutation precision } k: \quad k \in \{4, 5, \ldots, 20\} \\ &\text{mutation range } r: r \in [0.1, 10^{-6}] \end{aligned} \tag{5-2}$$

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

before mutation	0	1	1	***1***	0	0	1	1	0	1	0
				ß							
after mutation	0	1	1	***0***	0	0	1	1	0	1	0

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

scaling	linear		logarithmic	
coding	binary	gray	binary	gray
before mutation	5.0537	4.2887	2.8211	2.3196
after mutation	4.4910	3.3346	2.4428	1.8172

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 ([HOG95]: *derandomized adaptation of n individual step-sizes and one direction - A II*),

- Adaptation of *n* step-sizes and *n* directions ([HOG95]: *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 2*n* 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:

$$\text{mutation range } r : r \in \left[10^{-3}, 10^{-7}\right] \tag{5-3}$$

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.

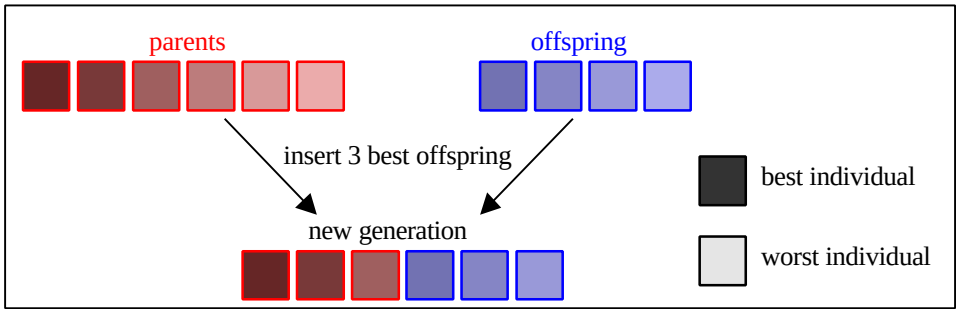
6.1 Global reinsertion

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.

Fig. 6-1: Scheme for elitist insertion



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 3.5). 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.

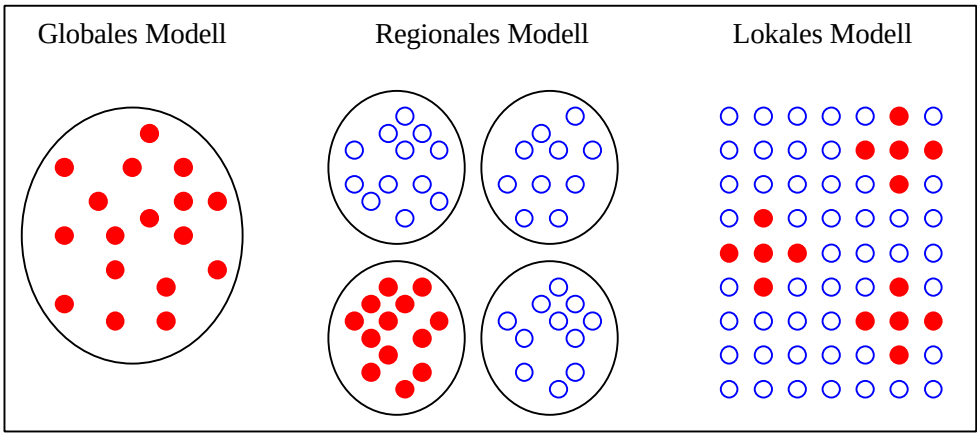
7 Population models - Parallel implementations

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)

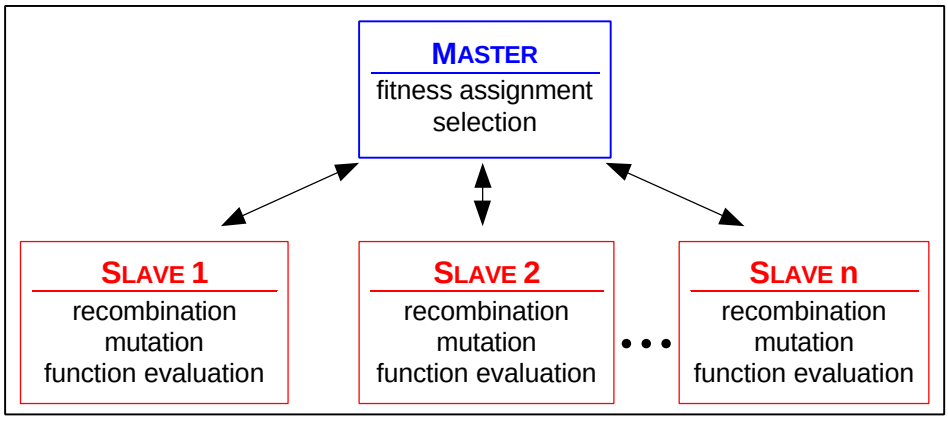


7.1 Global model - worker/farmer

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, mutation 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)

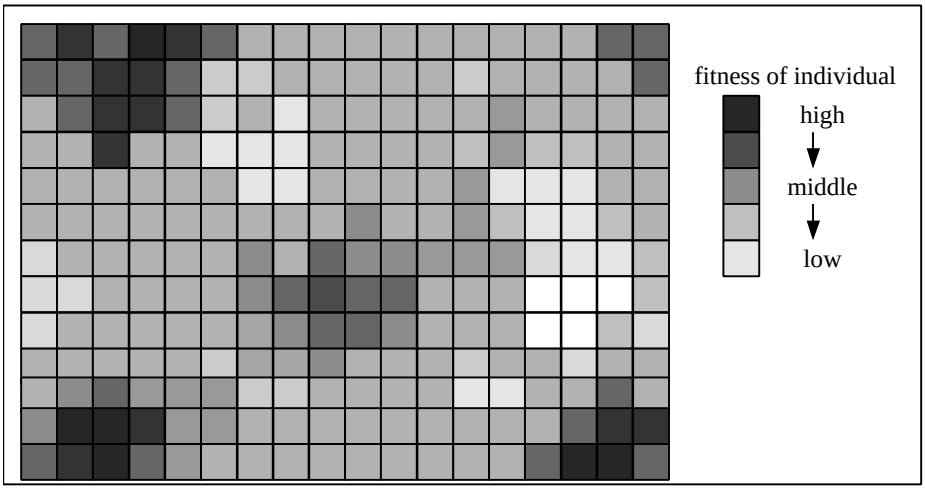


The slave calculations can be done in parallel. For most problems the evaluation of the objective function is the most time consuming part. In this case, the whole evolutionary algorithm is calculated by the master and only the objective function evaluation is distributed to the slaves. A nearly linear 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

Fig. 7-3: Local model (diffusion evolutionary algorithm)



The local model (diffusion model) handles every individual separately and selects the mating partner in a local neighborhood by local selection, see Section 3.5. 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).

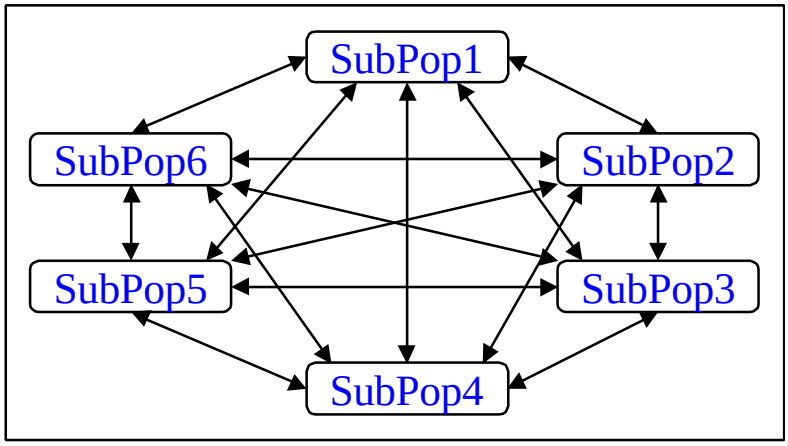
The selection of the individuals for migration can take place:

- 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.

Fig. 7-4: Unrestricted migration topology (Complete net topology)

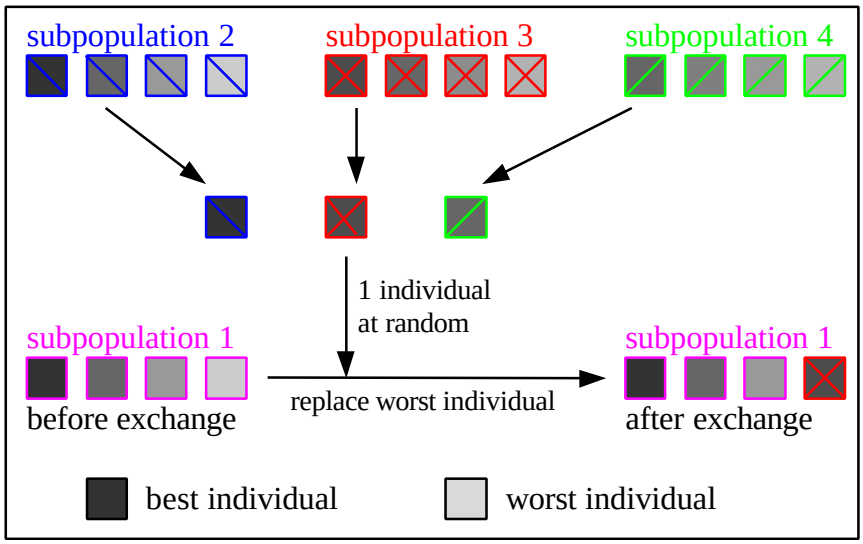


The most general migration strategy is that of unrestricted migration (complete net topology). Here, individuals may migrate from any subpopulation to another. For each subpopulation, a pool of potential immigrants is constructed from the other subpopulation. The individual migrants are then uniformly 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

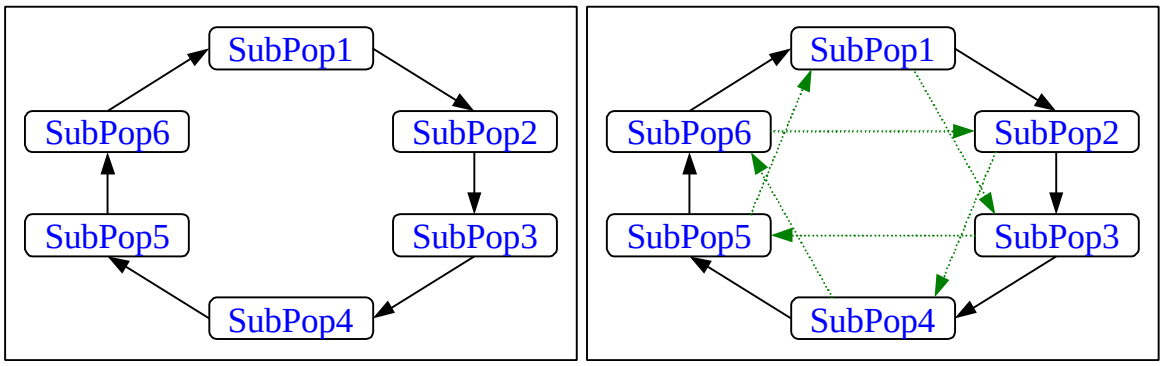
replaces the worst individual in subpopulation 1. This cycle is performed for every subpopulation. Thus, it is ensured that no subpopulation will receive individuals from itself.

Fig. 7-5: Scheme for migration of individuals between subpopulation



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.

Fig. 7-6: Ring migration topology; left: distance 1, right: distance 1 and 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).

Fig. 7-7: Neighbourhood migration topology (2-D grid)

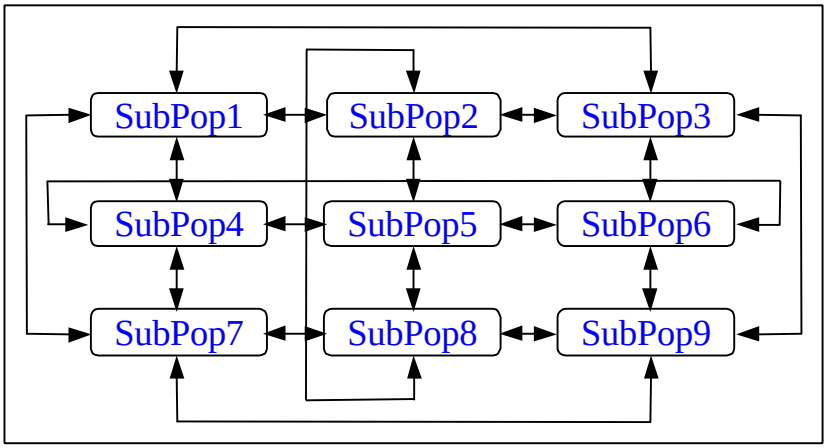


Figure 7-7 shows a possible scheme for a 2-D implementation of the neighbourhood topology. Sometimes this structure is called a torus.

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], [Tan89], [VBS91], [VSB92] and [Can95].

8 Application of different strategies

The optimization of complex systems is a challenging task. We can only master it by combining powerful optimization algorithms and analysis tools. Many of the known algorithms cannot be adjusted to the solution of large problems We have to find new and/or extended methods which are even able to tackle complex problems within huge search domains.

A promising approach for the development of these methods is the intelligent combination of several optimization methods by maintaining the advantageous characteristics of each method (hybridization).

This often results in a rigid combination of two optimization methods. For instance, the first optimization algorithm would work the whole time whereas the second one would only be switched on from time to time to exploit its specific features. Or both methods would be applied alternately according to a fixed schedule.

These rigid schemata lead to several disadvantages:

- The utility of each separate method compared to each other is not evaluated during an optimization run and can therefore not be taken into consideration.
- The methods are applied simultaneously without taking into account an adapted distribution of resources.
- The methods are applied successively.

A large number of problems can be solved successfully using these simple hybridization methods. However, confronted with the optimization of complex systems the listed disadvantages become noticeable. These complex systems often require the application of more than two optimization methods. Furthermore, the combination and chronological application of the methods constitutes a difficult question. We have to find a technique which automates the combination and interaction of several optimization algorithms, especially for real-world applications.

For the solution of real-world applications it is often difficult to decide which of the available Evolutionary Algorithms are best suited and how the operators and parameters should be combined.

We present two extensions to Evolutionary Algorithms. The first one allows the combination of several different Evolutionary Algorithms. It is called *application of different strategies*. The second extension enables the interaction of the different strategies as well as an efficient distribution of the resources. It is called *competing subpopulations*.

Both extensions are based on the regional population model. This population model is described in Section 7.3, p.43. The regional population model is often referred to as migration model, coarse grained model, or island model. Many papers were published about this population model. Quite a few focus on the parallel implementation. However, this population model is not only useful for a parallel implementation. Even when used in a serial manner the regional population model proves advantageous compared to a global population model (panmictic population).

