Computational Intelligence

Winter Term 2011/12

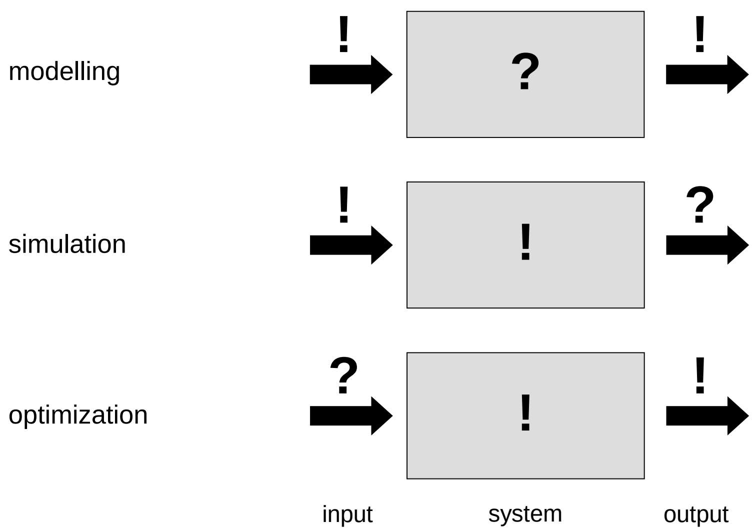
Prof. Dr. Günter Rudolph

Lehrstuhl für Algorithm Engineering (LS 11)

Fakultät für Informatik

TU Dortmund

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



given:

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

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**

<u>note:</u>

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

local solution $x^* \in X$:

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

if x^* local solution then

$f(x^*)$ **local optimum / minimum**

neighborhood of x^* =
bounded subset of X

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

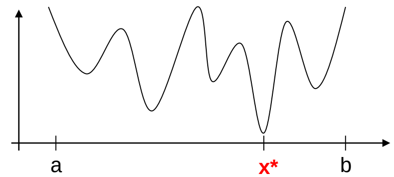
remark:

evidently, every global solution / optimum is also local solution / optimum;

the reverse is wrong in general!

example:

f: [a,b] $\to \mathbb{R}$, 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 ($\Rightarrow$ nondifferentiability, no gradients)
- lack of knowledge about problem ($\Rightarrow$ black / gray box optimization)

$f(x) = a_1 x_1 + ... + a_n x_n \to \max!$ with $x_i \in \{0,1\}$, $a_i \in \mathbb{R}$ $\Rightarrow x_i^* = 1$ if $a_i > 0$

add constaint $g(x) = b_1 x_1 + ... + b_n x_n \leq b$ $\Rightarrow$ NP-hard

add capacity constraint to TSP $\Rightarrow$ CVRP $\Rightarrow$ still harder

When using which optimization method?

mathematical algorithms

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

$\Rightarrow$ **don't apply EAs**

randomized search heuristics

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

$\Rightarrow$ **EAs worth a try**

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

$\Rightarrow$ interpretation of biological evolution as iterative method of improvement

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

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

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

$\Rightarrow$ structure of feasible region / search space defines **representation** of individual algorithmic skeleton

initialize population
↓
evaluation
↓
parent selection
↓
variation (yields offspring)
↓
evaluation (of offspring)
↓
survival selection (yields new population)
↓
stop? N → parent selection
↓ Y
output: best individual found

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

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

↑ parent ↑ offspring

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 $f(Y) \le f(X^{(t)})$ 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: $\mathbb{R}^n \to \mathbb{R}$

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

↑ parent ↑ offspring

compact set = closed & bounded

1. initialize $X^{(0)} \in \mathbb{C} \subset \mathbb{R}^n$ 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 $f(Y) \le f(X^{(t)})$ 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, ..., x_μ) 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 $s_i = \dfrac{f(x_i)}{\sum_{x \in P} f(x)}$

  don't use!

  problems: f(x) > 0 for all x ∈ X required ⇒ 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

Selection methods

population P = (x_1, x_2, ..., x_μ) with μ individuals

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

  outdated!

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

- ***k-ary tournament selection***
  draw k individuals uniformly at random (typically with replacement) from P
  choose individual with best fitness (break ties at random)

  ⇒ has all advantages of rank-based selection and
  probability that best individual does not survive: $\left(1 - \dfrac{1}{\mu}\right)^{k\,\mu} \approx e^{-k}$

Selection methods without replacement

population P = (x_1, x_2, ..., x_μ) with μ parents and

population Q = (y_1, y_2, ..., y_λ) 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

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

- **(μ+λ)-*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

  ⇒ 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
(μ + λ)	= 1	yes	yes
(μ , λ)	= 1	no	no

Variation operators: depend on representation

- mutation → alters a single individual
- recombination → creates single offspring from two or more parents

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 $k \in \{ 1, \ldots, n \}$ uniformly at random, flip bit k, i.e., $x_k = 1 - x_k$

b) global → for each index $k \in \{ 1, \ldots, n \}$: 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 and index

	a) k=2	b)	c) K=2	d) k_s, k_e
1	1	**0**	**0**	1
0	**1**	0	0	**1**
0	0	**1**	0	**0**
1	1	**0**	**0**	**0**
1	1	1	1	1

