



Machine Learning

Evolutionary Algorithms



Evolutionary Algorithms

Genetic Programming

Evolution Strategies

Genetic Algorithms

Classifier Systems

Evolutionary Programming

- genetic representation of candidate solutions
- genetic operators
- selection scheme
- problem domain



WWW-Resources

- Genetic Programming Notebook
 - <http://www.geneticprogramming.com>
software, people, papers, tutorial, FAQs
- Hitch-Hiker's Guide to Evolutionary Computation
 - <http://alife.santafe.edu/~joke/encore/www>
FAQ for comp.ai.genetic
- Genetic Algorithms Archive
 - <http://www.aic.nrl.navy.mil/galist>
Repository for GA related information, conferences, etc.
- EVONET European Network of Excellence on Evolutionary Comp. : <http://www.dcs.napier.ac.uk/evonet>



Literature

- Goldberg, D. "Genetic Algorithms in Search and Optimization"
Addison-Wesley, Reading MA, 1989
- Mitchell, M. "An Introduction to Genetic Algorithms"
MIT Press, Cambridge, MA, 1996
- Koza, J. "Genetic Programming II"
MIT Press, Cambridge, MA, 1994
- Holland, J. "Adaptation in Natural and Artificial Systems"
University of Michigan Press, Ann Arbor, 1975
- Bäck, Th. "Evolutionary Algorithms in Theory and Practice"
Oxford University Press, New York, 1996



Outline

- simple genetic algorithm (SGA)
- examples
 - brachystochone problem
 - lens optimization
 - prisoner's dilemma
 - traveling salesman problem
- schema theorem
- evolution strategies
- genetic programming



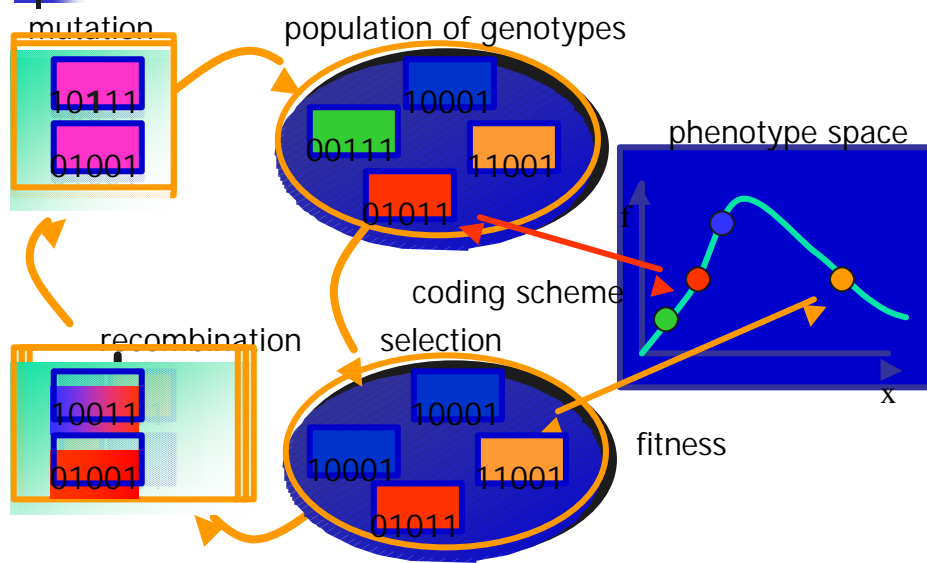
Biological Terminology

- 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
 - set of competing genomes/individuals

