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

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 σ */
$B_i \leftarrow A_i + u_i$
$B_i \leftarrow \max\{B_i, ug_i\}$ /* lower bound ug_i */
$B_i \leftarrow \min\{B_i, og_i\}$ /* upper bound og_i */
}
return B

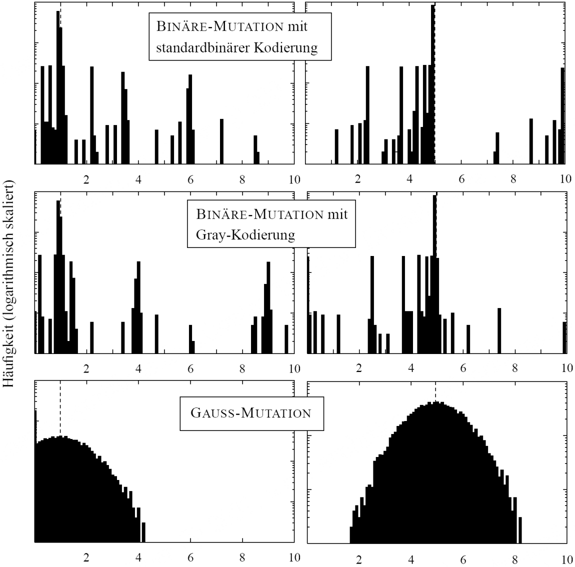
Comparison of the methods

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
Häufigkeit (logarithmisch skaliert)
2
4
6
8
10

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 Ω

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

Standard mutation and Pair swap

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

3	1	4	2	5	4	6	$\longrightarrow$	3	1	6	2	5	4	6

  - if necessary, multiple genes are mutated (see. n-Queens-Problem)
  - *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

3	1	4	2	5	4	6	$\longrightarrow$	3	5	4	2	1	4	6

  - *Precondition:* same allele sets of the exchanged genes
  - *Generalization:* cyclic change of 3, 4, . . . , k genes

Operations on subsequences

- **Shift:**

3	1	4	2	5	4	6	$\longrightarrow$	3	2	5	1	4	4	6

- **arbitrary permutation:**

3	1	4	2	5	4	6	$\longrightarrow$	3	2	1	5	4	4	6

- **Inversion:**

3	1	4	2	5	4	6	$\longrightarrow$	3	5	2	4	1	4	6

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

Outline

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

5	2	1	4	3	6	1
1	2	3	**4**	5	6	
3	1	4	2	5	4	6

$\longrightarrow$

5	2	1	4	5	4	6
1	2	3	**4**	5	6	
3	1	4	2	3	6	1

Two-point-Crossover

- Dertermining of two random cutting points
- Exchange of the gene sequences between both cutting points

5	2	1	4	3	6	1
1	2	3	4	**5**	6	
3	1	4	2	5	4	6

$\longrightarrow$

5	1	4	2	5	6	1
1	2	3	4	**5**	6	
3	2	1	4	3	4	6

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)

5	2	1	4	3	6	1
$+$	$-$	$+$	$-$	$-$	$+$	$-$
3	1	4	2	5	4	6

$\longrightarrow$

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

- **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

Permutation Crossover Unmix

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

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

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

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

- 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 occurence 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)

- chromosom is interpreted as a graph (chain or ring) each gene contains edges to its neighboors 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 |

A:	6	3	1	5	2	7	4

B:	3	7	2	5	6	1	4

Constructing the edge table

Allele	Neighbors in **A**	in **B**	aggregated
1	3, 5	6, 4	3, 4, 5, 6
2	5, 7	7, 5	5^*, 7^*
3	6, 1	4, 7	1, 4, 6, 7
4	7, 6	1, 3	1, 3, 6, 7
5	1, 2	2, 6	1, 2^*, 6
6	4, 3	5, 1	1, 3, 4, 5
7	2, 4	3, 2	2^*, 3, 4

- 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

6	5	2	7	4	3	1

Allele	Neighbor	Selection: 6	5	2	7	4	3	1
1	3, 4, 5, 6	3, 4, 5	3, 4	3, 4	3, 4	3		
2	5^*, 7^*	5^*, 7^*	7^*	**7***	—	—	—	—
3	1, 4, 6, 7	1, 4, 7	1, 4, 7	1, 4, 7	1, 4	1	**1**	—
4	1, 3, 6, 7	1, 3, 7	1, 3, 7	1, 3, 7	1, 3	**1, 3**	—	—
5	1, 2^*, 6	1, 2^*	**1, 2***	—	—	—	—	—
6	1, 3, 4, 5	**1, 3, 4, 5**	—	—	—	—	—	—
7	2^*, 3, 4	2^*, 3, 4	2^*, 3, 4	3, 4	**3, 4**	—	—	—