Variation in $\mathbb{B}^n$ — Individuals $\in \{ 0, 1 \}^n$

- Recombination (two parents)

a) 1-point crossover → draw cut-point $k \in \{1,\ldots,n-1\}$ 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

	1st parent	2nd parent	offspring
a)	1001	0111	⇒ 1111
b)	1001	0111	⇒ 1101
c)	1001	0111	⇒ 0001

Variation in $\mathbb{B}^n$ — Individuals $\in \{ 0, 1 \}^n$

- Recombination (multiparent: ρ = #parents)

a) diagonal crossover ($2 < \rho < n$)

→ choose $\rho - 1$ distinct cut points, select chunks from diagonals

Parents	Offspring
AAAAAAAAAA	ABBBCCDDDD
BBBBBBBBBB	BCCCDDAAAA
CCCCCCCCCC	CDDDAABBBB
DDDDDDDDDD	DAAABBCCCC

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 $X = \pi(1, \ldots, n)$

- Mutation

a) local → 2-swap / 1-translocation

5 3 2 4 1 → 5 4 2 3 1

5 3 2 4 1 → 5 2 4 3 1

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

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

expectation variance

Variation in $\mathbb{P}_n$

Individuals $X = \pi(1, \ldots, n)$

- Recombination (two parents)

a) order-based crossover (OB)

b) partially mapped crossover (PMX)

c) cycle crossover (CX)

Variation in $\mathbb{P}_n$

Individuals $X = \pi(1, \ldots, n)$

- Recombination (multiparent)

a) xx crossover

b) xx crossover

c) xx crossover

Variation in $\mathbb{R}^n$

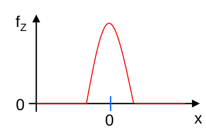
Individuals $X \in \mathbb{R}^n$

- Mutation

<u>additive:</u> Y = X + Z (Z: n-dimensional random vector)

offspring = parent + mutation

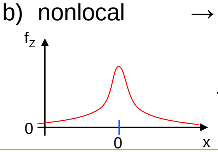
a) local → Z with bounded support



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

Definition
Let $f_Z: \mathbb{R}^n \to \mathbb{R}^+$ 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)

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: $\mathbb{R}^n \to \mathbb{R}$ be a strictly quasiconvex function. If f(x) = f(y) for some x ≠ 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 $0 < \xi < 1$

■

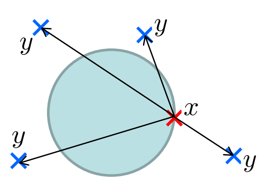
Theorem
Let f: $\mathbb{R}^n \to \mathbb{R}$ be a differentiable function and f(x) < f(y) for some x ≠ y.
If (y – x)‘ ∇f(x) < 0 then there is a positive probability that an offspring generated by intermediate recombination is better than both parents.

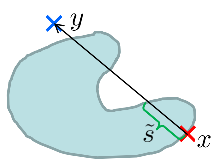
Proof:

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 $P\{ f(\xi\, x + (1-\xi)\, y) < f(x) \} \geq \tilde{s} > 0$. ■





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

Idea emerged independently several times: about late 1950s / early 1960s.

Three branches / “schools“ still active today.

- **Evolutionary Programming (EP)**:
  Pioneers: Lawrence Fogel, Alvin Owen, Michael Walsh (New York, USA).

  Original goal: Generate intelligent behavior through simulated evolution.
  Approach: Evolution of finite state machines predicting symbols.
  Later (~1990s) specialized to optimization in $\mathbb{R}^n$ by David B. Fogel.

- **Genetic Algorithms (GA)**:
  Pioneer: John Holland (Ann Arbor, MI, USA).

  Original goal: Analysis of adaptive behavior.
  Approach: Viewing evolution as adaptation. Simulated evolution of bit strings.
  Applied to optimization tasks by PhD students (Kenneth de Jong, 1975; et al.).

- **Evolution Strategies (ES)**:
  Pioneers: Ingo Rechenberg, Hans-Paul Schwefel, Peter Bienert (Berlin, Germany).

  Original goal: Optimization of complex systems.
  Approach: Viewing variation/selection as improvement strategy. First in $\mathbb{Z}^n$, then $\mathbb{R}^n$.

“Offspring“ from GA branch:

- **Genetic Programming (GP)**:
  Pioneers: Nichael Lynn Cramer 1985, then: John Koza (Stanford, USA).

  Original goal: Evolve programs (parse trees) that must accomplish certain task.
  Approach: GA mechanism transfered to parse trees.
  Later: Programs as successive statements → Linear GP (e.g. Wolfgang Banzhaf)

Already beginning early 1990s:

Borders between EP, GA, ES, GP begin to blurr ...

⇒ common term **Evolutionary Algorithm** embracing all kind of approaches

⇒ broadly accepted name for the field: **Evolutionary Computation**

scientific journals: *Evolutionary Computation* (MIT Press) since 1993,
IEEE Transactions on Evolutionary Computation since 1997,
several more specialized journals started since then.