ELTE | FACULTY OF INFORMATICS

Bacterial programming and its applications

János Botzheim

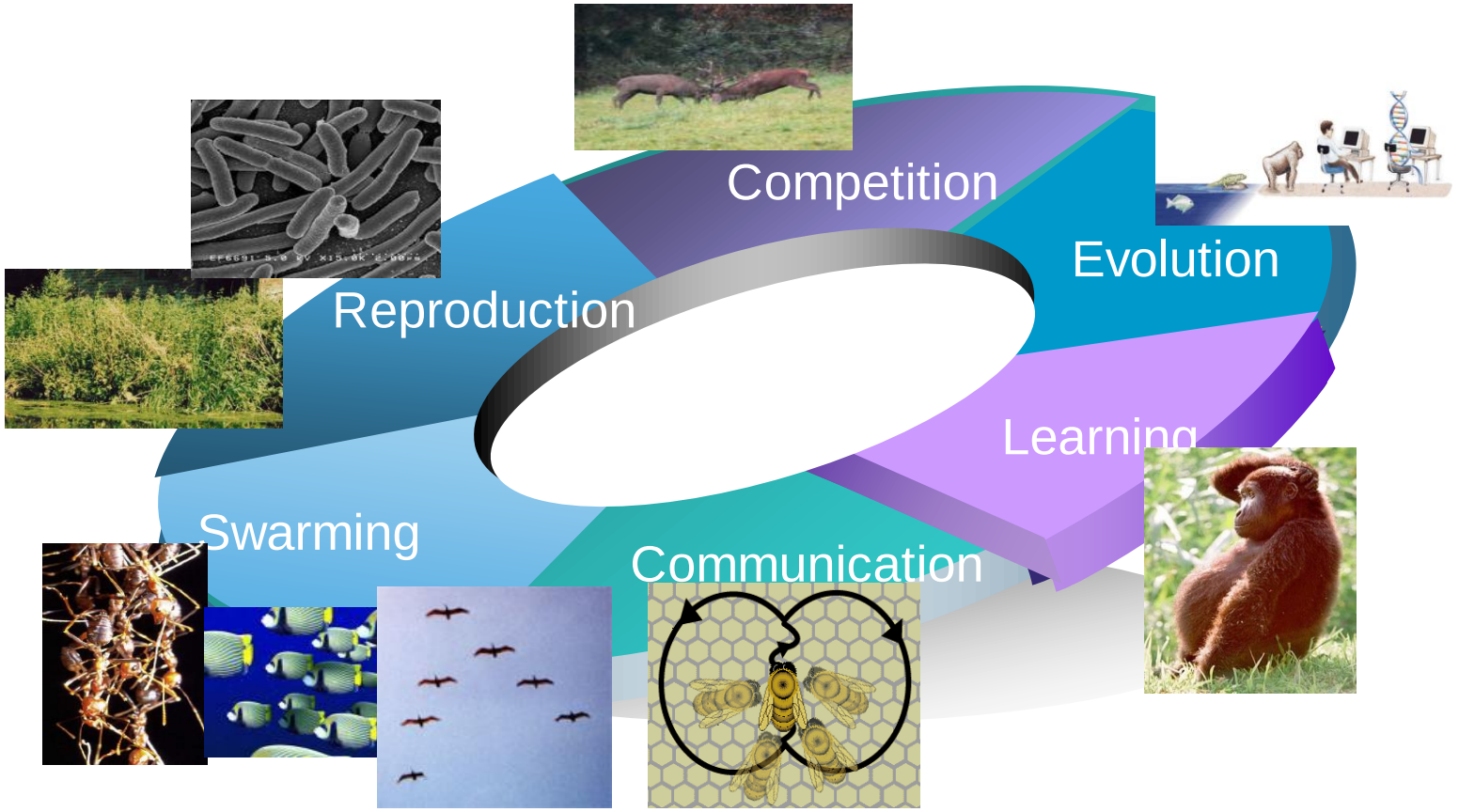
Eötvös Loránd University, Faculty of Informatics

Department of Artificial Intelligence

ELTE | IK

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Elements of intelligence in biological systems



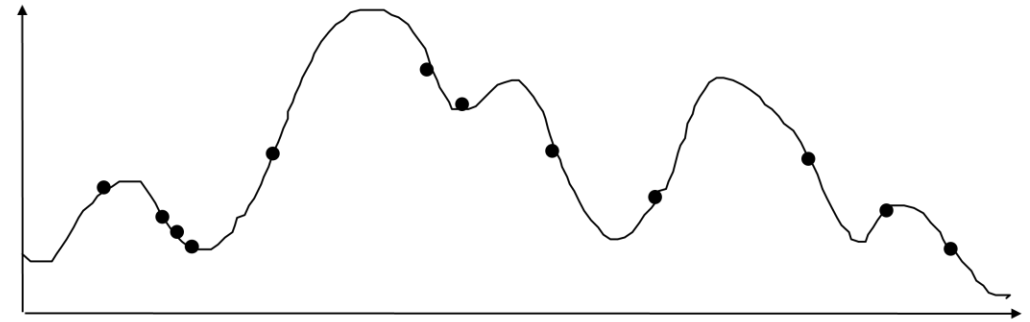
Learning, evolution, adaptation

- Learning:
  - Changes will be made after the evaluations
  - Learning is the purpose of change
- Evolution:
  - Evaluations happen after the changes
  - Evolution is the result of change
- Adaptation:
  - Changes occur after evaluations or as a result of adaptation to environmental conditions

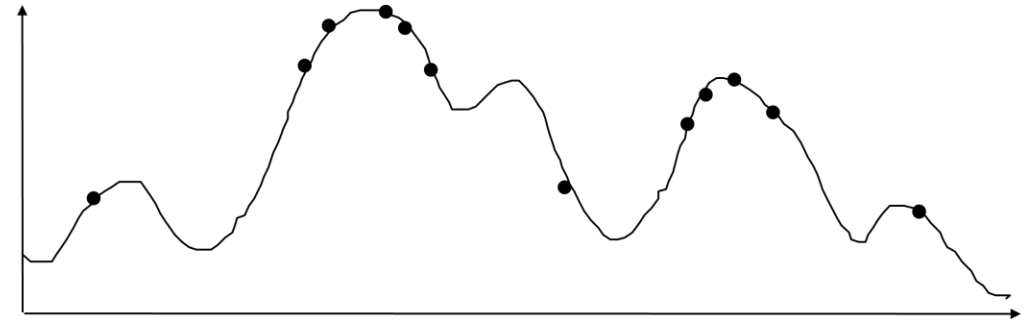
Evolutionary algorithms

- Their basic principle is the search on the population of solutions guided by laws known from biology
- The individuals in the population are the solutions of the given problem
- The population is evolving, we obtain better and better individuals

Evolution of the population



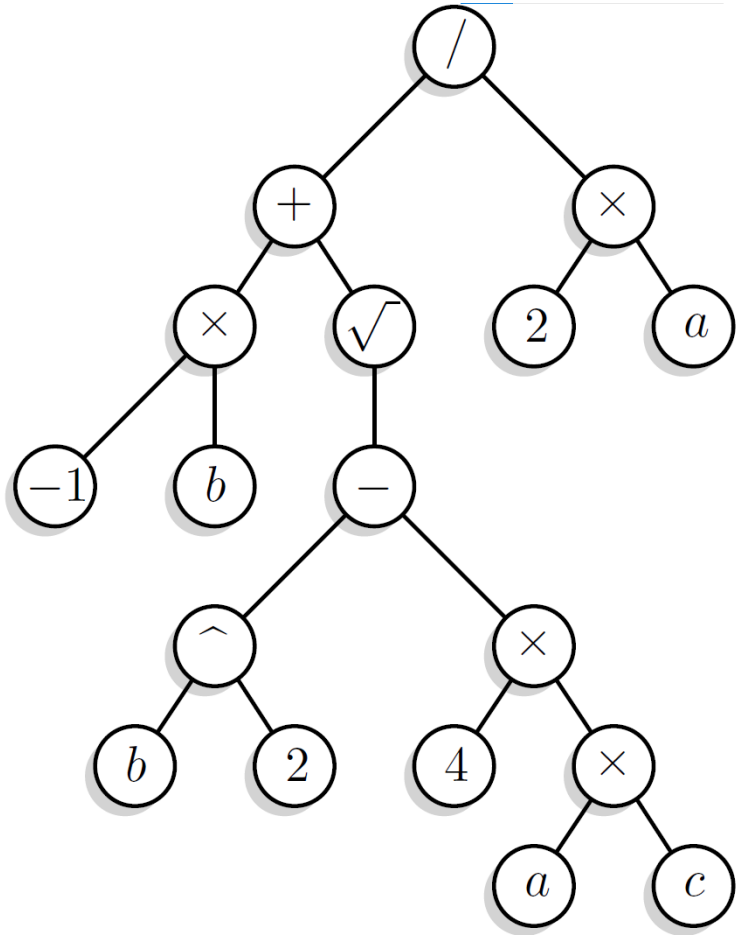
Distribution of Individuals in Generation 0



