Artificial Intelligence

Evolutionary Algorithms

Lesson 3: Elements of Evolutionary Algorithms

Vincenzo Piuri

Università degli Studi di Milano

Contents

- Encoding
  - Representation of similar phenotypes
  - Hamming cliffs
  - Similar candidate solutions
- Fitness
  - Selection intensity
  - Selective pressure
  - Roulette-wheel selection
  - Premature convergence
  - Vanishing selective pressure
  - Adapting of the fitness function
- Selection
  - Roulette-wheel selection
  - Rank-based selection
  - Tournament selection
  - Elitism
  - Prevention of crowding
  - Characterization of selection methods
- Genetic operators
  - One-parent operators: Mutation, Swap
  - Two- and Multi-parent operators: Crossover, Recombination
  - Self-adapting algorithms

Encoding

Encoding (1)

Encoding of a solution candidate

- Chosen problem-specific
- No general "recipe" to find a good encoding
- Some principles should be taken into consideration

Encoding (2)

Desirable properties

- Representation of similar phenotypes by similar genotype
- Similar fitness on similar candidates
- Closure on Ω under the used evolutionary operators

Representation of Similar Phenotypes (1)

Similar phenotypes should be represented by similar genotypes

- Mutations of certain genes result in similar genotypes
- If trait is not satisfied, obvious changes cannot be generated in some cases
- Huge change of the genotype to produce a similar (and perhaps better) phenotype

Representation of Similar Phenotypes (2)

Example:

- Optimization of a real function

$$y = f(x_1, \ldots, x_n)$$

- Representation of the (real) arguments by binary codes
- Problem: binary representation leads to "Hamming Cliffs"
  - Adjacent numbers are coded with bit strings having big Hamming distance
  - Mutations/Crossover overcome Hamming Cliffs very hardly

Similar Candidate Solutions (1)

Similarly encoded candidate solutions should have a similar fitness

- Problem of the epistasis
  - In biology
    - One allele of a (so-called epistatic) gene suppresses the effect of all possible alleles of another gene
  - In evolutionary algorithms
    - Interaction between genes of a chromosome
    - Changes of the fitness by modifying one gene strongly depends on the value(s) of (an)other gene(s)

Similar Candidate Solutions (2)

The Traveling Salesman Problem

- Find a round trip of n cities with respect to minimal costs
- Two different encodings
  1. Permutation of the cities
     - Visit city at the k-th position in the k-th step
     - **Low epistasis**: swapping two cities alters fitness (costs) by comparable amounts (only local changes)
  2. Specification of a list of numbers that state the position of the next city to be visited in a (sorted) list from which all already visited cities have been deleted
     - **High epistasis**: modifying a single gene (especially the closer to the top) may alter the complete round trip (global tour-change)
     - Leads mostly to large changes of the fitness

Similar Candidate Solutions (3)

- Second encoding: effect of a mutation

Mutation	Chromosome	Remaining cities	Round trip
before	5 3 3 2 2 1	1, 2, 3, 4, **5**, 6 1, 2, **3**, 4, 6 1, 2, **4**, 6 1, **2**, 6 1, **6** **1**	5 3 4 2 6 1
after	1 3 3 2 2 1	**1**, 2, 3, 4, 5, 6 2, 3, **4**, 5, 6 2, 3, **5**, 6 2, **3**, 6 2, **6** **2**	1 4 5 3 6 2

Similar Candidate Solutions (4)

Epistasis

- High epistatic encoding: no regularities
  - Mutation/Crossover leads to random fitness changes
  - Optimization problem is very hard to solve by EAs
- Very low epistatic encoding: other methods often more successful
- Epistasis = property of the encoding, not of the problem itself
  - ∃ encodings of a problem with higher and lower epistasis
  - ∃ problems with low epistatic encoding: too hard to solve by an EA

Closure under the Used Operators (1)

If possible, the search space Ω should be closed under the used genetic operators

- Search space is left, if
  - New chromosome cannot be meaningfully interpreted or decoded
  - A candidate solution does not fulfill certain basic requirements
  - A candidate solution is evaluated incorrectly by the fitness function

Closure under the Used Operators (2)

- Problem of coordination of encoding and EA-operators:
  - Choose or design encoding-specific genetic operators
  - Use mechanisms to "repair" chromosomes
  - Introduce a penalty term that reduces the fitness of such individuals $\notin \Omega$

Closure under the Used Operators (3)

Leaving the search space – example: the n-Queens Problem

- Two different encodings: chromosome of length n
  1. File positions of queens per rank (alleles $0,\ldots,n-1$)
     - Operators: One-point Crossover, standard mutation generates always valid vectors of file position
     - **Search space is not left**
  2. Numbers of the field (alleles $0,\ldots,\ n^2-1$) of the queens
     - Operators: One-point crossover, standard mutation generates chromosomes with more than one queen per field
     - **Search space is left**

Closure under the Used Operators (4)

- Use other encoding
  - First encoding avoids problem and Ω is considerably smaller
- Encoding-specific evolutionary operators
  - Mutation: Excluding already existing alleles on random
  - Crossover: look at first for field numbers of each chromosome which are not contained in other chromosomes and apply one-point crossover on shortened chromosomes

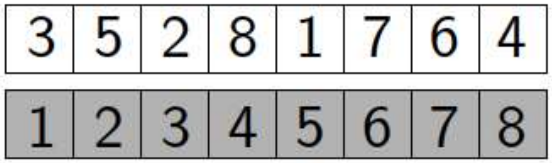
Closure under the Used Operators (5)

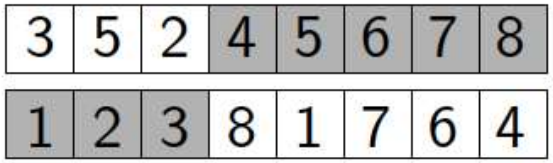
- Repair mechanisms
  - Find and replace multiple occurring field numbers until all field numbers are distinct
- Penalty term
  - Reduce fitness by amount of multiple allocations of fields multiplied with weight if necessary

Closure under the Used Operators (6)

Leaving search space – example: TSP

- Representation of the round trip by permutation of the cities (city at k-th position is visited in the k-th step)
- One-point crossover can exceed the space of permutations





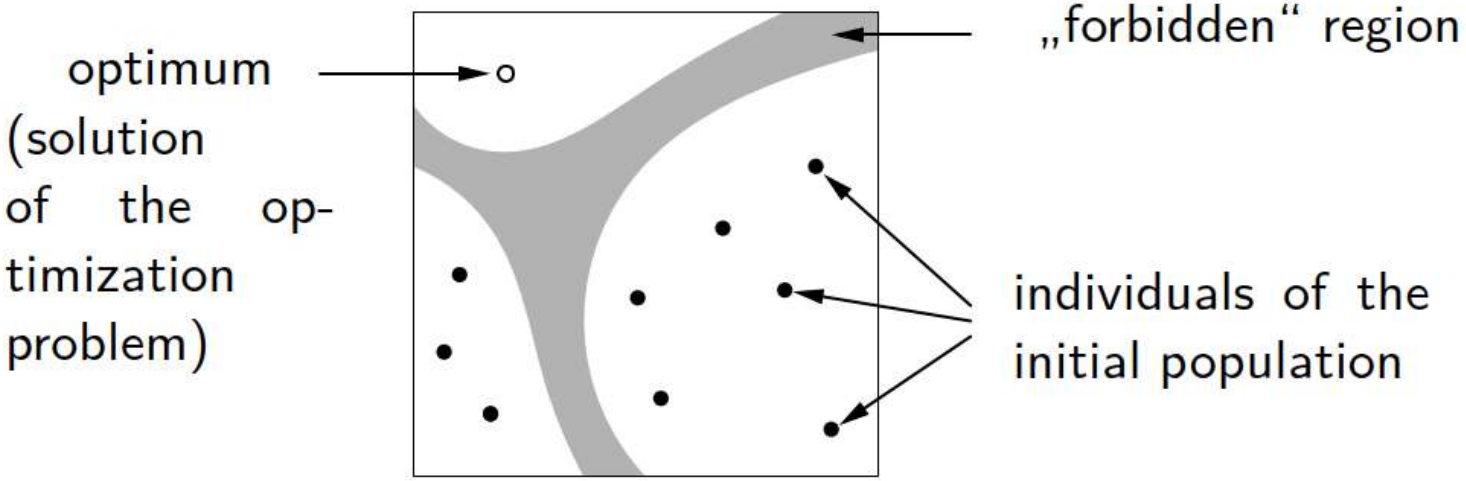
Closure under the Used Operators (7)

- Encoding-specific evolutionary operators:
  - Mutation: e.g. pair swaps, shift/cyclic permutation, inversion
  - Crossover: edge recombination
- Repair mechanisms
  - Remove twice occurring cities and append the missing cities at the end