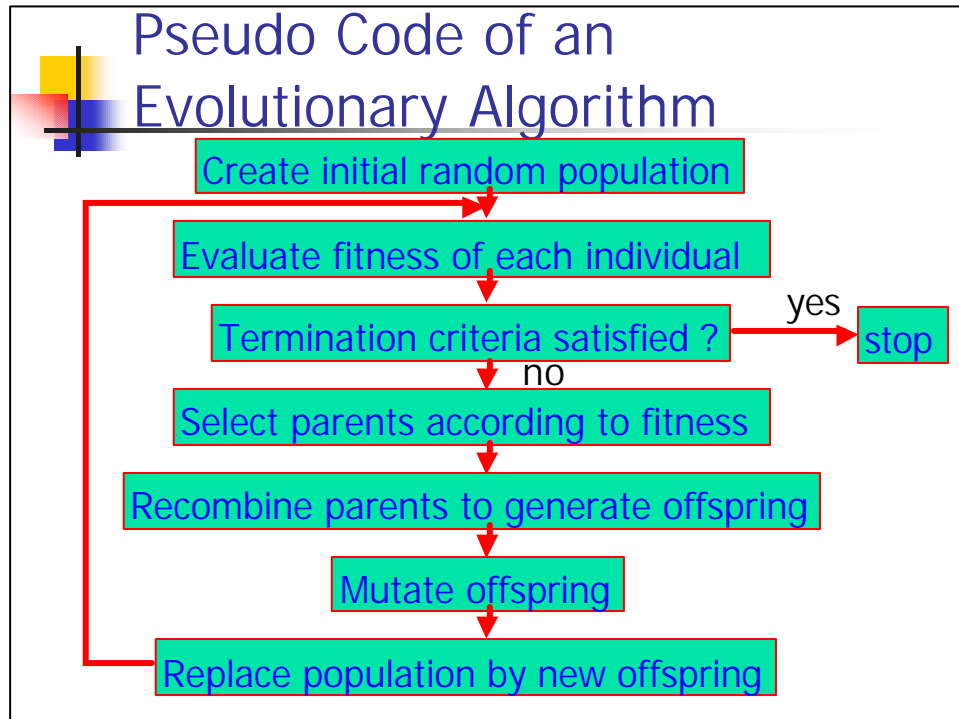
Genotype versus Phenotype

- 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")

Evolutionary Algorithm



Pseudo Code of an Evolutionary Algorithm



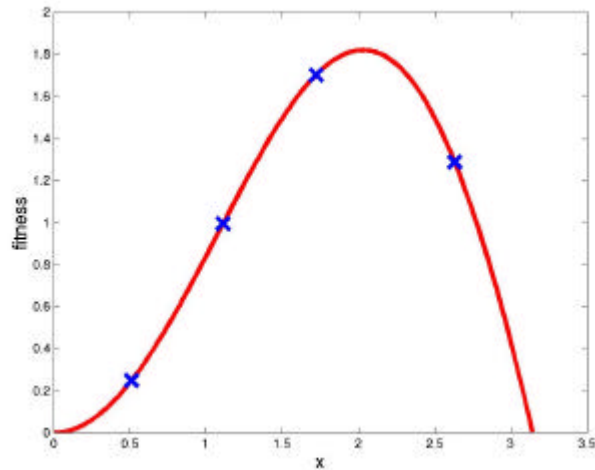
A Simple Genetic Algorithm

- optimization task : find the maximum of $f(x)$
for example $f(x) = x \cdot \sin(x)$ $x \in [0, \pi]$
- genotype: binary string $s \in \{0,1\}^5$ e.g. 11010, 01011, 10001
- mapping : genotype \rightarrow phenotype
binary integer encoding: $x = \sum_{i=1}^{n=5} s_i \cdot 2^{n-i} / (2^n - 1)$

Initial population

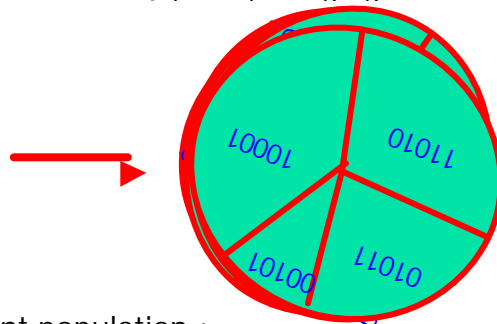
genotype	integ.	phenotype	fitness	prop. fitness
11010	26	2.6349	1.2787	30%
01011	11	1.1148	1.0008	24%
10001	17	1.7228	1.7029	40%
00101	5	0.5067	0.2459	6%