The main feature of the regional population model is that parents are selected in a regionally separated pool. The whole population is subdivided into subpopulations which are insulated from each other. An exchange of information (exchange of individuals) between the subpopulations takes place from time to time. This process is called migration.

Both extensions directly adapt themselves to the known structure of Evolutionary Algorithms (see figure 2-2, p.4). The *application of different strategies* is not detectable in this structure. It is part of the implementation of the high-level operators. For the *competing subpopulations* an extra operator competition is necessary and can directly be added to the structure. This extension is the only change made.

8.1 Different strategies for each subpopulation

Evolutionary Algorithms provide a large number of different operators and methods. Thus they can be applied broadly. The search strategies can be adapted to a wide range of problems. On the one hand, there are operators for parameter optimization, sequence optimization, up to the solution of very specific problems. On the other hand, it is possible to vary the search from a globally oriented search to a locally oriented search by setting appropriate parameters. Thus, the user is provided with an extensive toolbox. Now he need only select the appropriate methods.

If the user is well familiar with the type of the problem the choice of a good search strategy is not very difficult. However, in real-world application the necessary system knowledge is seldom available. Another point is that one search strategy is rarely best suited or sufficient. Mostly, the start of a search will have to be done with a different strategy than the end. The *application of different strategies* is an answer to this problem.

The application of different search strategies for each subpopulation is mentioned in a few of the early papers about parallel evolutionary algorithms. In [Tan87] TANESE describes the use of different mutation and recombination probabilities for some or all subpopulations. The results of the experiments indicate that the regional population model with different parameters for each subpopulation was more robust than the use of identical parameters for which the best values are not known.

The *application of different strategies* is the result of the definition of specific strategy parameters for every subpopulation (e.g. different mutation and recombination operators and/or parameters). In this way it is possible to specify a different behavior for every subpopulation of the Evolutionary Algorithm. To what extent the strategies differ from each other only depends on the application, the aims of the different strategies, as well as the possibilities of the optimization tool used.

On the one hand, strategies might differ with regard to only one parameter (e.g. mutation range which influences the size of the mutation steps). On the other hand, completely different types of Evolutionary Algorithms can be used. The principle of *application of different strategies* in itself does not pose any limitations.

The different strategies, however, do not only work concurrently. Rather, one strategy can be the basis for the success of another strategy. In this case the strategies support each other, i.e.

the simultaneous application of the different strategies leads to success. Thus, the *application of different strategies* leads to a cooperation between subpopulations.

8.1.1 Order of Subpopulations

It is necessary to find a measure for the evaluation of the application of different strategies which enables an assessment of the success of each strategy.

The success of a subpopulation is measured by its rank within the order of subpopulations. In this case, a low rank is better than a high rank. This approach is analogous to the ranking during the fitness assignment/selection process. From the rank of a subpopulation we are able to assess the utility of a subpopulation in comparison to the other subpopulations.

To calculate the order of subpopulations they have to be sorted according to one criterion. Since a subpopulation is made up of a number of individuals its quality results from the properties of its individuals.

The calculation of the order of subpopulations is carried out according to the following system:

1. A number of the best individuals is selected from each subpopulation for the evaluation. This can either be the best individual, a number of individuals, or all individuals of the subpopulation.
2. These individuals are ranked according to their quality (i.e. their fitness value). The evaluation of the individuals is identical to the fitness assignment procedure. We suggest the use of linear ranking. This assigns a quality value (fitness value) to each individual compared to the quality of the other individuals in the population (see Section 3.1, p.9).
3. The fitness of the individuals of each subpopulation is combined to an evaluation of the subpopulation (e.g. calculating the average fitness value of the individuals in a subpopulation).
4. By sorting the evaluation of the subpopulations we obtain their order.

The order of subpopulations offers a temporary picture. The order can fluctuate considerably within several generations, depending on the applied operators and parameters (especially if the strategies are similarly successful). It is therefore of advantage for the assessment and visualization if the order of the subpopulations is filtered. In this way a *position value* is calculated from the *ranking value*, which constitutes a weighted average of the rank of the subpopulation of previous generations.

$$position\ value_{generation} = 0.9 \cdot position\ value_{generation-1} + 0.1 \cdot rank_{generation} \tag{8-1}$$

The applied parameters of the filter in equation 8-1 prevent a strong fluctuation of the position values. The sum of both parameters must be 1. Higher values for the first parameter lead to a higher damping. Similar values are used in automatic control applications for this simple filter type.

The smaller the position value the greater the success of a subpopulation. If the rank of a subpopulation does not change for a long time the position value becomes equal to the rank. The position value is used for all following calculations and visualizations to illustrate the order of subpopulations.

8.2 Competition between subpopulations

A logical extension of the regional population model with the application of different strategies is the principle of competing subpopulations. During the application of different strategies within the regional model the size of every subpopulation (number of individuals) remains constant. Even when a strategy is not successful it still uses the same resources.

When using competing subpopulations this fixed amount of resources is not kept up. Instead the size of a subpopulation is made dependent on the current success of its strategy. Successful subpopulations receive more resources, less successful ones have to transfer resources to other subpopulations.

Every time a competition is executed between competing subpopulations the following steps have to be carried out:

- calculation of the order of subpopulations (see Section **Fehler! Verweisquelle konnte nicht gefunden werden.**),
- calculation of the division of resources,
- execution of distribution of resources.

By means of an appropriate algorithm the resources are efficiently redistributed as soon as there is a shift within the success of the subpopulations.

So far little has been published on the application of competing subpopulations. [SVM94] introduces a simple variant of competing subpopulations for strategy adaptation. Here, the order of subpopulations was not calculated. Instead, only the best subpopulation was determined on the basis of the best individual of the whole population. Only the best subpopulation received individuals from all other subpopulations. The competition selection took place uniform at random. In [SVM96] this concept was extended. By introducing the resource consumption parameter it becomes possible to control the relative size of a subpopulation. This version represents the basic model of competing subpopulations. The greatest simplification concerns the division of resources which just depends on the overall best individual. Thus, a subpopulation is always unsuccessful if it does not contain the best individual of the population. This method prefers only one strategy, all others are neglected. Especially during the transition between successful strategies this can lead to an ineffective distribution of resources.

8.2.1 Division of Resources

For every competition between subpopulations the available resources are redistributed. This can be done by determining successful and less successful subpopulations, or by a weighted division of the resources.

If we take a closer look at this process of the division of resources we find similarities to the fitness assignment of Evolutionary Algorithms (see Section 3.1, p.9). Fitness assignment means that every individual is assigned a reproduction probability/fitness (possible number of offspring) depending on its objective value compared to the objective values of the other individuals in the selection pool. The fitness of an individual is converted into resources enabling the individual to produce offspring.

The properties of the fitness assignment can directly be applied to the division of resources of competing subpopulations. This leads to a weighted division of the resources. It is known that

procedures based on the ranking of individuals are most robust. The (filtered) position value, see equation 8-1, serves as the basis for the division of resources.

In the next step every subpopulation is assigned a part of the available resources according to its rank/position value. Linear and non-linear ranking are well-known methods from the field of fitness assignment. The parameter of these methods is the selection pressure. In analogy we define the division pressure *DP*. This parameter determines how the resources are distributed. For a low division pressure the differences in the resources of every subpopulation are small. When applying a high division pressure, especially for non-linear ranking, the best subpopulation obtains a much larger share of resources. For the other subpopulations the share of resources is clearly smaller. Nevertheless, the other good subpopulations receive a larger share of resources than the less good subpopulations.

Fig. 8-1. Division of resources for subpopulations: linear and non-linear ranking and different values of division pressure

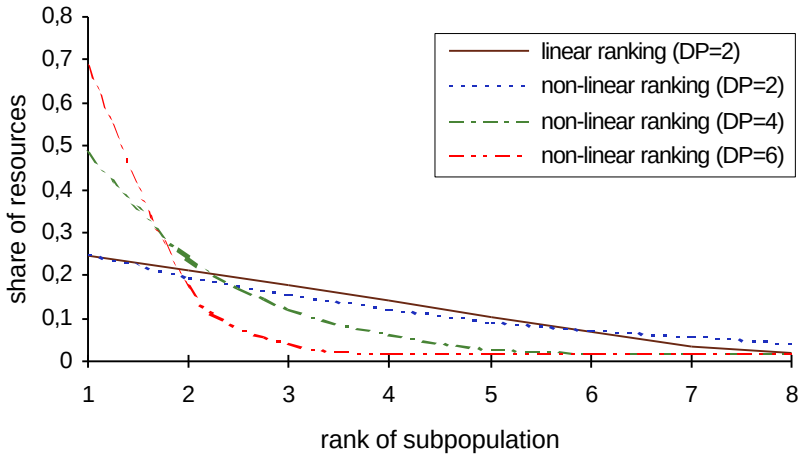


Figure 8-1 shows the division of resources of eight subpopulations for different values of the division pressure. A minimum of resources (subpopulation minimum, here 1%) needed for the survival of every subpopulation is taken into account.

When applying linear ranking a maximum division pressure of only 2 is possible. If the best subpopulation(s) is to be given stronger preference we have to use non-linear ranking. In this way values of division pressure up to *NumberofSubpopulations*-2 can be applied.

8.2.2 Distribution of Resources

The following parameters/methods have to be defined in order to execute the distribution of resources:

- resource consumption by the individuals,
- competition interval: frequency of a competition for resources,
- competition rate: maximum share of resources which has to be transferred by less successful subpopulations,
- competition selection: type of selection of resources to be transferred,
- subpopulation minimum: minimum size of an unsuccessful subpopulation.

However, all these parameters can be set with default values for nearly all applications. Only for very special problems different parameter values should be defined. Thus, the use of competing subpopulations imposes no additional parameters for the standard user.

8.2.3 Resource Consumption

So far we have only mentioned resources. However, what is missing is the conversion to the number of individuals used by the Evolutionary Algorithm. The resource consumption indicates how many resources are needed per individual.

The simplest variant is that every resource unit corresponds to one individual. This assumes that the individuals of all strategies consume the same amount of resources.

However, it is more realistic for individuals of different strategies to differ in their consumption of resources. This assumption permits a better modeling of the competition between subpopulations/strategies.

For instance, three different strategies with the same amount of resource units can provide for a varying number of individuals depending on the resource consumption parameter. The following example assumes 10 resource units for every strategy:

- resource consumption = 1: 10 individuals,
- resource consumption = 5: 2 individuals,
- resource consumption = 0.1: 100 individuals.

The parameter of resource consumption determines inverse proportionally to what extent a subpopulation will grow when assigned additional resources. At the same time the resource consumption controls whether a strategy will work with a small number of individuals (high consumption of resources), or with a large number of individuals (low consumption of resources).

8.2.4 Competition Interval and Competition Rate

The competition interval defines the points in time for a competition between subpopulations. It also indicates the span of time between competitions in which the subpopulations exist without any change of available resources and it permits the subpopulations to develop for the next competition.

It is easiest to specify a defined number of generations as competition interval. However, it is also possible to define the time of the competition depending on other events. One possibility is to carry out a competition every time a subpopulation has made a clear progress (we are aware that the definition of a “clear progress” is difficult and problem-specific).

The maximum amount of resources transferred by one subpopulation during a competition is determined by the competition rate. The competition rate is specified proportionally to the current resources of the subpopulation (and not as a fixed value). This ensures that subpopulations with few resources transfer a smaller amount of resources than subpopulations with many resources. The competition rate should always involve only a part of the resources of a subpopulation.

Following is a list of reference values for the competition interval (depending on the average or initial number of individuals per subpopulation) and the competition rate which proved to be suitable as predefined values.

$$\textit{competition interval} = 20\% \cdot \textit{NumIndSubPop}\ [\text{generations}] \quad (10\% - 50\% \Rightarrow 4 - 20\ \text{generations}) \tag{8-2}$$

$$\textit{competition rate} = 10\%\ [\text{resources of subpopulation}] \quad (5\%\text{–}20\%\ \text{of resources of subpopulation}) \tag{8-3}$$

If the competition interval has a very small value (< 6 generations) the competition rate should also be very small. This avoids a too rapid redistribution of resources between subpopulations. If a higher competition rate (> 10%) is used, a higher competition interval should be selected.

8.2.5 Competition Selection

The competition selection indicates how individuals are selected which are removed from less successful subpopulations.

There are several ways to select the individuals:

- the worst individuals,
- uniform at random selected individuals, and
- the best individuals.

A point in favor for selecting the worst individuals is that unsuccessful subpopulations would not be further weakened during a competition. Consequently, selecting the best individuals means that unsuccessful subpopulations would be further punished than just by the decrease in the number of individuals. The random selection of individuals is in-between the other two options. The selection of the best or worst individuals can be executed by means of one of the known selection methods of Evolutionary Algorithms (see Chapter 3, p.9).

8.2.6 Subpopulation Minimum

To avoid the complete disappearance of less successful subpopulations it is necessary to specify a minimum subpopulation size and/or a minimum amount of resources retained by the subpopulations. Resources can only be removed until this minimum is reached. Afterwards its size remains constant (until a possible success of the subpopulation).

The subpopulation minimum can be expressed by:

- a fixed amount of individuals,
- a proportion of the average resources/size of the subpopulation, or
- a proportion of all available resources

A better comparison between different-sized subpopulations is possible when the subpopulation minimum is specified proportionally to the subpopulation size rather than stating a fixed amount of individuals/resources. On the other hand, it is often known what minimum size a subpopulation needs in order to work. The most flexible option is the definition as proportion of the overall available resources.

$$subpopulation\ minimum = \begin{cases} 6\ (4-10)\ [\text{individuals}] \\ 20\%\ (10\%-30\%)\ [\%\ \text{of subpopulation}] \\ 4\%\ (1\%-10\%)\ [\%\ \text{of all resources}] \end{cases} \quad (8\text{-}4)$$

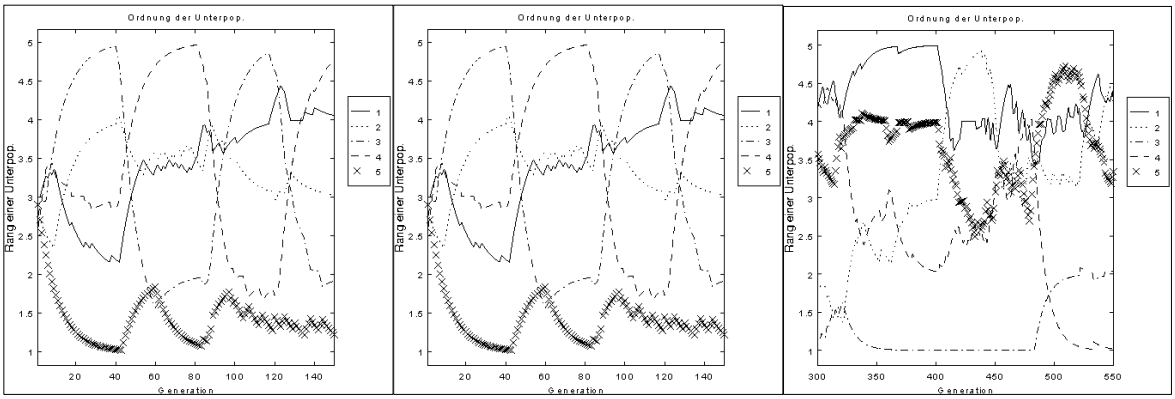
The successful application of most Evolutionary Algorithms can hardly be guaranteed if the subpopulation minimum is set to a very low value (4-6 individuals). A success of such a small subpopulation is only achievable if its strategy is superior to the strategies of the other subpopulations.