Distribution of Individuals in Generation N

Genetic programming

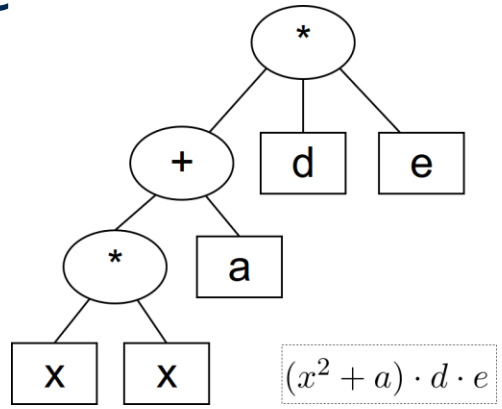
- John Koza (~1990)
- Genetic programming applies the approach of genetic algorithm to the space of possible computer programs
- A wide variety of seemingly different problems from many different fields can be reformulated as a search for a computer program to solve the problem
- Individuals are described by an expression tree



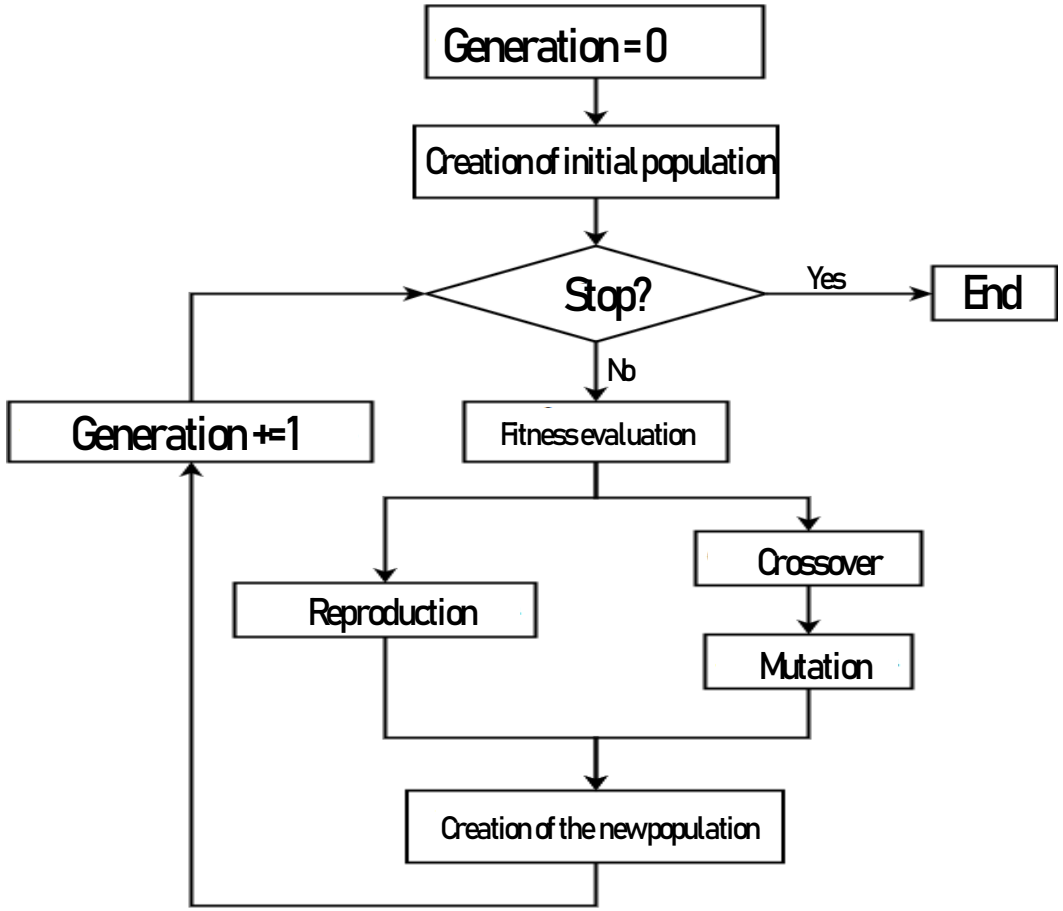
$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

Genetic programming

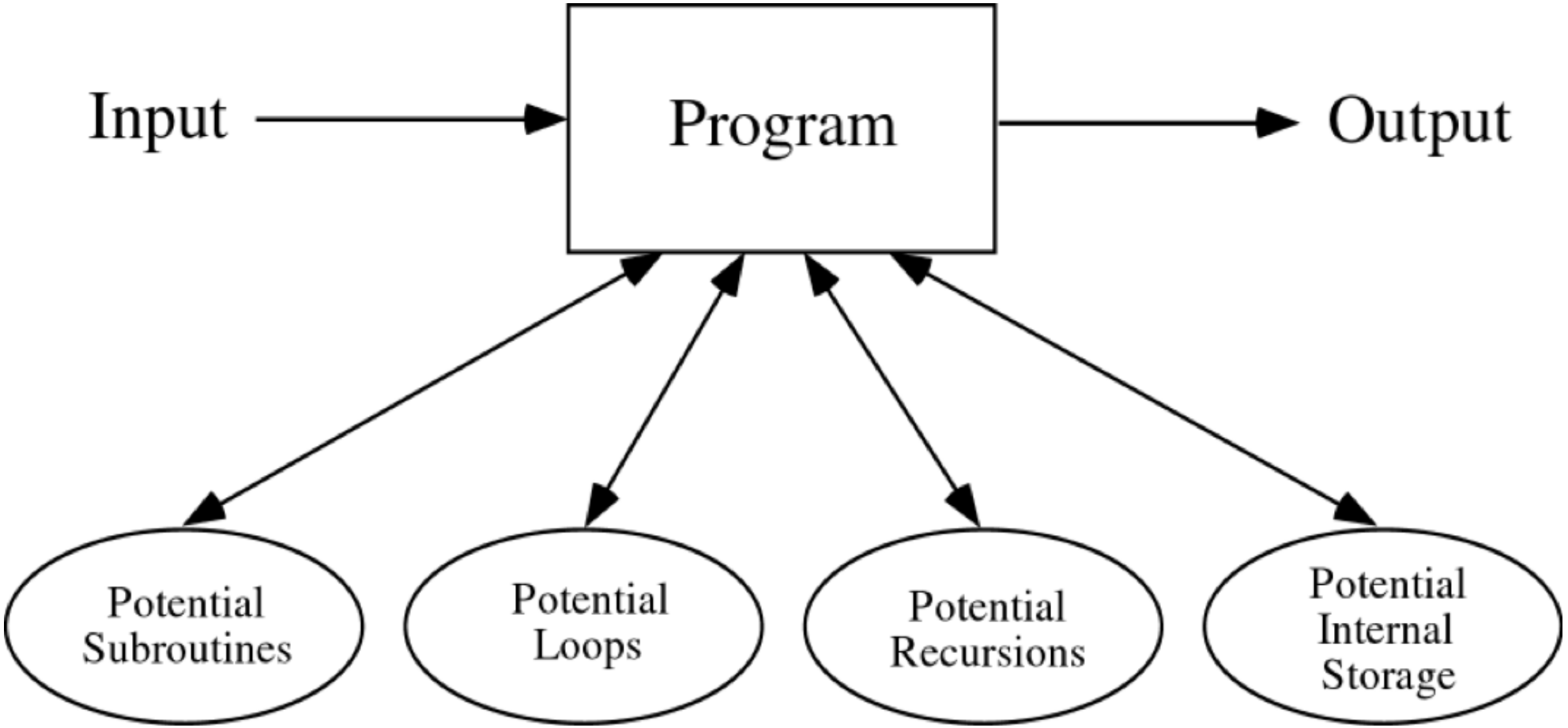
- The individual is a tree-based structure



- Evolutionary operators
  - Reproduction
  - Crossover
  - Mutation

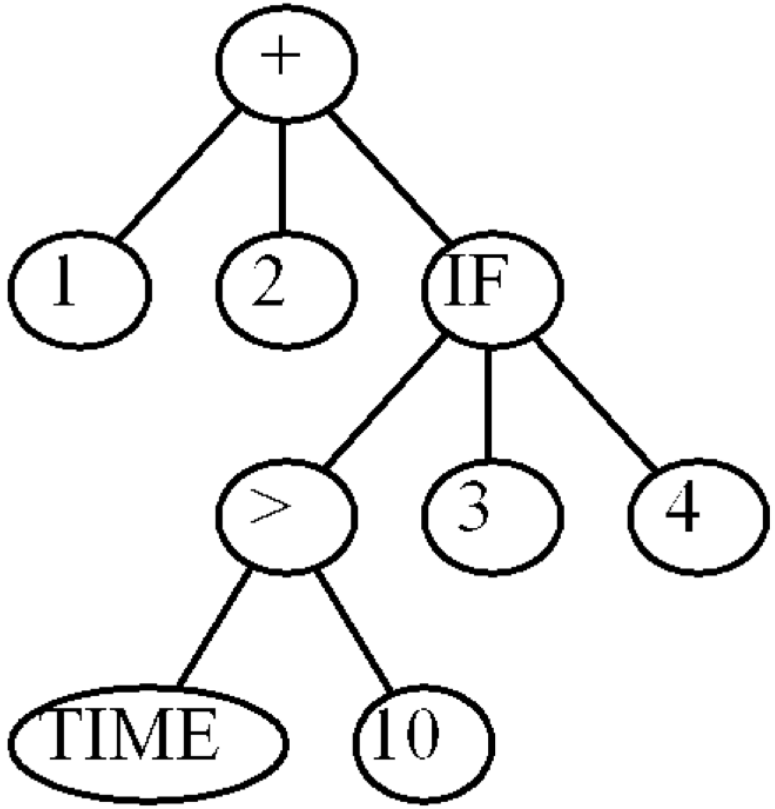


A computer program



A computer program in C and its tree representation

```
int foo (int time)
{
    int temp1, temp2;
    if (time > 10)
        temp1 = 3;
    else
        temp1 = 4;
    temp2 = temp1 + 1 + 2;
    return (temp2);
}
```



