

Computational Intelligence

Winter Term 2020/21

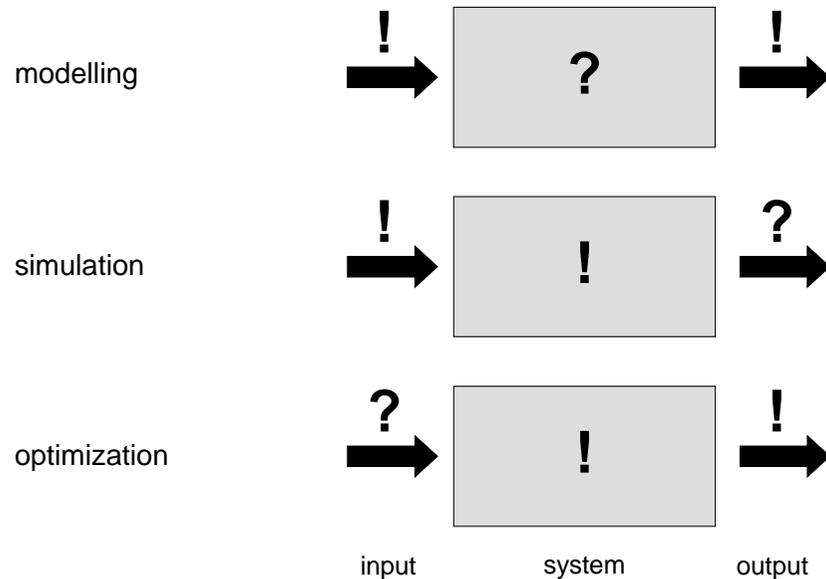
Prof. Dr. Günter Rudolph

Lehrstuhl für Algorithm Engineering (LS 11)

Fakultät für Informatik

TU Dortmund

- Evolutionary Algorithms (EA)
 - Optimization Basics
 - EA Basics



given:

objective function $f: X \rightarrow \mathbb{R}$

feasible region X (= nonempty set)

objective: find solution with *minimal* or *maximal* value!

optimization problem:

find $x^* \in X$ such that $f(x^*) = \min\{ f(x) : x \in X \}$

x^* **global solution**

$f(x^*)$ **global optimum**

note:

$\max\{ f(x) : x \in X \} = - \min\{ -f(x) : x \in X \}$

local solution $x^* \in X$:

$$\forall x \in N(x^*): f(x^*) \leq f(x)$$

neighborhood of x^* =
bounded subset of X

} if x^* local solution then
 $f(x^*)$ **local optimum / minimum**

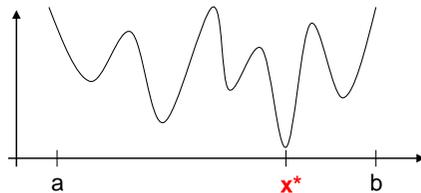
example: $X = \mathbb{R}^n, N_\epsilon(x^*) = \{x \in X: \|x - x^*\|_2 \leq \epsilon\}$

remark:

evidently, every global solution / optimum is also local solution / optimum;
the reverse is wrong in general!

example:

$f: [a,b] \rightarrow \mathbb{R}$, global solution at x^*



What makes optimization difficult?

some causes:

- local optima (is it a global optimum or not?)
- constraints (ill-shaped feasible region)
- non-smoothness (weak causality) → strong causality needed!
- discontinuities (⇒ nondifferentiability, no gradients)
- lack of knowledge about problem (⇒ black / gray box optimization)

$f(x) = a_1 x_1 + \dots + a_n x_n \rightarrow \max!$ with $x_i \in \{0,1\}, a_i \in \mathbb{R}$ ⇒ $x_i^* = 1$ iff $a_i > 0$
add constant $g(x) = b_1 x_1 + \dots + b_n x_n \leq b$ ⇒ NP-hard

add capacity constraint to TSP ⇒ CVRP ⇒ still harder

When using which optimization method?

mathematical algorithms

- problem explicitly specified
- problem-specific solver available
- problem well understood
- resources for designing algorithm affordable
- solution with proven quality required