8.3 Application of Different Strategies

This chapter illustrates the application of the introduced extensions using a simple and easily comprehensible objective function.

Figure 8-2 shows an example of the course and the results of an optimization of DEJONG's Function 1 (hyper sphere with 10 dimensions) using the *application of different strategies*. The optimization was executed with 5 subpopulations, with 15 individuals each. Each subpopulation used a different strategy (different parameters of the real-valued mutation operator): 1: large mutation steps; 2: middle-sized mutation steps; 3: small mutation steps; 4: tiny mutation steps; 5: middle-sized and small mutation steps (mutation range: 1e-2, 1e-4, 1e-6, 1e-8, 2e-1; mutation precision: 6, 6, 6, 6, 16). The smaller the mutation steps the more locally oriented the search. All strategies used the same recombination operator (discrete recombination), all other parameters were identical too (e.g. *generation gap*: 0,9). Every 40 generations a migration between subpopulations took place (complete net structure). See Section 7.3, p.43 for an extensive discussion of these operators.

Fig. 8-2. Application of different strategies, order of subpopulations; left: beginning of optimization run, middle: middle phase, right: final phase



The left diagram in figure 8-2 shows the beginning of the optimization. We can see that strategy 5 is the most successful one at the beginning. This changes after 200 generations. Now strategy 2 becomes better, taking up the leading position after 220 generations (see middle diagram). Strategy 3 becomes the best one after 300 generations, followed by strategy 4 after 500 generations (see right diagram). Strategy 1 shows a good behavior during the first 50 generations (see left diagram), however it then falls behind and will not be able to be successful at any time.

This is not surprising if one knows the properties of the example function. Large mutation steps are especially successful at the beginning. Later they only lead to deterioration. Similarly this applies to each of the strategies 2-4. If larger steps still make progress, small steps are too slow. Thus, each of the first four strategies is best during a specific period.

Strategy 5 takes up a special position. The left diagram shows clearly that strategy 5 is the most successful one at the beginning: larger steps now and then, and mostly smaller steps is a good strategy. After 200 generations strategy 5 runs out of breath. It rarely produces good individuals anymore. Now one of the more specialized strategies is advantageous.

8.4 Application of Competing Subpopulations

The following example demonstrates the application of competing subpopulation. It uses the same objective function and the same strategies as the example above with the addition of competition between subpopulations.

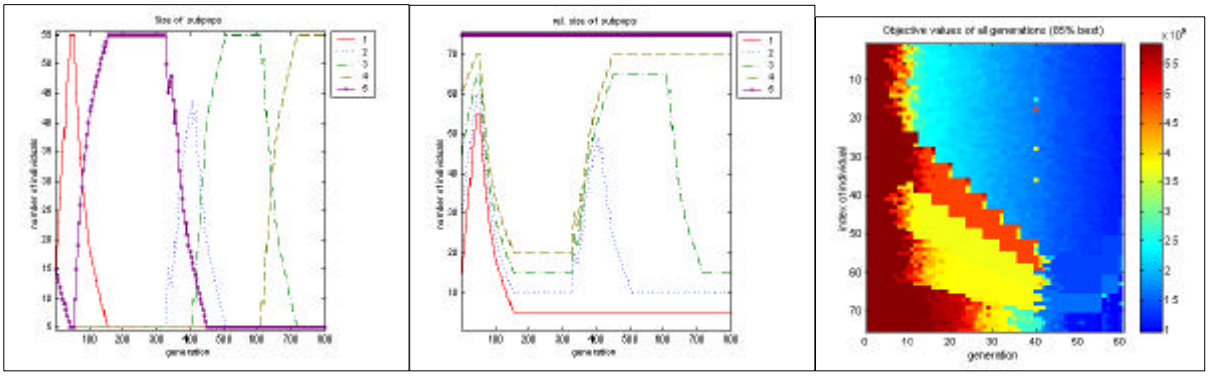
The competition parameters were set as follows:

- only the best subpopulation receives resources,
- the resource consumption for each individual is 1,
- the competition interval is 4 generations,
- the worst individuals are removed from less successful subpopulations,
- the subpopulation minimum is set to 5 individuals.

The example presents a powerful but still comprehensible variant of competing subpopulations. Figure 8-3 shows selected sections of the course and the results of the optimization run.

The left and middle diagram in figure 8-3 show the course of the competition between the subpopulations clearly. The left diagram displays the absolute size of the single subpopulations, and the middle one their relative size using a stacked line plot. The number of individuals in the middle diagram results from the distance between two contiguous lines, i.e. the size of each subpopulation can be identified without a legend.

Fig. 8-3. Competing subpopulations; left: size of subpopulations, middle: relative size of subpopulations, right: objective values of all individuals at the beginning of the optimization run



The example presents a powerful but still comprehensible variant of competing subpopulations. Figure 8-3 shows selected sections of the course and the results of the optimization run.

The left and middle diagram in figure 8-3 show the course of the competition between the subpopulations clearly. The left diagram displays the absolute size of the single subpopulations,

and the middle one their relative size using a stacked line plot. The number of individuals in the middle diagram results from the distance between two contiguous lines, i.e. the size of each subpopulation can be identified without a legend.

At the beginning of the optimization run the size of subpopulation 1 (large mutation steps) increases for a short time. After about 50 generations subpopulation 5 grows (large and middle-sized mutation steps). Subpopulation 2 is only successful after 320 generations (middle-sized mutation steps). Subpopulation 3 obtains more individuals after 400 generations (small mutation steps). For about the last 200 generations subpopulation 4 (tiny mutation steps) is the most successful one, having the most individuals. At all times the less successful subpopulations/strategies use only a small proportion of the overall population.

An additional insight into the course of the optimization is given by the right diagram in figure 8-3. It shows the objective values of all individuals over several generations (the individuals with the lowest index numbers form subpopulation 1, the individuals with the highest index numbers subpopulation 5, the other subpopulations are in-between). The division between the subpopulations is clearly visible and enables the user to indirectly recognize the size of every subpopulation. The diagram shows the beginning of the run where subpopulation 1 was successful and increased in size. The reason for this can also be seen: the greatest improvement of the objective values takes place in subpopulation 1. Subpopulations 2, 3, and 4, however, do not show much of an improvement. Only subpopulation 5 demonstrates a clear progress during the first 40 generations. The migration in generation 40 distributes the best individuals of the subpopulations. Thus, the less successful subpopulations are improved towards the level of quality of the best subpopulation. For further diagrams depicting the transitions between successful strategies please refer to [Poh99b], Section 4.6.

8.5 Conclusion

The *application of different strategies* permits the simultaneous application of different parameter settings. This accelerates and facilitates an optimization in contrast to the execution of multiple independent experiments. Results are presented more clearly and in a compact manner. It is very easy to identify successful and unsuccessful strategies as well as the varying success of strategies during an optimization run. Furthermore, the cooperation between strategies during a run can lead to their achieving better results than separated strategies. The success of a strategy at the beginning of a run often is the basis for the subsequent success of another strategy.

The application of different strategies and competing subpopulations serves multiple aims at the same time:

- During the initial evaluation of the systems to optimize we could easily test a number of parameter settings to determine promising strategies for the respective problem class. Simultaneously we excluded unsuccessful strategies (competition between strategies).
- During the productive search of the optimization all promising strategies were used at the same time. This is particularly important. We could use one configuration for a large class of optimization systems. The user of the test tool need not bother with the configuration of the optimization engine. Nevertheless, the optimization algorithm is still powerful.

- Not all of the applied strategies are successful for each single optimization run. However, the division of resources ensured only a small amount of resources being used for currently not successful strategies.
- Very often more than one of the applied strategies is successful during an optimization run. Each of these strategies is particularly successful at a specific point in time.

The use of different strategies enables a higher level of the application of Evolutionary Algorithms, especially within the following two areas:

- testing a number of parameter settings to determine successful strategies and simultaneously exclude unsuccessful ones (competition between strategies),
- the simultaneous application of different strategies where each strategy is particularly successful at a specific point in time, i.e. the strategies supplement each other (cooperation between strategies).

The extension *competing subpopulations* is based on the application of different strategies, however, it goes one step further. The different strategies are not only applied simultaneously but the successful ones receive more resources than the less successful ones. This leads to a dynamic distribution of resources.

The resource distribution leads to an indirect adaptation of the strategy parameters during an optimization run. This means the user does not have to specify the parameters beforehand because they are indirectly set depending on the success of the subpopulations. This opens up new dimensions to the application of Evolutionary Algorithms.

By applying competing subpopulations those strategies less suitable for solving the problem will receive less resources. Thus, only a small amount of resources are used for testing unsuccessful strategies which in turn enables the testing of additional promising strategies for very complex problems.

The approach presented in this chapter eliminates restrictions of earlier publication ([SVM94], [SVM96]). The inclusion of the best, multiple, or all individuals of the subpopulations enables the calculation of a weighted order of the subpopulations. This leads to a multilevel assessment of the success of the strategies/subpopulations. The weighted division of resources permits a distribution to all subpopulations and not only to the best subpopulation. Especially when employing similar or supplementary strategies this leads to a more equitable division of the available resources.

The presented extensions of Evolutionary Algorithms, *application of different strategies* and *competing subpopulations*, are very powerful, especially in real-world applications. These extensions are an integral part of the GEATbx. The extensions allow the setup of powerful optimization algorithms applicable to a large class of real-world problems.

Both extensions constitute another step towards the development of powerful Evolutionary Algorithms for the solution of large and complex systems.

9 Combination of Operators and Options to Produce Evolutionary Algorithms

In the preceding chapters individual methods and operators were described. In this chapter I shall explain how complete optimization algorithms can be created from these operators. Each of these optimization algorithms represents an evolutionary algorithm.

As there is a large number of problem classes, and as a different optimization procedure is best suited to each problem class, I would like to make some recommendations here which often hold good. These recommendations are based both on my experience with the optimization of various problems, as well as on other authors' results taken from the relevant literature.

By using this approach the user is provided with insights into the specific properties of the operators and the dependencies between the operators. The user can better understand how the interplay of these properties and dependencies defines an optimization algorithm with certain properties. Based on this, is it easier to adapt the proposed algorithms to new problem areas. Moreover, it can be better estimated why an algorithm has difficulties or fails when faced with a particular problem.

Besides the choice of an optimization algorithm (combination of operators), the behavior of the optimization algorithms can be controlled via a number of options (also termed parameters). Since a lot of the options of an optimization algorithm cannot be determined exactly, in most cases a range for their values or a dependency of these values on other factors are specified. This renders the optimization algorithm more adaptable to different problem sizes and makes it easier to recognise which mutual dependencies have to be considered.

In the case of some procedures and operators, the options can be stated relatively independently of the problem which has to be solved. These procedures and the corresponding options will be explained and described in section 9.1, p.60.

For application to specific problem classes, specific operators and their options will be presented in section 9.2-9.5.

The specific problem classes can be divided into the following groups and variants:

- Parameter optimization (e.g. parameter identification) with further differentiation according to the representation of the variables:
  - globally oriented parameter optimization of real or integer variables, section 9.2, p.62,
  - locally oriented parameter optimization real variables, section 9.3, p.63,
  - parameter optimization binary variables, section 9.4, p.65,
- combinatorial optimization, section 9.5, p.65, e.g.:
  - travelling sales person TSP,
  - production scheduling (e.g. job shop scheduling JSS, vehicle routing problems VRP).

When describing optimization algorithms for the specific problem classes, I shall illustrate only the particular operators together with their options. In every other case the procedures described in section 9.1, p.60, together with their options, will be applied.

The values suggested in this chapter for the individual parameters/options can, of course, be changed. The user should, however, then know what effect these changes have and whether they necessitate changes to other parameters. This is usually only possible after a relatively long period of practice with and examination of the way the individual operators work as well as the function and combined effects of the parameters.

For a more detailed explanation of the significance of the individual operators and their parameters refer to chapter 2, p.3 and chapter 7, p.41.

9.1 Generally Adjustable Operators and Options

This section describes procedures and operators for which the options can be adjusted relatively independently of the problem which has to be solved. With these adjustments a robust working is provided for many problems which have to be solved.

9.1.1 Operators and Options for Fitness Assignment and Selection

As explained in section 3.1, p.9, a rank based procedure (*ranking*) should always be employed for fitness assignment . All of the procedures presented in sections 3.3-3.7 can then be applied for selection . The selection procedure used is less decisive than the value of the corresponding option – the selection pressure. A comparison of the procedures and an examination of their differences was made in section 3.8, p.21.

Using selection pressure values in the area of 1.5 to 2 worked well in many cases. A higher selection pressure should only be applied in the case of very large populations (over 500 individuals).

The *generation gap* option has a greater influence on the course which the optimization takes. This option states how many offspring will be produced in comparison to the number of individuals in the population. A *generation gap* of less than 1 ensures that less offspring are produced than there are individuals in the population. As a result, some of the parents survive into the next generation, in this case the best individuals so far. This conforms to an elitist strategy . With a generation gap of exactly 1, exactly as many offspring as parents are produced and the offspring replace all the parents. A generation gap of a little less than 1 (0.8–0.99) ensures that very good solutions which have been found are only replaced by even better solutions.

9.1.2 Operators and Options for Application of Different Strategies and Competition between Subpopulations

Where, in the following, several procedures or variants of operators are described for a problem class, these can be applied together during an optimization run. This takes place via the application of different strategies as described in section 8.1, p.48. This application of different strategies allows the simultaneous deployment of various strategies. In this way, one can investigate which of these strategies is successful for the problem to be solved. It is then sufficient,

during subsequent optimisation, to limit oneself to the deployment of these successful strategies.

