Evolutionary Algorithms
Variation and genetic operators

Prof. Dr. Rudolf Kruse    Christian Moewes
{kruse,cmoewes}@iws.cs.uni-magdeburg.de
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Institut für Wissens- und Sprachverarbeitung
Outline

1. Motivation

2. One-Parent-Operators

3. Two- or Multiple-Parent-Operators

4. Interpolating and extrapolating recombination

5. Self-adapting algorithms

6. Summary
Variation by mutation [Weicker, 2007]

- Variations (mutations): small changes in biology

⇒ Mutation operator: changes as few as possible on the solution candidate concerning the fitness (function)

- below: investigation of the interaction with the selection
- here: behaviour of a simple optimization algorithm on a very simple optimization problem (comparison with a given bit string)
Meaning of mutation

**Exploration oder Erforschung**

- exploration at random
- also: further away regions of the space

**Exploitation oder Feinabstimmung**

- local improving of a solution candidate
- important: embedding of phenotypic neighborhood
Binary Mutation

**Algorithm 1** Binary Mutation

**Input:** individual $A$ with $A.G \in \{0, 1\}^l$

**Output:** individual $B$

$B \leftarrow A$

for $i \in \{1, \ldots, l\}$ {

$u \leftarrow$ choose randomly according to $U([0, 1))$

if $u \leq p_m$ {

/* probability of mutation $p_m$ */

$B.G_i \leftarrow 1 - A.G_i$

}

}

return $B$
Gaussian-Mutation

alternative real-valued mutation

- directly applied on real-valued numbers
- Addition of a normal distributed random number on each gene

Algorithm 2 Gaussian-Mutation

Input: individual $A$ mit $A.G \in \mathbb{R}^l$
Output: individual $B$

for $i \in \{1, \ldots, l\}$ {
    $u_i \leftarrow$ choose randomly according to $N(0, \sigma)$ /* standard deviation $\sigma$ */
    $B_i \leftarrow A_i + u_i$
    $B_i \leftarrow \max\{B_i, u_g_i\}$ /* lower bound $u_g_i$ */
    $B_i \leftarrow \min\{B_i, o_g_i\}$ /* upper bound $o_g_i$ */
}

return $B$
Comparison of the methods

Approach

- Optimizing of the simple function

\[ f_2(x) = \begin{cases} 
  x & \text{falls } x \in [0, 10] \subset \mathbb{R}, \\
  \text{undef.} & \text{sonst}
\end{cases} \]

- Individual of the parents (1.0 und 4.99)

- Determining the distribution of the descendants with 10000 mutations each
**Binäre-Mutation mit standardbinärer Kodierung**

**Binäre-Mutation mit Gray-Kodierung**

**Gauss-Mutation**
Comparison of the methods

- Gaussian-Mutation with lower $\sigma \Rightarrow$ well applicable on exploitation
- with higher $\sigma \Rightarrow$ wide exploration

- Hamming-Cliffs = break in frequency distribution

- Gray-Code succeeds on including phenotypical neighborhood
- tends to one part of the space, though

$\Rightarrow$ Gaussian-Mutation orients itself on phenotypical neighborhood
$\Rightarrow$ binary mutation faster detects interesting regions in $\Omega$
Genetic operators

- are applied on certain fraction of chosen individuals (intermediary population)
- generating variants and recombinations of already existing solution candidates
- gen. classification of genetic operators according to the number of parents:
  - One-Parent-Operators („Mutation“)
  - Two-Parent-Operators („Crossover“)
  - Multipe-Parent-Operators
- genetic operators have special properties (dep. on the encoding)
  - if solution candidates = permutations, then permutation-conserving genetic operators
  - gen.: if certain combination of alleles unreasonable, genetic operators should never create them
Outline

1. Motivation

2. One-Parent-Operators
   - Standard mutation and Pair swap
   - Operations on subsequences

3. Two- or Multiple-Parent-Operators

4. Interpolating and extrapolating recombination

5. Self-adapting algorithms

6. Summary
Standard mutation and Pair swap

- **Standard mutation:**
  Exchange the form/value of a gene by another allele

  \[
  \begin{array}{cccccc}
  3 & 1 & 4 & 2 & 5 & 4 \ 6 \\
  \end{array}
  \rightarrow
  \begin{array}{cccccc}
  3 & 1 & 6 & 2 & 5 & 4 \ 6 \\
  \end{array}
  \]