- Penalty term
  - Reduce fitness by value c for each missing city

Closure under the Used Operators (8)

- If Ω is not connected, repair mechanisms can complicate the search
  - Restoring "forbidden" $x \in \Omega$ in permitted regions



- Introduce penalty term
  - $x \in \Omega$ in "forbidden" region is penalized but not removed
  - Penalty term should be increased on time: suppresses $x \in \Omega$ in "forbidden" regions in following generations

Fitness

Fitness (1)

- Better individuals (better fitness) should have better chances to create offspring
- **Selective pressure**: strength of preferencing good individuals
  - Exploration of the space
    - Deviation of the individuals over Ω as wide as possible
    - Preferably big chances to find global optimum
    - Smaller selective pressure
  - Exploitation of good individuals
    - Strive for (perhaps local) optimum in the vicinity of good individuals
    - Convergence to optimum
    - Higher selective pressure

Fitness (2)

- Metrics for creating selective pressure
  - Time to takeover: number of generations until population converges (all individuals are identical)
  - **Selection intensity**: the differential between average quality before and after the selection

Selection Intensity (1)

Let $(\Omega, f, \succ)$ be a considered optimization problem and a selection operator $\mathsf{Sel}^{\xi} : (\mathcal{G} \times \mathcal{Z} \times \mathbb{R})^r \rightarrow (\mathcal{G} \times \mathcal{Z} \times \mathbb{R})^s$ is applied on a population P with an average quality μ_f and standard deviation σ_f. Then, let μ_f^{sel} be the average quality of the population P_{sel} and the selection operator has the *selection intensity*

$$I_{\mathsf{sel}} = \begin{cases} \frac{\mu_f^{\mathsf{sel}} - \mu_f}{\sigma_f} & \text{if} \quad \succ = \{=, >\} \\ \frac{\mu_f - \mu_f^{\mathsf{sel}}}{\sigma_f} & \text{otherwise} \end{cases}$$

Selection Intensity (2)

- 10 Indiv. with fitness: 2.0, 2.1, 3.0, 4.0, 4.3, 4.4, 4.5, 4.9, 5.5, 6.0
- selection leads to individuals with quality: 2.0, 3.0, 4.0, 4.4, 5.5

$$\mu_f = \frac{1}{|P|}\sum_{i=1}^{|P|} A^{(i)}.F \qquad \text{(Average of the fitness)}$$

$$\sigma_f = \sqrt{\frac{1}{|P|-1}\sum_{i=1}^{|P|}(A^{(i)}.F - \mu_f)^2} \qquad \text{(standard deviation)}$$

$$\Rightarrow \quad \mu_f = 4.07, \quad \sigma_f = 1.27, \quad \mu_f^{\text{sel}} = 3.78, \quad I_{\text{sel}} = \frac{4.07-3.78}{1.27} = 0.228$$

- Criticism on selection intensity
  - Requires a standard normal distribution of values
  - Rarely applicable on general optimization problems

Selective Pressure (1)

- Time-dependent selective pressure
  - Low selective pressure in prior generations
  - Higher selective pressure in later generations
  - First: good exploration of the space
  - Then: exploitation of the promising region
- Regulation of selective pressure
  - Adapting the fitness function
  - Adapting parameters of the selection method

Selective Pressure (2)

- Selection methods
  - Roulette-wheel selection
  - Rank-based selection
  - Tournament selection
- Adaptation methods
  - Variation of the fitness
  - Linear dynamical scaling
  - σ-scaling

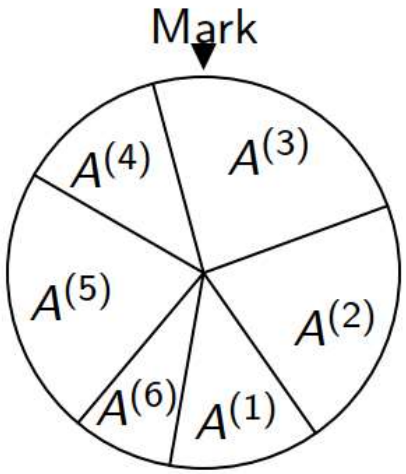
Roulette-Wheel Selection (1)

- Best known selection method
  - Computes the relative fitness of the individuals $A^{(i)}$

$$f_{\text{rel}}\left(A^{(i)}\right) = \frac{A^{(i)}.F}{\sum_{j=1}^{|P|} A^{(j)}.F}$$

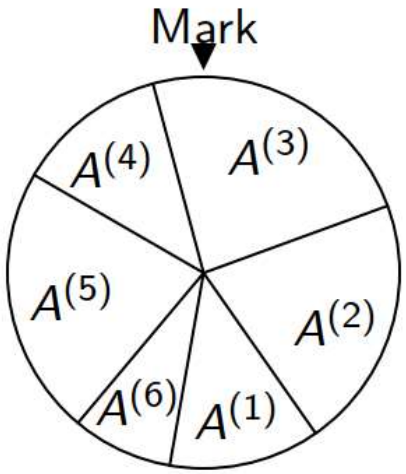
- **Fitness-proportionate selection**
  - Interprets $f_{rel}(A^{(i)})$ as a probability to be selected
  - Absolute fitness A.F may not be negative
  - Fitness has to be maximized
    - otherwise: selection of bad individuals with high probability

Roulette-Wheel Selection (2)



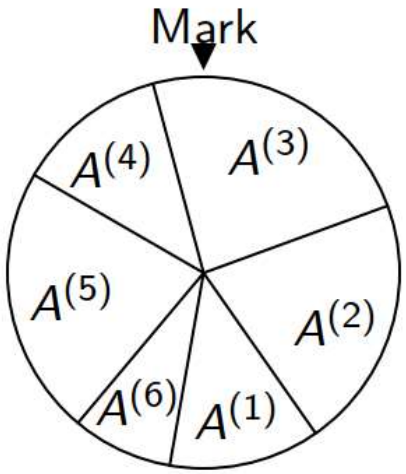
- Roulette-wheel with 1 sector per individual $A^{(i)}$
  - Sector size = relative fitness values $f_{rel}(A^{(i)})$
- Selection of an individual
  1. Set the roulette wheel into motion
  2. Choose the individual of the corresponding sector

Roulette-Wheel Selection (3)



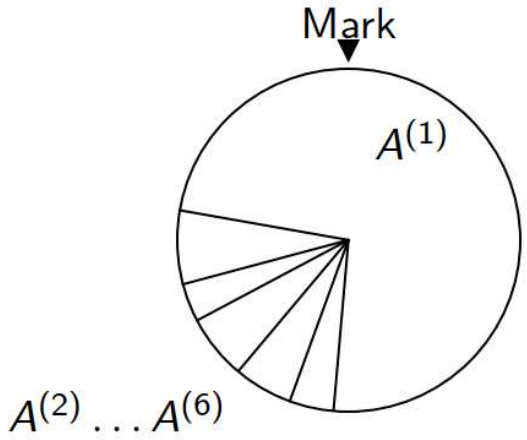
- Selection of the next population
  - Repeat selection for the number of desired individuals

Roulette-Wheel Selection (4)



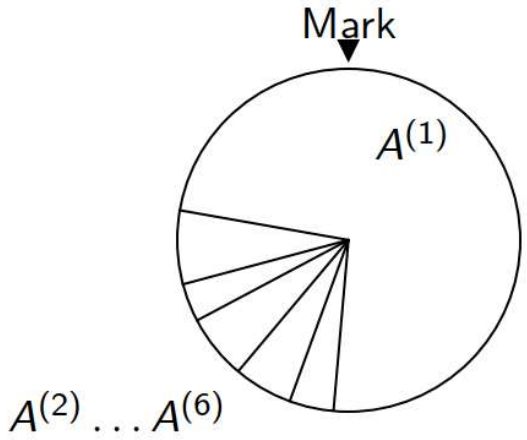
- Disadvantage
  - Computation of the relative fitness by summing up all fitness values (normalization factor)
  - Constant initial population during the selection
  - Complex parallelization of the implementation

Roulette-Wheel Selection (5)



- Individuals with very high fitness may dominate the selection
  - Many copies/very similar individuals: dominance may become stronger in subsequent generations
  - Crowding: population of very similar/identical individuals

Roulette-Wheel Selection (6)



- Problem: very fast find of the (local) optimum
  - Diversity of the population vanishes
  - Exploitation of worse individuals
  - No exploration of the space but local optimization
    - preferred in later generations, undesirable at the beginning