The simultaneous application of several mutation operators or letting these work with different parameters is an example of this. As a result, various search strategies are applied simultaneously. This often leads to better results. For example, three subpopulations can be used, whose parameters only differ in the mutation range (large, medium, small) – see section 5.1, p.33. In this way a rough, a medium and a fine search are carried out simultaneously. The rough search is usually successful at the beginning of a run and the fine search at the end.

Due to the additional deployment of *competition between the subpopulations*, section 8.2, p.50, a simultaneous and efficient distribution of the computational resources between the strategies, in favour of successful strategies, is achieved. This is then particularly noticeable if different strategies are successful at different times during an optimisation run. The strategies which are most successful at the respective times are allocated more resources and can, as a consequence, step up their search for better solutions. As soon as another strategy, and thus another subpopulation, becomes more successful, it is allocated more resources (in this case more individuals) and is able to carry out its search more effectively.

Both of these concepts, *application of different strategies* and the extension to *competition between strategies*, can be applied universally. They prove particularly useful when searching for suitable procedures and operators for the solution of new systems whose behaviour is not yet so well known. By applying these concepts, particularly in the initial phase of work on a system, better results are achieved more quickly than when different strategies are tried out consecutively. Moreover, it is quite possible that one single strategy is not successful, whereas strategies being run together are successful within a relatively short space of time.

9.1.3 Operators and Options for Regional Population Model (Migration between Subpopulations)

Both methods (*application of different strategies* and the extension to *competition between strategies*) are based on the subdivision of the population into subpopulations, the *regional population model* (see section 7.3, p.43). The development in the subpopulations takes place separately for a certain isolation period. From time to time an exchange of (good) individuals is made between the subpopulations. In this way, information found in one subpopulation also reaches the other subpopulations.

When applying the regional model, one usually works with a migration time of 20 (up to 40) generations, independently of how long a run actually needs in order to achieve results. If more than 1000 generations are being worked with, the migration time can be raised to every 50 generations, for short runs of around 50 generations a migration time of 5-10 generations is better. The migration rate (proportion of the population to be exchanged) is usually set at 10%, the migration structure at unlimited (exchange between all subpopulations – complete net). More detailed information on the parameters is provided in section 7.3, p.43.

9.1.4 Summary of generally adjustable operators and options

The explanations in this section were able to show that many options of an evolutionary algorithm can be adjusted to very sensible and robust values, without having to be readapted to every problem.

At this point I would like to provide a short summary of the procedures explained so far and of a robust parameterisation:

- **Fitness assignment and selection**:
  Fitness assignment via linear ranking with selection pressure of 1.5-2, generation gap of 0.95, stochastic universal sampling as the selection procedure,
- **Migration**: (in the case of several subpopulations, regional population model)
  Migration time of 20 generations, migration rate of 10%, migration structure: complete net,
- **Application of different strategies**:
  Definition of different procedures or options for the individual subpopulations (e.g. different mutation ranges),
- **Competition between subpopulations**:
  Competition interval of 4-10 generations, competition rate of 10%, subpopulation minimum at 10%-20% of the initial size of the subpopulation.

This covers the generally applicable explanations and comments. The information in the following sections is related to specific problem categories.

9.2 Globally Oriented Parameter Optimization

A large part of the technical application problems concerns parameter optimization problems with real and integer variables, for which there is usually no (strong) correlation between the variables present. In this section I shall present those evolutionary algorithms which are especially suited to the globally oriented optimization of these problems. Here a globally oriented search is defined as one in which no assumptions are made as to the type of the target function and the search is mainly carried out along the co-ordinate axes. Each variable is changed separately.

9.2.1 Recombination

Almost all recombination operators can be used (including those for binary variables).Discrete recombination (`recdis`) is the most favourable and (less often) line recombination (`reclin`).

For integer variables line recombination and intermediate recombination (`recint`) can only be used to a limited extent.

The recombination rate should, as a rule, be set at 1.

9.2.2 Mutation

The mutation operators available for real (`mutreal`) and integer variables (`mutint`), section 5.1, p.33, can exhibit very different behaviour as a result of appropriate parameterization. For this reason, there is only one operator per variable type, whereby both work (almost) iden-

tically. An adaptation to the application case (rough mutation up to fine mutation) is achieved by the continuous adjustment of the corresponding options (and thus also the corresponding search strategy).

For a rough mutation, a mutation range of 0.2-0.05 is used, for a fine search down to values of 10^{-8} and lower. The mutation range defines the value of the largest mutation step in relation to the definition range of the respective variable/parameter.

A mutation precision of 16 (range of 8-20) achieves a sensible distribution of the mutation steps (mostly small mutation steps, large steps only occasionally).

A possible parameterization of the mutation for real variables using 4 different strategies could involve mutation ranges of [0.1, 0.03, 0.001, 0.0003], all with a mutation precision of 10. Thus, a rough screening of the search area is connected with a fine search. This is an approach which has proved itself in many practical applications.

The mutation rate should be set at (1/number of variables) and should only be increased in exceptions or in the case of selection pressure larger than 2.

The parameterisation of the mutation operator usually has the greatest influence on the effectiveness and thus the course of the optimisation. For this reason, the search for an effective algorithm for the solution of the current problem should be concentrated on this operator. By using different mutation ranges in the individual subpopulations (and thus the application of different strategies), an adjustment to the problem can be carried out quickly.

The high-level toolbox functions for globally oriented parameter optimization are `tbx3real` (real-valued representation) and `tbx3int` (integer-valued representation).

9.3 Locally Oriented Parameter Optimization

In this section I shall present those evolutionary algorithms which are particularly suited to the locally oriented optimisation of real variables. This comprises, generally speaking, application problems for which there is a strong correlation between the (real) variables. Due to this, the variables have to be changed simultaneously (co-ordinated) and it must be possible to search in all directions within the search area.

To this end, those search directions which are most likely to be successful have to be “learned”. Special mutation operators, originating from the evolution strategies area, section 5.3, p.35, are available for such problems. These attempt to learn the direction of an improvement and thus render a goal-oriented search possible.

However, the application of these operators is limited to smooth functions and low-dimensional problems (the larger the problem is, the longer the learning of a direction takes). If a target function is noisy, it is highly likely that these evolutionary algorithms will get stuck in the next local minima. In exactly the same way, the algorithms have almost no chance of getting out of a local minima again. This makes them unsuitable for multi-modal problems. For more detailed information on the application of these strategies refer to [Ost97] and [Han98].

9.3.1 Recombination

These procedures (usually) do not involve recombination.

9.3.2 Mutation

Special mutation operators are available (`mutes1`, `mutes2`) which originate from the evolution strategies area, section 5.3, p.35. Only the size of the initial step is specified for these operators. Subsequently these functions carry out an independent adaptation of the step sizes or of the search direction.

The mutation range is no more than a specification for the range of the initial step size. Values of 10^{-3} to 10^{-7} (related to the definition range of the individual variables, as this is implemented in the GEATbx) have worked well. The determination of a sensible initial step size is problem-specific and cannot be done easily.

It is also possible to work with different strategies when using these operators. These would differ in the mutation range (e.g. large, medium, small). Each of the strategies (subpopulations) therefore starts with a different initial step size range. However, as soon as the adaptation of the step sizes has taken place, the different mutation ranges no longer have any influence. This, consequently, only conduces to the determination of sensible or successful initial step sizes.

9.3.3 Fitness Assignment and Selection

It should be noted that these procedures for locally oriented optimization only work with a small number of individuals and that a lot of offspring are produced, of which only the best are added to the population.

In order to achieve this we have a choice of two possibilities. On the one hand, the population can contain few individuals which each produce a lot of offspring. Only the best of these offspring replace the parents and form the new population. For this, the following parameters for population size and selection should be used:

- population size: 1-5 individuals,
- generation gap: 3-10 (number of offspring per parent),
- selection pressure: 1 (no selection pressure).

On the other hand, one works with a larger population size, in which only the very best individuals produce offspring. Here, (almost) all offspring replace the parents and form the new population:

- population size: 5-20 individuals,
- generation gap: 1 (produce as many offspring as parents),
- selection: truncation selection with a selection pressure of 3-10 (only the very best individuals are selected and each produce several offspring).

The first variant is based on the original “definition” of these procedures and has its complete justification. In the case of the second variant, however, there are some small advantages, which have particularly come to the fore during practical application. Firstly, all individuals produced (offspring) can be integrated in the (later) evaluation as they are included in the population for at least one generation (in the case of the first variant only a small part of the offspring produced are included in the population). Furthermore, in the case of the second variant, an elitist selection can be defined very easily, by which the hitherto best individual survives in the population. For this one simply has to define a generation gap of somewhat less than 1 (e.g. 0.99).

The high-level toolbox function for locally oriented parameter optimization is `tbx3es1`.

9.4 Parameter Optimization of Binary Variables

The parameter optimisation of binary variables is applied, above all, for classic genetic algorithms. Here, the real or integer variables are transformed into a larger number of binary variables via discretization. The genetic algorithm then works on these binary variables. Before the objective function is calculated, the binary variables have to be decoded into their original format.

As these genetic algorithms are still in use, a number of corresponding operators are available. Moreover, there are some real-world problems which use binary variables directly.

9.4.1 Recombination

All recombination operators for binary variables can be used. Discrete recombination (`recdis`), which is identical to uniform crossover, and the operators with *reduced surrogate* (`recsprs`, `recdprs`, `recshrs`,) are favourable. The recombination rate is usually set at 0.7 in the relevant literature.

9.4.2 Mutation

An operator (`mutbin`) is available for the mutation of binary variables. Only the mutation rate can be regulated for this operator, there can be no other possibilities for influencing binary variables. The mutation rate is usually set at (1/number of variables) and is only increased in exceptional cases.

The high-level toolbox function for parameter optimization of binary variables is `tbx3bin`.

9.5 Combinatorial Optimization

The evolutionary algorithms presented so far are meant to be used for parameter optimization. Beside that, there is a further problem category of considerable size; that of combinatorial optimization. Problem categories that are known within combinational optimisation are travelling sales person TSP, quadratic assignment problem QAP, production scheduling (e.g. job shop scheduling JSS) and vehicle routing problem VRP. These problems are often called ordering or permutation problems.

At this point, however, two restrictions must be placed. There exists an extremely large number of application fields for combinational optimization, all of which call for a slightly different procedure. Especially with regard to solving substantial problems special procedures and operators must be applied in order to achieve a result with justifiable effort. This cannot be covered comprehensively within this section.

Some information and hints shall be provided here that have proved themselves within the scope of this work. These are able to show how the ‘new’ application of combinational optimization falls into line with the structure of evolutionary algorithms used so far and is correspondingly filled with the specific operators. The statements from section 9.1, p.60, concerning the application and parameterization of the generally adjustable procedures and operators can be directly applied here.

9.5.1 Recombination

Within the scope of the GEATbx two recombination operators for combinatorial problems have been used: partial matching crossover PMX (`recpm`) and generalized position recombination GPX (`recgp`). These have been designed especially for the handling of sequence / ordering / permutation problems. All of the operators ensure the validity of the individuals generated by recombination.

The fundamental difference between the operators lies in the information they retain during recombination. Three properties are important for combinatorial optimization: the neighbourhood relation between the variables (relative relation), the absolute relation of the variables, and the absolute position of the variable. As regards the TSP the relative relation is crucial, the information on the absolute position is not. In the case of the QAP the absolute position of the variables is decisive but not the neighbourhood relation to other variables. As for the JSS again the absolute relation is important. The problem to be solved has to be examined with regard to which information should be retained or to which of the problem categories it belongs.

GPX particularly retains the absolute relation of the variables, but also shows good behaviour in retaining the relative relation. *PMX* is best suited to retaining the relative relation of the variables when working with locally optimized individuals.

By applying various strategies one can also quite simply determine which of these operators (or further recombination operators for combinatorial problems) is best suited for the solution of a problem. The subpopulations use one of the recombination operators each. The best subpopulation then provides an indication of which of the operators used should be applied for further optimization runs. Quite often this might be more than one operator.

The recombination rate is mostly set to values of 1 or a little below 1.

9.5.2 Mutation

There are several operators available for mutation (swap mutation (`mutswap`), move mutation (`mutmove`), invert mutation (`mutinvert`), scramble mutation (`mutrandperm`)).These mutation operators can be adjusted to mutation steps of varied sizes by specifying the mutation range and the mutation precision. This leads to a functionality in terms of large and small changes. The mutation range determines the maximum distance between the variables to be changed within the individual. The mutation precision serves to distribute the mutation steps in this area for a distribution between large and small mutation steps.

The mutation rate serves to control the number of changes per individual. In most cases appropriate behaviour should be obtained with a mutation rate of (1/number of variables), which corresponds to e.g. an exchange/swap of two variables or the move of one variable.

The high-level toolbox function for combinatorial optimization is `tbx3perm`.

9.6 Parameter Optimization of Variables of different Representations

I would like to end this chapter by looking at a problem which often occurs in practice and which the previous sections on parameter optimization in this chapter have in common. Up to

now it has been assumed that all the variables of a problem to be solved exist in the same representation. This is, however, not always the case. When solving some problems in practice variables of differing types have to be dealt with simultaneously. This means that some of an individual's variables are real and others are integer or binary.

As previously mentioned, each variable representation has different operators which are best suited to it. In addition, certain singularities have to be taken into account for each data type. A way has to be found to optimise all the variables simultaneously despite these different requirements.

There are two procedures for the processing of problems with variables of different representations:

- In the first case, the appropriate operators can be applied to the respective variables (as explained in the previous sections). If real and binary variables are present, two recombination and two mutation operators with the individuals' respective variables are called within the evolutionary algorithm. This leads, besides an increased complexity within the evolutionary algorithm, above all to a significant increase in complexity within the parameterisation of the evolutionary algorithm which particularly detracts from clarity. In effect, this is a question of complex implementation and will not be looked at further here.
- In the second case the different variable types are converted into one type with which the evolutionary algorithm works. On the face of it this seems to be a more cumbersome method. It has, however, proved itself, in practical application, to be much more easily manageable and simple to implement.

If all the variable types are converted into one representation, the different combinations of types present must be considered and it must be decided into which representation they will be convertd. In principle, it is possible to convert all the variables into each representation. Some of the advantages and disadvantages, as well as what should be taken into consideration, will be explained in the following.

9.6.1 Integer and Binary Variables

First I would like to look at the combination of integer and binary variables as it is possible to do this very simply and without restrictions. An integer variable within the limits [0, 1] is equivalent to a binary variable. The evolutionary operators for integer variables work well with these binary variables. The settings for integer variables are used, see section 9.2, p.62.

9.6.2 Use of Integer Representation

If real and integer/binary variables are present, the integer representation can be used. In this case there are no restrictions for binary and integer variables. These can retain their original representation.

