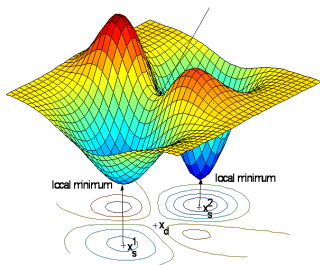


Metaheuristic algorithms – summary

Maciej Komosinski

Institute of Computing Science
Poznan University of Technology
www.cs.put.poznan.pl/mkomosinski

Search (state) space



$$\min_{\mathbf{x} \in M} f(\mathbf{x})$$

Local minimum



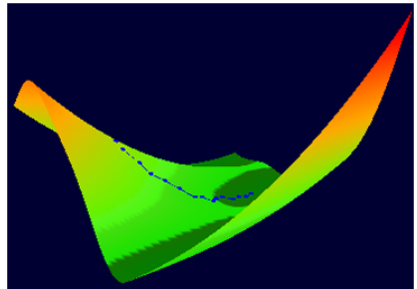
attractor

Hillclimbing methods

- Use an iterative improvement technique
- Start from a single point (current point) in the search space
- At each iteration a new point is selected from the *neighborhood* of the current point
- If new point is better, it becomes current point, otherwise another neighbor is selected and tested
- terminates when there is no more improvement in the neighborhood

Hillclimbing methods

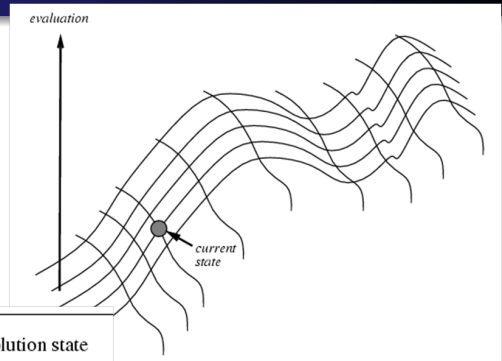
- Use an iterative improvement technique
- Start from a single point (current point) in the search space
- At each iteration a new point is selected from the *neighborhood* of the current point
- If new point is better, it becomes current point, otherwise another neighbor is selected and tested
- terminates when there is no more improvement in the neighborhood



Local Search (LS)

- It is called *greedy* when accepts the first better neighbor
- It is called *steepest descent* when accepts the best neighbor in the neighborhood
- The biggest problem of local search is getting stuck in a local optimum
- Pretty good, simple, fast and popular!

Local Search



function HILL-CLIMBING(*problem*) **returns** a solution state

inputs: *problem*, a problem

static: *current*, a node

next, a node

current \leftarrow MAKE-NODE(INITIAL-STATE[*problem*])

loop do

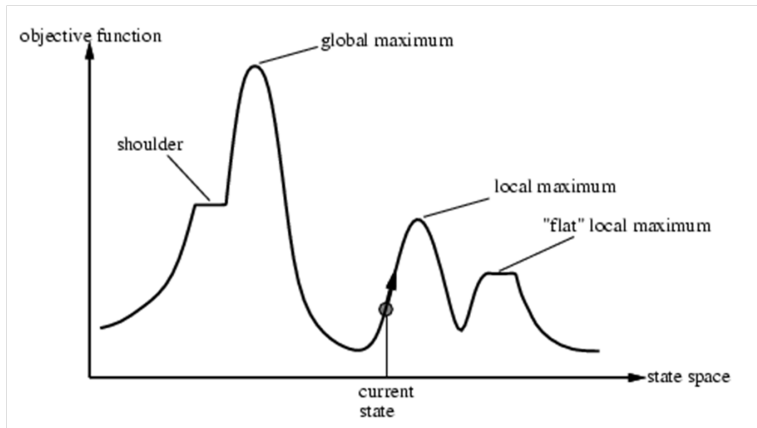
next \leftarrow a highest-valued successor of *current*

if VALUE[*next*] < VALUE[*current*] **then return** *current*

current \leftarrow *next*

end

Problems with Local Search



Simulated Annealing (SA)

Simulated Annealing (SA)

- Behaves like Greedy and always accepts improving moves

Simulated Annealing (SA)

- Behaves like Greedy and always accepts improving moves
- If a neighbor solution j is worse than the current solution i , j will be accepted (or not) depending on the “temperature” parameter c and the amount of deterioration $f(i) - f(j)$

Simulated Annealing (SA)

- Behaves like Greedy and always accepts improving moves
- If a neighbor solution j is worse than the current solution i , j will be accepted (or not) depending on the “temperature” parameter c and the amount of deterioration $f(i) - f(j)$
- Early on, when the “temperature” is high, accepting a non-improving move is more likely than later on

Simulated Annealing (SA)

