Evolutionary Algorithms

in

High Energy Physics and Computing

Liliana Teodorescu

Brunel UNIVERSITY WEST LONDON

Outline

- ***Introduction to evolutionary computation***
- ***Evolutionary algorithms***
  - ✓ *solution representation*
  - ✓ *fitness function*
  - ✓ *initial population generation*
  - ✓ *genetic and selection operators*
- ***Types of evolutionary algorithms***
  - ✓ *Genetic Algorithms*
  - ✓ *Evolutionary Strategies*
  - ✓ *Genetic Programming*
  - ✓ *Gene Expression Programming*
- ***Applications in HE Physics and Computing***
  - ✓ *data analysis tasks*
  - ✓ *job scheduling*
- ***Conclusions***

Evolutionary Computation

- *Evolutionary computation* simulates the natural evolution on a computer

  ↓

  process leading to maintenance or increase of a population ability to survive and reproduce in a specific environment

  ↓

  quantitatively measured by evolutionary fitness

- *Goal of natural evolution* - to generate a population of individuals with increasing fitness

- *Goal of evolutionary computation* - to generate a set of solutions (to a problem) of increasing quality

Terminology

- *Individual* – candidate solution to a problem

decoding ↑ ↓ encoding

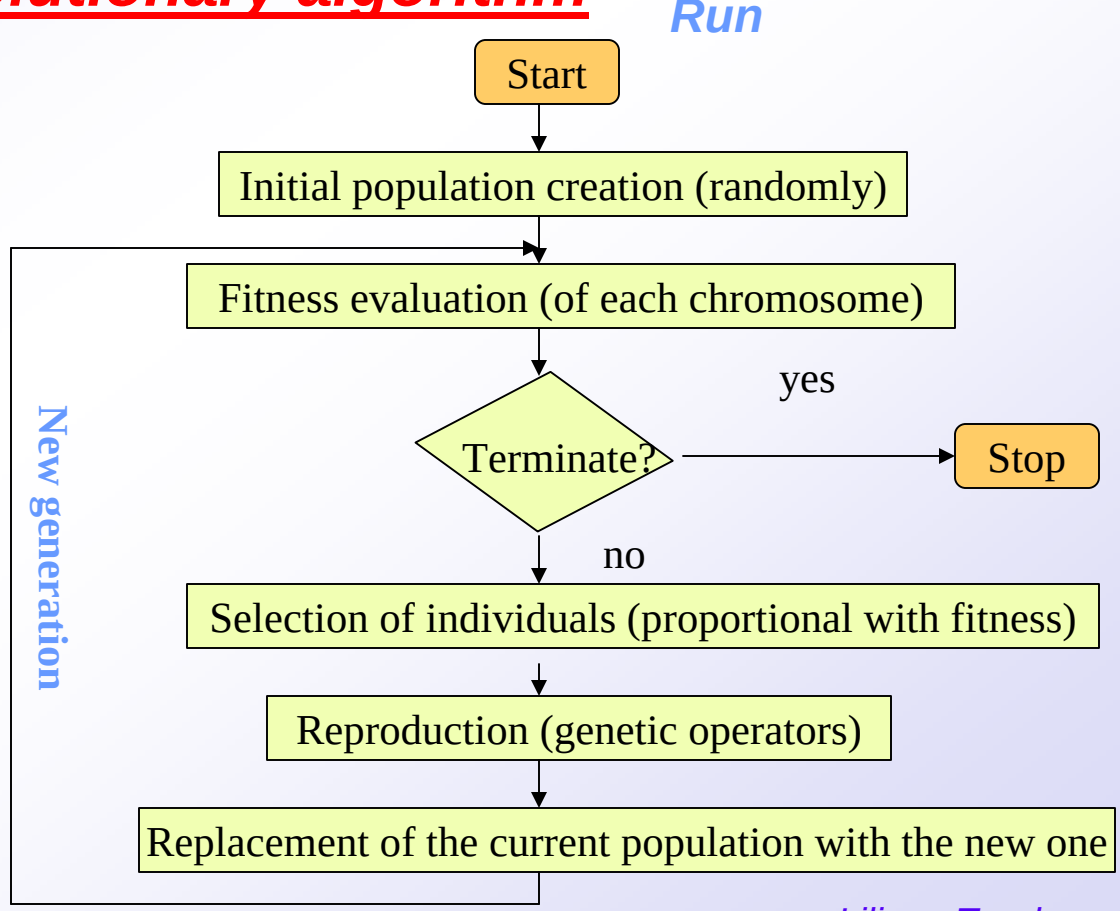
- *Chromosome* – representation of the candidate solution
- *Gene* – constituent entity of the chromosome
- *Population* – set of individuals/chromosomes
- *Fitness function* – representation of how good a candidate solution is
- *Genetic operators* – operators applied on chromosomes in order to create *genetic variation* (other chromosomes)

Evolutionary Algorithms

Natural evolution simulation - core of the evolutionary algorithms:
optimisation algorithms (iteratively improve the quality of the solutions until an optimal/feasible solution is found)

Basic evolutionary algorithm

- *Problem definition*
- *Solution representation (encoding the candidate solution)*
- *Fitness definition*
- *Run*
- *Decoding the best fitted chromosome = solution*



Solution representation

Chromosome – representation of the candidate solution

Each chromosome represents a point in search space

Appropriate chromosome representation

- ✓ *very important for the success of EA*
- ✓ *influence the efficiency and complexity of the search algorithm*

Representation schemes

- ❖ *Binary strings – each bit is a boolean value, an integer or a discretized real number*
- ❖ *Real-valued variables*
- ❖ *Trees*

Fitness function

The most important component of EA !

Fitness function *- representation of how good (close to the optimal solution) a candidate solution is*

- maps a chromosome representation into a scalar value

$$F : C^{I} \rightarrow \Re$$ I – chromosome dimension

Fitness function needs to model accurately the optimisation problem

Used:

- ✓ *in the selection process*
- ✓ *to define the probability of the genetic operators*

Includes:

- ✓ *all criteria to be optimised*
- ✓ *reflects the constraints of the problem penalising the individuals that violates the constraints*

Initial population

Generation of the initial population:

- ***random generation of gene values from the allowed set of values (standard method)***

  Advantage - ensure the initial population is a uniform representation of the search space

- ***biased generation toward potentially good solutions if prior knowledge about the search space exists.***

  Disadvantage – possible premature convergence to a local optimum

Size of the initial population:

- ***small population – represents a small part of the search space***
  - ✓ ***time complexity per generation is low***
  - ✓ ***needs more generations***
- ***large population – covers a large area of the search space***
  - ✓ ***time complexity per generation is higher***
  - ✓ ***needs less generations to converge***

Reproduction (genetic) operators

Purpose

- ***to produce offspring from selected individuals***
- ***to replace parents with fitter offspring***

Typical operators

- ***cross-over – creates new individuals combining genetic material from parents***
- ***mutation - randomly changes the values of genes (introduces new genetic material)***
  - has low probability in order not to distorts the genetic structure of the chromosome and to generate loss of good genetic material
- ***elitism/cloning – copies the best individuals in the next generation***

The exact structure of the operators – dependent on the type of EA

Selection operators

***Purpose** - to select individuals for applying reproduction operators*

- ***Random selection** – individuals are selected randomly, without any reference to fitness*
- ***Proportional selection** – the probability to select an individual is proportional with the fitness value*

$$P(C_n) = \frac{F(C_n)}{\sum_{n=1}^{N} F(C_n)}$$

$P(C_n)$ –*selection probability of the chromosome* C_n
$F(C_n)$ – *fitness value of the chromosome* C_n

- ✓ ***Normalised distribution by dividing to the maximum fitness - accentuate small differences in fitness values (roulette wheel method)***