  - if necessary, multiple genes are mutated (see. \textit{n}-Queens-Problem)
  - \textit{Parameter:} probability of mutation \( p_m, 0 < p_m \ll 1 \)
  for Bitstrings of length \( l \): \( p_m = 1/l \) approximately optimal

- **Pair swap:**
  Exchange the forms/values of two gene in a chromosome

  \[
  \begin{array}{cccccc}
  3 & 1 & 4 & 2 & 5 & 4 \ 6 \\
  \end{array}
  \rightarrow
  \begin{array}{cccccc}
  3 & 5 & 4 & 2 & 1 & 4 \ 6 \\
  \end{array}
  \]

  - \textit{Precondition:} same allele sets of the exchanged genes
  - \textit{Generalization:} cyclic change of 3, 4, \ldots, \( k \) genes
Operations on subsequences

- **Shift:**
  
  \[
  \begin{array}{ccccccc}
  3 & 1 & 4 & 2 & 5 & 4 & 6 \\
  \end{array}
  \quad \rightarrow 
  \begin{array}{ccccccc}
  3 & 2 & 5 & 1 & 4 & 4 & 6 \\
  \end{array}
  \]

- **arbitrary permutation:**
  
  \[
  \begin{array}{ccccccc}
  3 & 1 & 4 & 2 & 5 & 4 & 6 \\
  \end{array}
  \quad \rightarrow 
  \begin{array}{ccccccc}
  3 & 2 & 1 & 5 & 4 & 4 & 6 \\
  \end{array}
  \]

- **Inversion:**
  
  \[
  \begin{array}{ccccccc}
  3 & 1 & 4 & 2 & 5 & 4 & 6 \\
  \end{array}
  \quad \rightarrow 
  \begin{array}{ccccccc}
  3 & 5 & 2 & 4 & 1 & 4 & 6 \\
  \end{array}
  \]

- **Precondition:** same sets of alleles in the involved section
- **Parameter:** if necessary, probability distribution over the lengths
Outline

1. Motivation

2. One-Parent-Operators

3. Two- or Multiple-Parent-Operators
   - One-point- and Two-point-Crossover
   - n-point- and uniform crossover
   - Shuffle Crossover
   - Permutation-conserving crossover
   - Diagonal-Crossover
   - Characterization

4. Interpolating and extrapolating recombination
One-point- and Two-point-Crossover

One-point-Crossover

- Determining a random cutting line
- Exchange the gene sequences on one side of the cutting line

Two-point-Crossover

- Determining of two random cutting points
- Exchange of the gene sequences between both cutting points
n-point- and uniform crossover

n-point-crossover

- Generalization of the One- and Two-point-Crossover
- Determining of $n$ random cutting points
- alternating exchange / keep of the gene sequences between two following cutting points

Uniform crossover

- on each gene: determine whether to exchange or not ($+$: yes, $-$: no, Parameter: probability $p_x$ of exchange)

- **Attention**: uniform crossover not equivalent to the $(l - 1)$-point-crossover! number of the crossover points is chosen by random
**Shuffle Crossover**

- before One-Point-Crossover: random permutation of the genes
- after: Unmixing the genes

<table>
<thead>
<tr>
<th>Permutation</th>
<th>Crossover</th>
<th>Unmix</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 2 1 4 3 6</td>
<td>4 2 6 3 5 1</td>
<td>4 2 6 5 3 4</td>
</tr>
<tr>
<td>1 2 3 4 5 6</td>
<td>1 2 6 5 1 3</td>
<td>1 2 3 4 5 6</td>
</tr>
<tr>
<td>3 1 4 2 5 4</td>
<td>2 1 4 5 3 4</td>
<td>2 1 4 3 5 1</td>
</tr>
<tr>
<td></td>
<td>2 1 4 3 5 1</td>
<td>5 1 1 2 3 4</td>
</tr>
<tr>
<td>3 1 4 2 5 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Shuffle crossover is **not** equivalent to the uniform crossover!
- each count of gene exchanges between chromosomes has the same probability
- uniform crossover: count is binomial distributed with parameter $p_x$
- Shuffle crossover: one of the most recommending methods
Uniform order-based crossover