⇒ **don't** apply EAs

randomized search heuristics

- problem given by black / gray box
- no problem-specific solver available
- problem poorly understood
- insufficient resources for designing algorithm
- solution with satisfactory quality sufficient

⇒ **EAs worth a try**

idea: using **biological evolution** as **metaphor** and as **pool of inspiration**

⇒ interpretation of biological evolution as iterative method of improvement

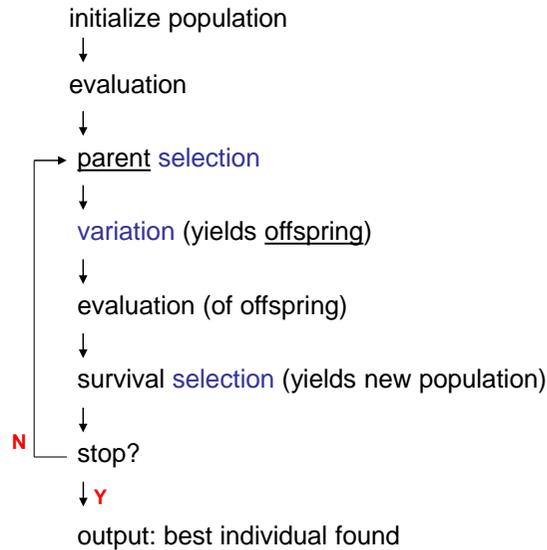
feasible solution $x \in X = S_1 \times \dots \times S_n$ = chromosome of **individual**
multiset of feasible solutions = **population**: multiset of individuals
objective function $f: X \rightarrow \mathbb{R}$ = **fitness function**

often: $X = \mathbb{R}^n, X = \mathbb{B}^n = \{0,1\}^n, X = \mathbb{P}_n = \{\pi : \pi \text{ is permutation of } \{1,2,\dots,n\}\}$

also : combinations like $X = \mathbb{R}^n \times \mathbb{B}^p \times \mathbb{P}_q$ or non-cartesian sets

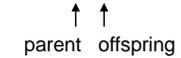
⇒ structure of feasible region / search space defines **representation** of individual

algorithmic skeleton



Specific example: (1+1)-EA in \mathbb{B}^n for minimizing some $f: \mathbb{B}^n \rightarrow \mathbb{R}$

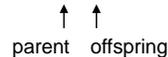
population size = 1, number of offspring = 1, selects best from 1+1 individuals



1. initialize $X^{(0)} \in \mathbb{B}^n$ uniformly at random, set $t = 0$
2. evaluate $f(X^{(t)})$
3. select parent: $Y = X^{(t)}$ no choice, here
4. variation: flip each bit of Y independently with probability $p_m = 1/n$
5. evaluate $f(Y)$
6. selection: if $f(Y) \leq f(X^{(t)})$ then $X^{(t+1)} = Y$ else $X^{(t+1)} = X^{(t)}$
7. if not stopping then $t = t+1$, continue at (3)

Specific example: (1+1)-EA in \mathbb{R}^n for minimizing some $f: \mathbb{R}^n \rightarrow \mathbb{R}$

population size = 1, number of offspring = 1, selects best from 1+1 individuals



compact set = closed & bounded

1. initialize $X^{(0)} \in C \subset \mathbb{R}^n$ uniformly at random, set $t = 0$
2. evaluate $f(X^{(t)})$
3. select parent: $Y = X^{(t)}$ no choice, here
4. variation = add random vector: $Y = Y + Z$, e.g. $Z \sim N(0, I_n)$
5. evaluate $f(Y)$
6. selection: if $f(Y) \leq f(X^{(t)})$ then $X^{(t+1)} = Y$ else $X^{(t+1)} = X^{(t)}$
7. if not stopping then $t = t+1$, continue at (3)

Selection

- (a) select parents that generate offspring → selection for **reproduction**
- (b) select individuals that proceed to next generation → selection for **survival**

necessary requirements:

- selection steps must not favor worse individuals
- one selection step may be neutral (e.g. select uniformly at random)
- at least one selection step must favor better individuals

typically : selection only based on fitness values $f(x)$ of individuals

seldom : additionally based on individuals' chromosomes x (→ maintain diversity)

Selection methods

population $P = (x_1, x_2, \dots, x_\mu)$ with μ individuals

two approaches:

1. repeatedly select individuals from population with replacement
2. rank individuals somehow and choose those with best ranks (no replacement)

- **uniform / neutral selection**

choose index i with probability $1/\mu$

- **fitness-proportional selection**

choose index i with probability $s_i = \frac{f(x_i)}{\sum_{x \in P} f(x)}$

problems: $f(x) > 0$ for all $x \in X$ required $\Rightarrow g(x) = \exp(f(x)) > 0$

but already sensitive to additive shifts $g(x) = f(x) + c$

almost deterministic if large differences, almost uniform if small differences

don't use!