- start with first allele of the chromosomes **A** (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 arbitrarly
  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 pefect, 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

$\longrightarrow$

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:

$$P(K=k) = \binom{n}{k} p_x^k \, (1-p_x)^{n-k} \qquad \text{mit } n \mathrel{\hat{=}} \text{ Gesamtzahl der Gene}$$

$\Rightarrow$ very small and very large numbers are less likely

Outline

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 Ω 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

$\Rightarrow$ 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 reckombination
- 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** $i \in \{1, \dots, l\}$ {
3: $\quad C.G_i \leftarrow u \cdot A.G_i + (1-u) \cdot B.G_i$
4: }
5: **return** C

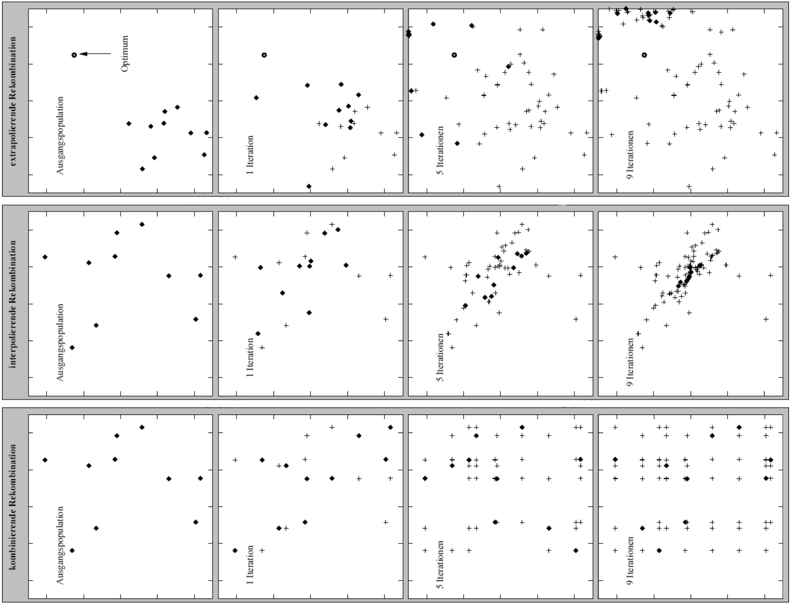
Extrapolating operators

- try to infer information from several individuals

$\Rightarrow$ create a prognosis in what direction one can expect fitness improvements

- extrapolating recombination may leave former Ω
- is only way of recombination which takes fitness values into account
- influence of diversity is hardly understandable
- example: arithmetic crossover with $u \in U([1, 2])$

Comparison


extrapolierende Rekombination
Optimum
Ausgangspopulation
1 Iteration
5 Iterationen
9 Iterationen
interpolierende Rekombination
Ausgangspopulation
1 Iteration
5 Iterationen
9 Iterationen
kombinierende Rekombination
Ausgangspopulation
1 Iteration
5 Iterationen
9 Iterationen

Outline

Self-adapting algorithms [Weicker, 2007]

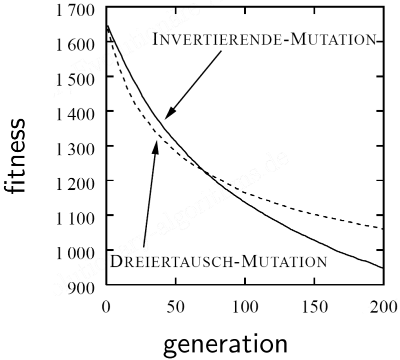
- so far: mutation should change phenotype as small as possible
- now: question if this is valid on every (time) step during the optimization

- control experiment
- solve TSP (here 51 cities) by Hillclimbing

$\Rightarrow$ no recombination

- differently local mutation operators are
  - inversion of a subsequence
  - cyclical exchange of three randomly chosen cities