• similar to uniform crossover: for each gene decide whether to keep it or not
  (+: yes, -: no, Parameter: probability $p_k$ of keeping the gene)

• fill gaps by missing alleles (in order of the occurrence in the other chromosome)

```
  5 7 2 4 6 3 1
+ - + + - - +
  4 2 3 1 5 7 6
```
```
  5 2 4 - - - 1
+ - + + - - +
  4 3 1 - - 6
```
```
  5 3 2 4 7 6 1
+ - + + - - +
  4 5 3 1 7 2 6
```

• preserves order information

• alternative: Keeping the „+“ resp. „−“ marked genes in one of the chromosomes
Edge recombination (developed for TSP)

- chromosome is interpreted as a graph (chain or ring) each gene contains edges to its neighbors in the chromosome
- Edges of the graphs of two chromosomes are mixed
- preserve *neighborhood information*

**Procedure:** 1. *Constructing an edge table*

- for every allele its neighbors (in both parents) are listed (including the last allele as a neighbor of the first and vice versa)
- if an allele has the same neighbor in both parents (where the side is irrelevant), this neighbor is listed only once (but marked)
Edge recombination

**Procedure:** 2. *Constructing a child*

- the first allele of a randomly chosen parent is taken for the first allele of the child
- chosen allele is deleted from all neighbor lists in the edge table and its own list of neighbors is retrieved
- From this neighbor list an allele is chosen respecting the following precedences:
  1. marked neighbors (i.e. neighbors that occur in both parents)
  2. neighbors with the shortest neighborhood list (marked neighbors count once)
  3. any neighbor

In analogy to this: a second child may be constructed from the first allele of the other parent (this is rarely done)
**Edge recombination**

**Example:**

A: 6 3 1 5 2 7 4

B: 3 7 2 5 6 1 4

**Constructing the edge table**

<table>
<thead>
<tr>
<th>Allele</th>
<th>Neighbors in A</th>
<th>Neighbors in B</th>
<th>aggregated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3, 5</td>
<td>6, 4</td>
<td>3, 4, 5, 6</td>
</tr>
<tr>
<td>2</td>
<td>5, 7</td>
<td>7, 5</td>
<td>5*, 7*</td>
</tr>
<tr>
<td>3</td>
<td>6, 1</td>
<td>4, 7</td>
<td>1, 4, 6, 7</td>
</tr>
<tr>
<td>4</td>
<td>7, 6</td>
<td>1, 3</td>
<td>1, 3, 6, 7</td>
</tr>
<tr>
<td>5</td>
<td>1, 2</td>
<td>2, 6</td>
<td>1, 2*, 6</td>
</tr>
<tr>
<td>6</td>
<td>4, 3</td>
<td>5, 1</td>
<td>1, 3, 4, 5</td>
</tr>
<tr>
<td>7</td>
<td>2, 4</td>
<td>3, 2</td>
<td>2*, 3, 4</td>
</tr>
</tbody>
</table>

• both chromosomes = ring (first gene is neighbor of the last gene): in A 4 is left neighbor of 6, 6 is right neighbor of 4; B analog to this

• in both: 5, 2 and 7 are next to each other – should be preserved (see marks)
### Edge recombination

#### Constructing a child

![Table](image)

- start with first allele of the chromosomes 6 (also 6) and delete 6 from all neighborhood lists (third column)
- as 5 has the shortest list of all neighbors of 6 (1, 3, 4, 5), 5 is selected for the second gene
- after that 2 is following, then 7 aso.
Edge recombination

- Child has often a new edge (from last to the first gene)
- can also be applied, if first and last gene are not seen as neighbors: Then, edges are not taken into the edge table
- if first and last gene are neighbors, first allele can be chosen arbitrarily
  if not, an allele which is located at the beginning of the chromosome should be chosen
- Construction of a child: neighborhood list of a currently chosen allele can be empty
  (priorities should limit the probability as low as possible; they are not perfect, though)
  in this case: random selection of the remaining alleles
Three- and Multi-Parent-Operators