(+ 1 2 (IF (> TIME 10) 3 4))

Creating random programs

- Available functions:
  - e.g. F = {+, -, *, %, IF}
- Available terminals:
  - e.g. T = {X, Y, constants}
- The random programs are:
  - syntactically valid
  - executable
- The trees may have different sizes and shapes

Genetic operators in GP

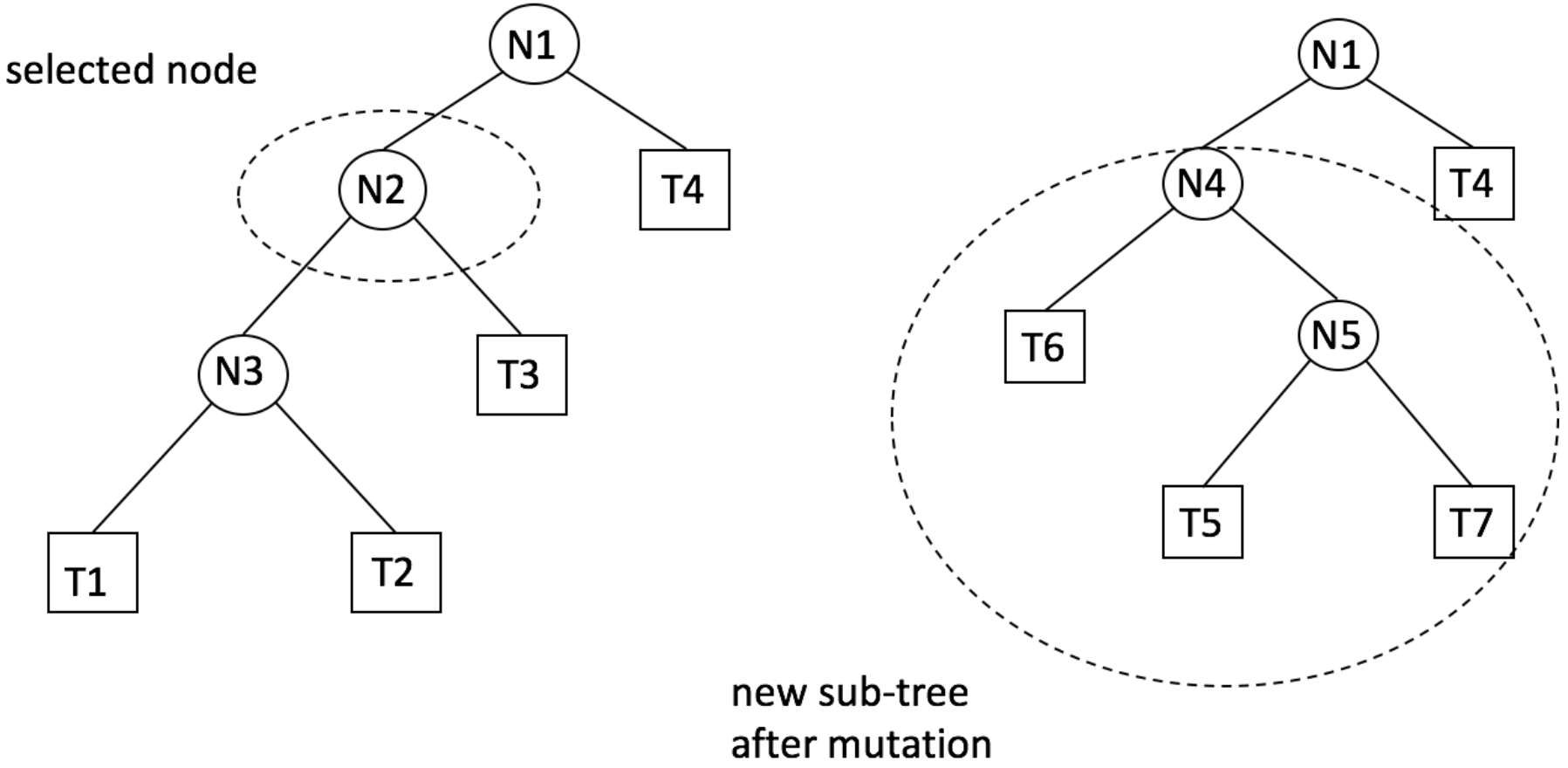
- Reproduction
- Mutation
- Crossover

- Reproduction:
  - select parent based on fitness
  - copy it (unchanged) into the next generation of the population

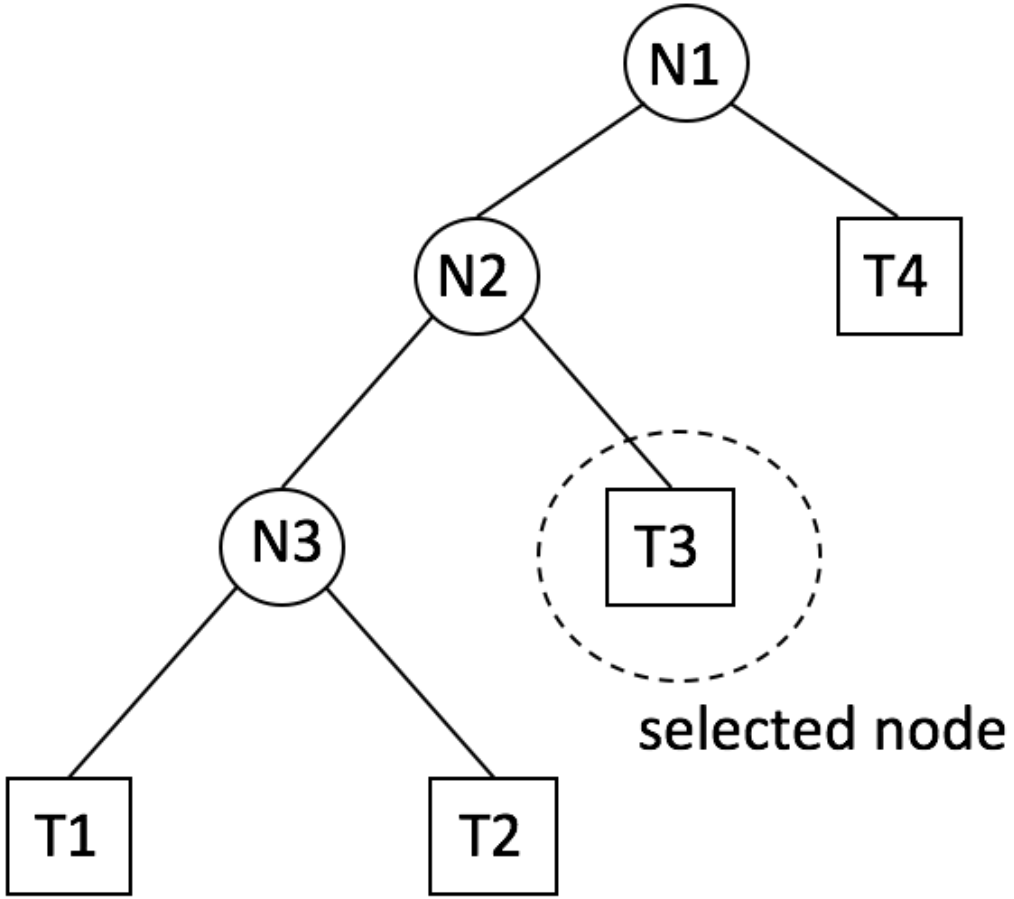
Mutation

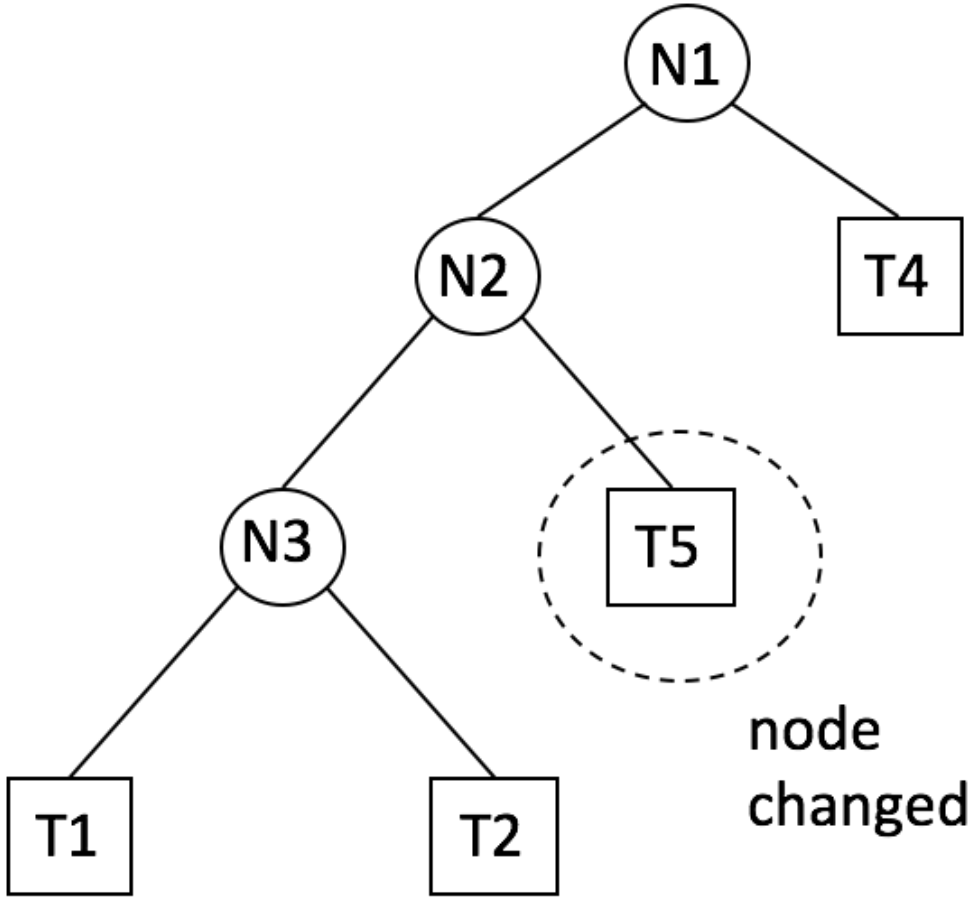
- Select 1 parent (based on fitness)
- Pick a point in the tree
- Delete subtree at the picked point
- Grow new subtree at the mutation point in same way as generated trees for initial random population
- The result is a syntactically valid executable program
- Put the offspring into the next generation of the population