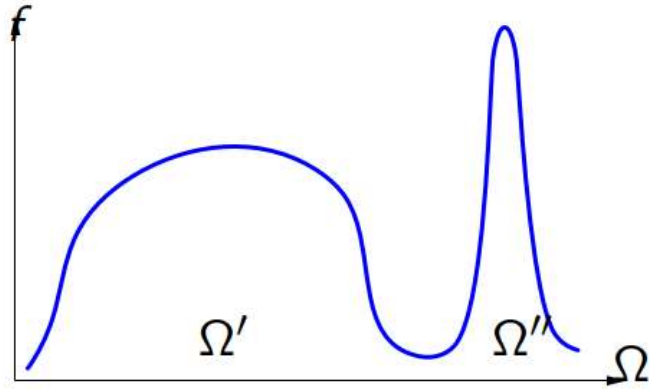
Roulette-Wheel Selection (7)

- When using a simple fitness-proportionate selection in a population with average quality μ_f and variation of the quality σ_f^2, the selection intensity is

$$I_{sel} = \frac{\sigma_f}{\mu_f}$$

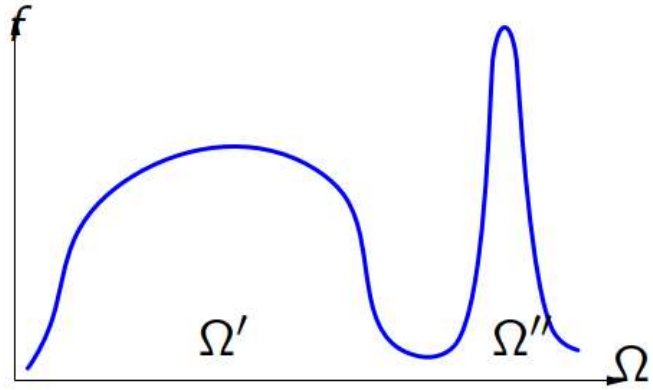
Premature Convergence (1)

- Strong influence of the fitness function on the effect of the fitness-proportionate selection
- Premature convergence
  - Range of the maximizing function is very huge
  - No chromosome at the beginning in the section Ω”
  - Population remains by selection in the vicinity of the local maximum in the section Ω’



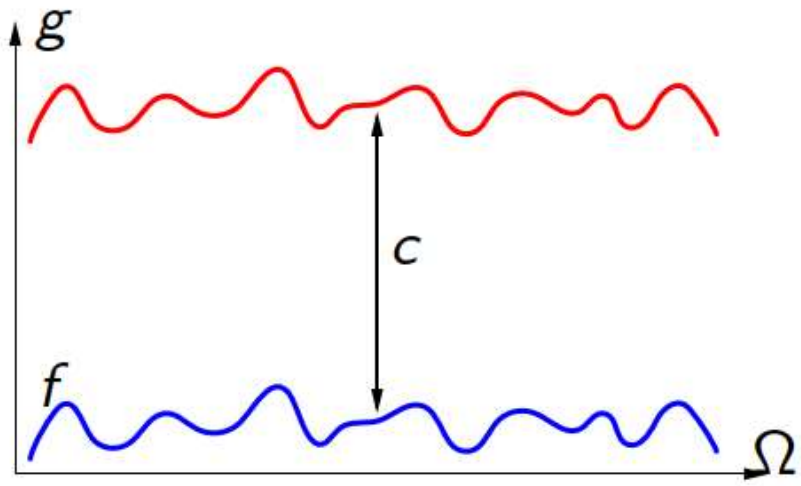
Premature Convergence (2)

- Individuals which converge to the section between Ω′ and Ω′′ have worse chances to create offspring



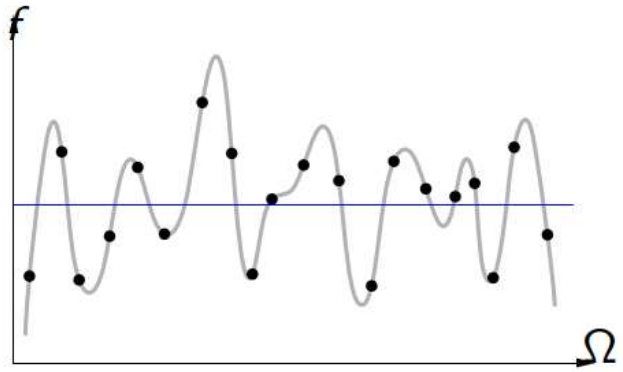
Vanishing Selective Pressure (1)

- Problem of the absolute height of the fitness values (in this case, the variation)
  - Maximizing of $f : \Omega \rightarrow \mathbb{R}$ is equivalent to the maximization of $g : \Omega \rightarrow \mathbb{R}$ with $g(x) = f(x) + c, \quad c \in \mathbb{R}$

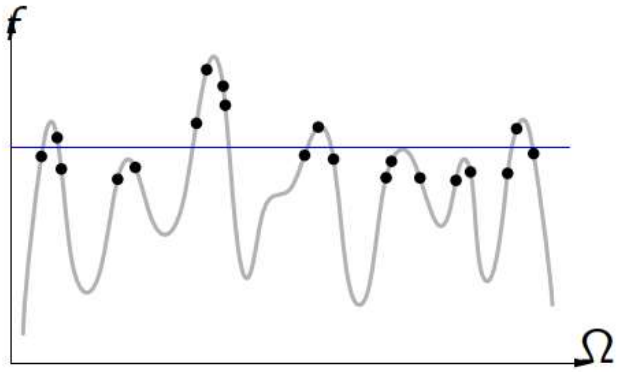


Vanishing Selective Pressure (2)

- Problem is perhaps grounded on the EA itself
  - It increases tendentially the (average) fitness of the individuals
  - Higher selective pressure at the beginning due to random fitness values
  - Later: smaller selective pressure (inverse way is preferred)



early generation



later generation

Adapting the Fitness Function (1)

- Scaling of the fitness
  - **Linear dynamical scaling**

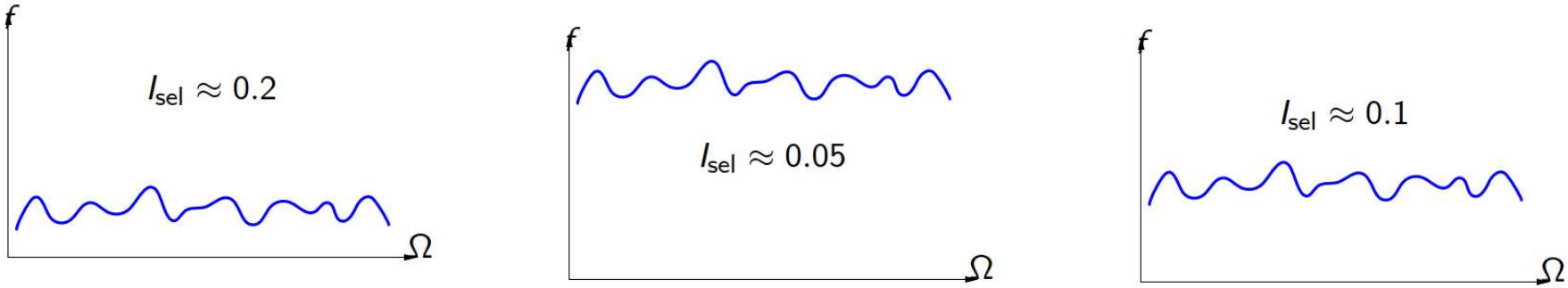
$$f_{\text{lds}}(A) = \alpha \cdot A.F - \min\left\{A^{(i)}.F \mid P(t) = \left\{A^{(1)}, \ldots, A^{(r)}\right\}\right\}, \quad \alpha > 0$$

    - Instead of minimum of $P(t)$, minimum of the last k generations can be used
    - Usually $\alpha > 1$
  - **σ-Scaling**

$$f_{\sigma}(A) = A.F - (\mu_f(t) - \beta \cdot \sigma_f(t)), \quad \beta > 0$$

    - Problem: choice of the parameter α and β

Adapting the Fitness Function (2)



- Too high I_{sel}
  - Premature convergence
- Too small I_{sel}
  - Vanishing selective pressure
- Appropriate: $I_{sel} \approx 0.1$

Adapting the Fitness Function (3)

- Dependence on time
  - Determines f_{rel} not directly from $f(x)$ but $g(x) \equiv (f(x))^{k(t)}$
  - Time-dependent exponent $k(t)$ regulates selective pressure

$$k(t) = \left(\frac{I^*_{\text{sel}}}{I_{\text{sel}}}\right)^{\beta_1} \left(\tan\left(\frac{t}{T+1} \cdot \frac{\pi}{2}\right)\right)^{\beta_2 \left(\frac{I_{\text{sel}}}{I^*_{\text{sel}}}\right)^{\alpha}}$$

    - Limits selection intensity in the vicinity of $I_{sel} \approx 0.1$
    - I_{sel}: coefficient of variation
    - T: max. number of remaining generations to be computed
    - t: current time step (number of generation)

Adapting the Fitness Function (4)

