technische universität
dortmund

Computational Intelligence

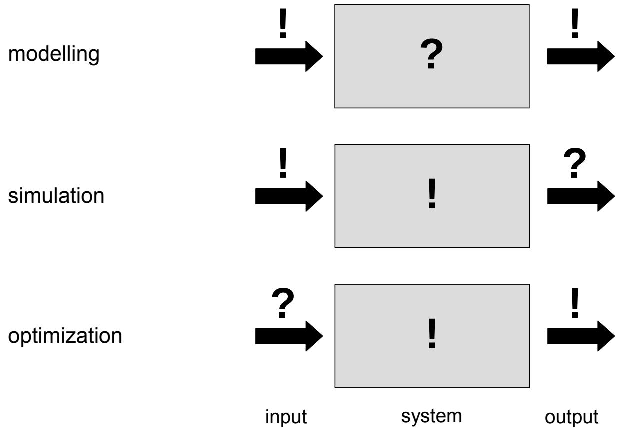
Winter Term 2009/10

Prof. Dr. Günter Rudolph
Lehrstuhl für Algorithm Engineering (LS 11)
Fakultät für Informatik
TU Dortmund

Plan for Today

- Evolutionary Algorithms
  - Optimization Basics
  - EA Basics

Optimization Basics



modelling ! ? !

simulation ! ! ?

optimization ? ! !

input system output

Optimization 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>

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

Optimization Basics

local solution $x^* \in X$:

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

$\downarrow$

neighborhood of x* = bounded subset of X

if x* local solution then

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

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

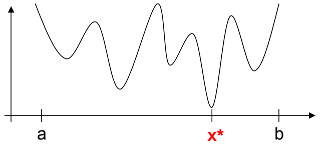
remark:

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

the reverse is wrong in general!

example:

f: [a,b] → ℝ, global solution at **x***



Optimization Basics

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)

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

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

add capacity constraint to TSP ⇒ CVRP ⇒ still harder

Optimization Basics

When using which optimization method?

mathematical algorithms	randomized search heuristics
• problem explicitly specified	• problem given by black / gray box
• problem-specific solver available	• no problem-specific solver available
• problem well understood	• problem poorly understood
• ressources for designing algorithm affordable	• insufficient ressources for designing algorithm
• solution with proven quality required	• solution with satisfactory quality sufficient
⇒ **don't apply EAs**	⇒ **EAs worth a try**

Evolutionary Algorithm Basics

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

⇒ interpretation of biological evolution as iterative method of improvement

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

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

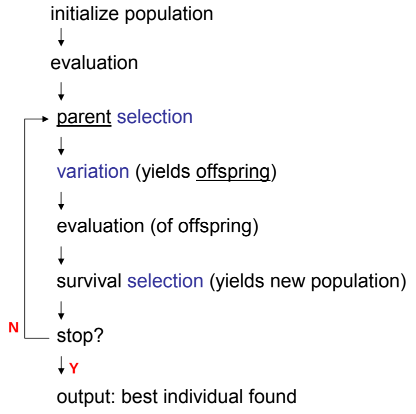
also : combinations like $X = \mathbb{R}^n \times \mathbb{B}^p \times \mathbb{P}_q$ or non-cartesian sets

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

Evolutionary Algorithm Basics

Lecture 10

algorithmic skeleton



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

Evolutionary Algorithm Basics

Lecture 10

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)

Evolutionary Algorithm Basics

Lecture 10

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)

Evolutionary Algorithm Basics

Lecture 10

Selection methods

population $P = (x_1, x_2, \ldots, x_\mu)$ 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 $s_i = \dfrac{f(x_i)}{\sum\limits_{x \in P} f(x)}$

  problems: f(x) > 0 for all $x \in X$ 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

don't use!

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

  ⇒ 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)

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

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