- ***Rank-based selection** – uses the rank order of the fitness value to determine the selection probability (not the fitness value itself)*

e.g. non-deterministic linear sampling – individual sorted in decreasing order of the fitness value are randomly selected

- ***Elitism** – k best individuals are selected for the next generation, without any modification*

k – called generation gap

EA vs classical optimisation

	EA	CO
Transition from one point to another in the search space	✓Probabilistic rules ✓Parallel search	✓ Deterministic rules ✓ Sequential search
Starting the search process	Set of points	One point
Search surface information that guides to the optimal solution	No derivative information (only fitness value)	Derivative information (first or second order)

Classes of Evolutionary Algorithms

- *Genetic Algorithms (GA) (J. H. Holland, 1975)*
- *Evolutionary Strategies (ES) (I. Rechenberg, H-P. Schwefel, 1975)*
- *Genetic Programming (GP) (J. R. Koza, 1992)*
- *Gene Expression Programming (GEP) (C. Ferreira, 2001)*

Main differences

- *Encoding method (solution representation)*
- *Reproduction method*

Genetic Algorithms

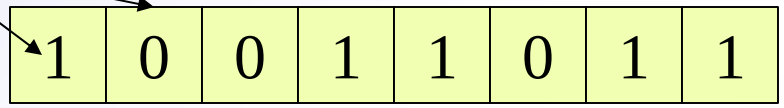
Solution representation

Chromosome - fixed-length binary string (common technique)

Gene - each bit of the string

genes

chromosome

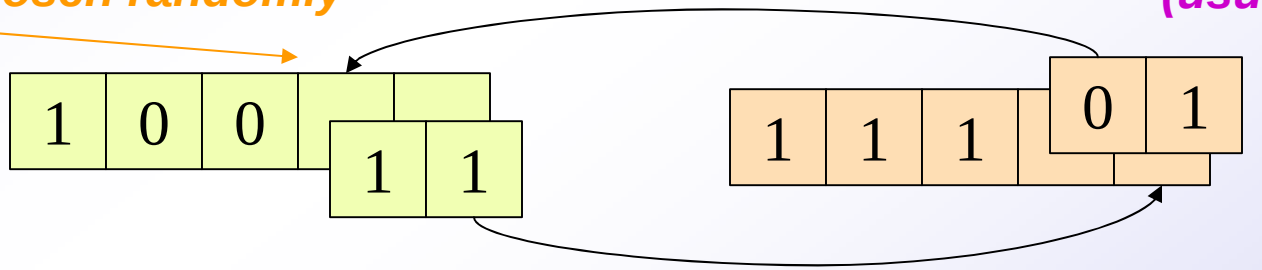


Reproduction

Cross-over (recombination) – exchanges parts of two chromosomes

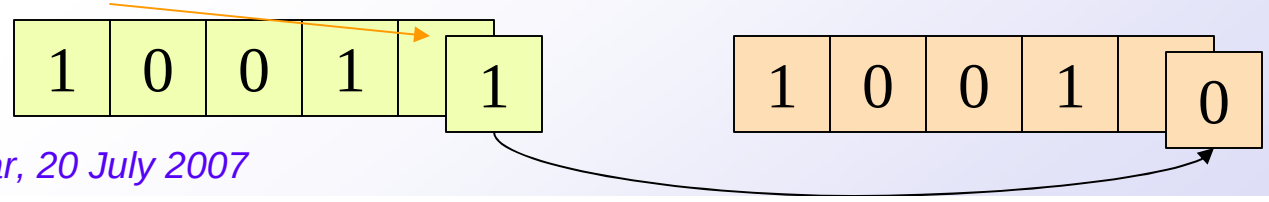
Point choosen randomly

(usual rate 0.7)



Mutation – changes the gene value (usual rate 0.001-0.0001)

Point choosen randomly



GA for job scheduling

Problem:

- *schedule m jobs on n resources (computer nodes)*
- *optimisation problem (GRID => large scale optimisation)*
- *optimisation objective:*
  - *uni-objective (e.g. job execution time)*
  - *multi-objective – more often (e.g. execution time, flow time, resources utilization etc.)*

GA specific to the problem

- *solution representation*
- *special genetic operators*

Typical GA for job scheduling

Solution representation

Chromosome – decimal string containing computer nodes

Computer nodes: ***P1 P2 P3 P4 … Pn***

Chromosome

P1	P2	P3	P3	P4	P4	P2	P1

represented as genes

Jobs ***J1 J2 J3 J4 J5 J6 J7 J8***

(position of a gene represents the sequence number of a job)

Fitness function $F = \frac{1}{Max\ (T_1, T_2, \ldots T_n)}$ T_i - execution time

Reproduction

Genetic operators – typical cross-over, mutation

Disadvantages – high convergence time

GA for job scheduling - improvements

PGGA – predictable and grouped GA for job scheduling

(M. Li et. al., Future Generation Computer Science 22 (2006) 588-599)

- ***classify computer nodes in groups based on their utilisable computing capabilities***
- ***dynamically predict an optimal fitness value using the divisible load theory***

optimal solution for job scheduling based on minimisation of the execution time - all the computing nodes finish their jobs at the same time

$$T = \frac{W}{\sum_{k=1}^{N} (F(G_k) \times N(G_k))}$$

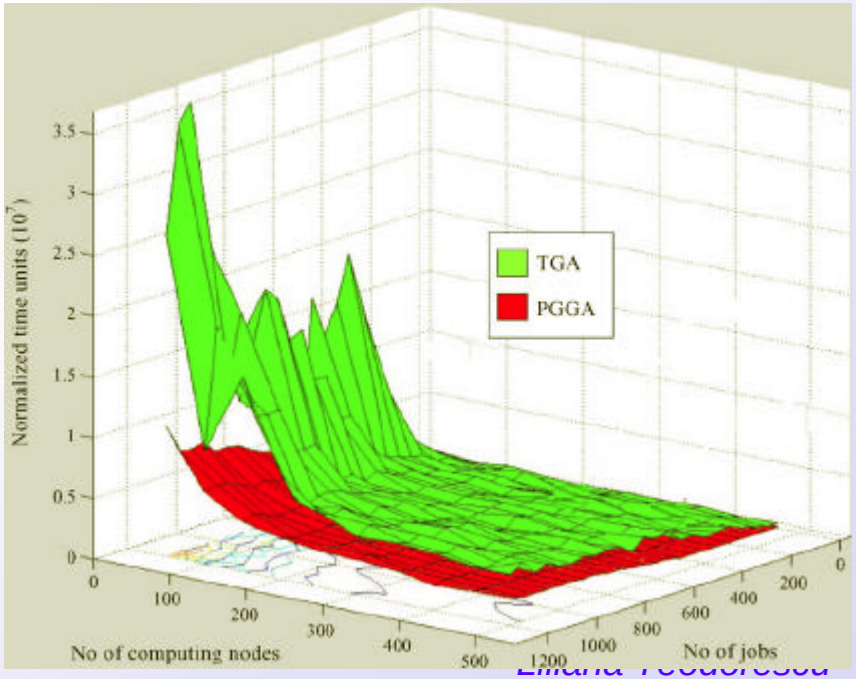
Total workload

Utilisable computing capability

Number of nodes in the group

Optimal solution – fitness value close to $\frac{1}{T}$

Speed improved by filtering out chromosomes with fitness values far away from the optimal value



GA for job scheduling - other improvements

Other versions

Specific genetic operators

e.g. mutation:

- *move: move a job from a node to another*
- *swap: interchange the jobs between nodes*

Multiple objective optimisation

- optimisation criteria defined hierarchically (e.g first execution time, then the flow time etc.)