- Behaves like Greedy and always accepts improving moves
- If a neighbor solution j is worse than the current solution i , j will be accepted (or not) depending on the “temperature” parameter c and the amount of deterioration $f(i) - f(j)$
- Early on, when the “temperature” is high, accepting a non-improving move is more likely than later on
- Poorer solutions are accepted based on probability,
 $\exp\left(\frac{f(i)-f(j)}{c}\right)$

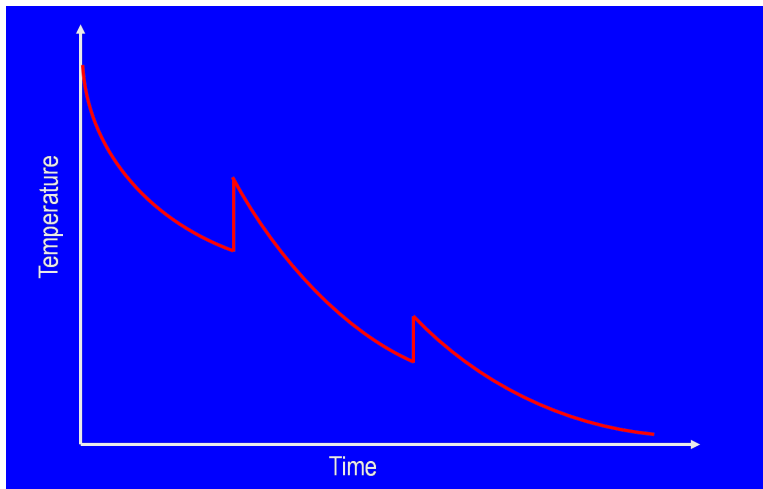
Simulated Annealing (SA)

- Behaves like Greedy and always accepts improving moves
- If a neighbor solution j is worse than the current solution i , j will be accepted (or not) depending on the “temperature” parameter c and the amount of deterioration $f(i) - f(j)$
- Early on, when the “temperature” is high, accepting a non-improving move is more likely than later on
- Poorer solutions are accepted based on probability,
 $\exp\left(\frac{f(i)-f(j)}{c}\right)$
- The system's *temperature* is controlled in accordance with a *cooling schedule*

Simulated Annealing (SA)

- Behaves like Greedy and always accepts improving moves
- If a neighbor solution j is worse than the current solution i , j will be accepted (or not) depending on the “temperature” parameter c and the amount of deterioration $f(i) - f(j)$
- Early on, when the “temperature” is high, accepting a non-improving move is more likely than later on
- Poorer solutions are accepted based on probability,
$$\exp\left(\frac{f(i)-f(j)}{c}\right)$$
- The system's *temperature* is controlled in accordance with a *cooling schedule*
 - A exponential cooling function may be optionally combined with periodic reheats to ensure a good “crystal” (final solution)

The SA “temperature” cooling schedule (with reheating)



SA procedure

- Periodic reheating (optional) increases the ability to break out of local optima

SA procedure

- Periodic reheating (optional) increases the ability to break out of local optima
- Starting temperature – depends on a particular instance; should be proportional to ruggedness of the landscape so that SA is initially not too greedy and not too random

SA procedure

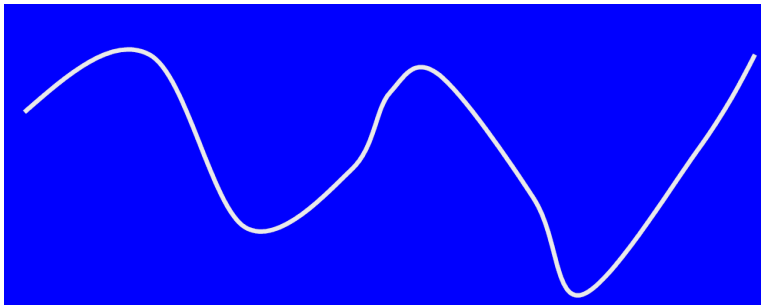
- Periodic reheating (optional) increases the ability to break out of local optima
- Starting temperature – depends on a particular instance; should be proportional to ruggedness of the landscape so that SA is initially not too greedy and not too random
- A stopping condition can be the number of iterations without improvement

Tabu Search (TS)

- Very similar to Steepest, the new aspect is adaptive memory
 - If there is no improvement in the neighborhood (Steepest would end), accept the neighbor that deteriorates the least
 - If search has already visited some solution in the search space, why waste computation time revisiting and re-evaluating that solution?
 - Such a solution (or area in the search space, or moves that lead to this area) is made “taboo” (to try to avoid it in the future)
 - Keep running forever! The only thing that changes in time is the contents of the “tabu list”