Selection methods

population $P = (x_1, x_2, \dots, x_\mu)$ with μ individuals

- **rank-proportional selection**

order individuals according to their fitness values
assign ranks
fitness-proportional selection based on ranks

\Rightarrow avoids all problems of fitness-proportional selection
but: best individual has only small selection advantage (can be lost!)

outdated!

- **k-ary tournament selection**

draw k individuals uniformly at random (typically with replacement) from P
choose individual with best fitness (break ties at random)

\Rightarrow has all advantages of rank-based selection and
probability that best individual does not survive: $\left(1 - \frac{1}{\mu}\right)^{k\mu} < e^{-k}$
 $\geq 4^{-k}$

Selection methods without replacement

population $P = (x_1, x_2, \dots, x_\mu)$ with μ parents and

population $Q = (y_1, y_2, \dots, y_\lambda)$ with λ offspring

- **(μ, λ) -selection** or **truncation selection on offspring** or **comma-selection**

rank λ offspring according to their fitness
select μ offspring with best ranks

\Rightarrow best individual may get lost, $\lambda \geq \mu$ required

- **$(\mu+\lambda)$ -selection** or **truncation selection on parents + offspring** or **plus-selection**

merge λ offspring and μ parents
rank them according to their fitness
select μ individuals with best ranks

\Rightarrow best individual survives for sure

Selection methods: Elitism

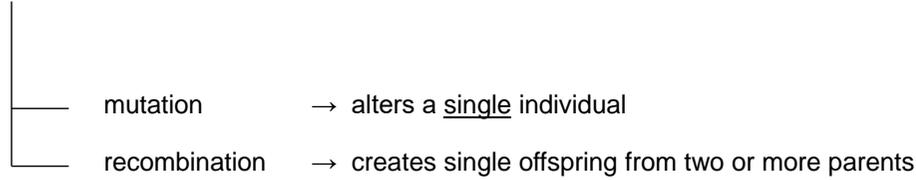
Elitist selection: best parent is not replaced by worse individual.

- *Intrinsic elitism:* method selects from parent and offspring,
best survives with probability 1

- *Forced elitism:* if best individual has not survived then re-injection into population,
i.e., replace worst selected individual by previously best parent

method	P{ select best }	from parents & offspring	intrinsic elitism
neutral	< 1	no	no
fitness proportionate	< 1	no	no
rank proportionate	< 1	no	no
k-ary tournament	< 1	no	no
$(\mu + \lambda)$	= 1	yes	yes
(μ, λ)	= 1	no	no

Variation operators: depend on representation



may be applied

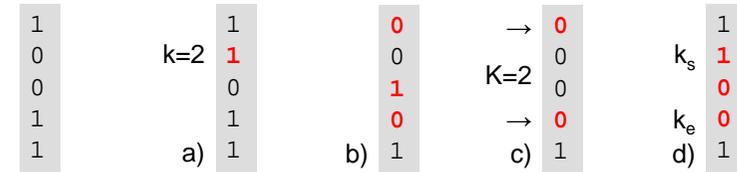
- exclusively (either recombination or mutation) chosen in advance
- exclusively (either recombination or mutation) in probabilistic manner
- sequentially (typically, recombination before mutation); for each offspring
- sequentially (typically, recombination before mutation) with some probability

Variation in \mathbb{B}^n

Individuals $\in \{0, 1\}^n$

• Mutation

- a) local → choose index $k \in \{1, \dots, n\}$ uniformly at random, flip bit k , i.e., $x_k = 1 - x_k$
- b) global → for each index $k \in \{1, \dots, n\}$: flip bit k with probability $p_m \in (0, 1)$
- c) “nonlocal” → choose K indices at random and flip bits with these indices
- d) inversion → choose start index k_s and end index k_e at random invert order of bits between start and end index