Influence



- 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

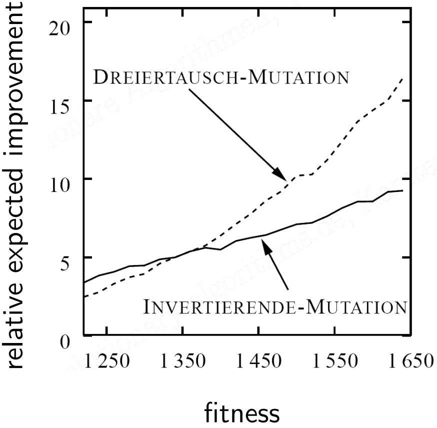
The *fitness improvement* of an individual $A \in \mathcal{G}$ to another individual $B \in \mathcal{G}$ is defined as

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

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

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

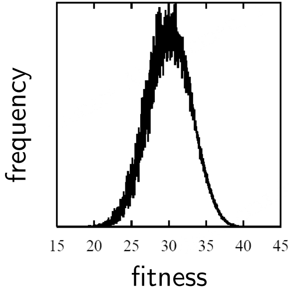
Influence

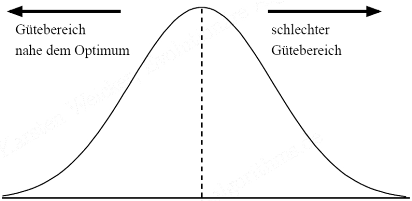


- determining the relative expected improvement in different fitness ranges by random samples from Ω
- responsible for illustrated effect

⇒ How frequent are the different fitness values in Ω?

Complete space

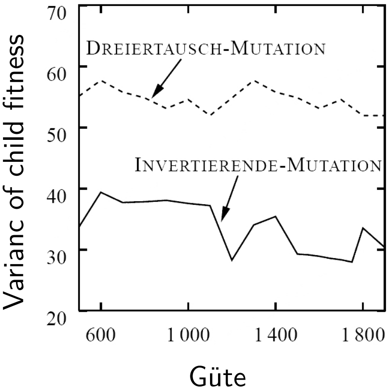




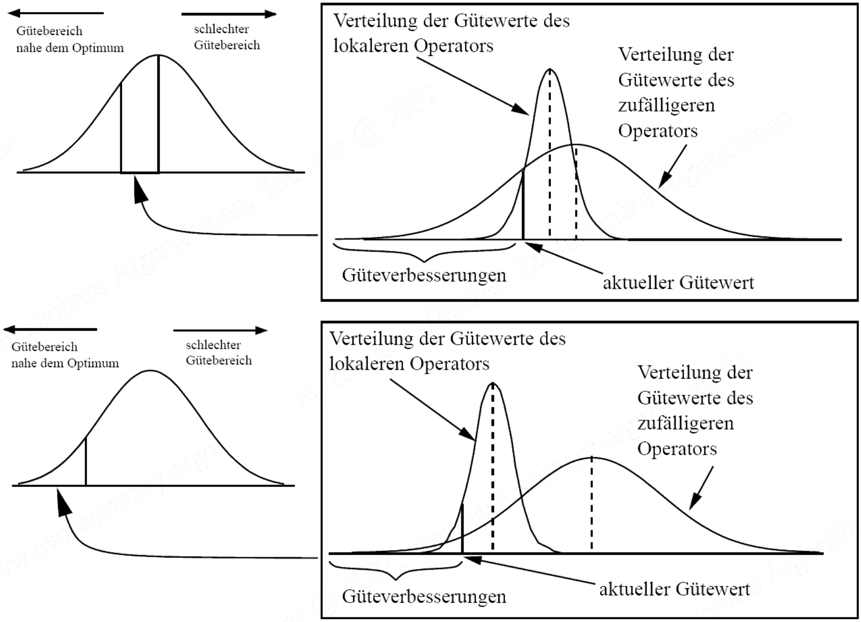
- left: density distribution of a TSP with 11 cities
- right: idealized density distribution of a minimization problem
- similar distribution on children (generated after mutation)

Variance of the generated fitness

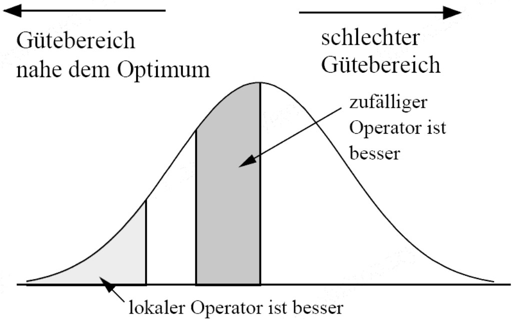
- *locality* of the mutation operator is very important
- very local $\Rightarrow$ fitness values in vicinity of the fitness of the parents
- less local $\Rightarrow$ bigger range of fitness values is covered