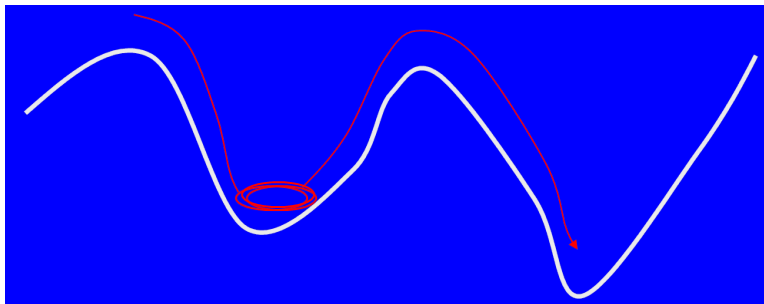
Tabu Search

- The **Moves** define the neighborhood
- The **Tabu List** contains taboo solutions or moves
- A stopping condition can be the number of iterations without improvement



Tabu Search

- The **Moves** define the neighborhood
- The **Tabu List** contains taboo solutions or moves
- A stopping condition can be the number of iterations without improvement



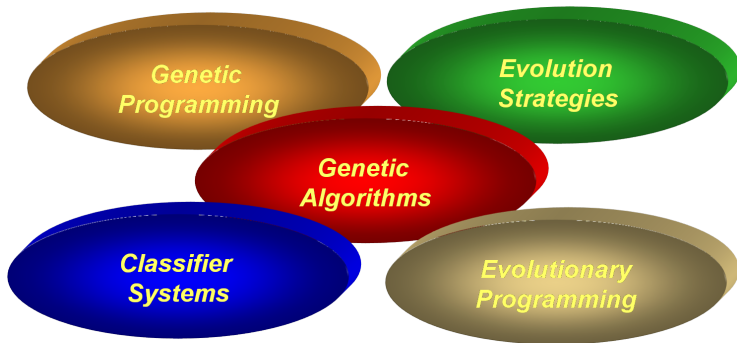
Tabu List: memory techniques that help identify (and prevent) cycling.

But...

LS, Multi-random-start LS, SA, TS are sometimes still not sufficient (local optima!)

Play with <http://en.alife.pl/opt/e/index.html> to see the limitations of LS, SA and TS.

Evolutionary Algorithms



- genetic representation of candidate solutions
- genetic operators
- selection scheme
- EA architecture

The idea behind EA

- Evolutionary Algorithms mimic some of the processes observed in natural evolution. Many people, biologists included, are astonished that life at the level of complexity that we observe could have evolved in the relatively short time suggested by the fossil record. The idea with EA is to use *evolution* to solve hard optimization problems.
- The father of the original Genetic Algorithm was John Holland who invented it in the early 1970's.

Biological terminology

- gene
 - functional entity that codes for a specific feature e.g. eye color
 - set of possible alleles
- allele
 - value of a gene e.g. blue, green, brown
 - codes for a specific variation of the gene/feature
- locus
 - position of a gene on the chromosome
- genome
 - set of all genes that define a species
 - the genome of a specific individual is called genotype
 - the genome of a living organism is composed of several chromosomes
- population/gene pool
 - set of competing individuals/genotypes

Genotype versus Phenotype

- genotype
 - blue print that contains the information to construct an organism e.g. human DNA
 - genetic operators such as mutation and recombination modify the genotype during reproduction
 - genotype of an individual is immutable (no Lamarckian evolution)
- phenotype
 - physical make-up of an organism
 - selection operates on phenotypes (Darwin's principle: "survival of the fittest")
- a phenotype is the meaning/interpretation of a genotype (e.g., permutation vs. route; matrix vs. assignment)

Genetic operators

- recombination (crossover)
 - combines two parent genotypes into a new offspring
 - generates new variants by mixing existing genetic material
 - stochastic selection among parent genes
 - “averaging” behavior (DPX: distance-preserving crossover – children should not be very different from parents)
- mutation
 - random, possibly small alteration of genes (one gene)
 - maintains genetic diversity

Crossover

- crossover applied to parent strings with some probability, e.g. $P_C : [0.6..1.0]$
- various methods: average, linear combination, multi-point. . .
- multi-point

- crossover site(s) chosen randomly
- one-point crossover

parent A 1101|0 offspring A 1101 1

parent B 1000|1 offspring B 1000 0

- two-point crossover

parent A 11|01|0 offspring A 11 00 0

parent B 10|00|1 offspring B 10 01 1

Mutation

- mutation applied to allele/gene with some probability, e.g. 1 divided by the number of genes, or $P_m : [0.001..0.1]$
- the role of mutation is to maintain genetic diversity
- corresponds to *neighbor operator* in local search algorithms (“minimal change”)