For real variables, on the other hand, a conversion or adaptation must take place. If the definition range of the real variables is very broad and the limitation to integer values does not cause any problems, this is the best method to choose. If, however, a higher accuracy of the real variables' values is necessary, then the real definition range must be discretized. These discrete values can then be mapped onto integer values which are used during optimization.

This conversion or adaptation can be achieved by means of a relatively simple transformation of the corresponding variables in the objective function of the problem. In doing so, one must determine with which accuracy each of the real variables is to be optimized. This accuracy can be determined by using a scaling factor for each variable. A scaling factor of 10 means that this variable is processed with a minimum resolution of 0.1 ($^1/_{10}$). Correspondingly, a scaling factor of 200 signifies a minimum resolution of 0.005 ($^1/_{200}$).

Subsequently, this is used as a basis to increase the definition range of the variables (for optimization) by this accuracy or scaling factor. The evolutionary algorithm and predetermined operators work with this definition range and integer variables. Before each evaluation of the objective function, the integer variables are converted with the determined scaling factor and are then available as real variables in the desired resolution.

I would like to illustrate the procedure described using two examples. In the first, a real variable is used with a definition range of [-1, +1]. An accuracy of 0.01 is desired for this variable. For this, the original limits of the definition range of this variable are multiplied by a scale factor of $^1/_{0,01}$=100. During optimization this variable is treated as an integer within the limits [-100, +100]. Before the target function is evaluated this variable is divided by the scaling factor of 100. The real variable which is to be used is available again with the desired accuracy.

In the second example we have 3 variables, the first and third as real variables, the second as an integer variable. The first variable should have a minimum resolution of 0.2, the third one of 0.005. The scale factor for all 3 variables must therefore be determined at [5, 1, 200]. By multiplying by this scaling factor the definition range for these variables is increased (or maintained for the integer variable). Before the evaluation of the objective function, division by the scaling factors results in a conversion of the integer variable values into real variable values of the desired resolution within the problem specific definition range.

The application of this approach is very flexible. On the one hand, the desired resolution of the variables can be influenced exactly. On the other, all the deployment possibilities for the evolutionary operators for integer variables continue to be available.

The only disadvantage is that, in the case of the formerly real variables, the direct relationship between the variables' values during optimization and the problem specific value is lost. The variable values from optimization first have to be converted with the scaling factor.

This disadvantage can, however, be put up with and does not outweigh the good manageability of this method. When solving systems with variables of different representations, the use of integer variables for optimization has proved itself and is recommended here for application.

For the combination of operators and their parameterization to evolutionary algorithms, the details given on parameter optimization of the corresponding representation in the previous sections, especially section 9.2, p.62, hold good. I shall, therefore, not go into it further at this point.

9.6.3 Use of Binary Representation

Besides a conversion of real variables into an integer representation, binary representation can also be used as the smallest common denominator. For this, all integer and real variables are transmuted into binary representation with which the evolutionary algorithm then works. This approach corresponds to classical genetic algorithms.

The desired resolution of each variable has to be predetermined for this conversion too. In the case of integer variables this occurs straightforward, an integer variable within the limits [10, 500] can be coded via $\log_2(491) < 9$ binary variables. In the case of real variables the data given above apply. One must, however, take into account that base 2 is being used. For a real variable, which is decoded with 10 binary variables, this means a resolution of around 0.001 ($1/2^{10}$) with respect to the definition range (in the case of linear scaling).

As pointed out in section 9.4, p.65, many users still work with this method. It has both historical importance and justification. It has been shown in many applications, however, that the use of evolutionary operators available for integer and real variables result in more effective evolutionary algorithms. Nevertheless, the decoding of binary variables into integer (`bin2int`) and real variables (`bin2real`) is supported in the GEATbx.

9.6.4 Use of real representation

In principle, it is also possible to work with a real representation and to carry out a limitation of the variable values in the objective function (rounding off the variables to the next integer value).

However, this can have the consequence that a mutation of originally binary or integer variables has no effect (due to the subsequent rounding off). This can be particularly extreme in the case of binary variables because only a change of, on average, a quarter of the definition range results in a change in the binary variable value. In the case of integer variables this problem becomes less serious, the broader the definition range of the corresponding variable is.

For this reason, real representation should only be used in special cases or when combining integer values with a broad value range (difference between lower and upper limit is greater than 1000) and real variables.

9.7 Summary

In this chapter I have presented details on the combination and parameterization of powerful evolutionary algorithms. For this, a subdivision into frequently occurring problem classes was undertaken (parameter optimization of real, integer and binary variables as well as sequence or ordering optimization). Details were provided at the beginning of the explanations which can be applied similarly to most of the problem categories (fitness assignment, selection, application of the regional population model with different strategies, competition). Thereafter, a compilation of the operators with robust parameters, which are specific to each problem class, were presented.

There are several possible solutions to problems with variables of different representations. According to each type of variable the advantages and disadvantages of the conversion and application of the different representations were addressed. During application the conversion of real variables into integer representation and its use for the optimization has proved successful and is thus recommended.

The evolutionary algorithms presented in this chapter provide the basis for their application to systems of various problem classes. After the analysis of the system to be solved the user may

select the appropriate evolutionary algorithm. An adaptation is only necessary if there are problem specific requirements.

10 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 10.1, p.71 contains papers and books about Evolutionary Algorithms in general. Section 10.2, p.82 collects publications about population models and parallel implementations of Evolutionary Algorithms. Section 10.3, p.84 presents papers on combinatorial optimization using Evolutionary Algorithms. Section 10.4, p.85 contains papers and books on visualization. Section 10.5, p.86 contains papers and books on the special topic polyploidy and Section 10.6, p.87 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.

10.1 Evolutionary Algorithms

[Ack87] *Ackley, D. H.*: A connectionist machine for genetic hillclimbing. Boston: Kluwer Academic Publishers, 1987.

[Ack93] *Ackermann, J.*: Robuste Regelung. Berlin, Heidelberg, New York: Springer-Verlag, 1993.

[AISB96] *Fogarty, T. C.*: Evolutionary Computing. Proceedings of AISB Workshop on Evolutionary Computing 1996, Vol. 1143 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1996.

[Alt95] *Altenberg, L.*: The Schema Theorem and Price's Theorem. In [FGA3], pp. 23-49, 1995.
http://dynamics.org/~altenber/PAPERS/STPT/

[AGP94] *Kinnear, K. E.*: Advances in Genetic Programming. Cambridge: MIT Press, 1994.

[AGP96] *Angeline, P. J. and Kinnear, K. E.*: Advances in Genetic Programming II. Cambridge: MIT Press, 1996.

[AGP99] *Spector, L., Langdon, W., O'Reilly, U.-M. and Angeline, P. J. (eds.)*: Advances in Genetic Programming III. Cambridge: MIT Press, 1999.

[Bäc93] *Bäck, T.*: Optimal Mutation Rates in Genetic Search. In [ICGA5], pp. 2-8, 1993.
http://lumpi.informatik.uni-dortmund.de/people/baeck/papers/icga93.ps.Z

[Bäc96] *Bäck, T.*: Evolutionary Algorithms in Theory and Practice – Evolution Strategies, Evolutionary Programming, Genetic Algorithms. New York, Oxford: Oxford University Press, 1996.
http://www.oup-usa.org/docs/0195099710.html

[BH91] *Bäck, T. and Hoffmeister, F.*: Extended Selection Mechanisms in Genetic Algorithms. In [ICGA4], pp. 92-99, 1991.

[BS93] *Bäck, T. and Schwefel, H.-P.*: An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation, 1(1), pp. 1-23, 1993.
http://lumpi.informatik.uni-dortmund.de/people/baeck/papers/ec93.ps.Z

[BS96] *Bäck, T. and Schütz, M.*: Intelligent Mutation Rate Control in Canonical Genetic Algorithms. In *Ras, Z. W. and Michalewicz, M.*: Foundation of Intelligent Systems. 9th International Symposium, ISMIS '96, pp. 158-167, Berlin: Springer-Verlag, 1996.
http://lumpi.informatik.uni-dortmund.de/people/baeck/papers/ismis.ps.Z

[Bak85] *Baker, J. E.*: Adaptive Selection Methods for Genetic Algorithms. In [ICGA1], pp. 101-111, 1985.

[Bak87] *Baker, J. E.*: Reducing Bias and Inefficiency in the Selection Algorithm. In [ICGA2], pp. 14-21, 1987.

[Bal94] *Baluja, S.*: Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report CMU-CS-94-163, Pittsburgh, Pennsylvania: School of Computer Science, Carnegie Mellon University, 1994.
ftp://reports.adm.cs.cmu.edu/1994/CMU-CS-94-163.ps

[Bey95] *Beyer, H.-G.*: Toward a Theory of Evolution Strategies: On the Benefits of Sex – the (μ/μ,λ) Theory. Evolutionary Computation, 3(1), pp. 81-111, 1995.

[BT95] *Blickle, T. and Thiele, L.*: A Comparison of Selection Schemes used in Genetic Algorithms (2. Edition). TIK Report No. 11, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, 1995.
http://www.tik.ee.ethz.ch/Publications/TIK-Reports/TIK-Report11abstract.html

[Bli97] *Blickle, T.*: Theory of Evolutionary Algorithms and Application to System Synthesis. Ph.D. thesis. TIK-Schriftenreihe Nr. 17., Zürich: vdf Verlag, 1997.
http://www.handshake.de/user/blickle/publications/index.html

[BEL95] *Böcker, J, Endrikat, C. and Liu, S.*: A Systematic Approach to State Feedback Controller Design for DC/DC Line-Side Traction Converters. Proc. EPE'95, Sevilla, Vol. 1, pp. 314-318, 1995.

[BW96] *Böcker, J. and Wu, Z.*: Symmetry Properties of Multi-Variable Control Systems. Technical Note, 25/96, Daimler Benz AG, 1996.

[Boo87] *Booker, L.*: Improving search in genetic algorithms. In [Dav87], pp. 61-73, 1987.

[Box57] *Box, G. E. P.*: Evolutionary operation: A method for increasing industrial productivity. In Journal of the Royal Statistical Society, C, 6(2), pp. 81-101, 1957.

[Bra72] *Branin, F. K.*: A widely convergent method for finding multiple solutions of simultaneous nonlinear equations. IBM J. Res. Develop., pp. 504-522, Sept., 1972.

[Bre62] *Bremermann, H. J.*: Optimization through evolution and recombination. In *Yovits, M. C. et al.*: Self-organizing systems. Washington, DC: Spartan Books, pp. 93-106, 1962.

[CEC99] *Angeline, P. J. (ed.)*: Proceedings of the Congress on Evolutionary Computation. Piscataway, New Jersey, USA: IEEE Press, 1999.

[Cha95] *Chambers, L. (ed.)*: Genetic Algorithm Applications Vol. I. New York: CRC Press, 1995.

[CS88] *Caruana, R. A. and Schaffer, J. D.*: Representation and Hidden Bias: Gray v. Binary Coding for Genetic Algorithms. In Fifth International Conference on Machine Learning, San Mateo, California, USA: Morgan Kaufmann Publishers, pp. 153-161, 1988.

[CES89] *Caruana, R. A., Eshelmann, L. A. and Schaffer, J. D.*: Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover. In *Sridharan, N. S. (ed.)*: Eleventh International Joint Conference on Artificial Intelligence, San Mateo, California, USA: Morgan Kaufmann Publishers, Vol. 1, pp. 750-755, 1989.

[Cav70] *Cavicchio, D. J.*: Adaptive search using simulated evolution. Unpublished doctoral dissertation, University of Michigan, Ann Arbor, 1970.

[CF94] *Chipperfield, A. J. and Fleming, P. J.*: Parallel Genetic Algorithms: A Survey. Technical Report No. 518, Department of Automatic Control and Systems Engineering, University of Sheffield, 1994.

[CFP94b] *Chipperfield, A. J., Fleming, P. J. and Pohlheim, H.*: A Genetic Algorithm Toolbox for MATLAB. Proc. Int. Conf. Sys. Engineering, Coventry, UK, 6-8 Sept., pp. 200-207, 1994.

[CFPF94a] *Chipperfield, A., Fleming, P. J., Pohlheim, H. and Fonseca, C. M.*: Genetic Algorithm Toolbox for use with Matlab. Technical Report No. 512, Department of Automatic Control and Systems Engineering, University of Sheffield, 1994.

[Coe99] *Coello, C. A.*: List of References on Evolutionary Multiobjective Optimization. Laboratorio Nacional de Informática Avanzada, México, 1999.
http://www.lania.mx/~ccoello/EMOO/EMOObib.html

[CMR91] *Cohoon, J. P., Martin, W. N. and Richards, D.S.*: Genetic Algorithms and Punctuated Equilibria in VLSI. In [PPSN1], pp. 134-144, 1991.

[Dav91a] *Davidor, Y.*: A Naturally Occurring Niche & Species Phenomenon: The Model and First Results. In [ICGA4], pp. 257-263, 1991.

[Dav91b] *Davidor, Y.*: Epistasis Variance: A Viewpoint on GA-Hardness. In [FGA1], pp. 23-35, 1991.

[Dav87] *Davis, L. D.*: Genetic Algorithms and Simulated Annealing. San Mateo, California, USA: Morgan Kaufmann Publishers, 1987.

[Dav91] *Davis, L. D.*: Handbook of Genetic Algorithms. Van Nostrand Reinhold, 1991.

[DJS93] *De Jong, K. and Spears, W.*: On the state of evolutionary computation. In [ICGA5], pp. 618-623, 1993.

[DeJ75] *De Jong, K.*: An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 36(10), 5140B, University Microfilms No. 76-9381, 1975.

[DS78] *Dixon, L. C. W. and Szego, G. P.*: The optimization problem: An introduction. In *Dixon, L. C. W. and Szego, G. P. (ed.)*: Towards Global Optimization II, New York: North Holland, 1978.

[Eas90] *Easom, E. E.*: A survey of global optimization techniques. M. Eng. thesis, Univ. Louisville, Louisville, KY, 1990.

[ECJ] *Whitley, D. (ed.)*: Evolutionary Computation. Journal, Cambridge, Massachusetts: MIT Press, 1993-1999.

[ECT] *Fogel, D. B. (ed.)*: IEEE Transactions on Evolutionary Computation. Journal, Piscataway, New Jersey, USA: IEEE Press.

[EP96] *Fogel, D. B. (ed.)*: Evolutionary Programming V, Proceedings of the Fifth Annual Conference on Evolutionary Programming. Cambridge, MA: MIT Press, 1996.

[EP97] *Angeline, P. J., Reynolds, R. G., McDonnell, J. R. and Eberhart, R. (eds.)*: Evolutionary Programming VI, Proceedings of the Sixth Annual Conference on Evolutionary Programming.

Vol. 1213 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1997.