- simultaneous optimisation of criteria

Other references

V. Di Martino, M. Mililotti – Sub optimal scheduling in a grid using GA, Parallel Computing, vol 30 (2004) 553-565

A. Abraham et. al., Nature's heuristic for scheduling jobs on computational Grids, 8th IEEE Int. Conf on Advanced Computing and Communications, 2000

A.Y. Zomaya, Y.H. The, Observations on Using GA for Dynamic Load-balancing, IEEE Transactions on Parallel and Distributed Systems, vol 12, no 9, 2001

GA in HEP

Mainly for large-scale optimisation and fitting problems

Experimental HEP

- ***event selection optimisation*** *(A. Drozdetskiy et. Al. Talk at ACAT2007)*
- ***trigger optimisation*** *(L1 and L2 CMS SUSY trigger – NIM A502 (2003) 693)*
- ***neural-netwok optimisation for Higgs search***
  (F. Hakl et.al., talk at STAT2002)

Theoretical/phenomenological HEP

- ***fitting isobar models to data for p(γ,K⁺)Λ*** *(NP A 740 (2004)147)*
- ***discrimination of SUSY models*** *(hep-ph/0406277)*
- ***lattice calculations*** *(NP B (Pric. Suppl.) 73 (1999) 847; 83-84 (2000)837*

Evolutionary Strategies

Based on the concept of evolution of the evolution:
the evolution optimises itself

Individual – represented by

- ***its genetic characteristics***
- ***a strategy parameter - models the behaviour of the individual in the environment***

Evolution – evolve both the genetic characteristics and the strategy parameter

Solution representation

$$C_n = (G_n, S_n)$$

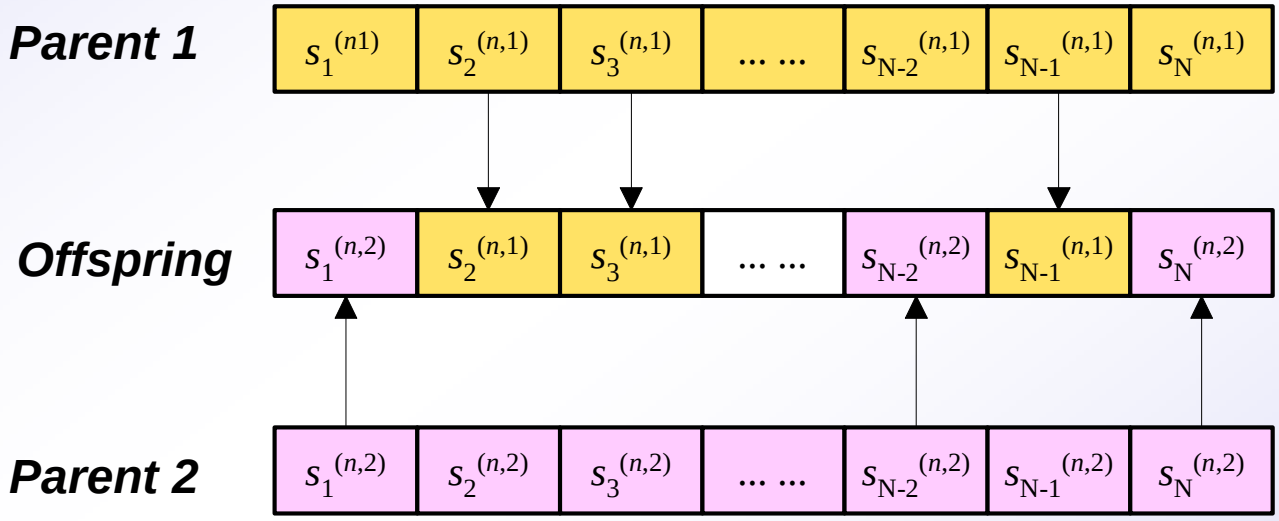
G_n – ***genetic material: floating-point values***

S_n – ***strategy parameter: standard deviation of a normal distribution associate with each individual***

Evolutionary Strategies (cont.)

Reproduction

- *Cross-over (recombination)* - ***offspring generated from material randomly selected from two parents***
  - ***Recombination of the selected material***
    - ✓ *discrete* – ***offspring's gene value is the gene value of the parents***

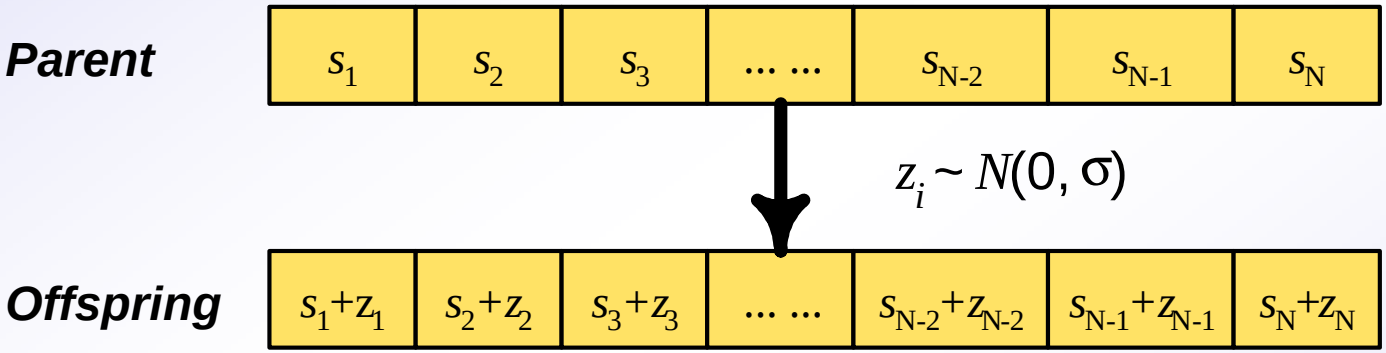


- ✓ *intermediate recombination* – ***offspring's gene value is the midpoint between the gene values of the parents***

Evolutionary Strategies (cont.)

Reproduction

- **Mutation**
  - ✓ *of the genetic material* – ***add a random number from a normal distribution to the each gene value***



$$G_{g+1,n} = G_{g,n} + s_{g+1,n} x \qquad x_t \propto N(0,1)$$

  - ✓ *of the strategy parameter – modify the standard deviation*

$$s_{g+1,n} = s_{g,n} e^{t x_t} \qquad x \propto N(0,1) \qquad t = \sqrt{I}$$

Mutated chromosome accepted only if it is fitter !

ES in HEP

ES (and GA) used mainly for large-scale optimisation problems

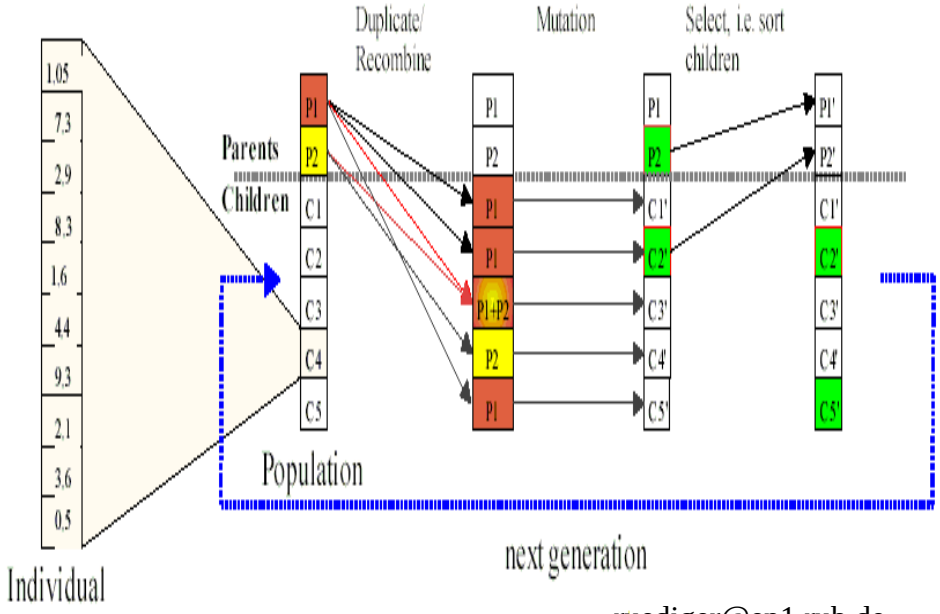
- ***event selection optimisation, NIM A534 (2004) 147***