Mutation

- mutation applied to allele/gene with some probability, e.g. 1 divided by the number of genes, or $P_m : [0.001..0.1]$
- the role of mutation is to maintain genetic diversity
- corresponds to *neighbor operator* in local search algorithms (“minimal change”)

offspring:

11000

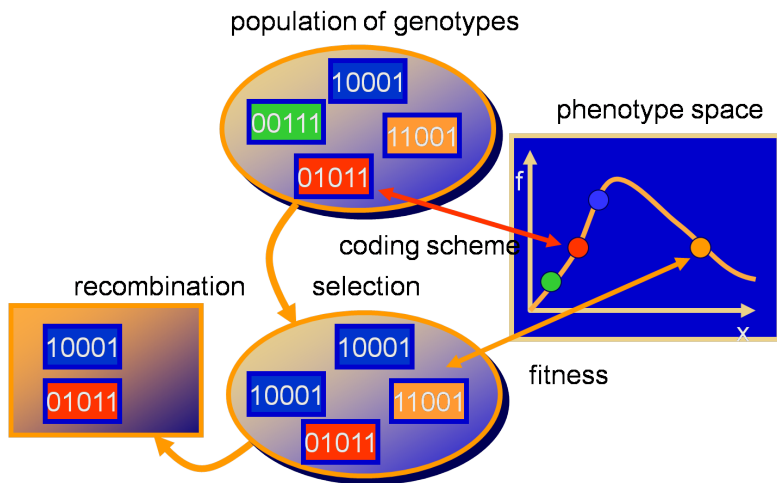


Mutate fourth allele (bit flip)

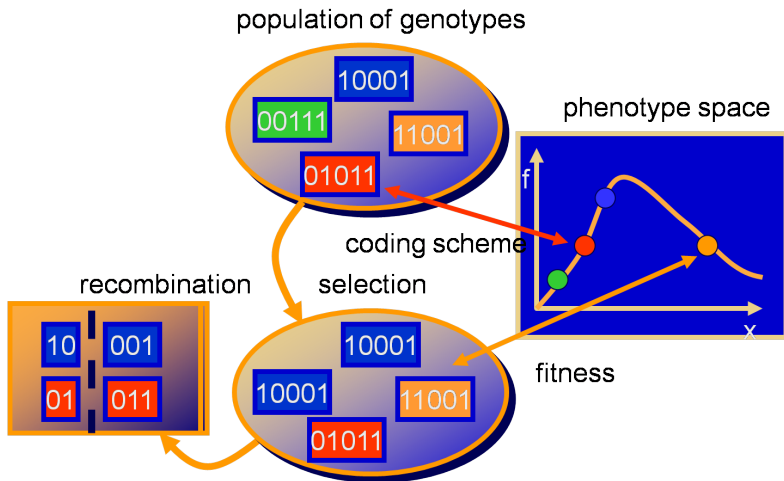
mutated offspring:

110 1 0

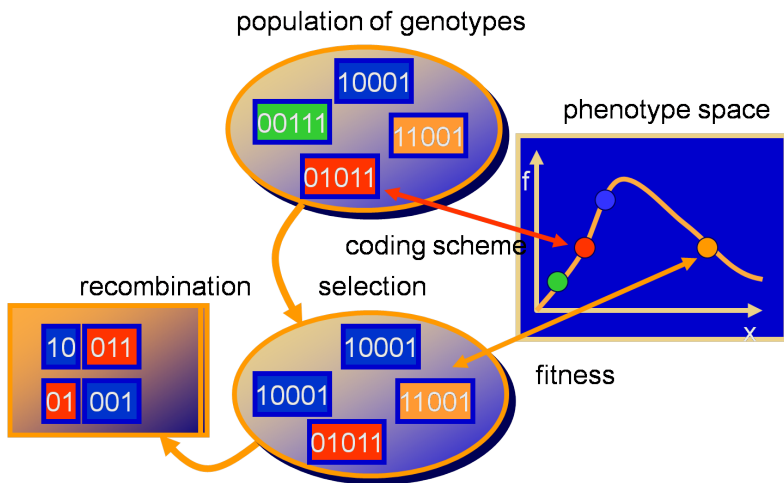
Idea of an Evolutionary Algorithm



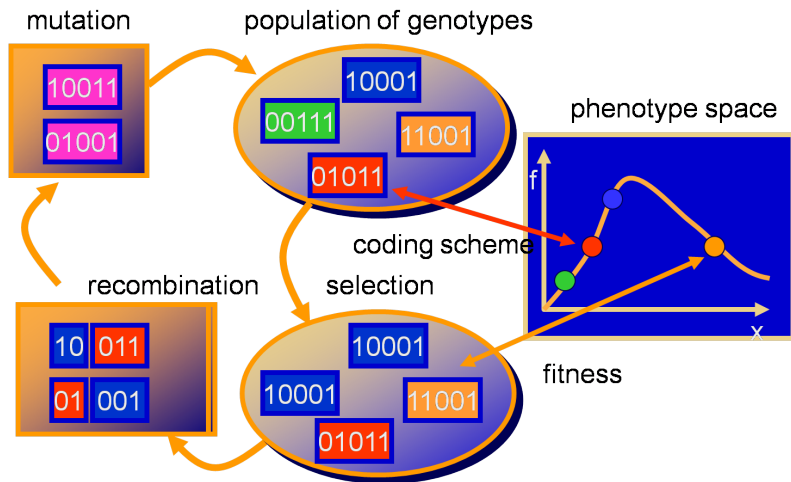
Idea of an Evolutionary Algorithm



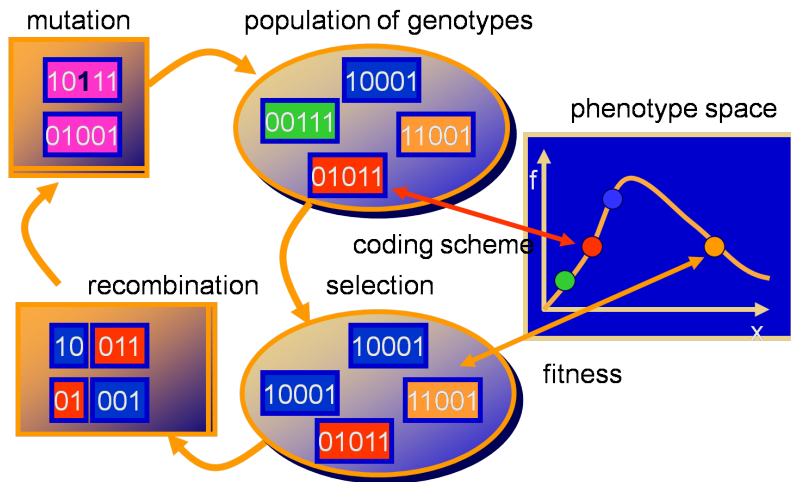
Idea of an Evolutionary Algorithm



Idea of an Evolutionary Algorithm



Idea of an Evolutionary Algorithm



Evolutionary Algorithm architectures

- generational
- incremental (steady-state)

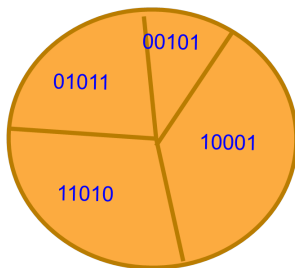
Selection

- positive
- negative (explicit in steady-state architecture)
- it is usually stochastic, but... (discussion about stochasticity and determinism)
- most popular: tournament selection (size k) and roulette wheel selection (probability of reproduction $p_i = f_i / \sum f_j$)
- more complex: island model, convection selection <http://www.cs.put.poznan.pl/mkomosinski/convection.html>

Selection

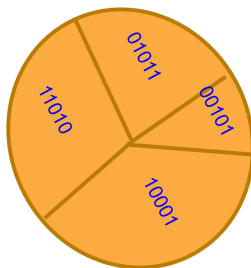
- positive
- negative (explicit in steady-state architecture)
- it is usually stochastic, but... (discussion about stochasticity and determinism)
- most popular: tournament selection (size k) and roulette wheel selection (probability of reproduction $p_i = f_i / \sum f_j$)
- more complex: island model, convection selection <http://www.cs.put.poznan.pl/mkomosinski/convection.html>
- the need for scaling of fitness function values f_i

Roulette wheel selection (example)



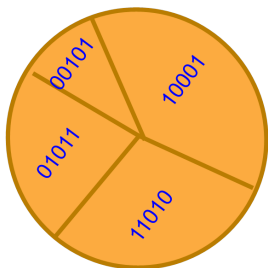
- selected parents: 01011, 11010, 10001, 10001

Roulette wheel selection (example)



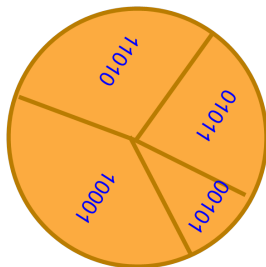
- selected parents: 01011, 11010, 10001, 10001

Roulette wheel selection (example)



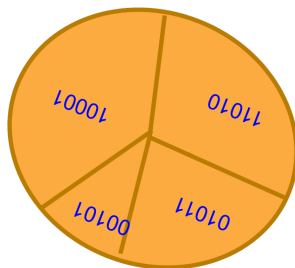
- selected parents: 01011, 11010, 10001, 10001

Roulette wheel selection (example)



- selected parents: 01011, 11010, 10001, 10001

Roulette wheel selection (example)