[EP98] *Porto, V. W., Saravanan, N., Waagen, D. and Eiben, A. E. (eds)*: Evolutionary Programming VII, Proceedings of the Seventh Annual Conference on Evolutionary Programming. Vol. 1447 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1998.

[Esh91] *Eshelmann, L. J.*: The CHC Adaptive Algorithm: How to have safe search when engaging in Nontraditional Genetic Recombination. In [FGA1], pp. 265-283, 1991.

[Fdb92] *Fogel, D. B.*: Evolving Artificial Intelligence. Dissertation, University of California, San Diego, 1992.

[Fdb94a] *Fogel, D. B.*: An Introduction to Simulated Evolutionary Optimization. IEEE Trans. on Neural Networks: Special Issue on Evolutionary Computation, Vol. 5, No. 1, pp. 3-14, 1994.

[Fdb94b] *Fogel, D. B.*: Applying Evolutionary Programming to Selected Control Problems. Comp. Math. App., 11(27), pp. 89-104, 1994.

[Fdb95] *Fogel, D. B.*: Evolutionary computation: toward a new philosophy of machine intelligence. New York: IEEE Press, 1995.
http://www.natural-selection.com/misc/evolCompBook.html

[FOW66] *Fogel, L. J., Owens, A. J. and Walsh, M. J.*: Artificial Intelligence through Simulated Evolution. New York: John Wiley, 1966.

[FF93] *Fonseca, C. M. and Fleming, P. J.*: Genetic Algorithms for Multiple Objective Optimization: Formulation, Discussion and Generalization. In [ICGA5], pp. 416-423, 1993.

[FF95a] *Fonseca, C. M. and Fleming, P. J.*: Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation. Research report 564, Dept. Automatic Control and Systems Eng., University of Sheffield, Sheffield, U.K., 1995.

[FF95b] *Fonseca, C. M. and Fleming, P. J.*: Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms II: Application Example. Research report 565, Dept. Automatic Control and Systems Eng., University of Sheffield, Sheffield, U.K., 1995.

[FF95c] *Fonseca, C. M. and Fleming, P. J.*: An Overview of Evolutionary Algorithms in Multiobjective Optimization. Evolutionary Computation, 3(1), pp. 1-16, 1995.

[Fon95] *Fonseca, C. M.*: Multiobjective Genetic Algorithms with Application to Control Engineering Problems. Ph.D. Thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U.K., 1995.

[FF96] *Fonseca, C. M. and Fleming, P. J.*: On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In [PPSN4], pp.584-593, 1996.

[For81] *Forsyth, R.*: BEAGLE – A Darwinian Approach to Pattern Recognition. Kybernetes, 10, pp. 159-166, 1981.

[Fra62] *Fraser, A. S.*: Simulation of genetic systems. Journal of Theoretical Biology, 2, pp. 329-346, 1962.

[Fri58] *Friedberg, R. M.*: A learning machine: Part I. IBM Journal, 2(1), pp. 2-13, 1958.

[FDN59] *Friedberg, R. M., Dunham, B. and North, J. H.*: A learning machine: Part II. IBM Journal, 3(7), pp. 282-287, 1959.

[FGA1] *Rawlins, G. J. E.*: Foundations of Genetic Algorithms. San Mateo, California, USA: Morgan Kaufmann Publishers, 1991.

[FGA2] *Whitley, L. D.*: Foundations of Genetic Algorithms 2. San Mateo, California, USA: Morgan Kaufmann Publishers, 1993.

[FGA3] *Whitley, L. D. and Vose, M. D.*: Foundations of Genetic Algorithms 3. San Francisco, California, USA: Morgan Kaufmann Publishers, 1995.

[FGA4] *Belew, R. K. and Vose, M. D.*: Foundations of Genetic Algorithms 4. San Francisco, California, USA: Morgan Kaufmann Publishers, 1997.

[FGA5] *Banzhaf, W. and Reeves, C.*: Foundations of Genetic Algorithms 5. San Francisco, California, USA: Morgan Kaufmann Publishers, 1999.

[GAOT] *Houck, C., Joines, J. and Kay, M.*: The Genetic Algorithm Optimization Toolbox (GAOT). North Carolina State University, Department of Industrial Engineering, NCSU-IE TR 95-09, 1995.
http://www.ie.ncsu.edu/gaot/

[GEC99] *Banzhaf, W. (ed.)*: GECCO'99 – Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA: Morgan Kaufmann, 1999.

[GP96] *Koza, J. R., Goldberg, D. E., Fogel, D. B. and Riolo, R. L.*: Genetic Programming 1996: Proceedings of the First Annual Conference. Cambridge: MIT Press, 1996.

[GP97] *Koza, J. R. et al. (eds.)*: Genetic Programming 1997: Proceedings of the Second Annual Conference. San Francisco, CA: Morgan Kaufmann, 1997.

[GP98] *Koza, J. R. et al. (eds.)*: Genetic Programming 1998: Proceedings of the Third Annual Conference. San Francisco, CA: Morgan Kaufmann, 1998.

[Gol83] *Goldberg, D. E.*: Computer-aided gas pipeline operation using genetic algorithms and rule learning. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 44(10), 3174B, University Microfilms No. 8402282, 1983.

[Gol87] *Goldberg, D. E.*: Simple Genetic Algorithms and the Minimal Deceptive Problem. In [Dav87], pp. 74-88, 1987.

[Gol89] *Goldberg, D. E.*: Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Mass.: Addison-Wesley, 1989.

[Gol99] *Goldberg, D. E.*: Genetic and Evolutionary Algorithms in the Real World. Technical Report IlliGAL No. 99013, University of Illinois at Urbana-Champaign, 1999.
ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99013.ps.Z

[GD91] *Goldberg, D. E. and Deb, K.*: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In [FGA1], pp. 69-93, 1991.

[GP71] *Goldstein, A. A. and Price, I. F.*: On descent from local minima. Math. Comput., Vol. 25, No. 115, 1971.

[Gre86] *Grefenstette, J. J.*: Optimization of Control Parameters for Genetic Algorithms. In IEEE Transactions on Systems, Man and Cybernetics, 16 (1986) 1, pp.122-128, 1986.

[GB89] *Grefenstette, J. J. and Baker, J. E.*: How Genetic Algorithms Work: A Critical Look at Implicit Parallelism. In [ICGA3], pp. 20-27, 1989.

[Gre93] *Grefenstette, J. J.*: Deception Considered Harmful. In [FGA2], pp. 75-91, 1983.

[HH94] *Haas, R. and Hunt, K. J.*: Genetic based optimisation of a fuzzy-neural vehicle controller. Technical Report, Daimler Benz AG, Research and Technology Berlin, 1994.

[Ham97] *Hammel, U.*: Evolutionary Computation Applications: Simulation models. In [HEC97], F1.8, pp. F1.8:1-F1.8:9, 1997.

[Han98] *Hansen, N.*: Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie – eine Untersuchung zur entstochastisierten, koordinatensystemunabhängigen Adaption der Mutationsverteilung. Berlin: Mensch & Buch Verlag, 1998

[HOG95] *Hansen, N., Ostermeier, A. and Gawelczyk, A.*: On the Adaptation of Arbitrary Mutation Distributions in Evolution Strategies: The Generating Set Adaptation. In [ICGA6], pp. 57-64, 1995.
ftp://ftp-bionik.fb10.tu-berlin.de/pub/papers/Bionik/GSAES.ps.Z

[HGO95] *Hansen, N., Gawelczyk, A. and Ostermeier, A.*: Sizing the Population with Respect to the Local Progress in (1,λ)-Evolution Strategies – A Theoretical Analysis. In [ICEC95], pp. 80-85, 1995.

[HO96] *Hansen, N. and Ostermeier, A.*: Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In [ICEC96], pp. 312-317, 1996.
ftp://ftp-bionik.fb10.tu-berlin.de/pub/papers/Bionik/CMAES.ps.Z

[HH98] *Haupt, R. L. and Haupt, S. E.*: Practical Genetic Algorithms. NewYork: John Wiley & Sons, 1998.

[HB91] *Hoffmeister, F. and Bäck, T.*: Genetic Algorithms and Evolutionary Strategies: Similarities and Differences. In [PPSN1], pp. 455-469, 1991.

[HEC97] *Bäck, T., Fogel, D. B. and Michalewicz, Z. (eds.)*: Handbook on Evolutionary Computation. Bristol: Institute of Physics Publishing and Oxford, New York: Oxford University Press, 1997.

[Hol75] *Holland, J. H.*: Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press, 1975.

[Hol95] *Holland, J. H.*: Hidden order: how adaptation builds complexity. Reading, Massachusetts: Addison-Wesley, 1995.

[Hoo95] *Hooker, J. N.*: Testing Heuristics: We Have It All Wrong. Journal of Heuristics, 1 (1995), pp. 33-42, 1995.

[HN93] *Horn, J. and Nafpliotis, N.*: Multiobjective optimization using the niched pareto genetic algorithm. IlliGAL Report 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, Champaign, 1993.

[HG95] *Horn, J. and Goldberg, D. E.*: Genetic Algorithm Difficulty and the Modality of Fitness Landscapes. In [FGA3], pp. 243-269, 1995.

[Hor97] *Horn, J.*: Evolutionary Computation Applications: Multicriterion decision making. In [HEC97], F1.9, pp. F1.9:1-F1.9:15, 1997.

[ICEC94] *Fogel, D. B.*: Proceedings of The First IEEE Conference on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Service Center, 1994.

[ICEC95] Proceedings of the Second IEEE Conference on Evolutionary Computation 1995, Piscataway, New Jersey, USA: IEEE Press, 1995.

[ICEC96] Proceedings of the 1996 IEEE Conference on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Press, 1996.

[ICEC97] Proceedings of the 1997 IEEE Int. Conf. on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Press, 1997.

[ICEC98] Proceedings of the 1998 IEEE Int. Conf. on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Press, 1998.

[ICGA1] *Grefenstette, J. J. (ed.)*: Proceedings of an International Conference on Genetic Algorithms and their Application, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates, 1985.

[ICGA2] *Grefenstette, J. J. (ed.)*: Proceedings of the Second International Conference on Genetic Algorithms and their Application, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates, 1987.

[ICGA3] *Schaffer, J. D. (ed.)*: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, 1989.

[ICGA4] *Belew, R. K. and Booker, L. B. (eds.)*: Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, 1991.

[ICGA5] *Forrest, S. (ed.)*: Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, 1993.

[ICGA6] *Eshelman, L. J. (ed.)*: Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, California, USA: Morgan Kaufmann Publishers, 1995.

[ICGA7] *Bäck, T. (ed.)*: Proceedings of the Seventh International Conference on Genetic Algorithms, San Francisco, California, USA: Morgan Kaufmann Publishers, 1997.

[Jac95] *Jacob, C.*: MathEvolvica – Simulierte Evolution von Entwicklungsprogrammen der Natur. Dissertation, Arbeitsberichte des Instituts für mathematische Maschinen und Datenverarbeitung (Informatik), Universität Erlangen-Nürnberg, Band 28, Nummer 10, 1995.

[JF95] *Jones, T. and Forrest, S.*: Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In [ICGA6], pp. 184-192, 1995.
http://www.santafe.edu/sfi/publications/Working-Papers/95-02-022.ps

[Koz92] *Koza, J. R.*: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.

[Koz94] *Koza, J. R.*: Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge: MIT Press, 1994.

[Koz99] *Koza, J. R., Bennett, F. H., Andre, D. and Keane, M. A.*: Genetic Programming III: Darwinian Invention and Problem Solving. San Francisco, CA: Morgan Kaufmann, 1999.

[KS79] *Kreisselmeier, G. and Steinhauser, R.*: Systematische Auslegung von Reglern durch Optimierung eines vektoriellen Gütekriteriums. In Regelungstechnik, 3, pp. 76-79, 1979.

[Lan95] *Langermann*: Definition of a test function for contest on Evolutionary Computation. In [ICEC95], 1995.

[Lit92] *Littger, K.*: Optimierung – Eine Einführung in rechnergestützte Methoden und Anwendungen. Berlin, Heidelberg: Springer-Verlag, 1992.

[MBC95] *Marenbach, P., Bettenhausen, K.D. and Cuno, B.*: Selbstorganisierende Generierung strukturierter Prozeßmodelle. at-Automatisierungstechnik 6 (1995), pp. 277-288, Berlin, 1995.

[MBF96] *Marenbach, P., Bettenhausen, K. D. and Freyer, S.*: Signal path oriented approach to generation of dynamic process models, in [GP96], pp. 327-332, 1996.

[MW94] *MathWorks, The*: Matlab – User Guide. Natick, Mass.: The MathWorks, Inc., 1994.
http://www.mathworks.com/

[MHF92] *Mecklenburg, K., Hrycej, T., Franke, U. and Fritz, H.*: Neural control of autonomous vehicle. In IEEE Vehicular Technology Conference, Denver, 1992.

[Men2] *Osmera, P.*: MENDEL'96 – 2nd International Conference on Genetic Algorithms. 26.-28. June 1996, Technical University of Brno, Czech Republic, 1996.

[Mic92] *Michalewicz, Z.*: Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Heidelberg, New York: Springer-Verlag, 1992.

[Mic94] *Michalewicz, Z.*: Genetic Algorithms + Data Structures = Evolution Programs, Second, Extended Edition. Berlin, Heidelberg, New York: Springer-Verlag, 1994.

[Mie99] *Miettinen, K. M.*: Nonlinear multiobjective optimization. Boston, London, Dordrecht: Kluwer Academic Publishers, 1999.

[Mit96] *Mitchell, M.*: An Introduction to Genetic Algorithms. Cambridge, Massachusetts: MIT Press, 1996.

[Mit90] *Mitschke, M.*: Dynamik der Kraftfahrzeuge: Band C, Fahrverhalten. Berlin, Heidelberg, New York: Springer-Verlag, 1990.

[Müh94] *Mühlenbein, H.*: The Breeder Genetic Algorithm – a provable optimal search algorithm and its application. Colloquium on Applications of Genetic Algorithms, IEE 94/067, London, 1994.

[Müh95a] *Mühlenbein, H.*: Adaptive Systeme in offenen Welten. GMD-Spiegel 2/95, 1995.
http://borneo.gmd.de/AS/gmdsp/editorial.html

[Müh95b] *Mühlenbein, H.*: Genetische Algorithmen und Evolutionsstrategien – Auf der Suche nach verschollenen Schätzen. GMD-Spiegel 2/95, 1995.
http://borneo.gmd.de/AS/gmdsp/muehlen.html

[MSV93a] *Mühlenbein, H. and Schlierkamp-Voosen, D.*: Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization. Evolutionary Computation, 1 (1), pp. 25-49, 1993.
ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-93_01.ps

[MSV95] *Mühlenbein, H. and Schlierkamp-Voosen, D.*: Analysis of Selection, Mutation and Recombination in Genetic Algorithms. In Banzhaf, W. and Eeckman, F. H.: Evolution as a Computational Process. Lecture Notes in Computer Science 899, pp. 142-168, Berlin: Springer-Verlag, 1995.
ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-95_03.ps

[Nis97] *Nissen, V.*: Einführung in evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution. Braunschweig, Wiesbaden: Vieweg, 1997.