Diagonal-Crossover

- similar two 1-, 2- and $n$-point-Crossover, but usable if more parents exist
- three parents: two crossover points
- shifts gene sequences diagonally on intersection points over the chromosomes

```
1 5 2 3 6 2 4
5 2 1 4 3 6 1
3 1 4 2 5 4 6
```

```
1 5 1 4 3 4 6
5 2 4 2 5 2 4
3 1 2 3 6 6 1
```

- Generalization for $> 3$ parents:
  choose $k - 1$ crossover points for $k$ parents
- leads to a strong exploration of the space,
  especially on large number of parents (10–15 parents)
Characterization of crossover operators

Positional bias (dt. ortsabhängige Verzerrung):

• if the probability that two genes are jointly inherited from the same parent depends on the (relative) position of these genes in the chromosome

• undesired because it can make the exact arrangement of the different genes in a chromosome crucial for the success or failure of an evolutionary algorithm

• **Example: One-Point-Crossover**
  • 2 genes are separated from each other (arrive in different childs), if crossover point lies between them
  • the closer 2 genes in the chromosome are located, the fewer crossover points can separate them
  \[\Rightarrow\] genes next to each other are jointly taken in the same child with higher probability than distant geness
Characterization of crossover operators

Distributional bias (dt. Verteilungsverzerrung):
- if the probability that a certain number of genes is exchanged between the parent chromosomes is not the same for all possible numbers of genes
- undesired, because it causes partial solutions of different lengths to have different chances of progressing to the next generation
- distributional bias is usually less critical than positional bias
- Example: uniform crossover
  - since for every gene it is decided with probability $p_x$ and independently of all other genes whether it is exchanged or not, the number $k$ of exchanged genes is binomially distributed with the parameter $p_x$:

  $$P(K = k) = \binom{n}{k} p_x^k (1-p_x)^{n-k}$$

  mit $n \equiv$ Gesamtzahl der Gene

  ⇒ very small and very large numbers are less likely
Outline

1. Motivation

2. One-Parent-Operators

3. Two- or Multiple-Parent-Operators

4. Interpolating and extrapolating recombination
   Interpolating operators
   Extrapolating operators

5. Self-adapting algorithms

6. Summary
Motivation [Weicker, 2007]

• so far: operators which recombines alleles that already exist in the parent chromosomes, but do not create any new alleles
  • One-point-, Two-point- und n-point-crossover
  • Uniform (order based) crossover
  • Shuffle Crossover
  • Edge recombination
  • Diagonal-Crossover

• depend crucially on the diversity of the population

• no construction of new alleles: only a fraction of $\Omega$ can be reached which is contained in the individuals of the population

• if a population is very diverse, recombination operators can explore the search space well
Interpolating operators

- can blend the traits of the parents in such a way that offspring with new traits is created

⇒ Ω is thus less explored

- interpol. Recombination focusses population on 1 main area
- benefits fine tuning of individuals with very good fitness
- to explore Ω sufficiently at the beginning: using a strong random and diversity-preserving mutation
Arithmetic crossover

- example for interpolating recombination
- works on real-valued genotypes
- geometric interpretation: can create all points on a straight line between both parents

**Algorithm 3** Arithmetic crossover

**Input:** Individuals $A, B$ with $A.G, B.G \in \mathbb{R}^l$

**Output:** new individual $C$

1: $u \leftarrow$ choose randomly from $U([0, 1])$
2: **for** $i \in \{1, \ldots, l\}$ **do**
3: $C.G_i \leftarrow u \cdot A.G_i + (1 - u) \cdot B.G_i$
4: **end for**
5: **return** $C$
Extrapolating operators

• try to infer information from several individuals

⇒ create a prognosis in what direction one can expect fitness improvements

• extrapolating recombination may leave former $\Omega$

• is only way of recombination which takes fitness values into account

• influence of diversity is hardly understandable

• example: arithmetic crossover with $u \in U([1, 2])$
Outline

1. Motivation

2. One-Parent-Operators

3. Two- or Multiple-Parent-Operators

4. Interpolating and extrapolating recombination

5. Self-adapting algorithms
   - Experiment based on the TSP
   - Locality of the mutation operator
   - Adaptation strategies

6. Summary
Self-adapting algorithms [Weicker, 2007]

• so far: mutation should change phenotype as small as possible
• now: question if this is valid on every (time) step during the optimization