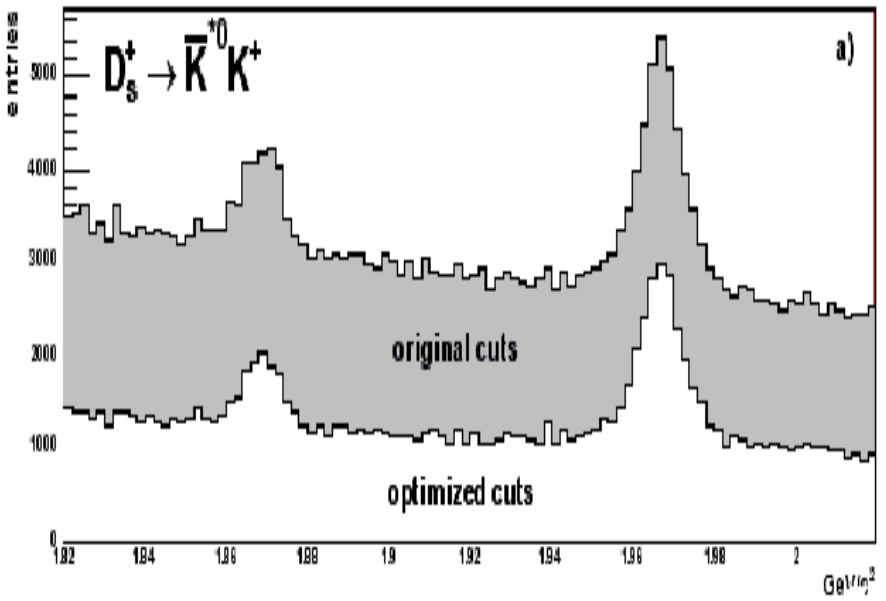
Chromosome: cut values

cos(q_H), p_{Ds} , mass constraint, vertex fit probability

Fitness function: sig²=S²/(S+2B)

45.4% improvement in sig²





ruediger@ep1.rub.de

Genetic Programming

GP search for the computer program to solve the problem, not for the solution to the problem.

Computer program - any computing language (in principle)
- LISP (List Processor) (in practice)

LISP - highly symbol-oriented

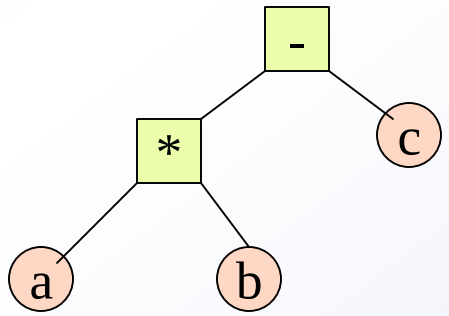
Mathematical expression

a*b-c

S-expression

(-(*ab)c)

Graphical representation of S-expression



functions (+,) and terminals (a,b,c)*

❖ Encoding

Chromosome: S-expression - variable length => more flexibility
- sintax constraints => invalid expressions
produced in the evolution process must be eliminated => waste of CPU

❖ Reproduction

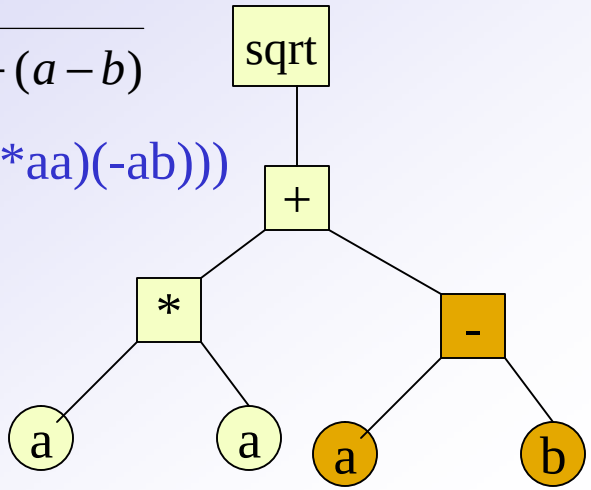
Cross-over (recombination) and Mutation (usualy)

Cross-over operator

Parents

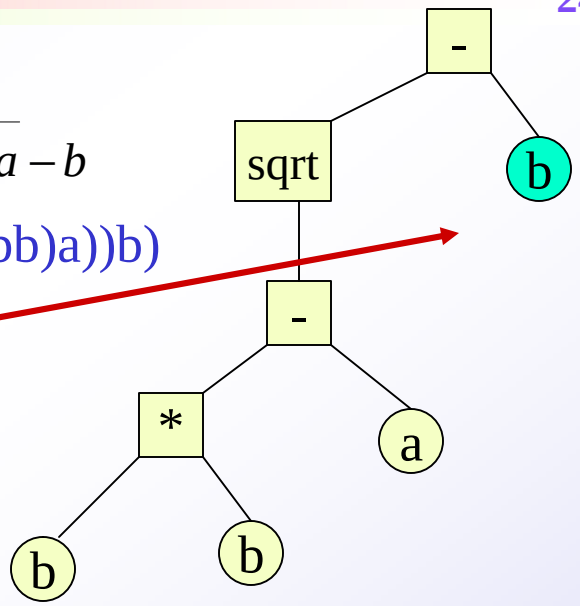
$\sqrt{a^2+(a-b)}$

(sqrt(+(*aa)(-ab)))



$\sqrt{b^2-a}-b$

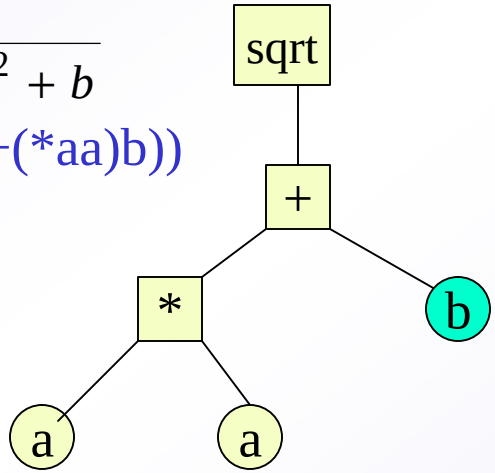
(-(sqrt(-(*bb)a))b)



Children

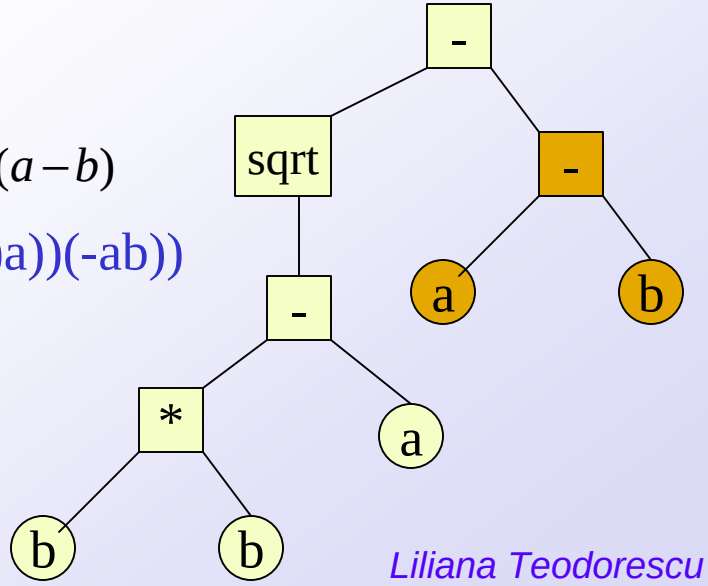
$\sqrt{a^2+b}$

(sqrt(+(*aa)b))