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


Gütebereich
nahe dem Optimum
schlechter
Gütebereich
Verteilung der Gütewerte des
lokaleren Operators
Verteilung der
Gütewerte des
zufälligeren
Operators
Güteverbesserungen
aktueller Gütewert
Gütebereich
nahe dem Optimum
schlechter
Gütebereich
Verteilung der Gütewerte des
lokaleren Operators
Verteilung der
Gütewerte des
zufälligeren
Operators
Güteverbesserungen
aktueller Gütewert

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 approximization 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
- σ determines average step width
- modifying parameter $0 < \alpha < 1$ lets decrease σ exponentially

Realization:

Algorithm 4 Predefined adaptation

Input: Standard deviation σ, modifying parameter α
Output: adapted standard deviation σ

1: $\sigma' \leftarrow \alpha \cdot \sigma$
2: **return** σ'

Adaptive adaptation

- Metric: fraction of improving mutations of last k generations
- if fraction is too „high“ σ should be increased

Algorithm 5 Adaptive adaptation

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

Output: adapted standard deviation σ

if $p_s > \theta$ {
    return $\alpha \cdot \sigma$
}
if $p_s < \theta$ {
    return σ / α
}
return σ

Self-adaption

Implementation:

- storing the standard deviation σ on generating the individual as additional information

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

- „good“ values for σ 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

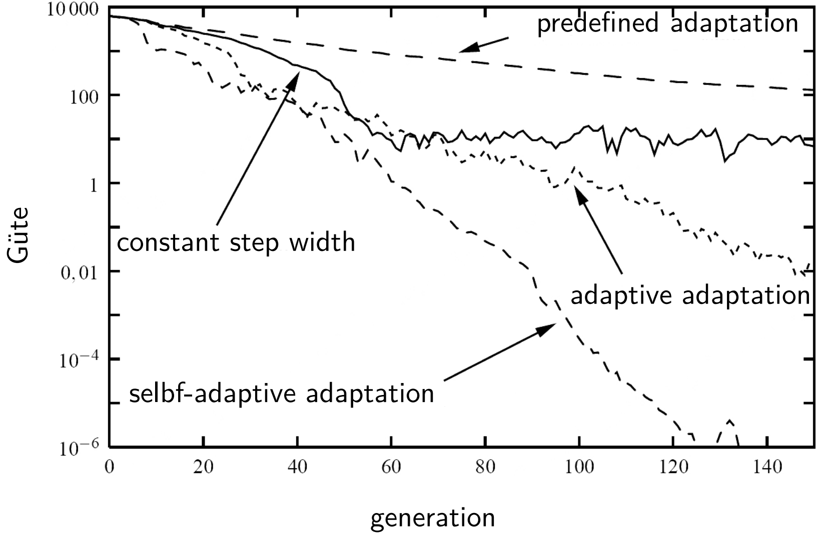
Algorithm 6 Self-adaptive Gaussian Mutation

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

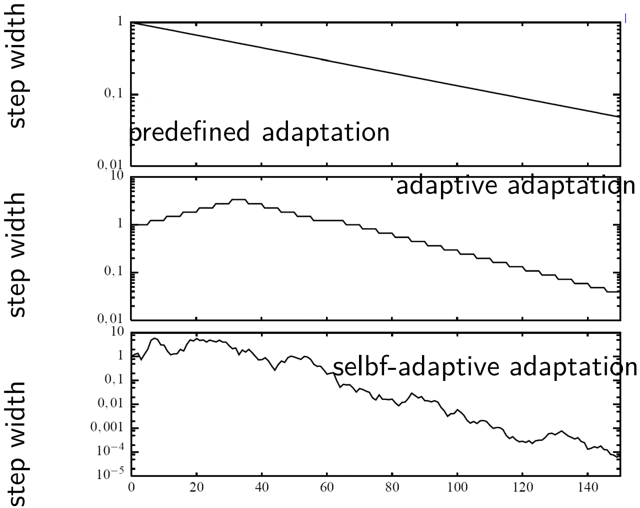
Output: varied individual B with $B.G \in \mathbb{R}^l$