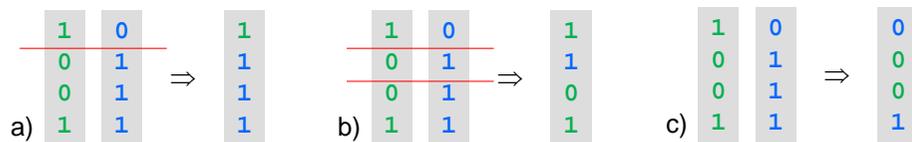


Variation in \mathbb{B}^n

Individuals $\in \{0, 1\}^n$

• Recombination (two parents)

- a) 1-point crossover → draw cut-point $k \in \{1, \dots, n-1\}$ uniformly at random; choose first k bits from 1st parent, choose last $n-k$ bits from 2nd parent
- b) K -point crossover → draw K distinct cut-points uniformly at random; choose bits 1 to k_1 from 1st parent, choose bits k_1+1 to k_2 from 2nd parent, choose bits k_2+1 to k_3 from 1st parent, and so forth ...
- c) uniform crossover → for each index i : choose bit i with equal probability from 1st or 2nd parent



Variation in \mathbb{B}^n

Individuals $\in \{0, 1\}^n$

• Recombination (multiparent: $\rho = \#$ parents)

- a) diagonal crossover ($2 < \rho < n$) → choose $\rho - 1$ distinct cut points, select chunks from diagonals
 AAAA|AA|AAAA ABBB|CC|DDDD
 BBBB|BB|BBBB BCC|DD|AAAA
 CCCC|CC|CCCC CDD|D|A|BBBB
 DDDD|DD|DDDD DAA|B|C|CCC
 can generate ρ offspring; otherwise choose initial chunk at random for single offspring
- b) gene pool crossover ($\rho > 2$) → for each gene: choose donating parent uniformly at random

Variation in \mathbb{P}_n

Individuals $\in X = \pi(1, \dots, n)$

• Mutation

a) local → 2-swap / 1-translocation



b) global → draw number K of 2-swaps, apply 2-swaps K times

K is positive random variable;
its distribution may be uniform, binomial, geometrical, ...;
E[K] and V[K] may control mutation strength



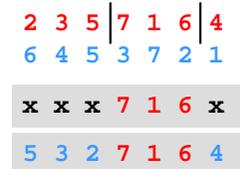
Variation in \mathbb{P}_n

Individuals $\in X = \pi(1, \dots, n)$

• Recombination (two parents)

a) order-based crossover (OBX)

- select two indices k_1 and k_2 with $k_1 \leq k_2$ uniformly at random
- copy genes k_1 to k_2 from 1st parent to offspring (keep positions)
- copy genes from left (pos. 1) to right (pos. n) of 2nd parent, insert after pos. k_2 in offspring (skip values already contained)



b) partially mapped crossover (PMX) [a version of]

- select two indices k_1 and k_2 with $k_1 \leq k_2$ uniformly at random
- copy genes k_1 to k_2 from 1st parent to offspring (keep positions)
- copy all genes not already contained in offspring from 2nd parent (keep positions)
- from left to right: fill in remaining genes from 2nd parent



Variation in \mathbb{P}_n

Individuals $\in X = \pi(1, \dots, n)$

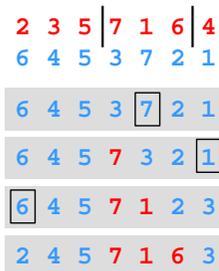
• Recombination (two parents)

c) partially mapped crossover (PMX) [Grefenstette et al. 1985]

→ consider array as ring!