- Boltzmann-Selection
  - Determines relative fitness not directly from $f(x)$ but $g(x) \equiv exp^{(\frac{f(x)}{kT})}$
    - Time-dependent temperature T controls selective pressure
    - k is a normalizing constant
    - Temperature decreases, e.g. linearly, to the predefined maximum number of generations

Selection

Roulette-Wheel Selection (1)

- **Variance problem**
  - Selection of individuals is proportional to the fitness, but random
  - No guarantee that "fitter" individuals are taken to the next generation, not even for the best individual
    - High deviation (high variance) of the offspring of an individual

Roulette-Wheel Selection (2)

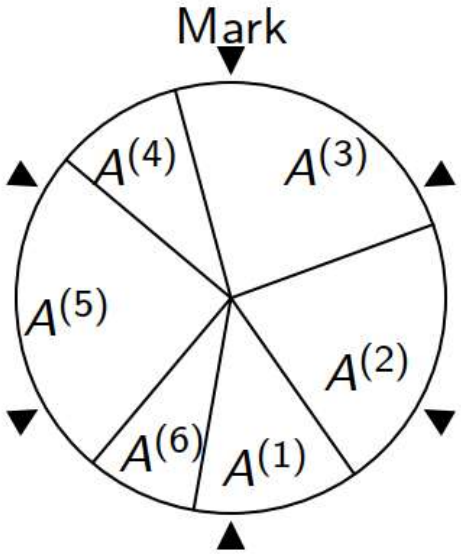
- First simple solution
  - Discretization of the fitness range
    - compute $\mu_f(t)$ and $\sigma_f(t)$ of P
    - if $\mu_f(t) - \sigma_f(t) > f(x)$: 0 offspring
    - if $\mu_f(t) - \sigma_f(t) \leq f(x) \leq \mu_f(t) + \sigma_f(t)$: 1 offspring
    - if $f(x) > \mu_f(t) + \sigma_f(t)$: 2 offsprings

Roulette-Wheel Selection (3)

- Expected value model
  - Generate $\lfloor f_{rel}(s) \cdot |P| \rfloor$ individuals for each solution candidate
  - Fill the population by Roulette-Wheel selection

Roulette-Wheel Selection (4)

- Stochastic Universal Sampling
- Selection of the next population
  - Rotate Roulette-Wheel once
  - Choose one chromosome per mark
    - 1 ×A(1), 1 ×A(2),
      2 ×A(3), 2 ×A(5).
  - Better-than-average individuals are taken into the next population definitely



Roulette-Wheel Selection (5)

- **Voting evaluation**
  - Roulette-Wheel selection but:
    - For each individual A with 1 offspring: $A.F' \leftarrow A.F - \Delta f$
    - If $A.F' < 0$, no further offspring of A
    - Choosing Δf: best individual gets at most predefined number k of offspring

$$\Delta f = \frac{1}{k} \max\{A.F \mid A \in P(t)\}$$

Rank-based Selection (1)

1. Sort individuals in descending order according to their fitness
2. Rank is assigned to each individual in the population
3. Define probability distribution over Rank scale: the lower the rank the lower the probability
4. Roulette-Wheel selection based on the distribution

Rank-based Selection (2)

- Advantage
  - Avoidance of dominance problem: decoupling of fitness value and selection probability
  - Regulation of the selective pressure by probability distribution on rank scale
- Disadvantage
  - Sorting of individuals (complexity: $|P| \cdot \log |P|$)

Tournament Selection (1)

1. Draw k individuals $(2 \leq k < |P|)$ randomly from $P(t)$
2. Individuals carry out the tournament and best individual wins:
   - Tournament winner receives a descendant in the next population
3. All participants (even the winner) of the tournament are returned to $P(t)$

Tournament Selection (2)

- Advantage
  - Avoidance of the dominance problem: decoupling of fitness value and selection probability
  - Regulation of the selective pressure by tournament size with limitations

Elitism (1)

- Only the expected value model ensures that the best individual enters the next generation
  - If best individual in next population: no protection from modifications by genetic operators
  - Fitness of the best individual can decrease from one generation to the next (=> undesired)
- Elitism
  - Unchanged transfer of the best individual into the next generation (or the $k, 1 \leq k < |P|$ best individuals)
  - Elite is not excluded from normal selection: genetic operator can improve them

Elitism (2)

- Many times offspring (Mutation/Crossover products) replace their parents
- "Local" elitism
  - Elitism between parents and offspring
  - Mutation
    - Mutated individual replaces its parents ↔ it has at least the same fitness
  - Crossover
    - Sort the four involved individuals (2 parents, 2 descendants) according to the fitness
    - Best 2 individuals → next generation

Elitism (3)

- Advantage
  - Better convergence as the local optimum is intended more consequently
- Disadvantage
  - Pretty high risk of getting stuck in local optima as no local degradation is possible

Prevention of Crowding (1)

- Deterministic Crowding
  - Generated offspring should always replace those individuals in the population that are most similar
  - Local density of individuals in Ω cannot grow so easily
  - Requires similarity or distance metric for individuals
    - e.g. Hamming-distance on binary coded chromosomes

Prevention of Crowding (2)

- Variant of deterministic crowding
  - Crossover: group individuals into two pairs (1 parents, 1 offspring)
  - Child is assigned to the parent to which it is more similar
  - Take the best individual of each pair
- Advantage
  - Much fewer similarity computations between individuals (only a part of the population is considered)

Sharing

- Idea: reduce the fitness of an individual if there are other individuals in its neighborhood
  - Intuitively: individuals share the resources of a niche
  - Requires similarity or distance metric for individuals

$$f_{\text{share}}(A) = \frac{A.F}{\sum_{B \in P(t)} g(d(A,B))}$$

  - d: Distance metric of the individuals
  - g: Weighting function, defines both shape and size of the niche

$$g(x) = \begin{cases} 1 - \left(\frac{x}{\varrho}\right)^{\alpha} & \text{if } x < \varrho, \\ 0, & \text{otherwise} \end{cases}$$

Characterization of Selection Methods

static	probability of selection remains constant
dynamic	probability of selection changes
extinguishing	probability of selection may be 0
preservative	all probabilities of selection must be >0
pure-bred	individuals can only have offsprings in one generation
under-bred	individuals are allowed to have offsprings in more than one generation
right	all individuals may reproduce
left	the best individuals may *not* reproduce
generational	parents are fixed until all offsprings are created
on the fly	created offsprings directly replace their parents

Genetic Operators

Genetic Operators

- Are applied on certain fraction of chosen individuals (intermediary population)
- Generating mutations and recombinations of already existing solution candidates
- Classification of genetic operators according to the number of parents
  - One-Parent Operators: Mutation
  - Two-Parent Operators: Crossover
  - Multiple-Parent Operators
- Genetic operators have special properties (depend on the encoding)
- If certain combination of alleles unreasonable, genetic operators should never create them

One-Parent Operators

Mutation

- Mutations (variations): small changes in biology
- Mutation operator in EA: changes the fitness (function) of the solution candidate as few as possible
- Exploration
  - Exploration at random
  - Exploration of further-away regions of the space
- Exploitation
  - Local improving of a solution candidate
  - Important: embedding of phenotypic neighborhood

Standard Mutation and Pair Swap

- Standard Mutation

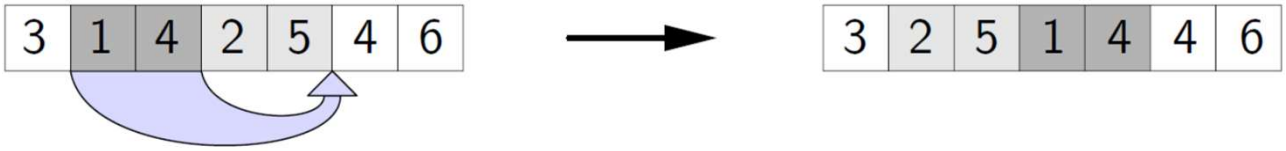
  Exchange the form/value of a gene by another allele

  - If necessary, multiple genes are mutated
  - *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
  - Precondition: same allele sets of the exchanged genes
  - Generalization: cyclic change of $3, 4, \ldots, k$ genes

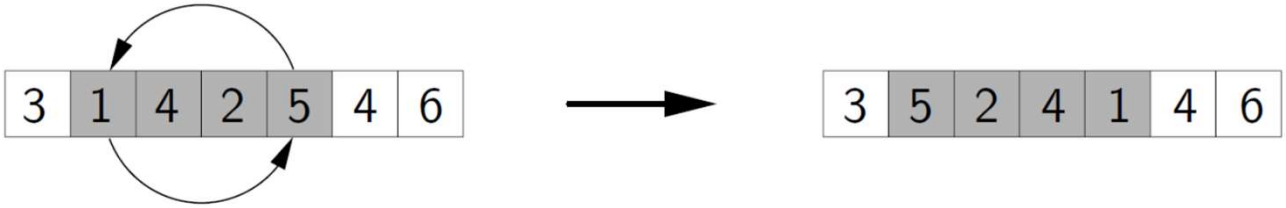
Operations on Subsequences

- Shift



- Arbitrary Permutation

- Inversion



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

Binary Mutation

Algorithm 1 Binary Mutation

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

```
B ← A
for i ∈ {1, ..., l} {
    u ← choose randomly according to U([0,1))
    if u ≤ p_m {                    /* probability of mutation p_m */
        B.G_i ← 1 − A.G_i
    }
}
return B
```