$\sqrt{b^2-a}-(a-b)$

(-sqrt(-(*bb)a))(-ab))



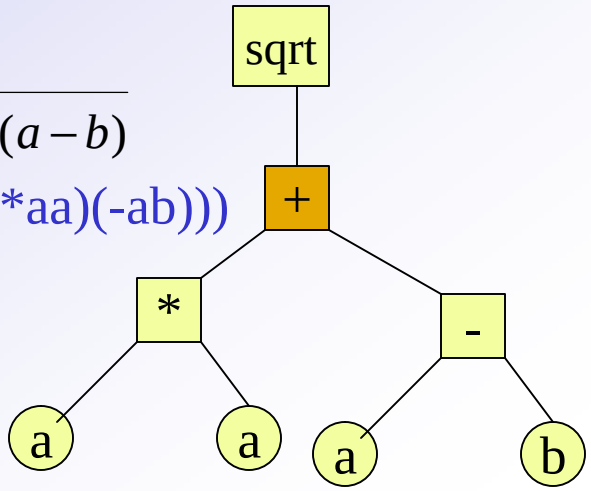
Mutation operator

- *function* replaced by another *function*
- *terminal* replaced by another *terminal*

Parents

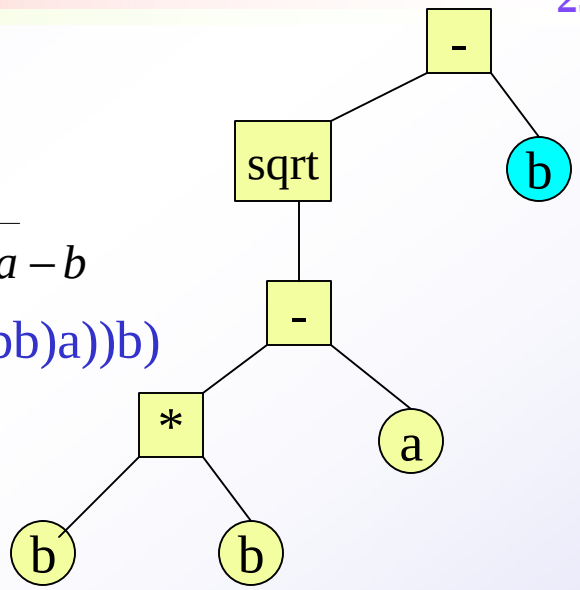
$\sqrt{a^2+(a-b)}$

(sqrt(+(*aa)(-ab)))



$\sqrt{b^2-a}-b$

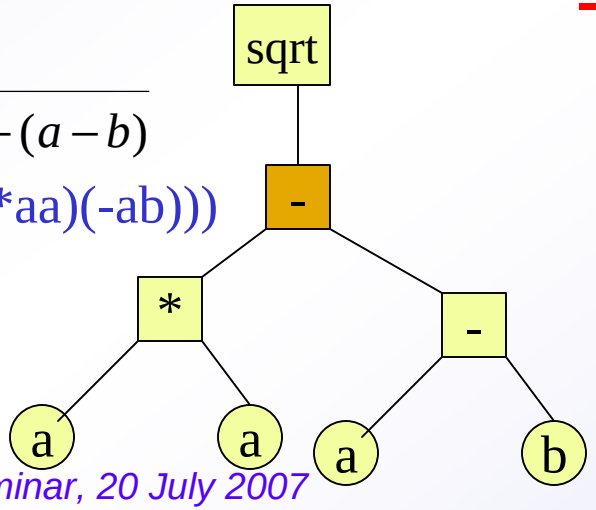
(-(sqrt(-(*bb)a))b)



Children

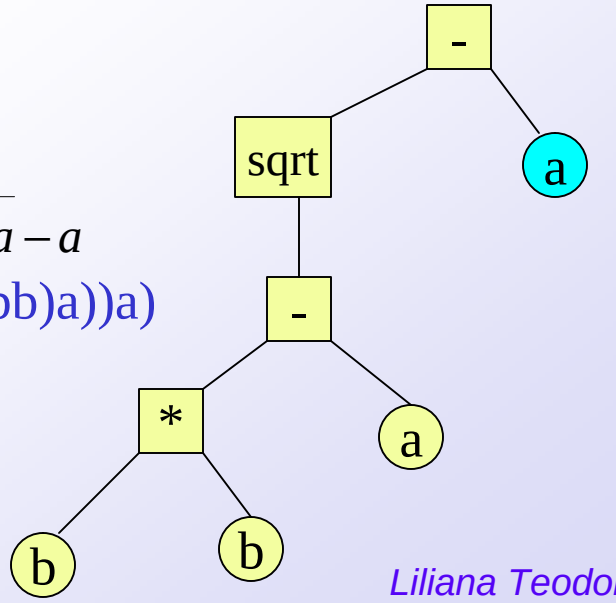
$\sqrt{a^2-(a-b)}$

(sqrt(-(*aa)(-ab)))



$\sqrt{b^2-a}-a$

(-sqrt(-(*bb)a))a)



GP in HEP

Experimental HEP - event selection

- Higgs search in ATLAS (physics/0402030)
- D, D_s and L_c decays in FOCUS (hep-ex/0503007, hep-ex/0507103)

e.g. Search for $D^+ \to K^+ p^+ p^-$ (hep-ex/0503007)

Chromosome: candidate cuts - tree of:

- functions: mathematical functions and operators, boolean operators
- variables: vertexing variables, kinematical variables, PID variables
- constants: reals (-2,2), integers (-10,+10)

In total: 55

Fitness function (will be minimised)

$$\frac{S+B}{S^2} \times 10000(1+0.005\times n)$$

n- number of tree nodes

penalty based on the size of the tree

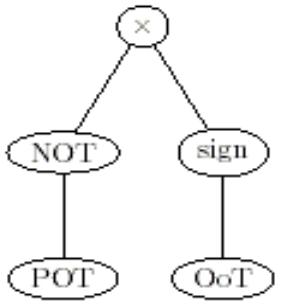
(big trees must make significant contribution to bkg reduction or signal increase)

GP in HEP (cont.)

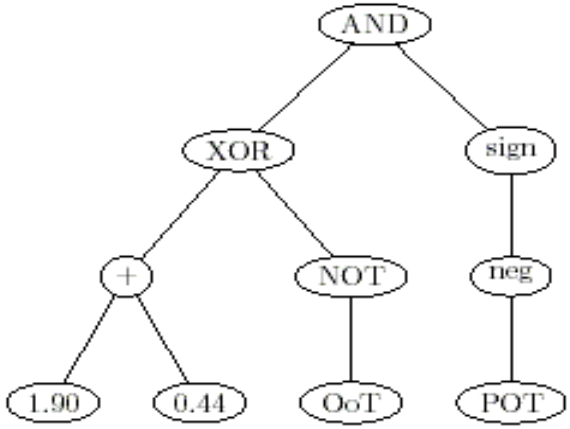
Basic procedure:

1. *Generates (almost randomly) a population of chromosomes*
2. *Loop over events and calculate the fitness for each chromosome*
   - *loop over each event and keep events where the tree evaluates to > 0*
   - *for survival events, fit signal (S) and bkg. (B)*
   - *calculate fitness of each chromosome*
3. *Select chromosomes, apply genetic operators and create the next generation*
4. *Repeat for the desired number of generations (40)*