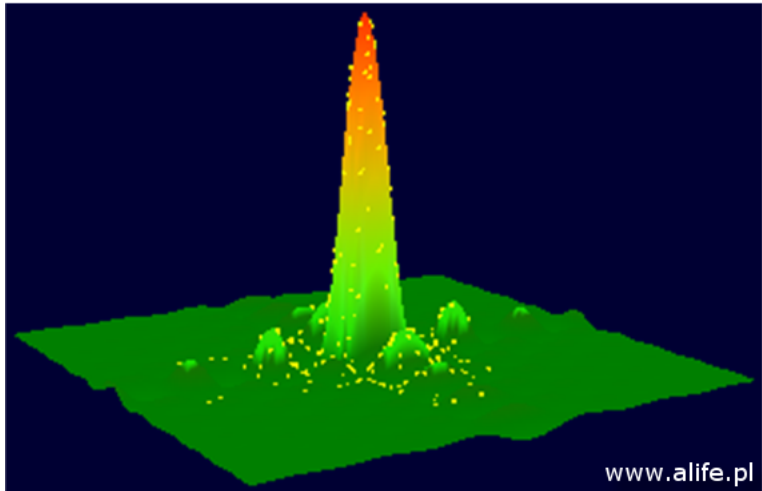
- selected parents: 01011, 11010, 10001, 10001

EA in the search space

- avoid converging to local optima
- balance *exploration* of the search space and *exploitation* of promising areas
- hardly dependent on initial starting points
- starts search from many points in the search space
- problems: time and accuracy

EA in the search space

<http://en.alife.pl/opt/e/index.html>



Genetic Programming

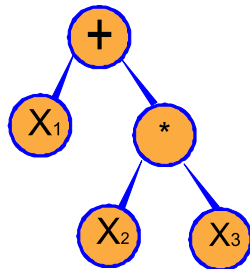
<http://en.alife.pl/function-generator>

- automatic generation of computer programs by means of evolutionary algorithms
- programs can be represented by a parse tree (RPN – Reverse Polish Notation, as in LISP)

Genetic Programming

<http://en.alife.pl/function-generator>

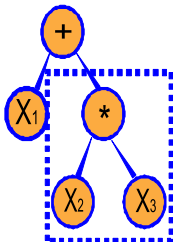
- automatic generation of computer programs by means of evolutionary algorithms
- programs can be represented by a parse tree (RPN – Reverse Polish Notation, as in LISP)
- tree nodes correspond to functions:
 - arithmetic functions $\{+, -, *, /\}$
 - logarithmic functions $\{\sin, \exp\}$
 - algorithmic (if, for, ...)
- leaf nodes correspond to terminals:
 - input variables $\{X_1, X_2, X_3\}$
 - constants $\{0.1, 0.2, 0.5\}$
 - commands (go_left!)



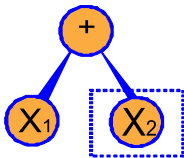
tree is parsed from left to right:
 $(+X_1(*X_2X_3)) \longrightarrow X_1 + (X_2 * X_3)$

Genetic Programming: Crossover

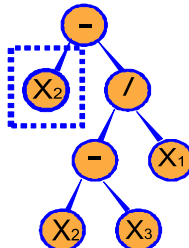
Parent A



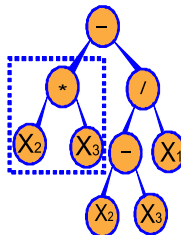
Offspring A



Parent B

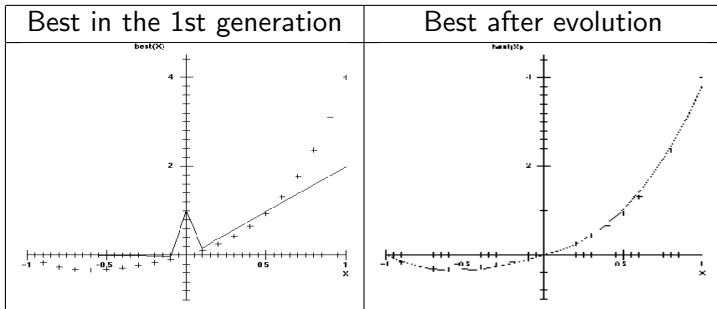


Offspring B



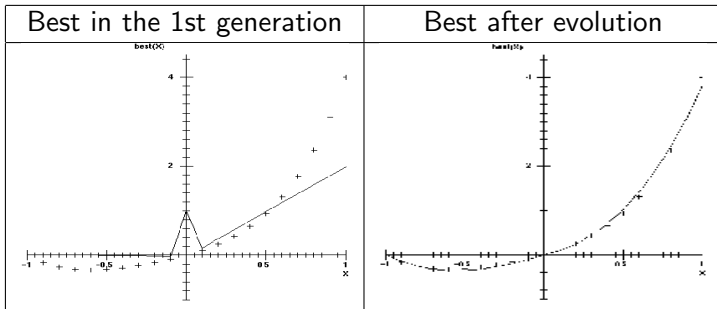
GP in symbolic regression, #1

$$f(x) = x + x^2 + x^3 + x^4$$



GP in symbolic regression, #1

$$f(x) = x + x^2 + x^3 + x^4$$



[+ [- [plog [pexp x]] [+ [sin x] [- x x]]] [+ [pexp
[plog x]] [sin [plog x]]]