Subtree mutation



Node mutation

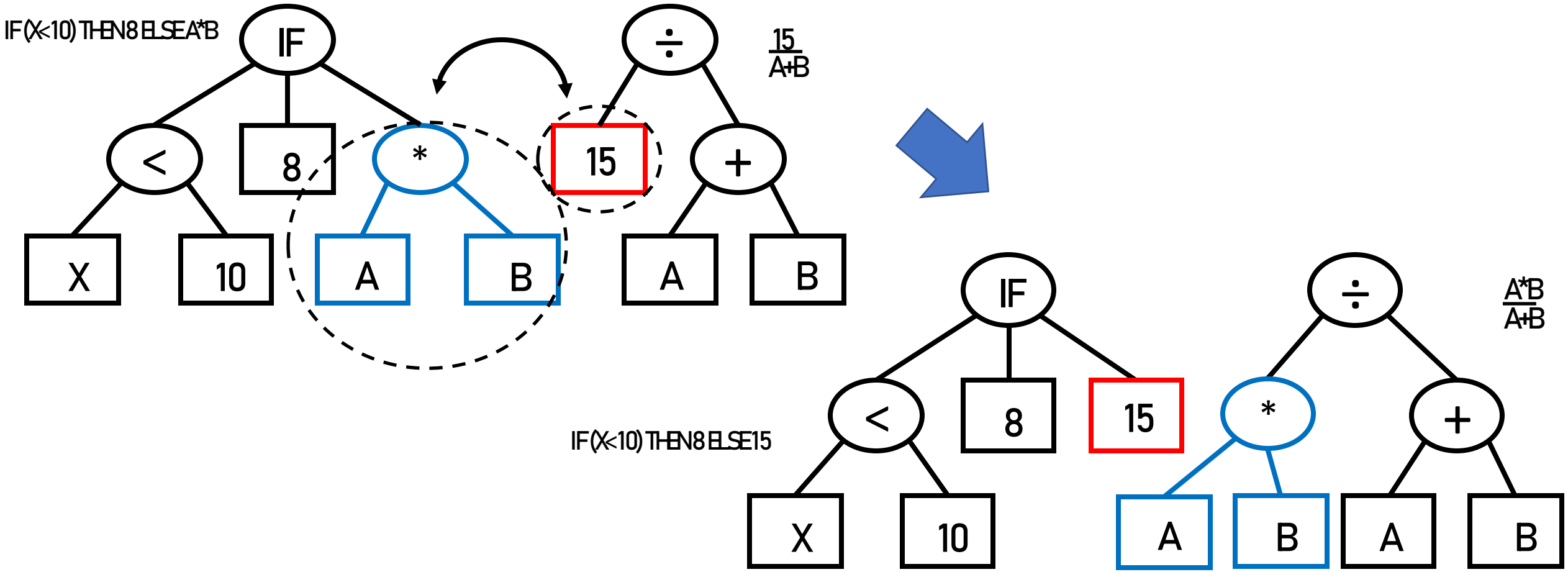




Crossover

- Select 2 parents (based on fitness)
- Randomly pick a node in the tree for first parent
- Independently randomly pick a node for second parent
- Exchange the subtrees at the two picked points
- The result is a syntactically valid executable program
- Put the offspring into the next generation of the population

Crossover

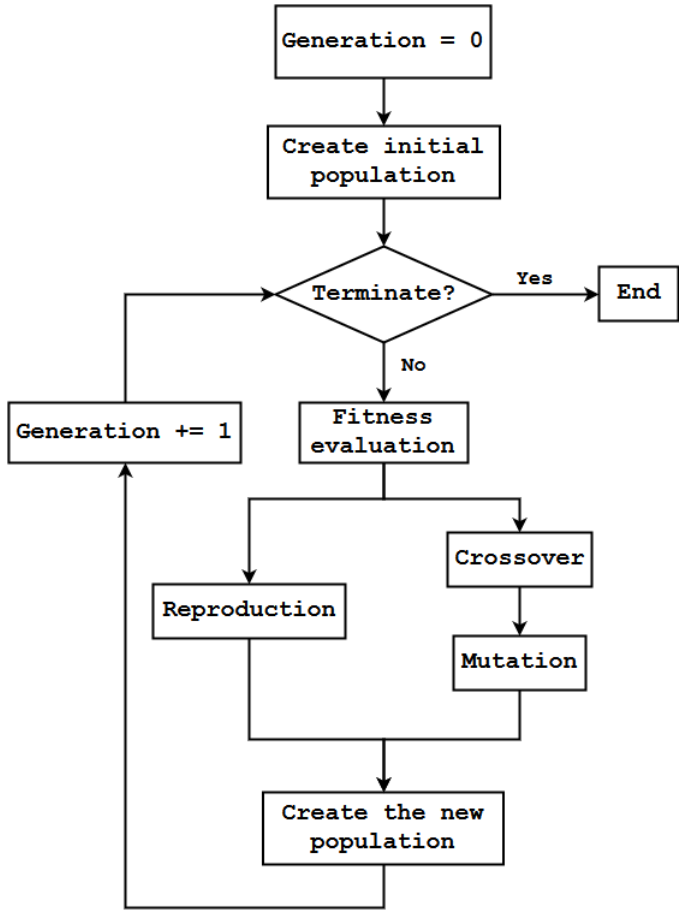

IF(X<10)THEN8ELSEA*B
IF
<
8
*
÷
15
+
X
10
A
B
A
B
15
A+B
IF(X<10)THEN8ELSE15
IF
<
8
15
X
10
÷
*
+
A
B
A
B
A*B
A+B

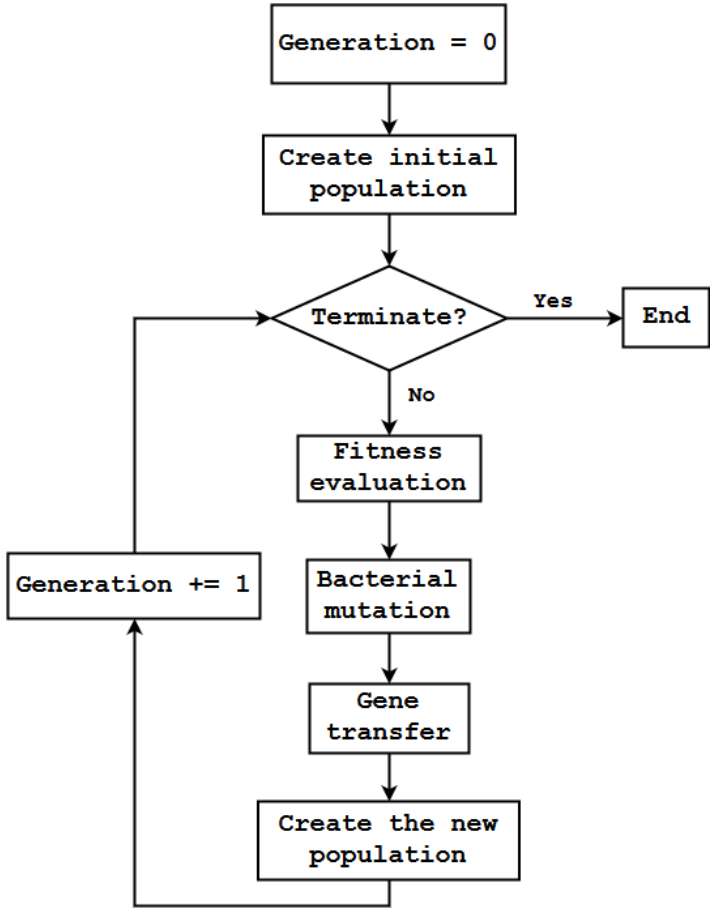
Preparatory steps

- Determining the set of terminals
- Determining the set of functions
- Determining the fitness measure
- Determining the parameters for the run
- Determining the criterion for terminating a run