Best fitted chromosomes from generation 0



(a) Fitness 0.458

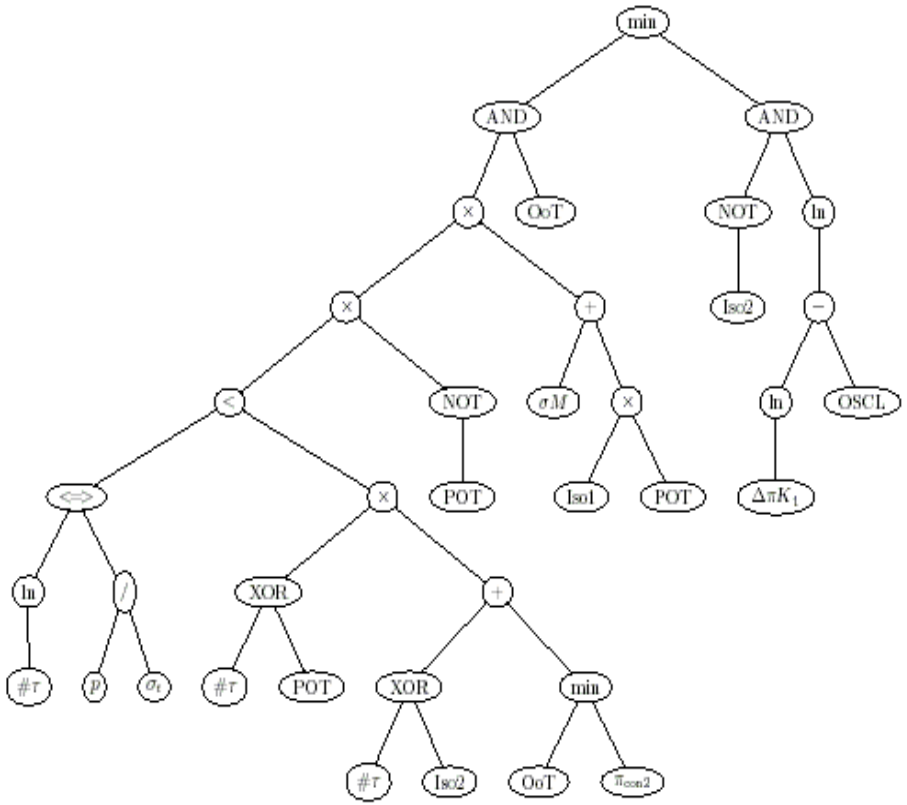


(b) Fitness 0.469

Inter point in target (POT<0)
and
Decay vertex out of target (OoT>0)

GP in HEP (cont.)

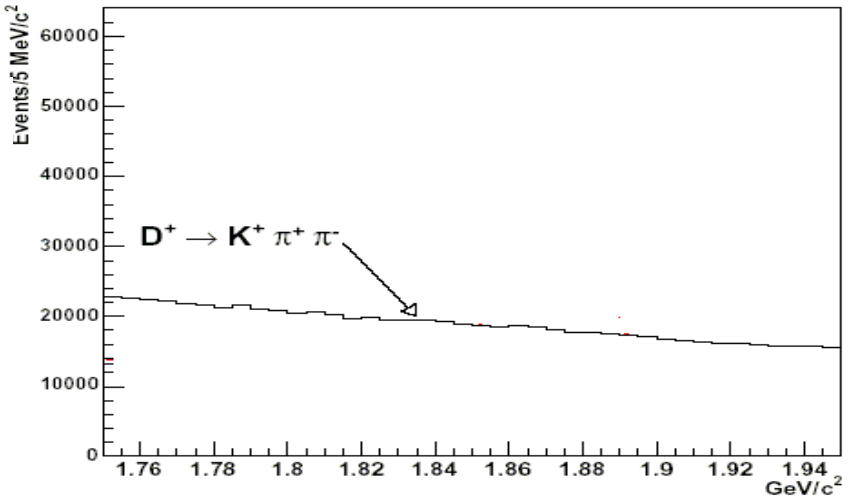
Best candidate, after 40 generations = final selection criteria