[Ost97] *Ostermeier, A.*: Schrittweitenadaption in der Evolutionsstrategie mit einem entstochastisierten Ansatz. Dissertation, Technische Universität Berlin, Fachbereich 6, 1997.

[OGH93] *Ostermeier, A., Gawelczyk, A. and Hansen, N.*: A Derandomized Approach to Self Adaptation of Evolution Strategies. Technical Report TR-93-003, TU Berlin, 1993.
ftp://ftp-bionik.fb10.tu-berlin.de/pub/papers/Bionik/tr-03-93.ps.Z

[OGH94] *Ostermeier, A., Gawelczyk, A. and Hansen, N.*: Step-size adaptation based on non-local use of selection information. In [PPSN3], pp. 189-198, 1994.

[Pad97] *Institut für angewandte Daten- und Wissenstechnik*: Prognose der minimalen Ausführungszeiten von Echtzeit-Systemen auf Basis von genetisch erzeugten Testfallmengen – Version 1.0. Institut für angewandte Daten- und Wissenstechnik AD/WT, Universität Paderborn, 1997.

[Poh93] *Pohlheim, H.*: Simulation und Optimierung eines Blaualgen-Wachstums-Modells. Diplomarbeit, Technische Universität Ilmenau, 1993.

[GEATbx] *Pohlheim, H.*: GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab. www.geatbx.com, 1994-2004.
http://www.geatbx.com/index.html

[Poh95] *Pohlheim, H.*: Ein genetischer Algorithmus mit Mehrfachpopulationen zur Numerischen Optimierung. at-Automatisierungstechnik 3 (1995), pp. 127-135, 1995.

[Poh97] *Pohlheim, H.*: Advanced Techniques for the Visualization of Evolutionary Algorithms. Proceedings of 42. International Scientific Colloquium Ilmenau, Vol. 3, pp. 60-68, 1997.
http://www.pohlheim.com/publications.html

[Poh98] *Pohlheim, H.*: Entwicklung und systemtechnische Anwendung Evolutionärer Algorithmen. Aachen, Germany: Shaker Verlag, 1998. (Development and Engineering Application of Evolutionary Algorithms)
http://www.pohlheim.com/diss/index.html

[Poh99a] *Pohlheim, H.*: Visualization of Evolutionary Algorithms – set of standard techniques and multidimensional visualization. In [GEC99], p. 533-540, 1999.
http://www.pohlheim.com/publications.html

[Poh99b] *Pohlheim, H.*: Evolutionäre Algorithmen - Verfahren, Operatoren, Hinweise aus der Praxis. Berlin, Heidelberg, New York: Springer-Verlag, 1999. (Evolutionary Algorithms – methods, operators and practical directions)
http://www.pohlheim.com/eavoh/index.html

[PH96a] *Pohlheim, H. und Heißner, A.*: Optimale Steuerung der Zustandsgrößen im Gewächshaus mit Genetischen Algorithmen: Grundlagen, Verfahren und Ergebnisse. Technischer Bericht, Technische Universität Ilmenau, 1996.
http://www.pohlheim.com/publications.html

[PH96b] *Pohlheim, H. and Heißner, A.*: Anwendung genetischer Algorithmen zur optimalen Steuerung des Gewächshausklimas. In GMA-Kongreß'96, VDI-Berichte 1282, pp. 799-809, Düsseldorf: VDI-Verlag, 1996.

[PH97a] *Pohlheim, H. and Heißner, A.*: Optimal Control of Greenhouse Climate using a Short Time Greenhouse Climate Model and Evolutionary Algorithms. In Proceedings of 3rd IFAC/ISHS Workshop on „Mathematical and Control Applications in Agriculture & Horticulture", pp. 113-118, 1997.

[PM96] *Pohlheim, H. and Marenbach, P.*: Generation of structured process models using genetic programming. In [AISB96], pp. 102-109, 1996.
http://www.pohlheim.com/publications.html

[PPW99] *Pohlheim, H., Pawletta, S. and Westphal, A.*: Parallel Evolutionary Optimization under Matlab on standard computing networks. In [GEC99], Evolutionary Computation and Parallel Processing Workshop, pp. 174-176, 1999.
http://www.pohlheim.com/publications.html

[PWS2000] *Pohlheim, H., Wegener, J. and Sthamer, H.*: Testing the Temporal Behavior of Real-Time Engine Control Software Modules using extended Evolutionary Algorithms. In CI'2000 Computational Intelligence im industriellen Einsatz, Baden-Baden Mai 2000, vdi-Verlag, 2000.
http://www.pohlheim.com/publications.html

[PS98] *Pohlheim, H. und Schütte, A.*: Optimierung der Parameter in einem Verbrennungsmodell für einen Dieselmotor mit Evolutionären Algorithmen. interner Technischer Bericht FT3/A-1998-001, Daimler Benz AG, 1998.

[PW99] *Pohlheim, H. and Wegener, J.*: Testing the Temporal Behavior of Real-Time Software Modules using Extended Evolutionary Algorithms. In [GEC99], p. 1795, 1999.
http://www.pohlheim.com/publications.html

[PL96] *Poli, R. and Logan, B.*: Evolutionary Computation Cookbook: Recipes for Designing New Algorithms. In Second Online Workshop on Evolutionary Computation, Japan, 1996.
http://www.bioele.nuee.nagoya-u.ac.jp/wec2/papers/index.html

[PPA93] *Puta, H., Pohlheim, H. and Affa, I.*: Simulation und Entscheidungshilfe für das Ökosystem Barther Bodden. In VDI-Berichte 1067, pp. 429-440, Düsseldorf: VDI-Verlag, 1993.

[PPSN1] *Schwefel, H.-P. and Männer, R.*: Parallel Problem Solving from Nature – PPSN I. Vol. 496 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1991.

[PPSN2] *Männer, R. and Manderick, B.*: Parallel Problem Solving from Nature, 2. Amsterdam: Elsevier Science Publishers, 1992.

[PPSN3] *Davidor, Y., Schwefel, H.-P. and Männer, R.*: Parallel Problem Solving from Nature – PPSN III: International Conference on Evolutionary Computation. Vol. 866 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1994.

[PPSN4] *Voigt, H.-M., Ebeling, W., Rechenberg, I. and Schwefel, H.-P.*: Parallel Problem Solving from Nature – PPSN IV: International Conference on Evolutionary Computation. Vol. 1141 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1996.

[PPSN5] *Voigt, H.-M., Ebeling, W., Rechenberg, I. and Schwefel, H.-P.*: Parallel Problem Solving from Nature – PPSN V: International Conference on Evolutionary Computation. Vol. 1498 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1998.

[Ray94] *Ray, T. S.*: An evolutionary approach to synthetic biology: Zen and the art of creating life. Artificial Life, 1 (1/2), pp. 195-226, 1994.

[Rec73] *Rechenberg, I.*: Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart: Frommann-Holzboog, 1973.

[Rec94] *Rechenberg, I.*: Evolutionsstrategie 94. Stuttgart: Frommann-Holzboog, 1994.

[RB94] *Renders, J.-M. and Bersini, H.*: Hybridizing Genetic Algorithms with hill-climbing Methods for Global Optimization: Two Possible Ways. In [ICEC94] Vol. I, pp. 312-317, 1994.

[Ric95] *Richter, G.*: Adaptive Systeme: Computer passen sich an. In GMD-Spiegel 2/95, 1995.
http://borneo.gmd.de/AS/gmdsp/richter.html

[RB93] *Riedmiller, M. and Braun, H.*: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *H. Ruspini (ed.)*: Proceedings of the IEEE International Conference on Neural Networks (ICNN), pp.586-591, 1993.

[Sch68] *Schwefel, H.-P.*: Projekt MHD-Staustrahlrohr: Experimentelle Optimierung einer Zweiphasendüse, Teil I. Technischer Bericht 11.034/68, 35, AEG Forschungsinstitut, Berlin, 1968.

[Sch75] *Schwefel, H.-P.*: Evolutionsstrategie und numerische Optimierung. Dissertation, Technische Universität Berlin, 1975.

[Sch81] *Schwefel, H.-P.*: Numerical optimization of computer models. Chichester: Wiley & Sons, 1981.

[Sch95] *Schwefel, H.-P.*: Evolution and optimum seeking. New York: John Wiley & Sons, 1995.

[SK92] *Schwefel, H. P. and Kursawe, F.*: Künstliche Evolution als Modell für natürliche Intelligenz. In Nachtigall, W. (Ed.): Technische Biologie und Bionik 1, Proceedings 1. Bionik-Kongreß, BIONA report 8, Stuttgart: G. Fischer, pp. 73-91, 1992.

[SHF94] *Schöneburg, E., Heinzmann, F. and Feddersen, S.*: Genetische Algorithmen und Evolutionsstrategien. Bonn, Paris, Reading, Mass.: Addison-Wesley, 1994.

[SDJ91a] *Spears, W.M. and De Jong, K. A.*: On the Virtues of Parameterised Uniform Crossover. In [ICGA4], pp. 230-236, 1991.

[SDJ91b] *Spears, W.M. and De Jong, K. A.*: An Analysis of Multi-Point Crossover. In [FGA1], pp. 301-315, 1991.

[SD94] *Srinivas, N. and Deb, K.*: Multiobjective Optimization using nondominated sorting in genetic algorithms. Evolutionary Computation, 2(3), pp. 221-248, 1994.

[SE92] *Stuckmann, B. E. and Easom, E. E.*: A Comparison of Bayesian Sampling and Global Optimization Techniques. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 5, pp.1024-1032, 1992.

[Sum95] *Sumser, S.*: Verbrennungsmodell für direkteinspritzende Dieselmotoren. interner Technischer Bericht F1M/ST 95-0036, Daimler Benz AG, 1995.

[SR96] *Surry, P. D. and Radcliffe, N. J.*: Innoculation to Initialise Evolutionary Search. In [AISB96], pp. 260-275, 1996.

[Sys89] *Syswerda, G.*: Uniform crossover in genetic algorithms. In [ICGA3], pp. 2-9, 1989.

[TZ89] *Törn, A. and Zilinskas, A.*: Global Optimization. Vol. 350 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1989.

[Vel99] *Veldhuizen, D. A. van.*: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.
http://www.lania.mx/~ccoello/EMOO/veldhuizen99a.ps.gz

[Vös97] *Vössner, S.*: Convergence Measures for Genetic Algorithms. Technical Report, Stanford University, Department of EES, Operations Research, 1997.

[VA94] *Voigt, H.-M. and Anheyer, T.*: Modal Mutations in Evolutionary Algorithms. In [ICEC94] Vol. I, pp. 88-92, 1994.

[VMC95] *Voigt, H.-M., Mühlenbein, H. and Cvetkovi, D.*: Fuzzy recombination for the continuous Breeder Genetic Algorithm. In [ICGA6], pp. 104-111, 1995.
ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-95_01.ps

[Wei70] *Weinberg, R.*: Computer simulation of a living cell. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 31(9), 5312B, University Microfilms No. 71-4766, 1970.

[Why89] *Whitley, D.*: The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In [ICGA3], pp. 116-121, 1989.

[WM95] *Wolpert, D. H. and Macready, W. G.*: No free lunch theorems for search. Technical report SFI-TR-95-02-010, The Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM, 87501, USA, 1995.
http://www.santafe.edu/sfi/publications/Working-Papers/95-02-010.ps

[Wri91] *Wright, A. H.*: Genetic Algorithms for Real Parameter Optimization. In [FGA1], pp. 205-218, 1991.

[ZT98] *Zitzler, E. and Thiele, L.*: An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 1998.
ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/ZT1998a.ps

[ZDT99] *Zitzler, E., Deb, K. and Thiele, L.*: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Technical Report 70, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 1999.
ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/ZDT1999.ps

10.2 Population models and parallel EA

[Bel95] *Belding, T. C.*: The Distributed Genetic Algorithm Revisited. In [ICGA6], pp.114-121, 1995.

[Can95] *Cantú-Paz, E.*: A Summary of Research on Parallel Genetic Algorithms. Technical Report IlliGAL No. 95007, July 1995, University of Illinois at Urbana-Champaign, 1995.
ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/95007.ps.Z

[CJ91] *Collins, R. J. and Jefferson, D. R.*: Selection in Massively Parallel Genetic Algorithms. In [ICGA4], pp. 249-256, 1991.

[CPR96] *Corno, F., Prinetto, P., Rebaudengo, M. and Reorda, M. S.*: Exploiting Competing Subpopulations for Automatic Generation of Test Sequences for Digital Circuits. In [PPSN4], pp. 792-800, 1996.

[DJS95] *DeJong, K. and Sarma, J.*: On Decentralizing Selection Algorithms. In [ICGA6], pp. 17-23, 1995.

[FH91] *Fogarty, T. C. and Huang, R.*: Implementing the Genetic Algorithm on Transputer Based Parallel Processing Systems. In [PPSN1], pp. 145-149, 1991.

[GW91] *Gordon, V. S. and Whitley, D.*: Serial and Parallel Genetic Algorithms as Function Optimizers. In [ICGA5], pp. 177-183, 1993.

[GS89] *Gorges-Schleuter, M.*: ASPARAGOUS An Asynchronous Parallel Genetic Optimization Strategy. In [ICGA3], pp. 422-427, 1989.

[GS91] *Gorges-Schleuter, M.*: Explicit Parallelism of Genetic Algorithms through Population Structures. In [PPSN1], pp. 150-159, 1991.

[GS98] *Gorges-Schleuter, M.*: A Comparative Study of Global and Local Selection in Evolution Strategies. In [PPSN5], pp. 367-377, 1998.

[HM94] *Hauser, R. and Männer, R.*: Implementation of Standard Genetic Algorithms on MIMD Machines. In [PPSN3], pp. 504-513, 1994.

[Her92] *Herdy, M.*: Reproductive Isolation as Strategy Parameter in Hierarchical Organized Evolution Strategies. In [PPSN2], pp. 207-217, 1992.

[KSR94] *Kapsalis, A., Smith, G.D. and Rayward-Smith, V.J.*: A unified parallel genetic algorithm. AISB Workshop Evolutionary Computation, April 11-13, Leeds, 1994.

[Loh91] *Lohmann, R.*: Application of Evolution Strategy in Parallel Populations. In [PPSN1], pp. 198-208, 1991.