Phenotype Space



Roulette Wheel Selection

- selection is a stochastic process
- probability of reproduction $p_i = f_i / \sum_k f_k$



- inter-mediate parent population :

01011 11010 10001 10001

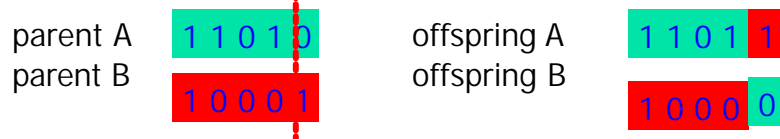
Genotype Operators

- recombination (crossover)
 - combines two parent genotypes into a new offspring
 - generates new variants by mixing existing genetic material
 - stochastic selection among parent genes
- mutation
 - random alteration of genes
 - maintain genetic diversity
- in genetic algorithms crossover is the major operator whereas mutation only plays a minor role

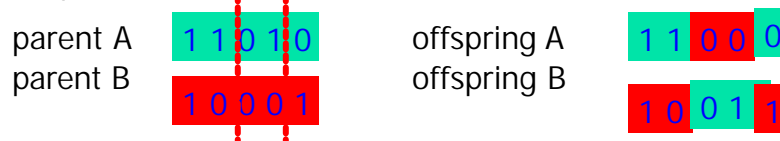
Crossover

- crossover applied to parent strings with probability p_c ? [0.6..1.0]
- crossover site chosen randomly

- one-point crossover

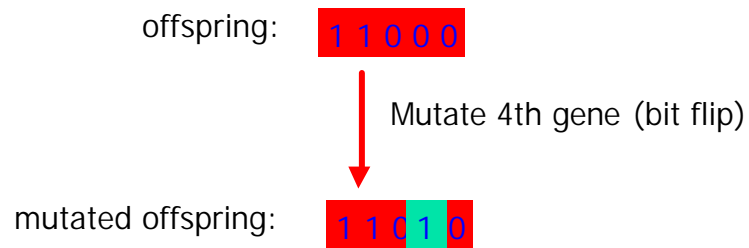


- two-point crossover

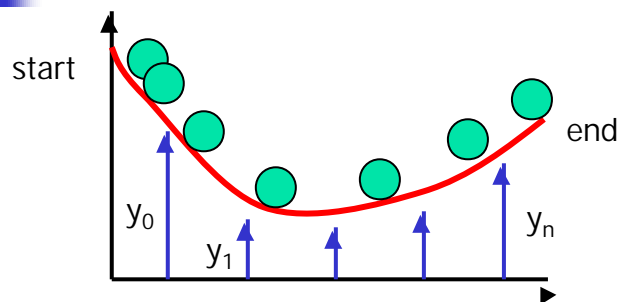


Mutation

- mutation applied to allele/gene with probability P_m ? [0.001..0.1]
- role of mutation is to maintain genetic diversity