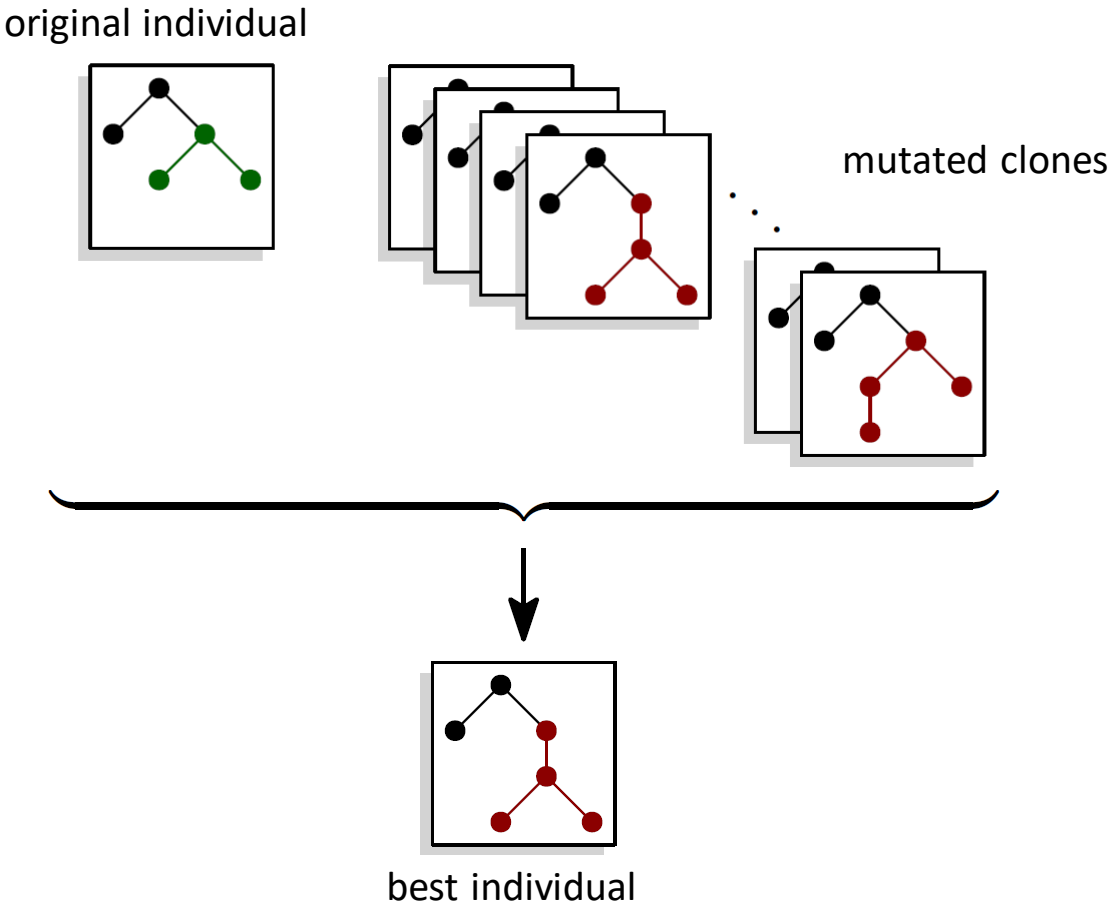
Genetic programming & Bacterial programming

- Evolutionary operators in GP:
  - Reproduction
  - Mutation
  - Crossover
- Evolutionary operators in BP:
  - Bacterial mutation
  - Gene transfer

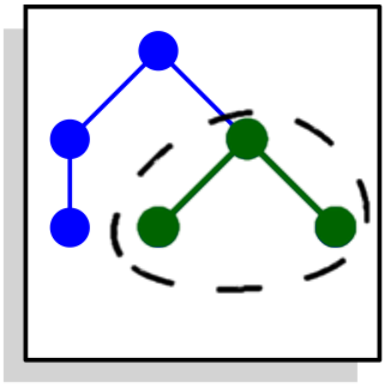




Bacterial mutation

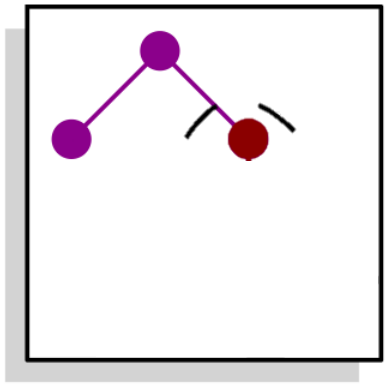


Gene transfer



Donor

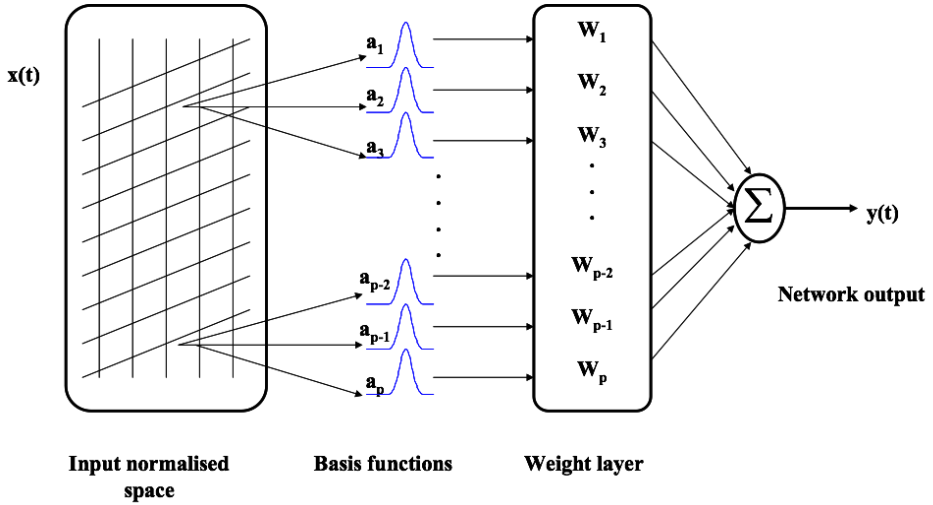
+



Acceptor

B-spline neural networks

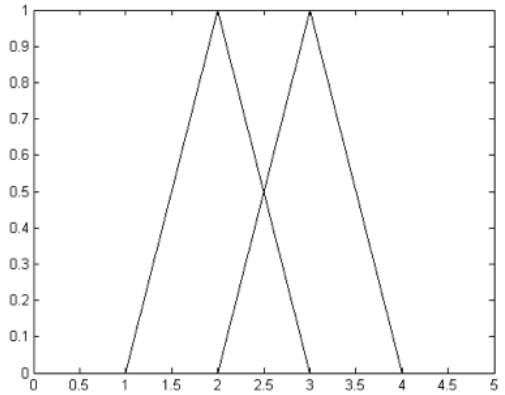
- Basic Structure:



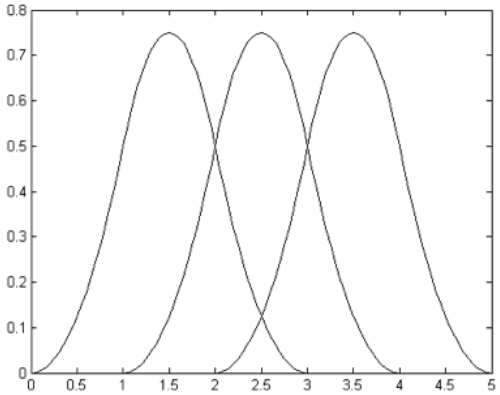
- Normalized Input Layer
  - Grid defined by vectors of knots: $x_i^{\min} < \lambda_{i,1} \leq \lambda_{i,2} \leq \cdots \leq \lambda_{i,r_i} < x_i^{\max}$
  - Total number of cells in the grid: $p' = \prod_{i=1}^{n} (r_i + 1)$

B-spline neural networks

- Basis Functions Layer
  - In BSNNs, the order of the spline sets the shape and the support of the basis functions