[* [+ [+ [+ [* [+ [pexp [plog x]] [sin [plog [+ x
x]]]] x] [pexp [plog x]]] [pdiv x x]] x] [plog [pexp [+
[plog [- [sin [+ [plog [pexp [plog [- [sin [+ [+ [+ x
x] [pdiv [* [+ [+ [+ x [pexp [plog x]]] [pdiv x x]] x]
x] x]] x]] [pexp x]]]]]] x]] [pexp [plog [- [sin [+ [+ [+
x x] [pdiv [* [+ [+ [+ x [pexp [plog x]]] [pdiv x x]]
x] x] x]] x]] [pexp x]]]]]] [plog [- x x]]]]]]]

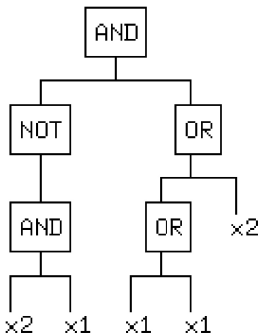
GP in symbolic regression, #2

$$f(x_1, x_2) = XOR(x_1, x_2)$$

AND, OR, NOT gates only	NAND gates only
AND [NOT [AND x2 x1]] [OR [OR x1 x1] x2]	NAND [NAND [NAND x2 x1] x2] [NAND x1 [NAND x1 x2]]

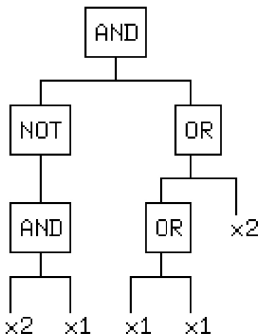
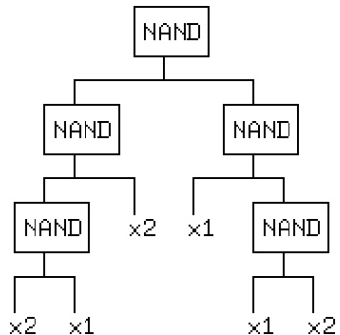
GP in symbolic regression, #2

$$f(x_1, x_2) = XOR(x_1, x_2)$$

AND, OR, NOT gates only	NAND gates only
 <pre>graph TD AND1[AND] --- NOT1[NOT] AND1 --- OR1[OR] NOT1 --- AND2[AND] AND2 --- x2_1[x2] AND2 --- x1_1[x1] OR1 --- OR2[OR] OR1 --- x2_2[x2] OR2 --- x1_2[x1] OR2 --- x1_3[x1]</pre>	
AND [NOT [AND x2 x1]] [OR [OR x1 x1] x2]	NAND [NAND [NAND x2 x1] x2] [NAND x1 [NAND x1 x2]]

GP in symbolic regression, #2

$$f(x_1, x_2) = \text{XOR}(x_1, x_2)$$

AND, OR, NOT gates only	NAND gates only
 <pre>graph TD AND1[AND] --> NOT[NOT] AND1 --> OR1[OR] NOT --> AND2[AND] AND2 --> x2_1[x2] AND2 --> x1_1[x1] OR1 --> OR2[OR] OR1 --> x2_2[x2] OR2 --> x1_2[x1] OR2 --> x1_3[x1]</pre>	 <pre>graph TD NAND1[NAND] --> NAND2[NAND] NAND1 --> NAND3[NAND] NAND2 --> NAND4[NAND] NAND2 --> x2_1[x2] NAND4 --> x2_2[x2] NAND4 --> x1_1[x1] NAND3 --> x1_2[x1] NAND3 --> NAND5[NAND] NAND5 --> x1_3[x1] NAND5 --> x2_3[x2]</pre>
AND [NOT [AND x2 x1]] [OR [OR x1 x1] x2]	NAND [NAND [NAND x2 x1] x2] [NAND x1 [NAND x1 x2]]

Advanced EA mechanisms

- Speciation
- Hybridization with LS (= GLS/memetic)
- Genetic-based machine learning (GBML) / classifier systems (CFS)
- Coevolution
 - Competitive
 - Cooperative
- Directed/undirected and open/closed-ended evolution

So, in summary...

what (general) approaches and optimization algorithms do you know?