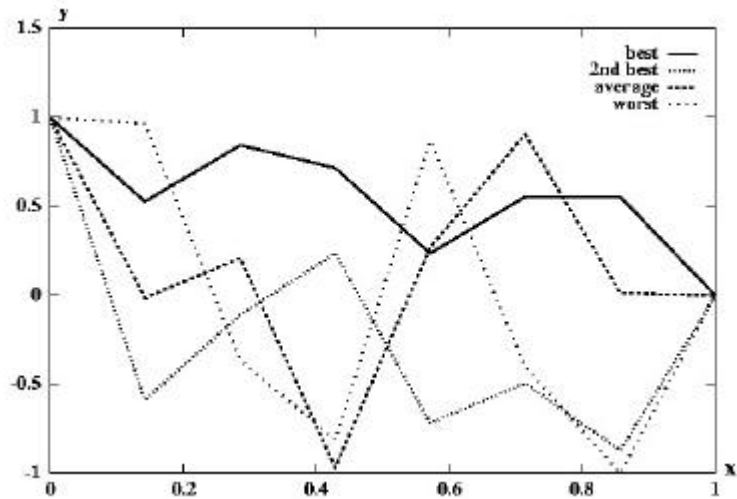


Evolution of a Brachystochrone

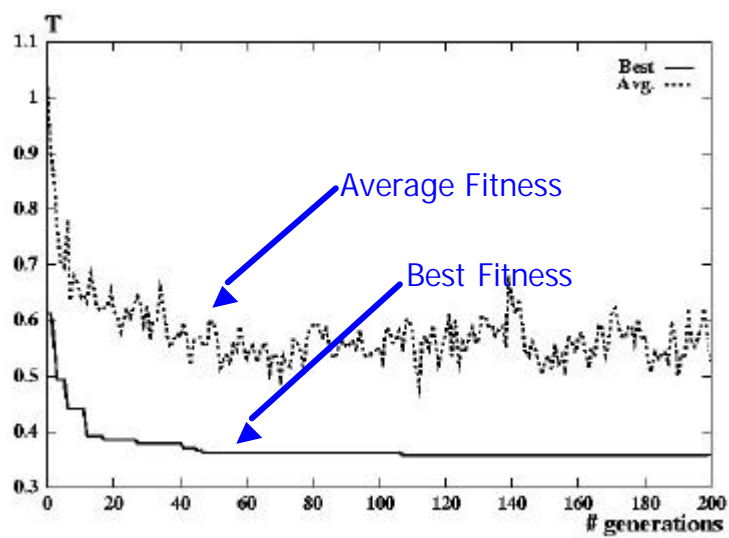


- objective: optimize track between start and end point such that a frictionless point mass travels the path in minimal time
- phenotype parameters y_i : height h of track at point x_i
- fitness: time expired to get from start to end point
- parameter encoding: gray-encoded bit strings

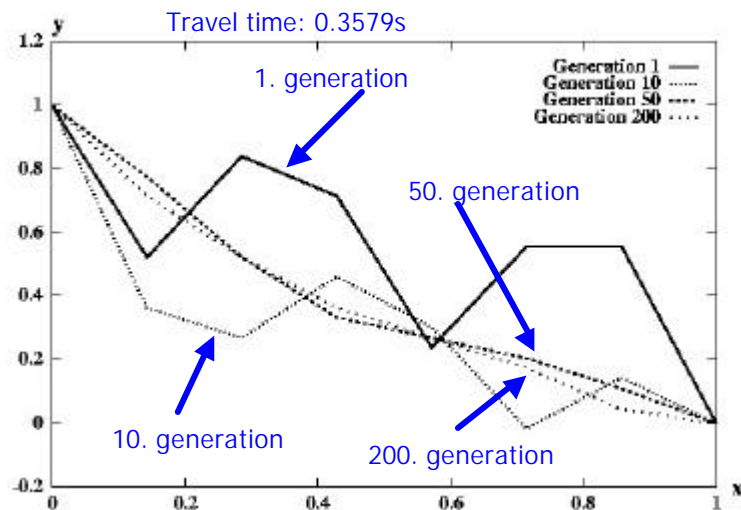
Brachystochrone with a GA



Evolution of Fitness



Brachystochrone Best Solution



Evonet Flying Circus Brachy.

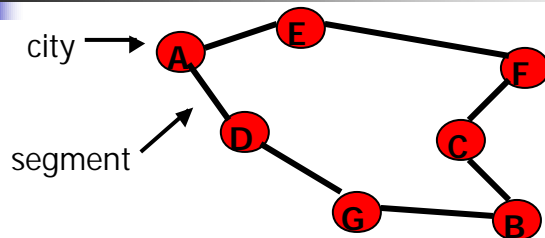
http://www.wi.leidenuniv.nl/~gusz/Flying_Circus

- ✦ Brachystochrone
- ✦ Prism Lens
- ✦ Traveling Salesman
- ✦ Symbolic Regression
- ✦ ...