k=2



k=3

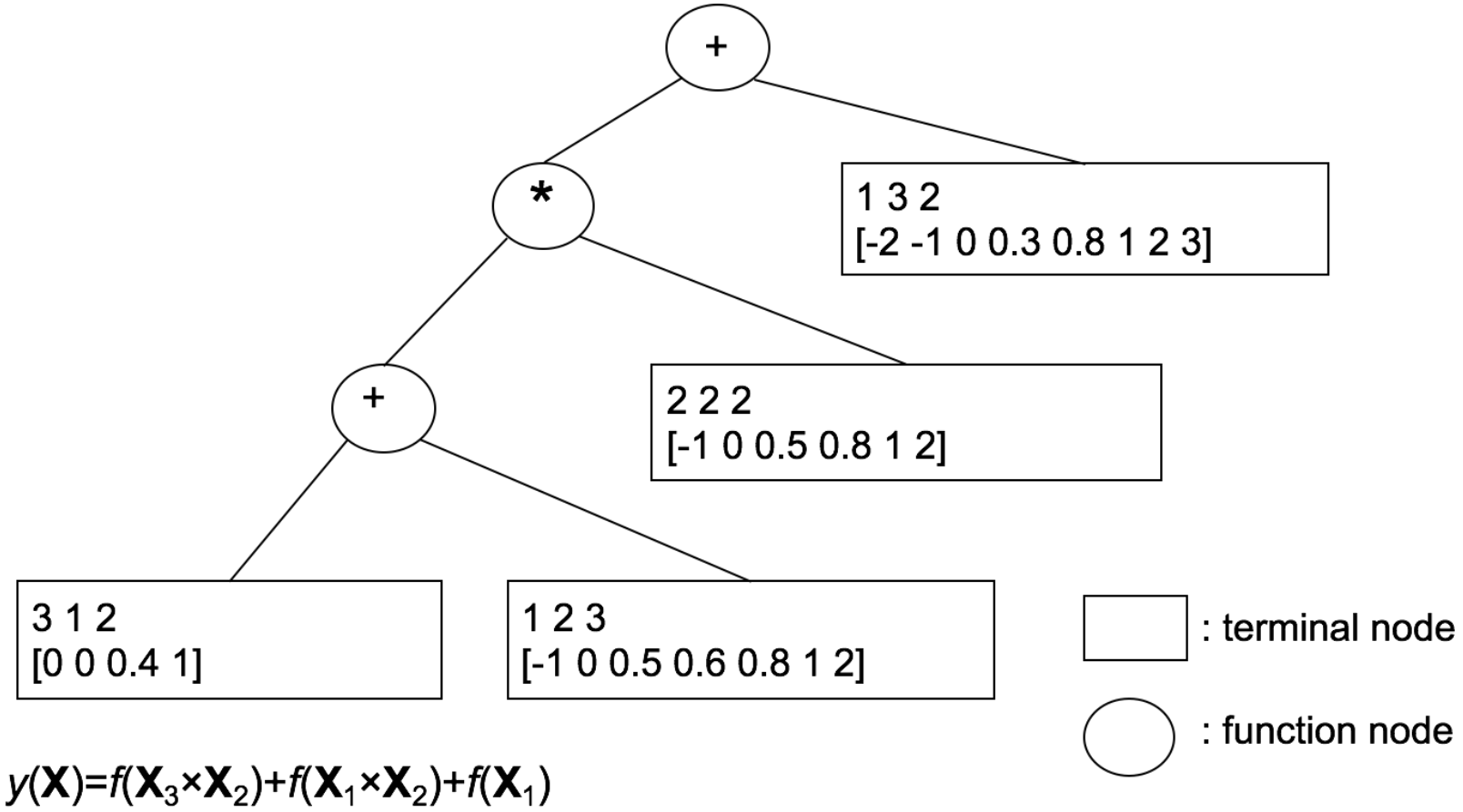
$$N_k^j(x) = \left(\frac{x - \boldsymbol{\lambda}_{j-k}}{\boldsymbol{\lambda}_{j-1} - \boldsymbol{\lambda}_{j-k}} \right) N_{k-1}^{j-1}(x) + \left(\frac{\boldsymbol{\lambda}_j - x}{\boldsymbol{\lambda}_j - \boldsymbol{\lambda}_{j-k+1}} \right) N_{k-1}^j(x)$$

$$N_1^j(x) = \begin{cases} 1 & if\ x \in I_j \\ 0 & otherwise \end{cases}$$

B-spline neural networks design

- Determination of the number of its sub-modules
- For each sub-model, the set of its inputs
- Order of the splines for each input
- Number of the interior knots for each input
- Location of the interior knots for each input
- Values of the linear output weights

A sample expression tree for B-spline networks



$y(\mathbf{X})=f(\mathbf{X}_3\times\mathbf{X}_2)+f(\mathbf{X}_1\times\mathbf{X}_2)+f(\mathbf{X}_1)$

Nodes

- Function nodes:
  - „+”: sub-modules must be *added*
  - „*”: sub-modules must be *created* from smaller sub-modules
  - „/”: sub-modules must be split into lower dimensional sub-modules
- Terminal node:
  - input variable
  - spline order
  - number of interior knots
  - location of knots

Terminal mutation

- Terminal mutation can be one of 6 different types:
  1. Full replacement of the terminal
  2. Variable identification replacement
  3. Splines order replacement
  4. Random displacement of an interior knot
  5. Addition of *N* interior knots placed randomly
  6. Removal of *N* interior knots
- The terminal mutation rate is described as a vector p_{term_mut}=[%1 %2 %3 %4 %5 %6], where %i designates the i^{th} type mutation rate

Simulation result

- A 6 dimensional problem (Nawa and Furuhashi, 1999) is to approximate the following function:

$$f_{6dim} = x_1 + \sqrt{x_2} + x_3 x_4 + 2e^{2(x_5 - x_6)}$$

$$x_1, x_2 \in [1,5], \quad x_3 \in [0,4], \quad x_4 \in [0,0.6], \quad x_5 \in [0,1], \quad x_6 \in [0,1.2]$$

- Error measures:

- *Mean Squared Error (MSE)*:

$$\frac{1}{m}\sum_{i=1}^{m}(d_i - y_i)^2$$

- *Mean Squared Relative Error (MSRE)*:

$$\frac{1}{m}\sum_{i=1}^{m}\frac{(d_i - y_i)^2}{y_i^2}$$

- *Mean Relative Error Percentage (MREP)*:

$$\frac{100}{m}\sum_{i=1}^{m}\left|\frac{d_i - y_i}{y_i}\right|$$

d_i: target output for the i^{th} pattern
y_i: model output for the i^{th} pattern
m: number of training samples
n: the model complexity (number of basis functions)
RMS: Root-Mean-Square

$$BIC = m\ln(RMS) + n\ln(m)$$

Simulation results

- Algorithms parameters definition:

Parameters	GP	BP
N_inf	-	4
N_clones	-	8
N_ind	80	10
N_gen	10	10
Crossover rate	50% of population	-
Mutation Rate	0.8	-

- The terminal type mutation rate is [5%, 10%, 5%, 10%, 60%, 10%]

Results

- Average of 10 runs for the 6-variable problem:

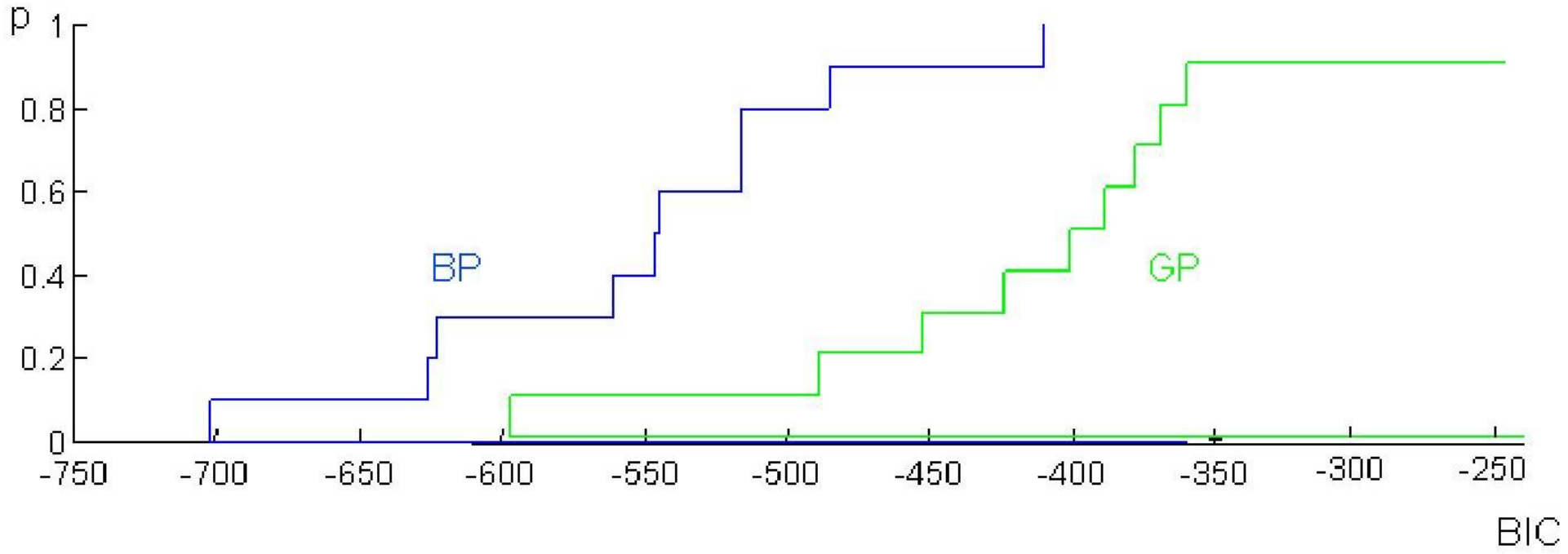
	GP	BP
BIC	$-380{,}1$	$-552{,}9$
MSE	$2{,}0 \cdot 10^{-1}$	$2{,}7 \cdot 10^{-2}$
MSRE	$2{,}1 \cdot 10^{-3}$	$5{,}2 \cdot 10^{-4}$
MREP	$2{,}1$	$0{,}98$
complexity	$38{,}8$	$44{,}6$

	GP	BP
submodels	$(5)(4)(2)$ $(3 \times 4)(5 \times 3 \times 6)$ (3×1)	$(6 \times 5)(5 \times 2)$ $(6 \times 1)(3 \times 6)$ $(3 \times 4)(1)$
complexity	98	156
BIC	$-593{,}2$	-702
MSE	$3{,}8 \cdot 10^{-3}$	$4{,}8 \cdot 10^{-4}$
MSRE	$7{,}3 \cdot 10^{-5}$	$1{,}2 \cdot 10^{-5}$
MREP	$6{,}6 \cdot 10^{-1}$	$2{,}4 \cdot 10^{-1}$
$\lvert W \rvert$	$423{,}8$	$128{,}4$

The best (lowest BIC) individual for the 6-variable problem:

Results from the simulation

- The best-run population empirical probability distribution function for the 6-variable problem:



Hierarchical fuzzy rule base

R_0: **If** z_0 is D_1 **then** use R_1
If z_0 is D_2 **then** use R_2
...
If z_0 is D_n **then** use R_n

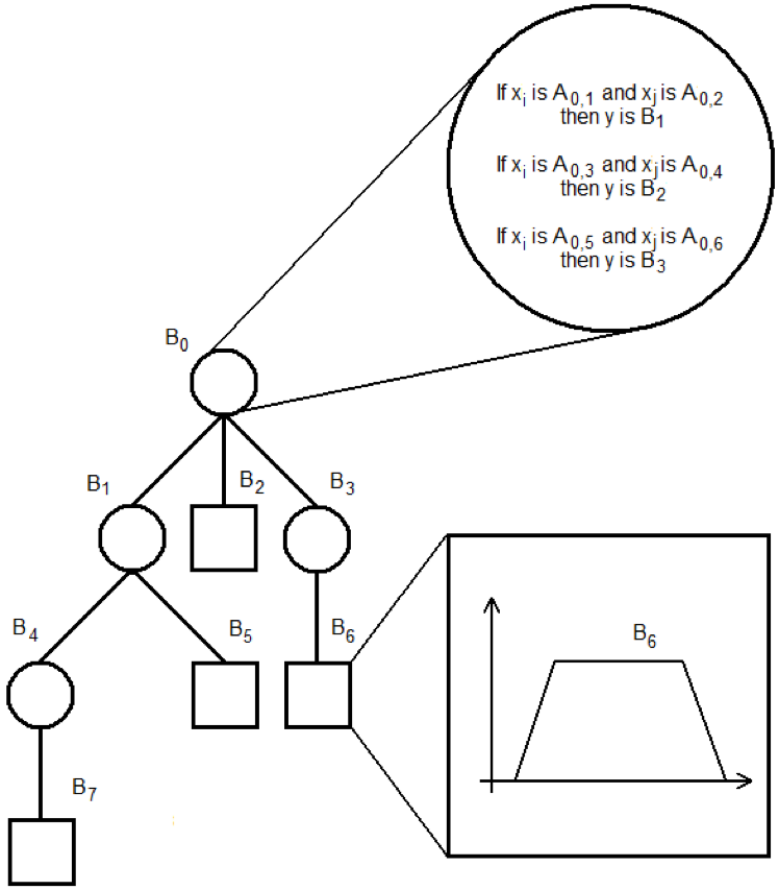
R_1: **If** z_1 is A_{11} **then** y is B_{11}
If z_1 is A_{12} **then** y is B_{12}
...
If z_1 is A_{1r_1} **then** y is B_{1r_1}

R_2: **If** z_2 is A_{21} **then** y is B_{21}
If z_2 is A_{22} **then** y is B_{22}
...
If z_2 is A_{2r_2} **then** y is B_{2r_2}

...

R_n: **If** z_n is A_{n1} **then** y is B_{n1}
If z_n is A_{n2} **then** y is B_{n2}
...
If z_n is A_{nr_n} **then** y is B_{nr_n}

Encoding method



Expression tree building restrictions

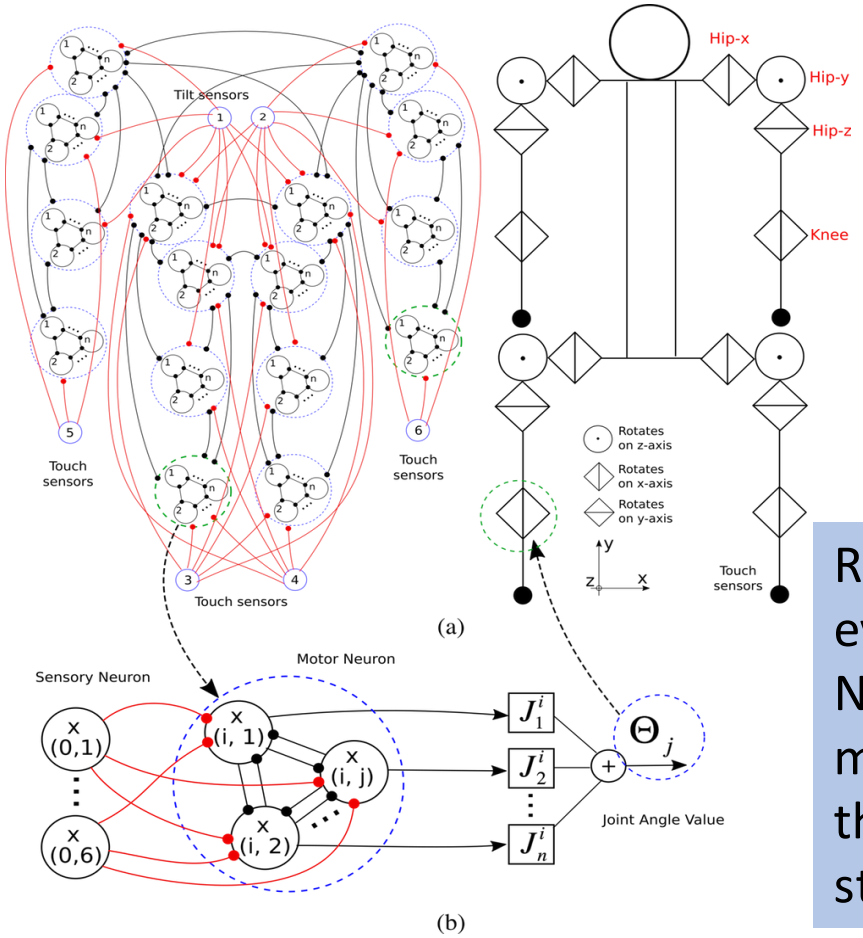
- The maximum number of decision variables can be limited (maximum number of input dimensions in the non-terminal nodes)
- The maximum number of rules can be limited (maximum number of children in the non-terminal nodes)
- Decreasing these values decreases the degrees of freedom of the rule base
  - Lower expression power
  - Smaller state space
  - This may result in a lower accuracy and higher convergence speed

Time boundary

RESULTS FOR THE SIX DIMENSIONAL LEARNING PROBLEM IN CASE OF THE 1000 SECONDS TIME LIMIT

	GProg	BProg
MSE	3.6465	1.0334
MSRE	0.0576	0.0157
MREP	18.0081	9.5196
No. of gen.	1172084	20183

Our previous model



Representation of the evolving neural model (a) Neural structure and mechanical structure of the robot. (b) Neuron structure in joint angle

$$\tau \dot{u}_i = \left(u_0 - u_i - \sum_{j=1}^{n} w_{ij} y_j J_j^k + \sum_{l=1}^{n} m_{il} s_l - b v_i \right) \tau_f$$

$$\tau' \dot{v}_i = (y_i - v_i) \tau_f$$

$$y_i = \max(u_i, 0)$$

$$\Theta_i^{(l,r)} = \sum_{j=1}^{N_i^{neuron}} J_j^{l,r} y_j$$

Inner state calculation

Notation	Definition
w_{ij}	motor neurons interconnection
m_{il}	sensory motor neurons interconnection
u_i	inner state
y_i	output value
v_i	variable of self-inhibition effect of neurons
J_j^k	operator representing +1, -1, or 0
s_l	output of the lth sensory neuron
b	rate of the adaptation value
u_0	external input for coupled neurons
τ and τ'	time constant of the inner state and the adaptation effect

$$\theta_0 = \begin{cases} \theta_1^{(l)} - \theta_1^{(r)} & \text{if } N^{joint} < 3 \\ \Theta_3 & \text{otherwise} \end{cases}$$

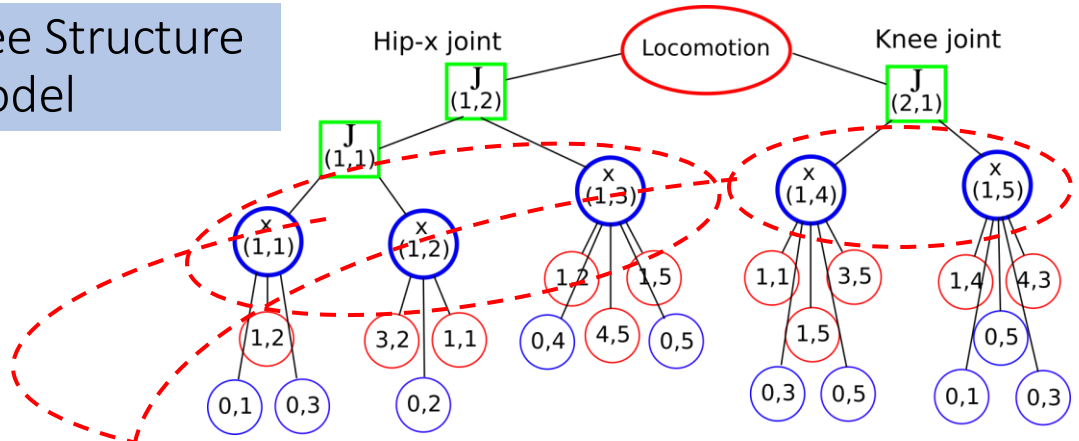
$$\theta_1^{(l,r)} = \Theta_0$$

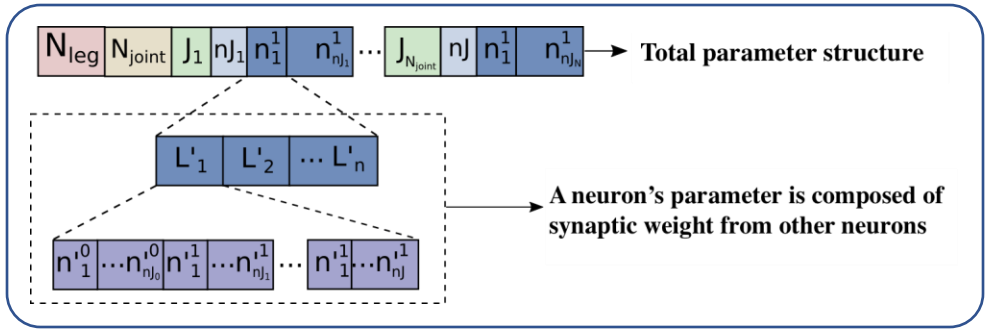
$$\theta_3 = \begin{cases} \theta_2^{(l,r)} - \theta_1^{(l,r)} & \text{if } N^{joint} < 4 \\ \Theta_4 & \text{otherwise} \end{cases}$$