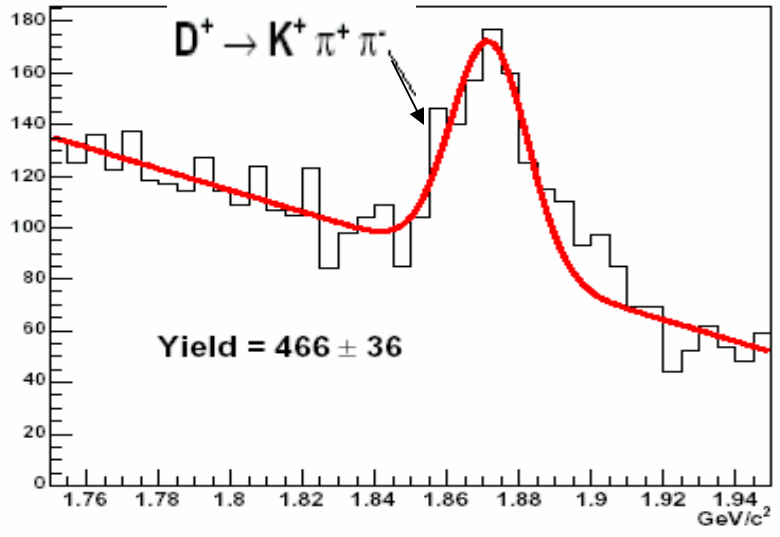
(a) Most fit tree: fitness 0.1234

Skim criteria

Initial selection

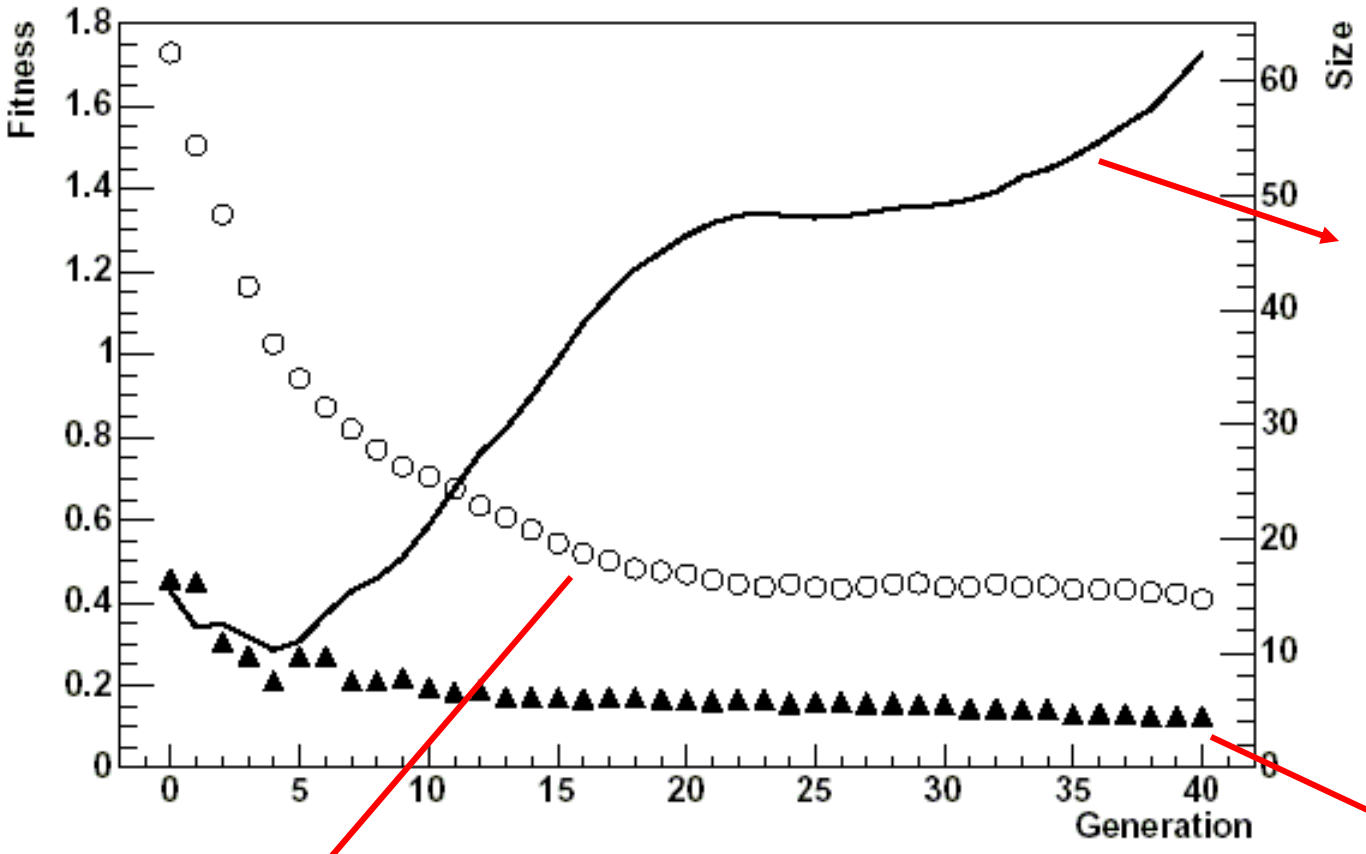


Final selection



GP in HEP (cont.)

Evolution graph



average size of the individuals

Fitness of the best individual

Average fitness of the population

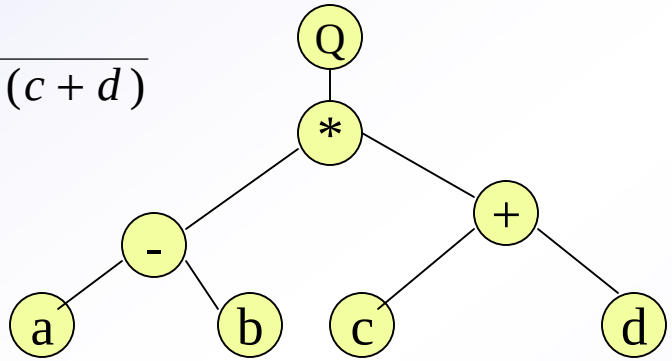
Gene Expression Programming

- ***search for the computer program that solve the problem (as GP)***
- ***works with two entities: chromosomes and expression trees***

Solution representation

Candidate solution represented by an expression tree (ET)
(similar with GP tree)

$$\sqrt{(a-b)\cdot(c+d)}$$



ET encoded in a chromosome:
read ET from left to right
and from top to bottom

Q*-+abcd
Q means sqrt

Decoding the chromosome (translates the chromosome in an ET)
- ✓***first line of ET (root) – first element of the chromosome***
- ✓***next line of ET – as many arguments needed by the element in the previous line***

GEP (cont.)

***Chromosome* – has one or more genes of equal length**

Gene* – *head:* contains both functions and terminals *(length h)
- tail:* contains only terminals *(length t)

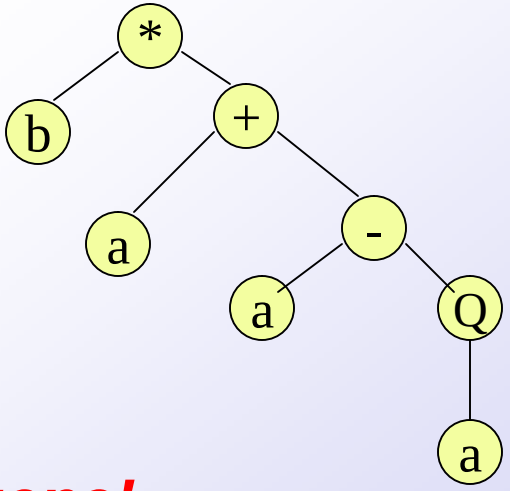
$t=h(n-1)+1$

n – number of arguments of the function with the highest number of arguments

e.g. set of functions: Q,,/,-,+*
set of terminals: a,b

n=2; h=15 (choosen) =>t =16 =>
length of gene=15+16=31

***b+a-aQab+//+b+babbabbbababbaaa**



ET ends before the end of the gene!

GEP (cont.)

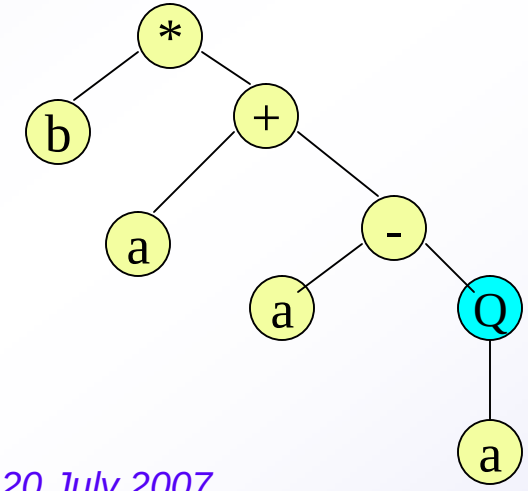
Reproduction

Genetic operators applied on chromosomes not on ET =>
always produce sintactically correct structures!

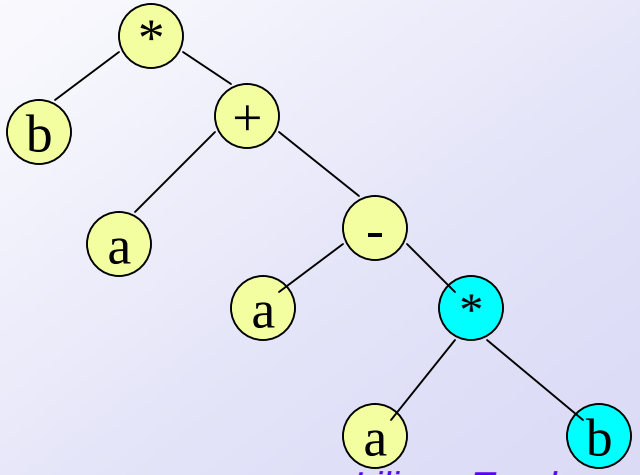
- ***Cross-over – exchanges parts of two chromosomes***
- ***Mutation – changes the value of a node***
- ***Transposition – moves a part of a chromosome to another location in the same chromosome***

*e.g. Mutation: Q replaced with **

***b+a-aQab+//+b+babbabbbababbaaa**



***b+a-a*ab+//+b+babbabbbababbaaa**



GEP in HEP

GEP for event selection

L. Teodorescu, IEEE Trans. Nucl. Phys., vol. 53, no.4, p. 2221 (2006)
also talks at CHEP06 and ACAT 2007

- *cuts/selection criteria finding*
- ***classification problem (signal/background classification)***
- *statistical learning approach*

Data samples:

- *Monte-Carlo simulation from BaBar experiment*
- *Ks production in e^+e^- (~10 GeV)*
- *8 or 20 event variables used in a standard analysis for* $K_S \rightarrow p^+p^-$