<..\Demos\index.html>

..\Demos\index_prism.htm

Traveling Salesman Problem



- objective: find the shortest tour that visits each city (NP-hard)
- a tour is encoded by a sequence AEFGBGD which specifies the order in which cities are visited
- fitness: overall length of the tour
- phenotype (tour) is defined by the permutation of genes rather than their values

Traveling Salesman Problem

- usual crossover results in invalid tours

AEF|CBGD ? AEF DFAB

ECG|DFAB ? ECG CBGD

- edge recombination operator

parent 1 AEFGBDG

parent 2 ECGDFAB

City	Connections
A	E,D,F,B
B	C,G,A,E
C	F,B,E,G
D	G,A,F
E	A,F,B,C
F	E,C,D,A
G	B,D,C

1. Select a start city
2. Select the neighbor with the minimum number of connections (break ties randomly)
3. Repeat 2 until no more cities are left

..\Demos\index_salesman.htm

Prisoners Dilemma

		player B	
		cooperate	defect
player A	cooperate	3,3	0,5
	defect	5,0	1,1

- objective: maximize payoff in prisoners dilemma
- encode next action (d=0, c=1) for next move depending on the possible outcome (5-bits) in the previous round against the same opponent

Last move : cc cd dc dd initial move

Action: 1 0 1 0 1 Tit for Tat
 0 0 0 0 0 always defect

- fitness: payoff obtained in a tournament against the other members in the current population (dynamic fitness function)

Extensions to the Simple GA

Encoding schemes

- gray encoding
- messy genetic algorithms

Replacement schemes

- generational replacement
- steady state replacement

Fitness scaling

- linear scaling
- ? - truncation
- ranking

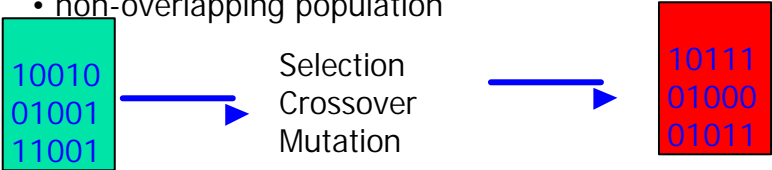
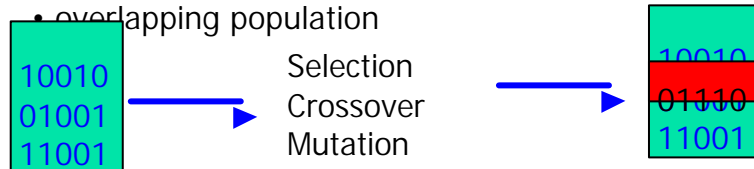
Selection schemes

- stochastic sampling
- tournament selection

Fitness Scaling

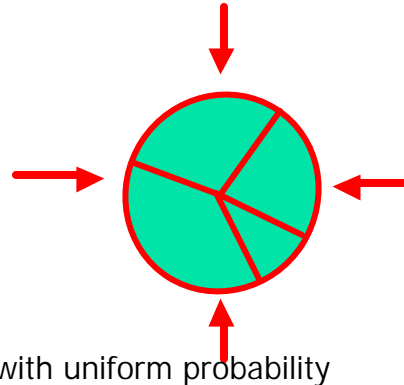
- linear scaling : $f_k^* = a f_k + b$
 - choose a,b such that the best individual gets m (m ~ two) offspring and the average individual gets one offspring
- σ -truncation : $f_k^* = f_k - (f_{avg} - c \sigma)$ σ : standard deviation
 - selection pressure related to the fitness distribution
- normalizing : $f_k^* = (f_k - f_{min} + \sigma) / (f_{max} - f_{min} + \sigma)$
 - dynamic scaling
- ranking : $p_k = q - (k-1) r$
 - choose q and r such that $\sum p_k = 1$
 - sort the population from the best (k=1) to the worst (k=n)
 - assign selection probability according to ranking