Gaussian Mutation

- Alternative real-valued mutation
  - Directly applied on real-valued numbers
  - Adding a normal distributed random number on each gene

Algorithm 2 Gaussian-Mutation

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

```
for i ∈ {1, ..., l} {
    u_i ← choose randomly according to N(0, σ) /* standard deviation
    σ */
    B_i ← A_i + u_i
    B_i ← max{B_i, ug_i}                    /* lower bound ug_i */
    B_i ← min{B_i, og_i}                    /* upper bound og_i */
}
return B
```

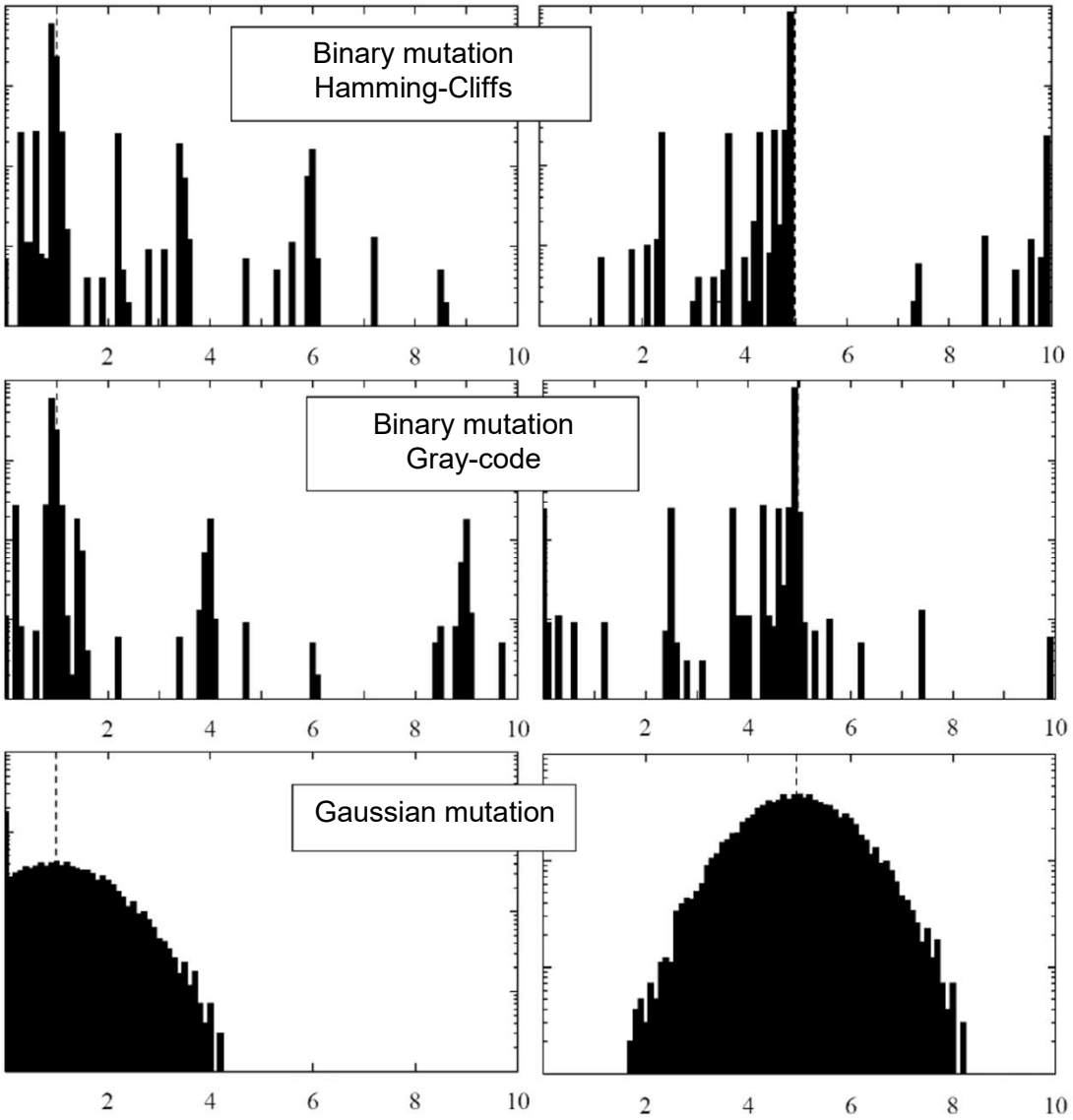
Comparison of the Methods (1)

- Approach
  - Optimizing of the simple function

$$f(x) = \begin{cases} x & \text{if } x \in [0, 10] \subset \mathbb{R} \\ \text{undefined} & \text{otherwise} \end{cases}$$

  - Parents: 1.0 and 4.99
  - Determining the distribution of the descendants with 10000 mutations each

Comparison of the Methods (2)



Comparison of the Methods (3)

- Gaussian mutation
  - with lower $\sigma \Longrightarrow$ well applicable on exploitation
  - with higher $\sigma \Longrightarrow$ wide exploration
- Hamming cliffs = break in frequency distribution
- Gray code succeeds on including phenotypical neighborhood
  - Tends to one part of the space

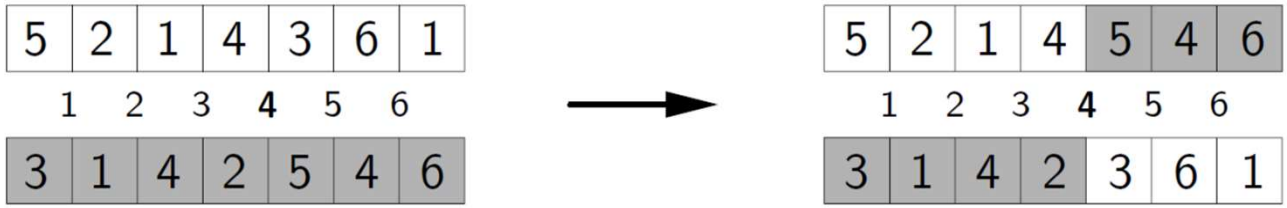
  $\Longrightarrow$ Gaussian mutation oriented to phenotypical neighborhood

  $\Longrightarrow$ Binary mutation faster, detects interesting regions in Ω

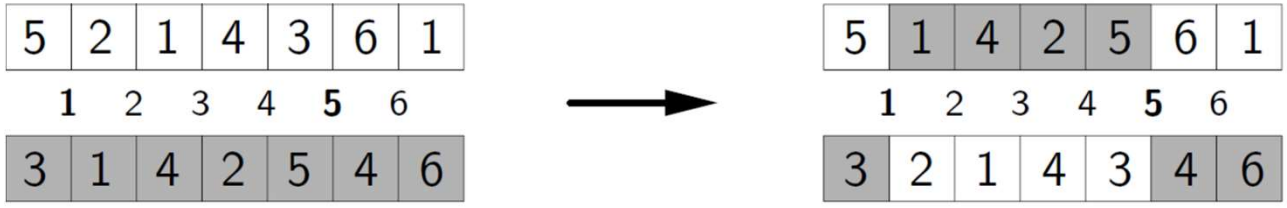
Two- or Multiple- Parents Operators

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

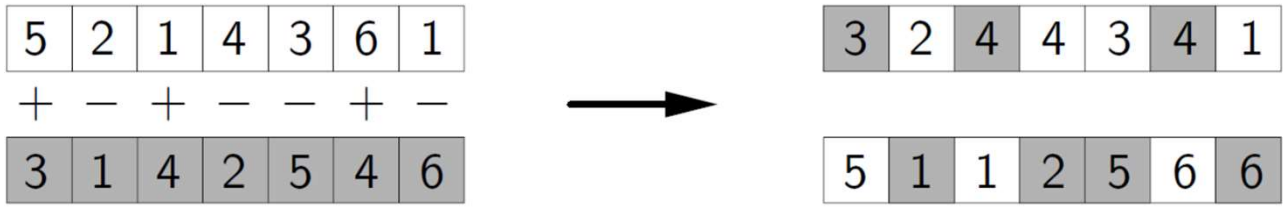


- Two-Point Crossover
  - Determining of two random cutting points
  - Exchange of the gene sequences between both cutting points