- given: 2 permutations **a** and **b** of length n
- select 2 indices k_1 and k_2 uniformly at random
- copy **b** to **c**
- procedure =

```
i = k1
repeat
  j = findIndex(a[i], c)
  swap(c[i], c[j])
  i = (i + 1) mod n
until i == k2
```



Variation in \mathbb{R}^n

Individuals $X \in \mathbb{R}^n$

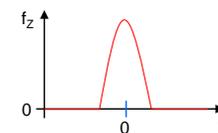
• Mutation

additive:

$$Y = X + Z \quad (Z: n\text{-dimensional random vector})$$

↑ ↑ ↑
parent mutation
offspring = parent + mutation

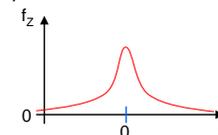
a) local → Z with bounded support



$$f_Z(x) = \frac{4}{3} (1 - x^2) \cdot 1_{[-1,1]}(x)$$

Definition
Let $f_Z: \mathbb{R}^n \rightarrow \mathbb{R}^+$ be p.d.f. of r.v. Z.
The set $\{x \in \mathbb{R}^n : f_Z(x) > 0\}$ is termed the support of Z.

b) nonlocal → Z with unbounded support



$$f_Z(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

most frequently used!

Variation in \mathbb{R}^n

Individuals $X \in \mathbb{R}^n$

- Recombination (two parents)

a) all crossover variants adapted from \mathbb{B}^n

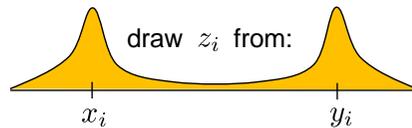
b) intermediate $z = \xi \cdot x + (1 - \xi) \cdot y$ with $\xi \in [0, 1]$

c) intermediate (per dimension) $\forall i : z_i = \xi_i \cdot x_i + (1 - \xi_i) \cdot y_i$ with $\xi_i \in [0, 1]$

d) discrete $\forall i : z_i = B_i \cdot x_i + (1 - B_i) \cdot y_i$ with $B_i \sim B(1, \frac{1}{2})$

e) simulated binary crossover (SBX)

→ for each dimension with probability p_c



Variation in \mathbb{R}^n

Individuals $X \in \mathbb{R}^n$

- Recombination (multiparent), $\rho \geq 3$ parents

a) intermediate $z = \sum_{k=1}^{\rho} \xi^{(k)} x_i^{(k)}$ where $\sum_{k=1}^{\rho} \xi^{(k)} = 1$ and $\xi^{(k)} \geq 0$

(all points in convex hull)

b) intermediate (per dimension) $\forall i : z_i = \sum_{k=1}^{\rho} \xi_i^{(k)} x_i^{(k)}$

$$\forall i : z_i \in \left[\min_k \{x_i^{(k)}\}, \max_k \{x_i^{(k)}\} \right]$$

Theorem

Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be a strictly quasiconvex function. If $f(x) = f(y)$ for some $x \neq y$ then every offspring generated by intermediate recombination is better than its parents.

Proof:

f strictly quasiconvex $\Rightarrow f(\xi \cdot x + (1 - \xi) \cdot y) < \max\{f(x), f(y)\}$ for $0 < \xi < 1$

since $f(x) = f(y) \Rightarrow \max\{f(x), f(y)\} = \min\{f(x), f(y)\}$

$\Rightarrow f(\xi \cdot x + (1 - \xi) \cdot y) < \min\{f(x), f(y)\}$ for $0 < \xi < 1$ ■

Theorem

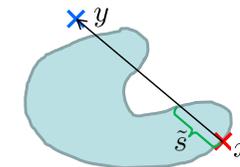
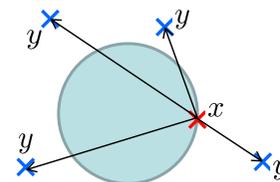
Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function and $f(x) < f(y)$ for some $x \neq y$. If $(y - x)' \nabla f(x) < 0$ then there is a positive probability that an offspring generated by intermediate recombination is better than both parents.

Proof:

If $d' \nabla f(x) < 0$ then $d \in \mathbb{R}^n$ is a direction of descent, i.e.

$\exists \tilde{s} > 0 : \forall s \in (0, \tilde{s}] : f(x + s \cdot d) < f(x)$.

Here: $d = y - x$ such that $P\{f(\xi x + (1 - \xi) y) < f(x)\} \geq \frac{\tilde{s}}{\|d\|} > 0$. ■



sublevel set $S_\alpha = \{x \in \mathbb{R}^n : f(x) < \alpha\}$