Replacement Schemes

- generational replacement
 - entire population is replaced each generation
 - non-overlapping population
- 
- steady state replacement
 - a single individual (worst, random) is replaced by one offspring
 - overlapping population
- 

Selection Schemes

- stochastic sampling
 - roulette wheel selection
 - spin wheel N times
- stochastic universal sampling
 - roulette wheel selection
 - single spin, wheel has N equally spaced markers
- tournament selection
 - choose k candidates at random with uniform probability
 - pick best one for reproduction
 - expected number of offspring
 - best : k , average $k/2^{k-1}$, worst $k/2^{k-1}$



Gray-Coding

- proper genetic representation of candidate solutions is crucial for the success of an evolutionary algorithm
- in general similar phenotypes should correspond to similar genotypes
- Hamming distance : number of different bits among two binary strings
- Standard base two encoding : e.g. 1000 and 0111 have maximal Hamming distance but correspond to the adjacent integers 7 and 8 so called Hamming-cliff

Gray-Coding

- adjacent integers are represented by bit strings that only differ in one bit
- given a binary string s_1, \dots, s_n coding the integer in the standard way the conversion to a Gray coded string g_1, \dots, g_n is :

$$g_k = \begin{cases} s_1 & : \text{if } k=1 \\ s_{k+1} \oplus s_k & : \text{if } k>1 \end{cases}$$

\oplus Denotes addition modulo 2 :

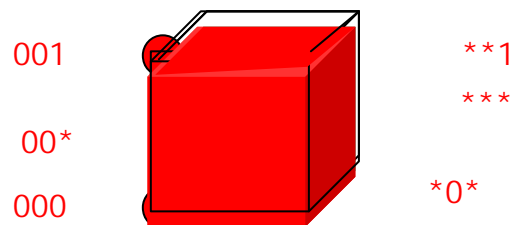
$$0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0$$

Integer	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111
Gray	000	001	011	010	110	111	101	100

Concept of a Schema

Schema

- template of '1', '0' and '*' (wild cards)
- instances of a schema are strings that match the template
- a schema corresponds to a subspace in genotype space



Schema $h = **1 ? \{001, 011, 101, 111\}$



Concept of a Schema

- order $o(H)$ of a schema H : #defined (0,1) bits
- defining length $d(H)$ of a schema H : distance between outermost defined bits

Schema H	$o(H)$	$d(H)$
1^*0^*	2	3
$***0^*$	1	0
1^*10	3	4



Schema Theorem

- $m(h,t)$: #instances of schema h in the population at generation t
- $f(h,t)$: observed average fitness of schema h at generation t
 $f(h) = \sum_{x \in h} f(x) / m(h,t)$
- $E(m(h,t))$: expected #instances of schema h
- selection :
 $E(m(h,t+1)) = \sum_{x \in h} f(x) / \langle f \rangle = m(h,t) f(h,t) / \langle f \rangle$
- crossover: probability that a schema h of defining length $d(h)$ is not destroyed by crossover: $(1 - p_c d(h) / (l-1))$
- mutation: probability that schema h of order $o(h)$ is not destroyed by mutation : $(1 - p_m)^{o(h)}$

Schema Theorem

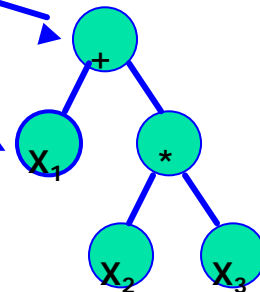
$$E(m(h,t+1)) = m(h,t) f(h,t)/\langle f \rangle (1 - p_c d(h)/(l-1)) (1 - p_m)^{o(h)}$$

- simple GA increases the number of schemata with
 - low order
 - short defining length
 - above average fitness
- implicit parallelism
 - simultaneous evaluation of a large number of schemata in a population of n strings
 - one string implicitly samples 2^l different schemata
- building block hypothesis
 - GA works by recombining instances of good schemata to form instances of equally good or better higher-order schemata by means of crossover