n-Point and Uniform Crossover

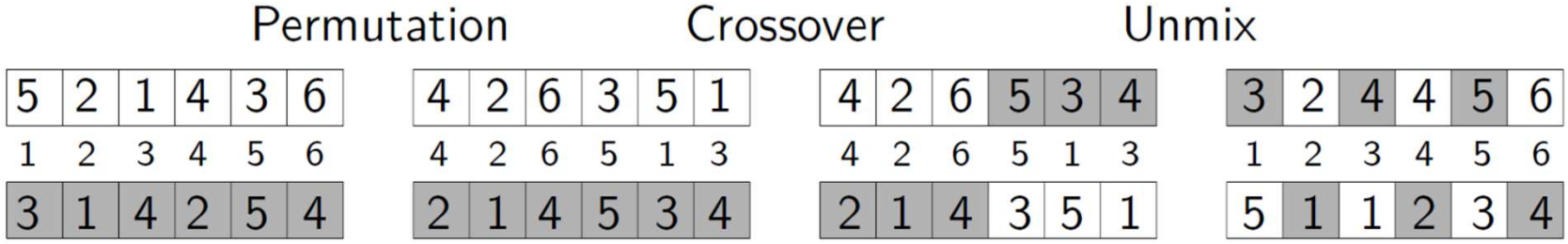
- n-Point Crossover
  - Generalization of the One-Point and Two-Point Crossover
  - Determining of *n* random cutting points
  - Alternating exchange / keep of the gene sequences between two following cutting points
- Uniform Crossover
  - On each gene: determine whether to exchange or not (+:yes, −:no, parameter: probability p_x of exchange)



  - Uniform crossover is not equivalent to the $(l-1)$-point-crossover because the number of the crossover points is chosen randomly

Shuffle Crossover

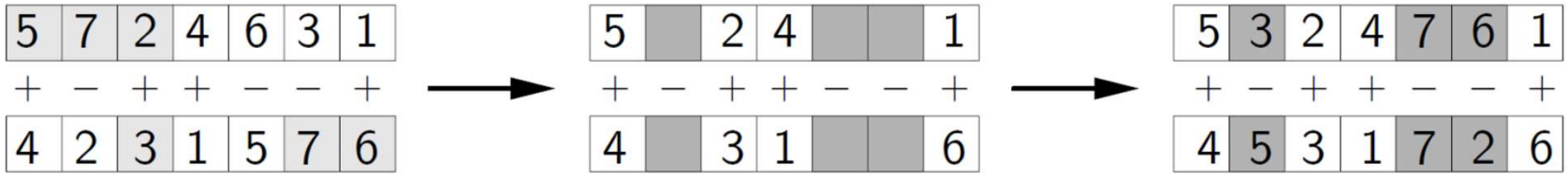
1. Random permutation of the genes
2. One-Point Crossover
3. Unmixing the genes



- 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)



- Preserves order information
- Alternative: Keeping the "+" respect to "−" marked genes in one of the chromosomes

Edge Recombination (1)

- 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

Edge Recombination (2)

- 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)
- 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 precedence
    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

Edge Recombination (3)

- Example

A: | 6 | 3 | 1 | 5 | 2 | 7 | 4 |

B: | 3 | 7 | 2 | 5 | 6 | 1 | 4 |

- Constructing the edge table

Allele	Neighbors in **A**	Neighbors 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 (4)

- 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 and so on

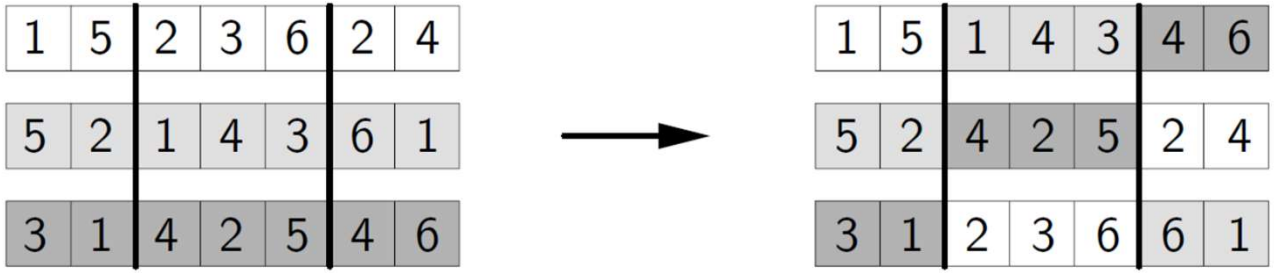
Edge Recombination (5)

- Child has often a new edge

  (from last to the first gene)
- If first and last gene are not seen as neighbors, 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, even though they may not perfect
  - Random selection of the remaining alleles

Three- and Multi-Parent Operators

- Diagonal Crossover
  - similar to 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



  - 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 (1)

- Positional bias
  - 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 since 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 children), if crossover point lies between them
  - The closer 2 genes in the chromosome are located, the fewer crossover points can separate them

    ⇒ genes next to each other are jointly taken in the same child with higher probability than distant genes

Characterization of Crossover Operators (2)

- Distributional bias
  - 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 since 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

$$P(K = k) = \binom{n}{k} p_x^k (1 - p_x)^{n-k}, \qquad \text{with } n \mathrel{\widehat{=}} \text{total number of genes}$$

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

Interpolating Recombination

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

  $\Longrightarrow \Omega$ is thus less explored
- Creates new alleles
- Focuses 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** $i \in \{1, \ldots, l\}$ {
3: $\quad C.G_i \leftarrow u \cdot A.G_i + (1-u) \cdot B.G_i$
4: }
5: **return** C

Extrapolating Recombination

- Try to infer information from several individuals

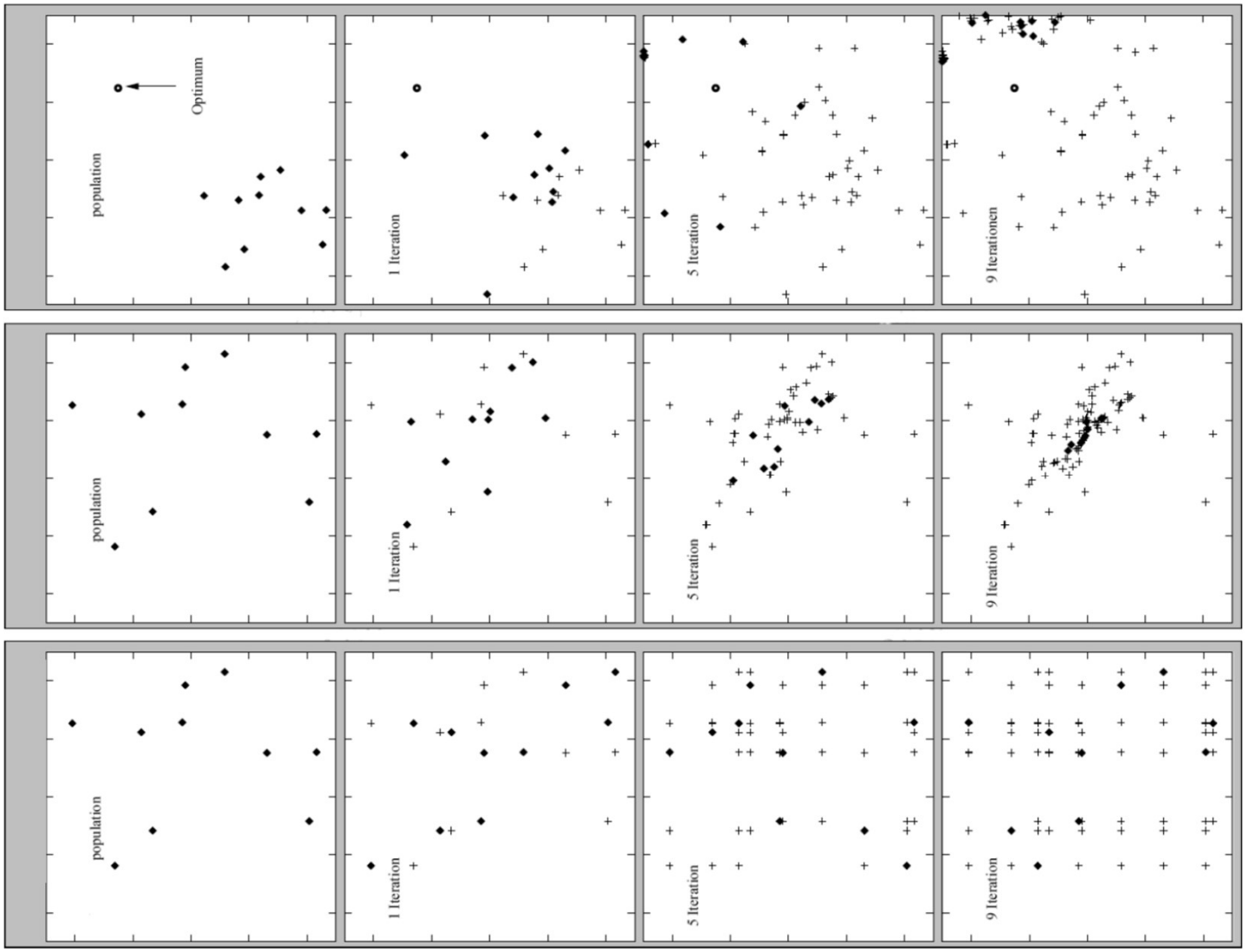
  $\Longrightarrow$ create a prognosis in what direction one can expect fitness improvements
- Creates new alleles
- May leave former Ω
- Is the only way of recombination which takes fitness values into account
- Influence of diversity is hardly understandable
- Example: arithmetic crossover with $u \in \mathrm{U}([1,2])$

Comparison

Extrapolating recombination

Interpolating recombination

Arithmetic recombination



Adaptation Strategies

Adaption Strategies (1)

- Should mutation change phenotype as small as possible in every (time) step during optimization?