• control experiment
• solve TSP (here 51 cities) by Hillclimbing
⇒ no recombination
• differently local mutation operators are
  • inversion of a subsequence
  • cyclical exchange of three randomly chosen cities
• supposed inappropriate triple exchange: more successful in first 50 generations than favored inversion
• therefore: definition of the relative expected improvement as metric of what improvement an operator enables
Relative expected improvement

Definition

The *fitness improvement* of an individual $A \in \mathcal{G}$ to another individual $B \in \mathcal{G}$ is defined as

$$\text{Improvement}(A, B) = \begin{cases} |B.F - A.F| & \text{if } B.F \succ A.F, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the *relative expected improvement* of an operator Mut concerning individual $A$ can be defined as

$$\text{relEV}_{\text{Mut},A} = E \left( \text{Improvement}(A, \text{Mut}^\xi(A)) \right).$$
Influence

- determining the relative expected improvement in different fitness ranges by random samples from $\Omega$
- responsible for illustrated effect

$\Rightarrow$ How frequent are the different fitness values in $\Omega$?
Complete space

- left: density distribution of a TSP with 11 cities
- right: idealized density distribution of a minimization problem
- similar distribution on children (generated after mutation)
Variance of the generated fitness

- *locality* of the mutation operator is very important
- very local $\Rightarrow$ fitness values in vicinity of the fitness of the parents
- less local $\Rightarrow$ bigger range of fitness values is covered

- inverting mutation is more local over the complete fitness range than triple exchange
Results of consideration

- quality of a mutation operator cannot be judged independently of the current fitness level
- operator is never optimal over the complete process of optimization
- on increasing approximation to the optimum: more local operators!
Adaptation strategies: 3 techniques

Predefined adaptation:
• define change before

Adaptive adaptation:
• define measure of appropriateness
• deduce adapting from rules

Selbst-adaptive adaptation:
• use additional information in individual
• parameter should align individually by a random process
Predefined adaptation

Considered parameter:

- real valued gaussian mutation
- $\sigma$ determines average step width
- modifying parameter $0 < \alpha < 1$ lets decrease $\sigma$ exponentially

Realization:

**Algorithm 4** Predefined adaptation

**Input:** Standard deviation $\sigma$, modifying parameter $\alpha$

**Output:** adapted standard deviation $\sigma$

1: $\sigma' \leftarrow \alpha \cdot \sigma$

2: **return** $\sigma'$
Adaptive adaptation

- Metric: fraction of improving mutations of last $k$ generations
- if fraction is too "high" $\sigma$ should be increased

Algorithm 5 Adaptive adaptation

Input: standard deviation $\sigma$, success rate $p_s$, threshold $\theta$, modifying parameter $\alpha > 1$

Output: adapted standard deviation $\sigma$

1: if $p_s > \theta$ {
2: return $\alpha \cdot \sigma$
3: }
4: if $p_s < \theta$ {
5: return $\sigma / \alpha$
6: }
7: return $\sigma$
Self-adaption

Implementation:

• storing the standard deviation $\sigma$ on generating the individual as additional information

⇒ using a *strategy parameter*
  (will be varied on mutation by random very likely)

• „good“ values for $\sigma$ win through better quality of the childs
Experimental comparison

testing environment

- 10-dimensional sphere
- Hillclimber
- **but:** $\lambda = 10$ child individuals per generation will be generated
- real-valued Gaussian-Mutation with $\sigma = 1$
- Environment selection of the best of parents and children
- $\theta = \frac{1}{5}$ und $\alpha = 1.224$
Self-adaptive Gaussian Mutation

\textbf{Algorithm 6} Self-adaptive Gaussian Mutation

\textbf{Input:} individual $A$ with $A.G \in \mathbb{R}^l$

\textbf{Output:} varied individual $B$ with $B.G \in \mathbb{R}^l$

1: $u \leftarrow$ choose randomly according to $\mathcal{N}(0, 1)$

2: $B.S_1 \leftarrow A.S_1 \cdot \exp\left(\frac{1}{\sqrt{l}} u\right)$

3: \hspace{1em} \textbf{for each} $i \in \{1, \ldots, l\}$ \{ 

4: \hspace{2em} $u \leftarrow$ choose randomly according to $\mathcal{N}(0, B.S_1)$