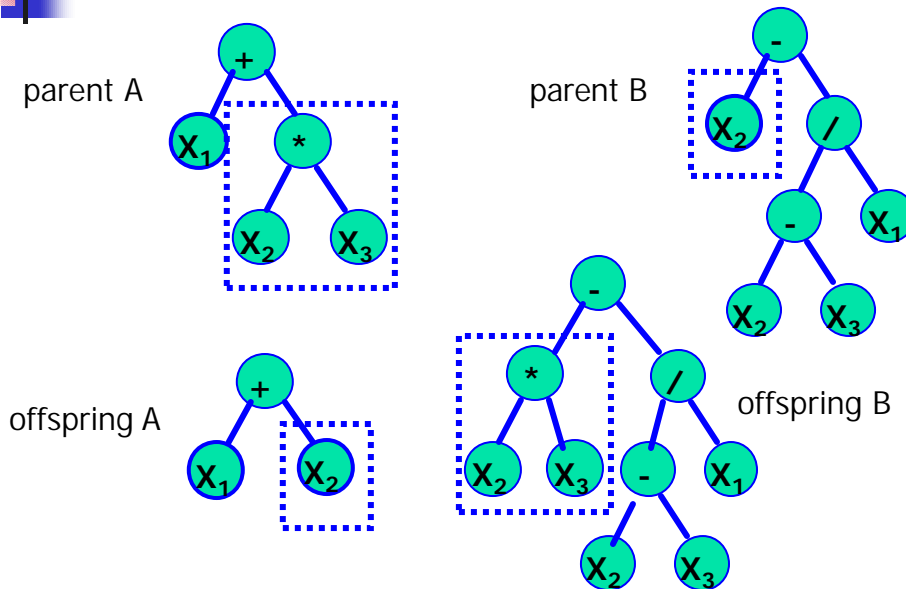
Genetic Programming

- automatic generation of computer programs by means of natural evolution see Koza 1999
- programs are represented by a parse tree (LISP expression)
- tree nodes correspond to functions :
 - arithmetic functions {+, -, *, /}
 - logarithmic functions {sin, exp}
- leaf nodes correspond to terminals :
 - input variables { X_1, X_2, X_3 }
 - constants {0.1, 0.2, 0.5}

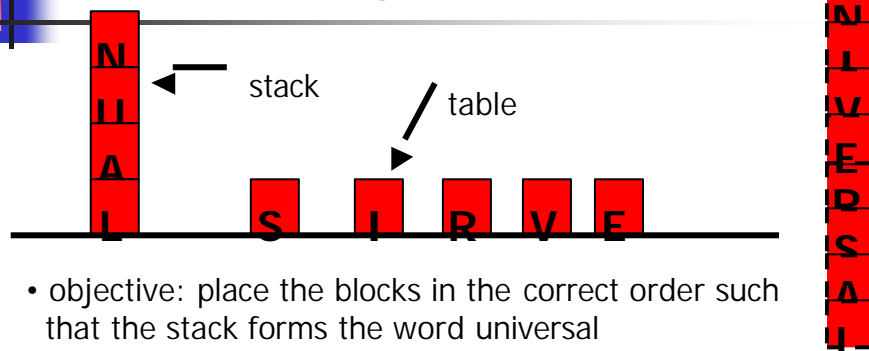
tree is parsed from left to right:
 $(+ X_1 (* X_2 X_3)) \rightarrow X_1 + (X_2 * X_3)$