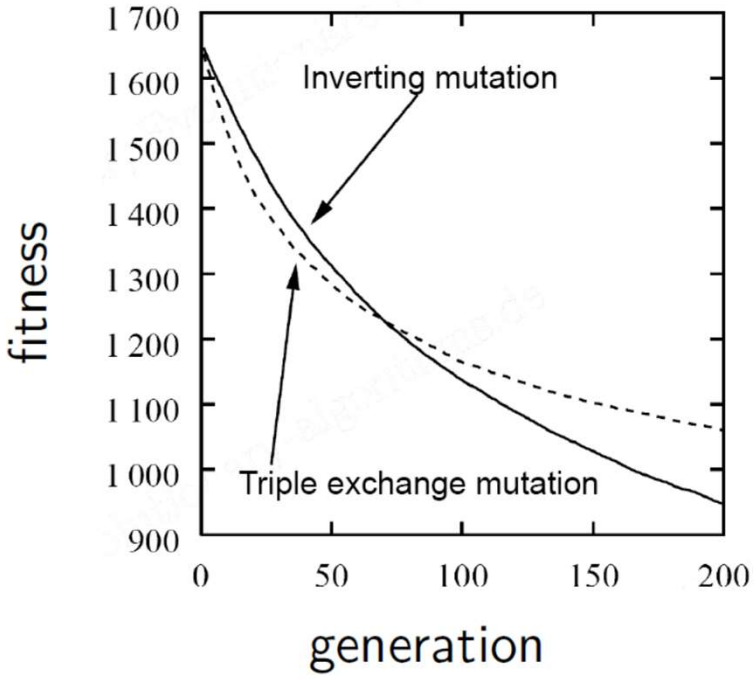
- Local mutation operators
  - Inversion of a subsequence
  - Cyclic exchange of three genes

Adaption Strategies (2)

- Study case: Solve TSP (here 51 cities) by Hill Climbing

  $\Rightarrow$ no recombination

- Is triple exchange mutation always inappropriate? More successful in first 50 generations than favored inversion
- Definition of the relative expected improvement as metric of what improvement an operator enables



Relative Expected Improvement (1)

- Definition: Fitness Improvement

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

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

- Definition: Relative Expected Improvement

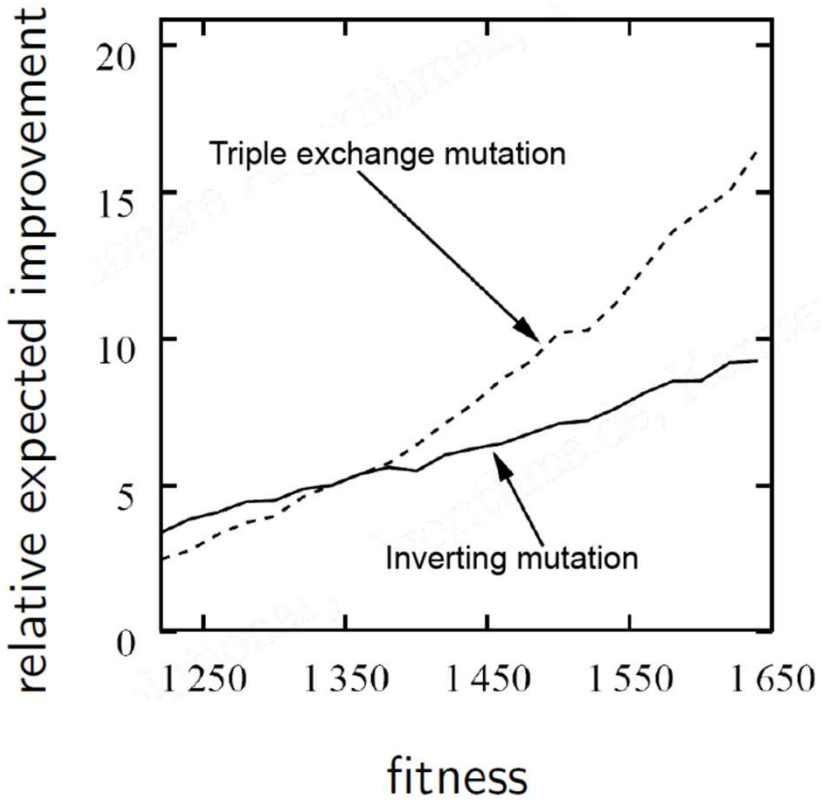
  The relative expected improvement of an operator Mut concerning individual A can be defined as

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

Relative Expected Improvement (2)

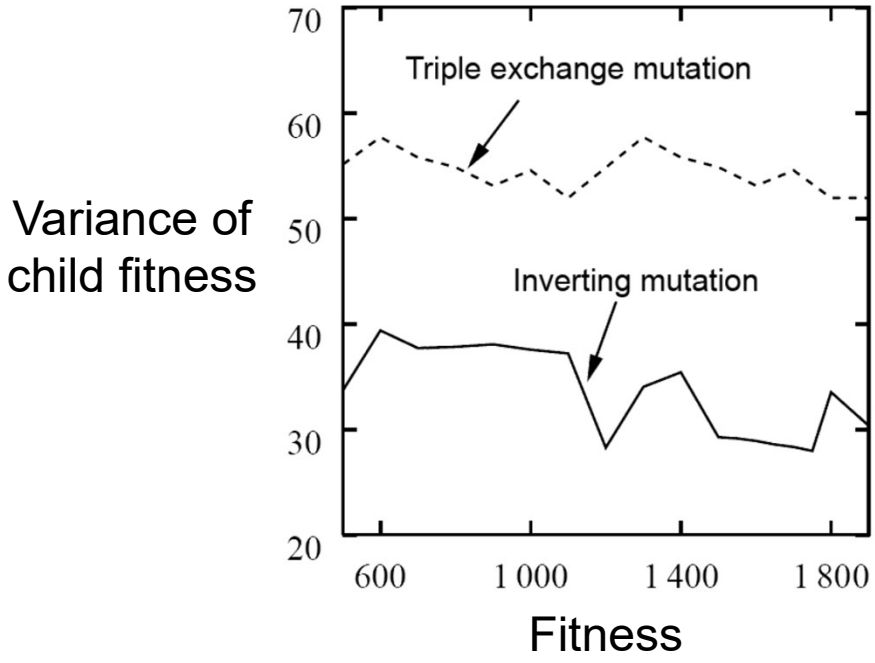
- Determining the relative expected improvement in different fitness ranges by random samples from Ω

  $\Longrightarrow$ How frequent are the different fitness values in Ω?



Relative Expected Improvement (3)

- Locality of the mutation operator is very important
  - Very local ⟹ fitness values in vicinity of the fitness of the parents
  - Less local ⟹ bigger range of fitness values is covered



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

Relative Expected Improvement (4)

- 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!