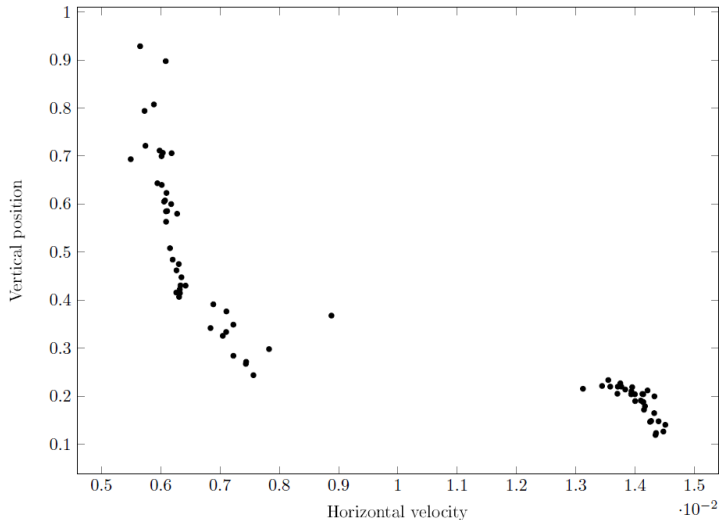
Considering multiple criteria

- Examples: TSP, timetable

Considering multiple criteria

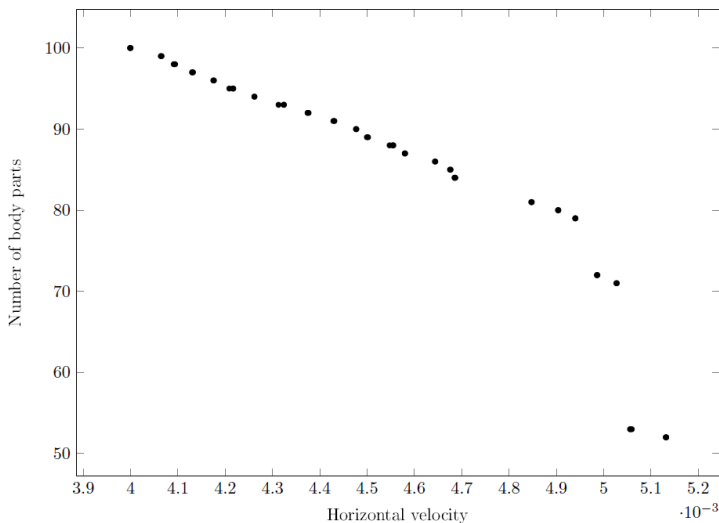
- Examples: TSP, timetable
- Transforming multiple criteria into a single criterion
 - Lexicographical order
 - Weighted sum
 - Dominance relation and using some characteristic value
- Considering multiple criteria all together – the dominance relation
 - Pareto Local Search – we start from many random solutions, (*) we identify non-dominated solutions, for each non-dominated solution we evaluate all its neighbors using multiple criteria, then go back to (*)
 - Dedicated mechanisms and techniques of algorithms – for example selection in EA (NSGA, SPEA and others)

Multi-criteria optimization, population, example 2D (a)



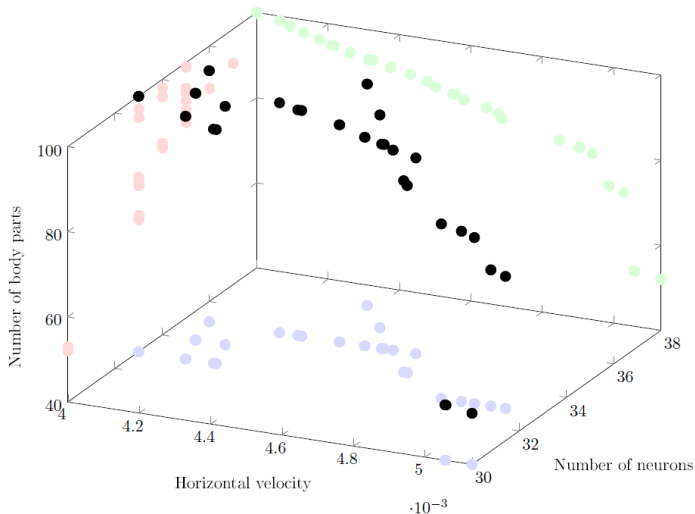
(a) maximization of horizontal velocity and vertical position

Multi-criteria optimization, population, example 2D (b)



(b) maximization of horizontal velocity and number of creature's body parts

Multi-criteria optimization, population, example 3D



(b) maximization of horizontal velocity, number of creature's body parts and minimization of number of neurons