5: \hspace{2em} $B.G_i \leftarrow A.G_i + u_i$

6: \hspace{2em} $B.G_i \leftarrow \max\{B.G_i, uG_i\}$ \hspace{1em} /* lower range bound $uG_i$ */

7: \hspace{2em} $B.G_i \leftarrow \min\{B.G_i, uG_i\}$ \hspace{1em} /* upper range bound $uG_i$ */

8: \hspace{1em} \}

9: \hspace{1em} \textbf{return} $B$
Result of comparison

![Graph showing various lines representing different types of adaptation: predefined, constant step width, adaptive, and selbf-adaptive. The y-axis represents Güte (quality) on a logarithmic scale, ranging from 10^-6 to 10^0, and the x-axis represents generation, ranging from 0 to 140. The lines illustrate the performance and adaptation across generations.]

- Predefined adaptation
- Constant step width
- Adaptive adaptation
- Selbf-adaptive adaptation
Result of comparison

- **Step width**
  - Predefined adaptation
  - Adaptive adaptation
  - Self-adaptive adaptation
Outline

1. Motivation
2. One-Parent-Operators
3. Two- or Multiple-Parent-Operators
4. Interpolating and extrapolating recombination
5. Self-adapting algorithms
6. Summary
### Relation 1

<table>
<thead>
<tr>
<th>Condition</th>
<th>Target value</th>
<th>Expected impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>genotype</td>
<td>mutation</td>
<td>influences vicinity of mutation operator</td>
</tr>
<tr>
<td>mutation</td>
<td>exploration</td>
<td>random mutations support exploration</td>
</tr>
<tr>
<td>mutation</td>
<td>fine tuning</td>
<td>local mutations (w.r.t fitness) support fine tuning</td>
</tr>
<tr>
<td>mutation</td>
<td>diversity</td>
<td>mutation increases diversity</td>
</tr>
<tr>
<td>mutation</td>
<td>local optima</td>
<td>local mutations (w.r.t fitness) preserve local optima of the phenotype (random mutations can introduce more optima)</td>
</tr>
<tr>
<td>recombination</td>
<td>exploration</td>
<td>extrapolating operators strengthen exploration</td>
</tr>
<tr>
<td>recombination</td>
<td>fine tuning</td>
<td>interpolating operators strengthen fine tuning</td>
</tr>
<tr>
<td>Condition</td>
<td>Target value</td>
<td>Expected impact</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Div./Recomb.</td>
<td>mutation</td>
<td>small diversity and interpolating recombination damp outlier of the mutation</td>
</tr>
<tr>
<td>Diversity</td>
<td>Recombination</td>
<td>high diversity support mechanism of the recombination</td>
</tr>
<tr>
<td>Selection</td>
<td>Exploration</td>
<td>small selection pressure strengthen the exploration</td>
</tr>
<tr>
<td>Selection</td>
<td>fine tuning</td>
<td>high selection pressure strengthen fine tuning</td>
</tr>
<tr>
<td>Selection</td>
<td>Diversity</td>
<td>Selection mostly decreases diversity</td>
</tr>
<tr>
<td>Div./Recomb.</td>
<td>Exploration</td>
<td>combinating recombination strengthen exploration on high diversity</td>
</tr>
<tr>
<td>Div./Recomb.</td>
<td>fine tuning</td>
<td>combinating recombination strengthen fine tuning on high diversity</td>
</tr>
<tr>
<td>Condition</td>
<td>Target value</td>
<td>Expected impact</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Exploration</td>
<td>Diversity</td>
<td>exploring operations increase diversity</td>
</tr>
<tr>
<td>Fine tuning</td>
<td>Diversity</td>
<td>fine tuning operations decrease diversity</td>
</tr>
<tr>
<td>Diversity</td>
<td>Selection</td>
<td>small diversity decreases selection pressure of the fitness-proportional selection</td>
</tr>
<tr>
<td>Local optima</td>
<td>search progress</td>
<td>huge ammount of local optima inhibits search progress</td>
</tr>
<tr>
<td>Expl./Fine tun./Sel.</td>
<td>search progress</td>
<td>Counterbalancing of all factors is required</td>
</tr>
</tbody>
</table>
Further reading