Genetic Programming : Crossover

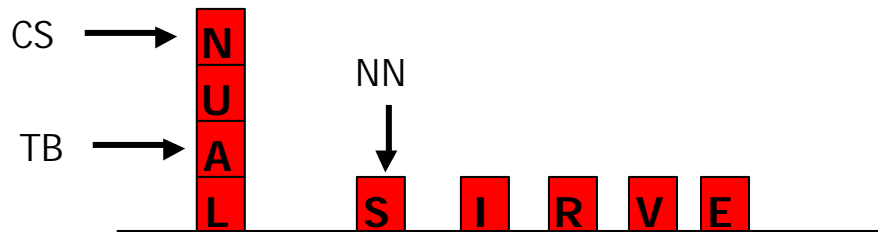


Block-Stacking Problem



- objective: place the blocks in the correct order such that the stack forms the word universal
- functions: set of actions, logical operators, do-until loop
- terminals: set of sensors that indicate top block on stack, next suitable block on table etc.
- each program tree is tested on 166 different initial configurations
- fitness: #configurations for which the stack was correct after program execution

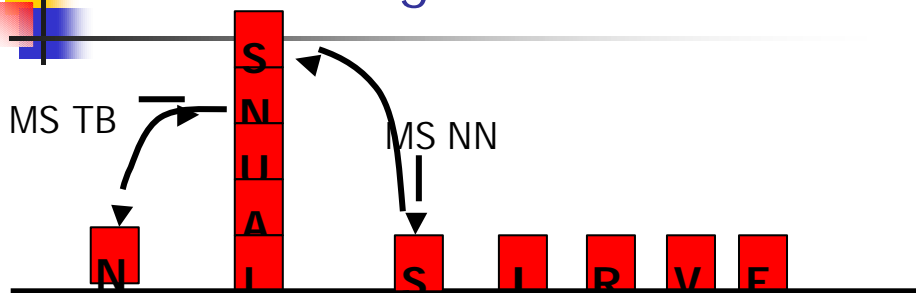
Block-Stacking Problem



Sensors:

- CS: current stack, name of the top block of the stack
- TB: top correct block, name of the topmost block on the stack such that it and all blocks underneath are in correct order
- NN: next block needed, name of the block needed above TB

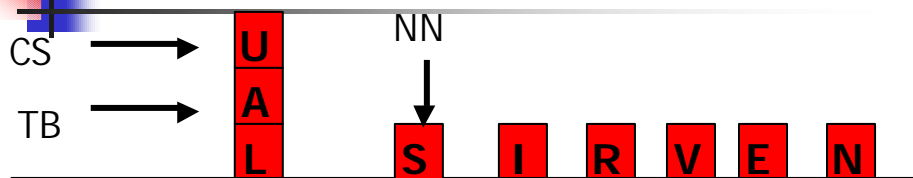
Block-Stacking Problem



Functions:

- MS(X): move block X to the top of the stack, return value X
- MT(X): moves the block on top of the stack to the table if X is anywhere in the stack returns X
- DU(exp1, exp2): execute exp1 until the predicate exp2 becomes true
- NOT(exp1) : negation of expression exp1
- EQ(exp1, exp2) : returns true if exp1 and exp2 are equal

Block-Stacking Problem



- (EQ (MS NN) (EQ (MS NN) (MS NN)))
move next needed block to the stack three times in a row (succeeds with a stack VERSAL and U N I on the table)
- (DU (MS NN) (NOT NN))
move next needed block to the stack until no more blocks are needed
- (EQ (DU (MT CS) (NOT CS)) (DU (MS NN) (NOT NN)))
empty the stack on the table and then build the stack in the correct order
- (EQ (DU (MT CS) (EQ (CS TB))) (DU (MS NN) (NOT NN)))

Evolution Strategy vs. Genetic Algorithms

	<i>evolutionary strategy</i>	<i>genetic algorithm</i>
<i>genetic representation:</i>	<i>real valued</i>	<i>binary</i>
<i>importance of genetic operators:</i>	<i>1. mutation 2. recombination</i>	<i>1. recombination 2. mutation</i>
<i>self adaptation:</i>	<i>yes</i>	<i>no</i>
<i>selection:</i>	<i>deterministic</i>	<i>stochastic</i>
<i>problem domain:</i>	<i>continuous optimisation problems</i>	<i>discrete, combinatorial optimisation problems</i>