Functions and constants to be used in the classification rules

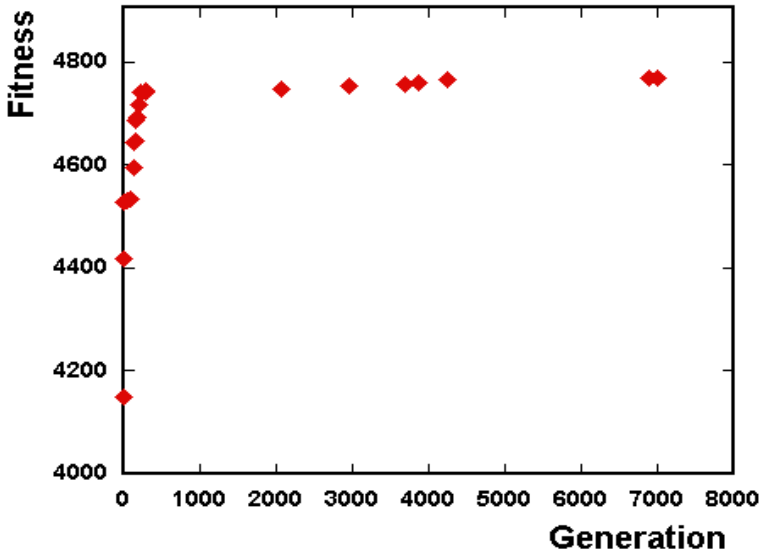
- *18 functions – logical functions => cut type rules*
- *38 functions - common mathematical functions*
- *constants - floating point constants (-10,10)*

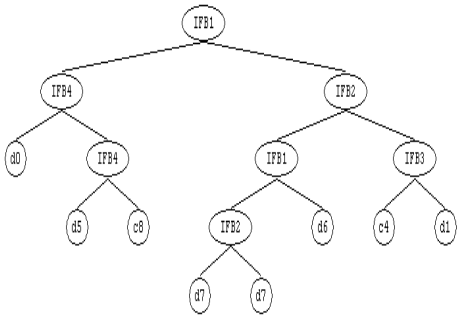
Fitness function – number of events correctly classified as signal or bkg. (maximise classification accuracy)

Model evolution

Data sample: S/N =0.25; 18 functions, 5000 events

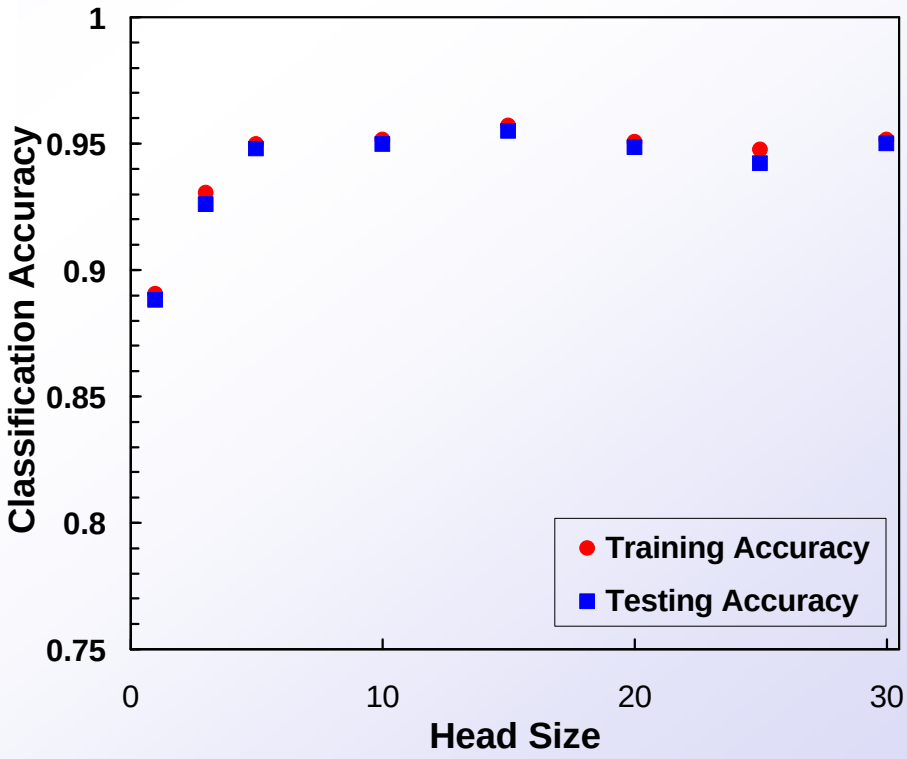
No. of genes = 1, Head length =10





Fsig ³ 5.26,
Rxy < 0.19,
doca <1,
Pchi > 0

Model complexity



Classification Accuracy = 95%

Classification rules

GEP analysis – optimises classification accuracy

Data sample: S/N =0.25, 18 functions, 5000 events

Head	Selection criteria
1	Fsig ³ 9.93
2	Fsig ³ 8.80, doca <1
3	Fsig > 3.67, Rxy £ Pchi
4	Fsig > 3.67, Rxy £ Pchi
5	Fsig ³ 3.63, |Rz| £ 2.65, Rxy < Pchi
7	Fsig ³ 3.64, Rxy < Pchi, Pchi > 0
10	Fsig ³ 5.26, Rxy < 0.19, doca <1, Pchi > 0
20	Fsig > 4.1, Rxy £ 0.2, SFL > 0.2, Pchi > 0, doca > 0, Rxy £ Mass

Cut-based (standard) analysis – optimises signal significance

Fsig ³ 4.0
Rxy £ 0.2cm
SFL ³ 0cm
Pchi > 0.001

Reduction
S: 15%
B: 98%

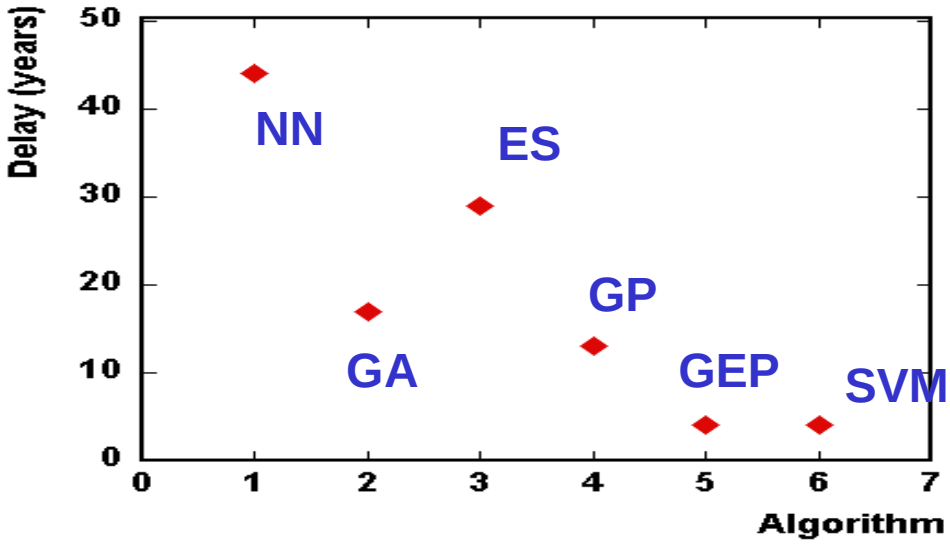
doca £ 0.4cm
|Rz| £ 2.8cm

Reduction
S: 16%
B: 98.3%

Conclusions - final remarks

***Evolutionary algorithms** in HE Physics & Computing*

- *used but not extensively at present*
- *good performance – optimal solutions*
- *main desadvantage – high computational time*
- *prospects for changes – new, faster algorithms, more computing power*



Conclusions - final remarks

Used/developed by who ? … Your colleague !!

Yellow Report (this summer) – lectures from iCSC

Computational Intelligence in HEP

- ***Statistical learning – Anselm Vossen***
- ***Machine learning – Jarek Przybyszewski***
- ***Support Vector Machine – Anselm Vossen***
- ***Neural Networks - Liliana Teodorescu***
- ***Evolutionary Algorithms – Liliana Teodorescu***
- ***Data Mining – Petr Olmer***

Computing topics

- ***Parallel Programming – Marek Biskup***
- ***Database performance pitfalls – Michal Kwiatek***
- ***Debugging techniques – Paolo Adragna***
- ***Code review – Gerhard Brandt***