[MS89] *Manderick, B. and Spiessens, P.*: Fine-grained Parallel Genetic Algorithms. In [ICGA3], pp. 428-433, 1989.

[Müh89] *Mühlenbein, H.*: Parallel genetic algorithms, population genetics and combinatorial optimization. In [ICGA3], pp. 416-421, 1989.

[Müh91] *Mühlenbein, H.*: Evolution in Time and Space – The Parallel Genetic Algorithm. In [FGA1], pp. 316-337, 1991.
ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-91_01.ps

[MGK88] *Mühlenbein, H., Gorges-Schleuter, M. and Krämer, O.*: Evolution algorithms in combinatorial optimization. Parallel Computing, 7, pp.65-85, 1988.

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

[MSV93a] see page 78

[PLG87] *Pettey, C. B., Leuze, M. R. and Grefenstette, J. J.*: A Parallel Genetic Algorithm. In [ICGA2], pp. 155-161, 1987.

[Rob87] *Robertson, G. G.*: Parallel Implementation of Genetic Algorithms in a Classifier System. In [ICGA2], pp. 140-147, 1987.

[Rud91] *Rudolph, G.*: Global Optimization by Means of Distributed Evolution Strategies. In [PPSN1], pp. 209-213, 1991.

[SDJ96] *Sarma, J. and DeJong, K.*: An Analysis of the Effects of Neighbourhood Size and Shape on Local Selection Algorithms. In [PPSN4], pp. 236-244, 1996.

[Swm96] *Schwehm, M.*: Globale Optimierung mit massiv parallelen genetischen Algorithmen. Dissertation, Universität Erlangen-Nürnberg, 1996.
http://www-ra.informatik.uni-tuebingen.de/mitarb/schwehm/Abstracts.html#Schweh97

[SM91] *Spiessens, P. and Manderick, B.*: A Massively Parallel Genetic Algorithms – Implementation and First Analysis. In [ICGA4], pp. 279-286, 1991.

[SVM94] *Schlierkamp-Voosen, D. and Mühlenbein, H.*: Strategy adaptation by competing subpopulations. In [PPSN3], pp. 199-208, 1994.
ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-94_14.ps

[SVM96] *Schlierkamp-Voosen, D. and Mühlenbein, H.*: Adaptation of Population Sizes by Competing Subpopulations. In [ICEC96], pp. 330-335, 1996.
ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-96_01.ps

[SWM91] *Starkweather, T., Whitley, D. and Mathias, K.*: Optimization using Distributed Genetic Algorithms. In [PPSN1], pp. 176-185, 1991.

[Tan87] *Tanese, R.*: Parallel Genetic Algorithm for a Hypercube. In [ICGA2], pp. 177-183, 1987.

[Tan89] *Tanese, R.*: Distributed Genetic Algorithms. In [ICGA3], pp. 434-439, 1989.

[VBS91] *Voigt, H.-M., Born, J. and Santibanez-Koref, I.*: Modeling and Simulation of Distributed Evolutionary Search Processes for Function Optimization. In [PPSN1], pp. 373-380, 1991.

[VSB92] *Voigt, H.-M., Santibanez-Koref, I. and Born, J.*: Hierarchically Structured Distributed Genetic Algorithm. In [PPSN2], pp. 145-154, 1992.

10.3 Combinatorial optimization

[AV97] *Aarts, E. and Verhoeven, M.*: Evolutionary Computation in Practice: Genetic local search for the traveling salesman problem. In [HEC97], G9.5, pp. G9.5:1-G9.5:7, 1997.

[AM96] *Asveren, T. and Molitor, P.*: New Crossover Methods For Sequencing Problems. In [PPSN4], pp. 290-299, 1996.

[BUM91] *Bagchi, S., Uckun, S., Miyabe, Y. and Kawamura, K.*: Exploring problem-specific recombination operators for job shop scheduling. In [ICGA4], pp. 10-17, 1991.

[BMK96] *Bierwirth, C., Mattfeld, D. C. and Kopfer, H.*: On Permutation Representations for Scheduling Problems. In [PPSN4], pp. 310-318, 1996.

[Bru97] *Bruns, R.*: Evolutionary Computation Applications: Scheduling. In [HEC97], F1.5, pp. F1.5:1-F1.5:9, 1997.

[Dav85] *Davis, L.*: Applying adaptive algorithms to epistatic domains. In Proc. Int. Joint Conf. on Artificial Intelligence, 1985.

[DW94] *Dzubera, J. and Whitley, D.*: Advanced Correlation Analysis of Operators for the Traveling Salesman Problem. In [PPSN3], pp. 68-77, 1994.

[GL85] *Goldberg, D. and Lingle, R.*: Alleles, loci, and the traveling salesman problem. In [ICGA1], 1985.

[GS89] see p.82

[GS97a] *Gorges-Schleuter, M.*: Asparagos96 and the Traveling Salesman Problem. In [ICEC97], pp. 171-174, 1997.

[GS97b] *Gorges-Schleuter, M.*: On the power of evolutionary optimization at the example of ATSP and large TSP Problems. In European Conference on Artificial Life `97, Brighton, U.K., 1997.

[LK73] *Lin, S. and Kernighan, B.*: An efficient heuristic procedure for the traveling salesman problem. Operations Res. 21, pp. 498-516, 1973.

[MW92] *Mathias, K. and Whitley, D.*: Genetic operators, the fitness landscape and the traveling salesman problem. In [PPSN2], pp. 219-228, 1992.

[Mat96] *Mattfeld, D. C.*: Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling. Heidelberg: Physica-Verlag, 1996.

[Müh89] see p.83

[MGK88] see p.83

[NK97] *Nagata, Y. and Kobayashi, S.*: Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In [ICGA7], pp. 450-457, 1997.

[Nis97] *Nissen, V.*: Evolutionary Computation in Practice: Quadratic assignment. In [HEC97], G9.10, pp. G9.10:1-G9.10:8, 1997.

[OSH87] *Oliver, I. M., Smith, D. J. and Holland, J. R. C.*: A study of permutation crossover operators on the traveling salesman problem. In [ICGA2], pp.224-230, 1987.

[Ron95] *Ronald, S.*: Routing and scheduling problems. In [Cha95], pp. 367-430, 1995.

[SMW91] *Starkweather, T., McDaniel, S., Mathias, K., Whitley, D. and Whitley, C.*: A Comparison of Genetic Sequencing Operators. In [ICGA4], pp. 69-76, 1991.

[Sys91] *Syswerda, G.*: Schedule Optimization Using Genetic Algorithms. In [Dav91], pp. 332-349, 1991.

[TM98] *Tao, G. and Michalewicz, Z.*: Inver-over Operator for the TSP. In [PPSN5], pp. 803-812, 1998.

[TSPBIB] *Moscato, P.*: TSPBIB Home page. 1996.
http://www.densis.fee.unicamp.br/~moscato/TSPBIB_home.html

[TSPLIB] TSPLIB – A Traveling Salesman Problem Library. 1995.
http://www.iwr.uni-heidelberg.de/groups/comopt/soft/TSPLIB95/TSPLIB.html

[WRE98] *Watson, J. P., Ross, C., Eisele, V., Denton, J., Bins, J., Guerra, C., Whitley, D. and Howe, A.*: The Traveling Salesrep Problem, Edge Assembly Crossover, and 2-opt. In [PPSN5], pp. 823-832, 1998.
http://www.cs.colostate.edu/~genitor/1998/ppsn98b.pdf

[WSF89] *Whitley, D., Starkweather, T. and Fuquay, D.*: Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. In [ICGA3], pp. 133-140, 1989.

[WSS91] *Whitley, D., Starkweather, T. and Shaner, D.*: Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination. In [Dav91], pp. 350-372, 1991.

[WY95] *Whitley, D. and Yoo, N.-W.*: Modeling simple genetic algorithms for permutation problems. In [FGA3], pp. 163-184, 1995.

[Whi97a] *Whitley, D.*: Search Operators: Mutation: Permutations. In [HEC97], C3.2.3, pp. C3.2:5-C3.2:7, 1997.

[Whi97b] *Whitley, D.*: Search Operators: Recombination: Permutations. In [HEC97], C3.3.3, pp. C3.3:14-C3.3:20, 1997.

[Whi99] *Whitley, D.*: A free lunch proof for gray versus binary encodings. In [GEC99], pp. 726-733, 1999.

10.4 Visualization

[CB96] *Beardah, C. C. and Baxter, M.*: The archaeological use of Kernel Density Estimates. Internet Archaeology 1, 5.1, 1996.
http://intarch.ac.uk/journal/issue1/beardah_toc.html

[Col93] *Collins, T. D.*: The Visualisation of Genetic Algorithms. Msc. Thesis, De Montfort University, Leicester, GB, 1993.

[Col95] *Collins, T. D.*: The Visualization of Genetic Algorithms – Related Work. Technical Report, KMI-TR-19, Knowledge Media Institute, The Open University, Milton Keynes, UK, 1995.
http://kmi.open.ac.uk/publications/tr.cfm?trnumber=19

[Col97a] *Collins, T. D.*: Genotypic-Space Mapping: Population Visualization for Genetic Algorithms. Technical Report, KMI-TR-39, Knowledge Media Institute, The Open University, Milton Keynes, UK, 1997.
http://kmi.open.ac.uk/publications/tr.cfm?trnumber=18

[Col97b] *Collins, T. D.*: Using Software Visualization technology to help Genetic Algorithms Designers. In Proceedings of The Ninth Annual Conference of the Psychology of Programming Interest Group (PPIG 9), pp. 43-51, 1997.
http://kmi.open.ac.uk/people/trevor/publications/PPIG-97.ps.gz

[CC94] *Cox, T. F. and Cox, M. A. A.*: Multidimensional Scaling. London: Chapman & Hall, 1994.

[DH73] *Duda, R. O. and Hart, P. E.*: Pattern Classification and Scene Analysis. New York: John Wiley & Sons, 1973.

[DCW96] *Dybowski, R., Collins, T. D. and Weller, P. D.*: Visualization of binary string convergence by Sammon mapping. In [EP96], pp. 377-383, 1996.
http://kmi.open.ac.uk/people/trevor/publications/EP96.ps.gz

[JF95] see page 77

[NEA94] *Nassersharif, B., Ence, D. and Au, M.*: Visualization of Evolution of Genetic Algorithms, In Proceedings of World Congress on Neural Networks WCNN'94, San Diego, CA, USA, Hillside, NJ, USA: Lawrence Erlbaum Associates, pp. 1/560-1/565, 1994.

[Poh97] see page 79

[Poh99a] see page 79

[Rip96] *Ripley, B. D.*: Pattern Recognition and Neural Networks. Cambridge, GB: Cambridge University Press, 1996.

[RC93] *Routen, T. W. and Collins, T. D.*: The Visualisation of AI Techniques. In Proceedings of Third International Conference on Computational Graphics and Visualisation Techniques COMPUGRAPH'93, Alvor, Portugal, New York, USA: ACM Press, pp. 274-282, 1993

[Rou94] *Routen, T. W.*: Techniques for the Visualisation of Genetic Algorithms. In [ICEC94] Vol. II, pp. 846-851, 1994.

[Sam69] *Sammon, J. W. jr.*: A Nonlinear Mapping for Data Structure Analysis. IEEE Transactions on Computers, Vol. C-18, no. 5, pp. 401-409, 1969.

[Swm96] see page 83

10.5 Polyploidy and Evolutionary Algorithms

[Bag67] *Bagley, J. D.*: The behavior of adaptive systems which employ genetic and correlation algorithms. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 28(12), 5106B, University Microfilms No. 68-7556, 1967.

[Bri81] *Brindle, A.*: Genetic algorithms for function optimization. unpublished doctoral dissertation, University of Alberta: Edmonton, 1981.

[CCR96] *Collingwood, E., Corne, D. and Ross, P.*: Useful Diversity via Multiploidy. In [AISB96], Workshop Proceedings, pp. 49-53, 1996.

[GS87] *Goldberg, D. E. and Smith, R. E.*: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In [ICGA2], pp. 59-68, 1987.

[Gol89] see page 75

[Hol71] *Hollstien, R. B.*: Artificial genetic adaptation in computer control systems. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 32(3), 1510B, University Microfilms No. 71-23,773, 1971.

[NW95] *Ng, K. P. and Wong, K. C.*: A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization. In [ICGA6], pp. 159-166, 1995.

[Ros67] *Rosenberg, R. S.*: Simulation of genetic populations with biochemical properties. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 28(7), 2732B, University Microfilms No. 67-17,836, 1967.

[Rya96] *Ryan, C.*: Reducing Premature Convergence in Evolutionary Algorithms. Ph.D. thesis, University College Cork, Ireland, 1996.
ftp://odyssey.ucc.ie/pub/genetic/thesis.ps.Z

[Smi87] *Smith, R. E.*: Diploid genetic algorithms for search in time varying environments. Proceedings of the 25th Annual Southeast Regional Conference of the ACM, pp. 175-178, 1987.

[Smi88] *Smith, R. E.*: An investigation of diploid genetic algorithms for adaptive search of nonstationary functions. Unpublished master's thesis, University of Alabama: Tuscaloosa, 1988.

[YA94] *Yoshida, Y. and Adachi, N.*: A Diploid Genetic Algorithm for Preserving Population Diversity – pseudo-Meiosis GA. In [PPSN3], pp. 36-45, 1994.

10.6 Biology, Genetics and Population genetics

[CK70] *Crow, J. F. and Kimura, M.*: An Introduction to Population Genetics Theory. New York: Harper and Row, 1970.

[Dar1859] *Darwin, C.*: On the origin of species by means of natural selection. London: Murray, 1859. (Deutsche Übersetzung: Die Entstehung der Arten durch natürliche Zuchtwahl. 1860. Stuttgart: Reclam, 1974.)

[Fwb63] Fremdwörterbuch. Leipzig: Bibliographisches Institut, 1963.

[Hag91] *Hagemann, R.*: Allgemeine Genetik. Jena: Gustav Fischer Verlag, 1991.

[Ode97] *Odenbach, W.*: Biologische Grundlagen der Pflanzenzüchtung. Berlin: Parey Buchverlag, 1997.

[Hen95] *Hennig, W.*: Genetik. Berlin, Heidelberg: Springer-Verlag, 1995.

[RM54] *Rieger, R. and Michaelis, A.*: Genetisches und Cytogenetisches Wörterbuch. Der Züchter, 2. Sonderheft, Berlin, Göttingen, Heidelberg: Springer-Verlag, 1954.

[Wer68] *Werner, F.*: Wortelemente lateinisch-griechischer Fachausdrücke in den biologischen Wissenschaften. Halle (Saale): Max Niemeyer Verlag, 1968.

List of Figures

List of Tables