1: $u \leftarrow$ choose randomly according to $\mathcal{N}(0, 1)$
2: $B.S_1 \leftarrow A.S_1 \cdot \exp(\frac{1}{\sqrt{l}} u)$
3: **for each** $i \in \{1, \ldots, l\}$ $\{$
4: $\quad u \leftarrow$ choose randomly according to $\mathcal{N}(0, B.S_1)$
5: $\quad B.G_i \leftarrow A.G_i + u_i$
6: $\quad B.G_i \leftarrow \max\{B.G_i,\ ug_i\}$ /* lower range bound ug_i */
7: $\quad B.G_i \leftarrow \min\{B.G_i,\ ug_i\}$ /* upper range bound og_i */
8: $\}$
9: **return** B

Result of comparison

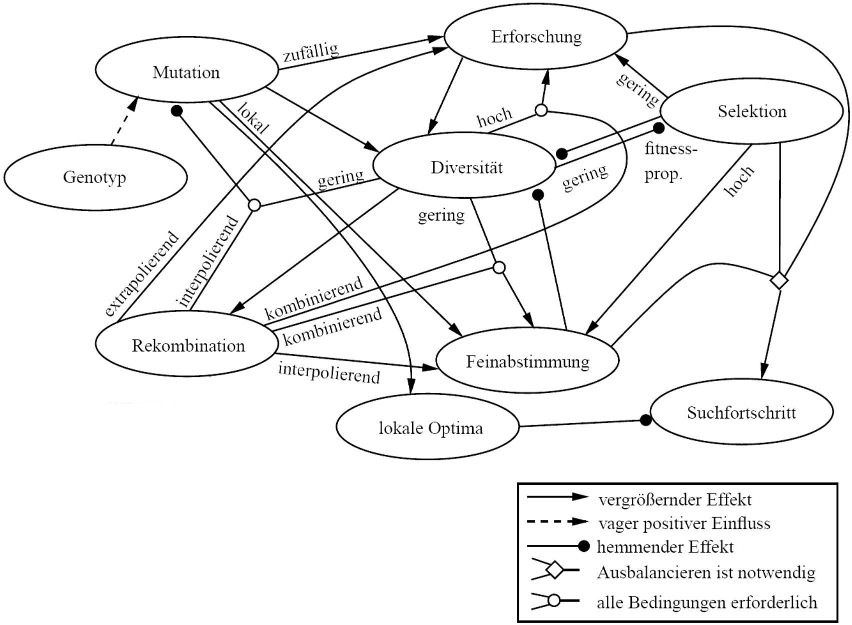


Result of comparison



Outline

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


Erforschung
Mutation
zufällig
Selektion
gering
Genotyp
hoch
lokal
Diversität
fitness-
prop.
gering
gering
hoch
gering
extrapolierend
interpolierend
kombinierend
kombinierend
Rekombination
Feinabstimmung
interpolierend
lokale Optima
Suchfortschritt
vergrößernder Effekt
vager positiver Einfluss
hemmender Effekt
Ausbalancieren ist notwendig
alle Bedingungen erforderlich

Relation I

Condition	Target value	Expected impact
genotype	mutation	influences vicinity of mutation operator
mutation	exploration	random mutations support exploration
mutation	fine tuning	local mutations(w.r.t fitness) support fine tuning
mutation	diversity	mutation increases diversity
mutation	local optima	local mutations(w.r.t fitness) preserve local optima of the phenotype (random mutations can introduce more optima)
recombination	exploration	extrapolating operators strengthen exploration
recombination	fine tuning	interpolating operators strengthen fine tuning

Relations II

Condition	Target value	Expected impact
Div./Recomb.	mutation	small diversity and interpolating recombination damp outlier of the mutation
Diversity	Recombination	high diversity support mechanism of the recombination
Selection	Exploration	small selection pressure strengthen the exploration
Selection	fine tuning	high selection pressure strengthen fine tuning
Selection	Diversity	Selection mostly decreases diversity
Div./Recomb.	Exploration	combinating recombination strengthen exploration on high diversity
Div./Recomb.	fine tuning	combinating recombination strengthen fine tuning on high diversity

Relation III

Condition	Target value	Expected impact
Exploration	Diversity	explorating operations increase d versity
Fine tuning	Diversity	fine tuning operations decrease d versity
Diversity	Selection	small diversity decreases selecti pressure of the fitness-proportion selection
local optima	search progress	huge ammount of local optima inh bits search progress
Expl./Fine tun./Sel.	search progress	Counterbalancing of all factors is r quired

Further reading

Weicker, K. (2007).
Evolutionäre Algorithmen.
Teubner Verlag, Stuttgart, Germany, 2nd edition.