Predefined Adaptation

- Defines change before
- Considered parameter
  - Real valued gaussian mutation
  - σ determines average step width
  - Modifying parameter $0 < \alpha < 1$ lets decrease σ exponentially
- **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

- Define metrics for appropriateness: fraction of improving mutations of last k generations
- Deduce adaptation from rules: if this 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

1: **if** $p_s > \theta$ {
2: **return** $\alpha \cdot \sigma$
3: }
4: **if** $p_s < \theta$ {
5: **return** σ / α
6: }
7: **return** σ

Self Adaption

- Use additional information in individuals
- Parameter should align individually by a random process
- Implementation:
  - Storing the standard deviation σ on generating the individual as additional information

    $\Longrightarrow$ Using a strategy parameter (will be varied on mutation by random very likely)
  - "Good" values for σ win through better quality of the children

Self-Adaptive Gaussian Mutation

Algorithm 6 Self-adaptive Gaussian Mutation

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

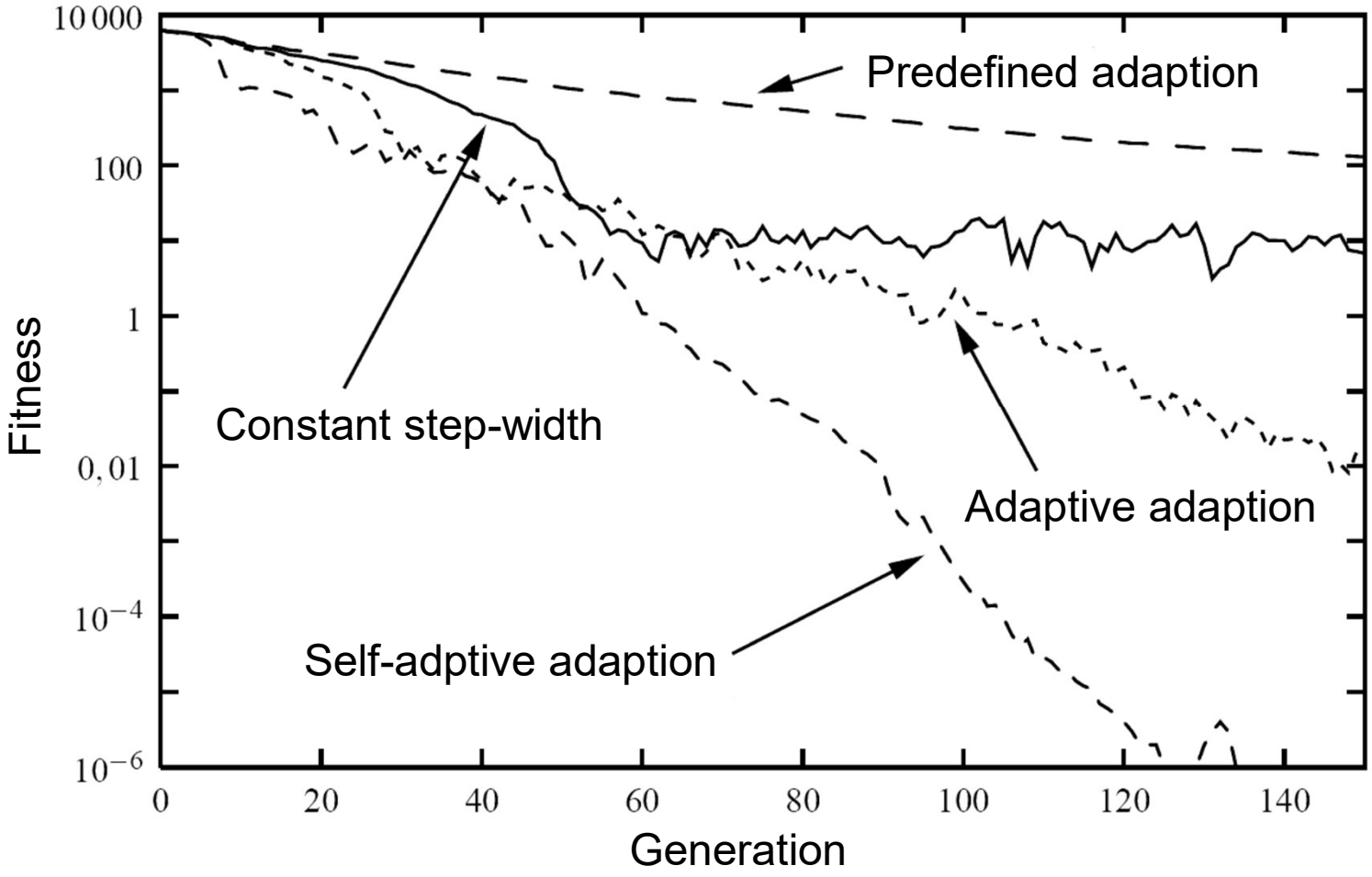
Output: varied individual B with $B.G \in \mathbb{R}^l$

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: $\quad B.G_i \leftarrow \min\{B.G_i,\ og_i\}$ /* upper range bound og_i */
8: $\}$
9: **return** B

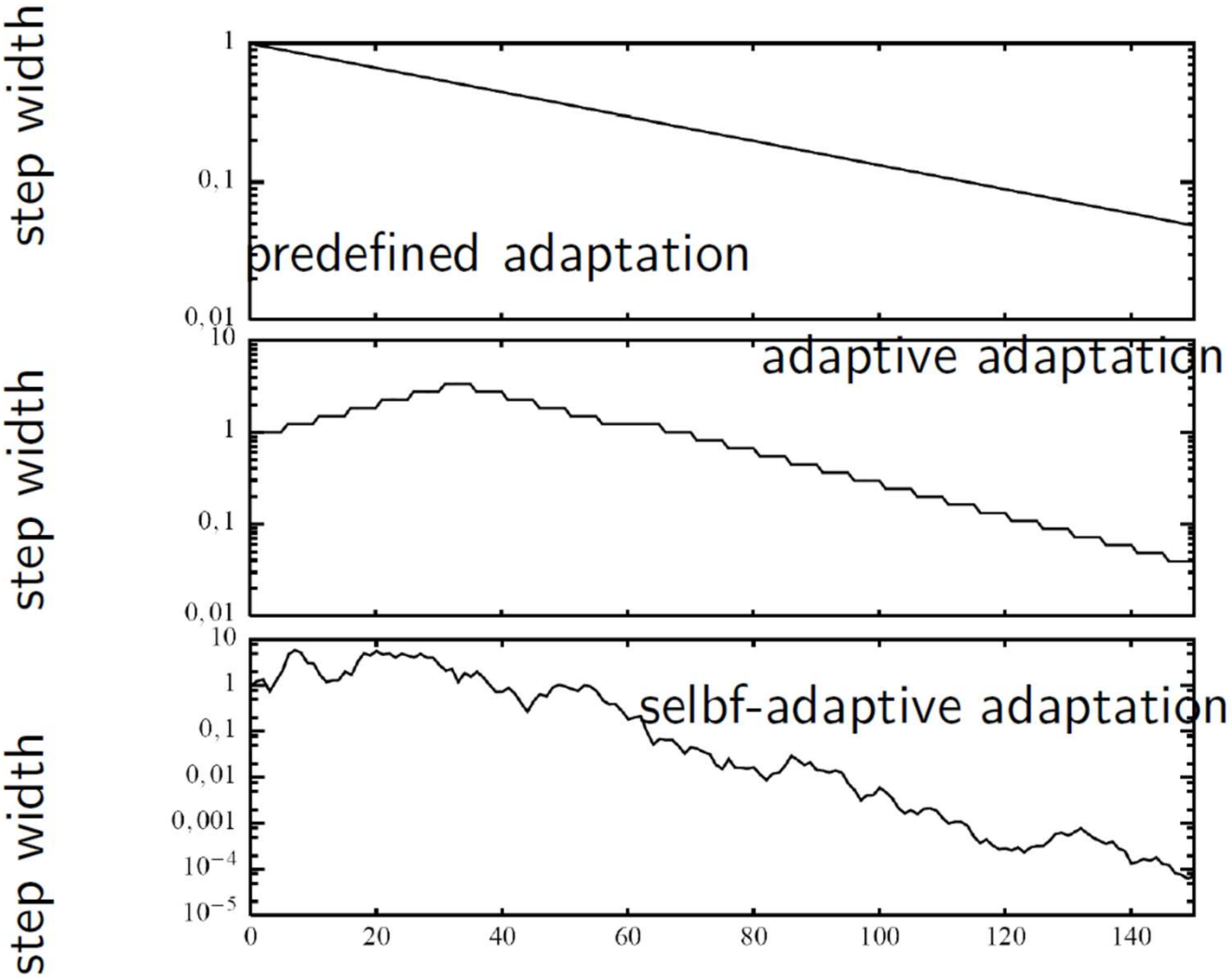
Experimenting Adaptation Strategies (1)

- Experimental 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 $\lambda = \frac{1}{5}$ and $\sigma = 1.224$

Experimenting Adaptation Strategies (2)



Experimenting Adaptation Strategies (3)



Relations and Impacts (1)

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
Diversity/ Recombination	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
Diversity/ Recombination	Exploration	Combining recombination strengthen exploration on high diversity
Diversity/ Recombination	Fine tuning	Combining recombination strengthen fine tuning on high diversity

Relations and Impacts (2)

Condition	Target value	Expected impact
Fine tuning	Diversity	Fine tuning operations decrease diversity
Diversity	Selection	Small diversity decreases selection pressure of the fitness-proportion selection
Local optima	Search progress	Huge amount of local optima inhibits search progress
Exploration/Fine tuning/Selection	Search progress	Counterbalancing of all factors is required