Joint angle calculation

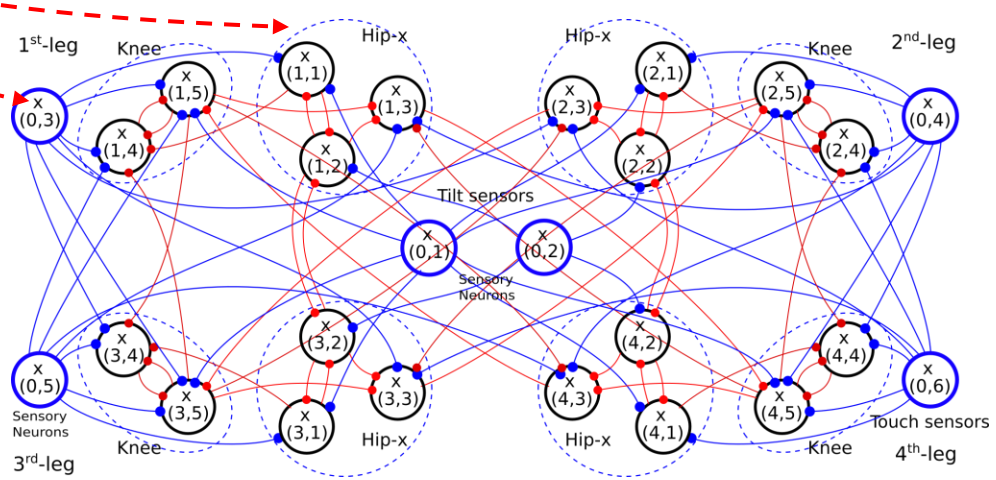
Tree Structure Model

Tree Structure model

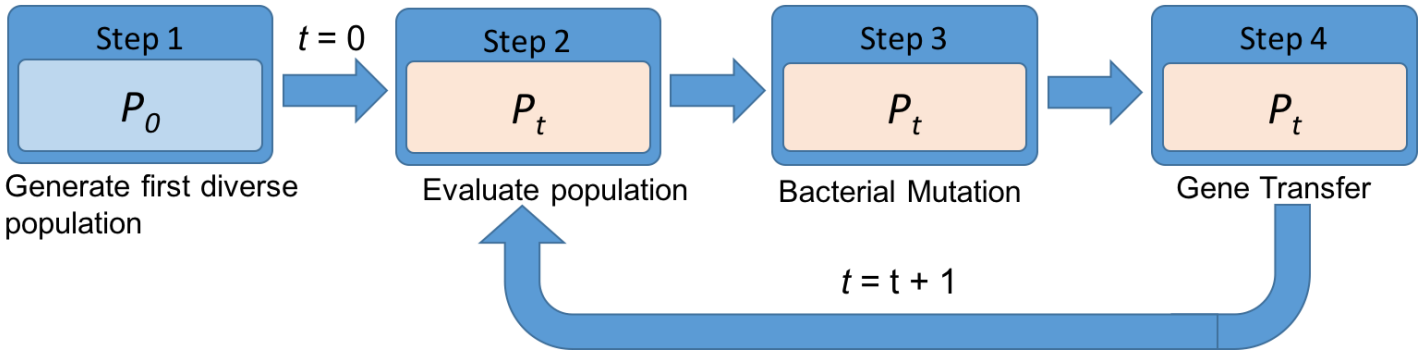




Neuron inter-connection structure



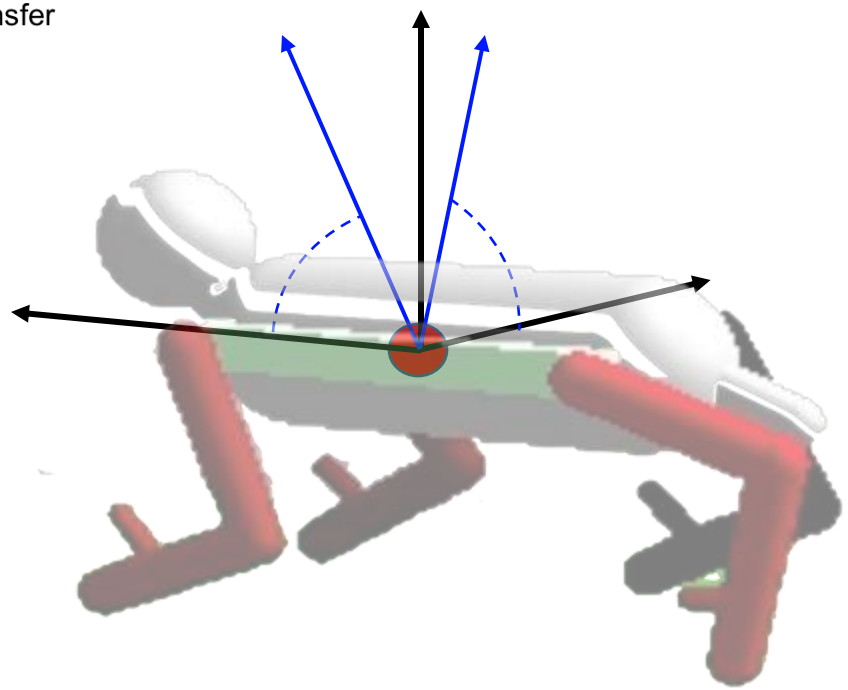
Bacterial Programming - Evaluation



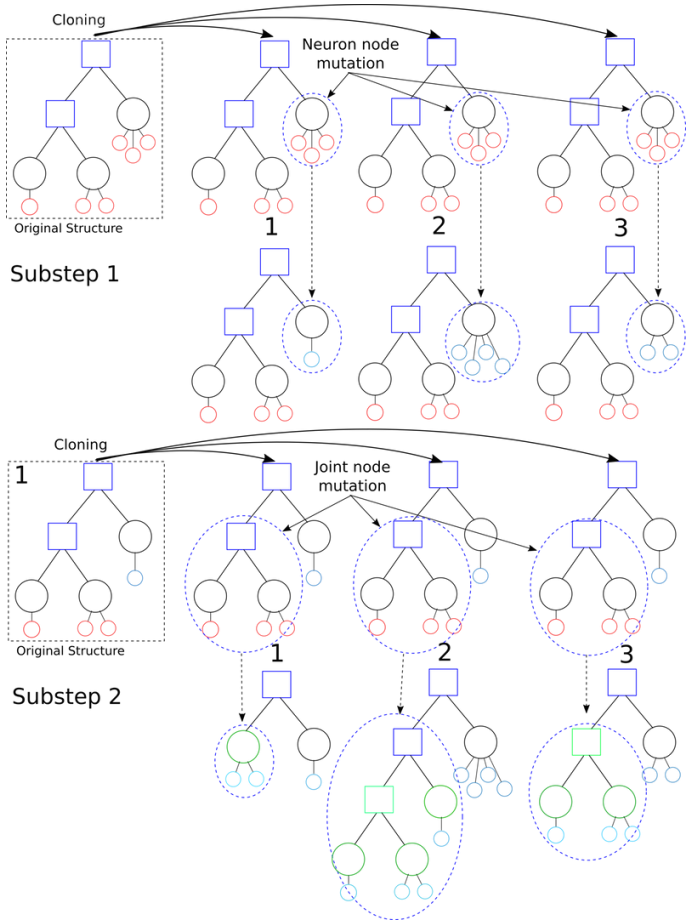
$$\bar{\dot{\beta}} = \frac{1}{T}\sum_{t=0}^{T}\left(\beta_{pitch}(t) + \beta_{roll}(t)\right)$$

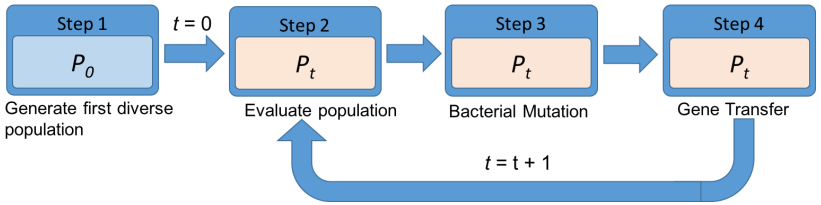
$$\bar{v}_{(w_{ij})} = \Upsilon - \frac{1}{T}\sum_{t=0}^{T}\frac{\delta}{\delta t}\ell\left(t, w_{ij}, \mathrm{x}(t), \mathrm{y}(t)\right)$$

$$f = \bar{\dot{\beta}}\alpha_1 + \bar{v}\alpha_2$$



Bacterial Programming - Bacterial Mutation

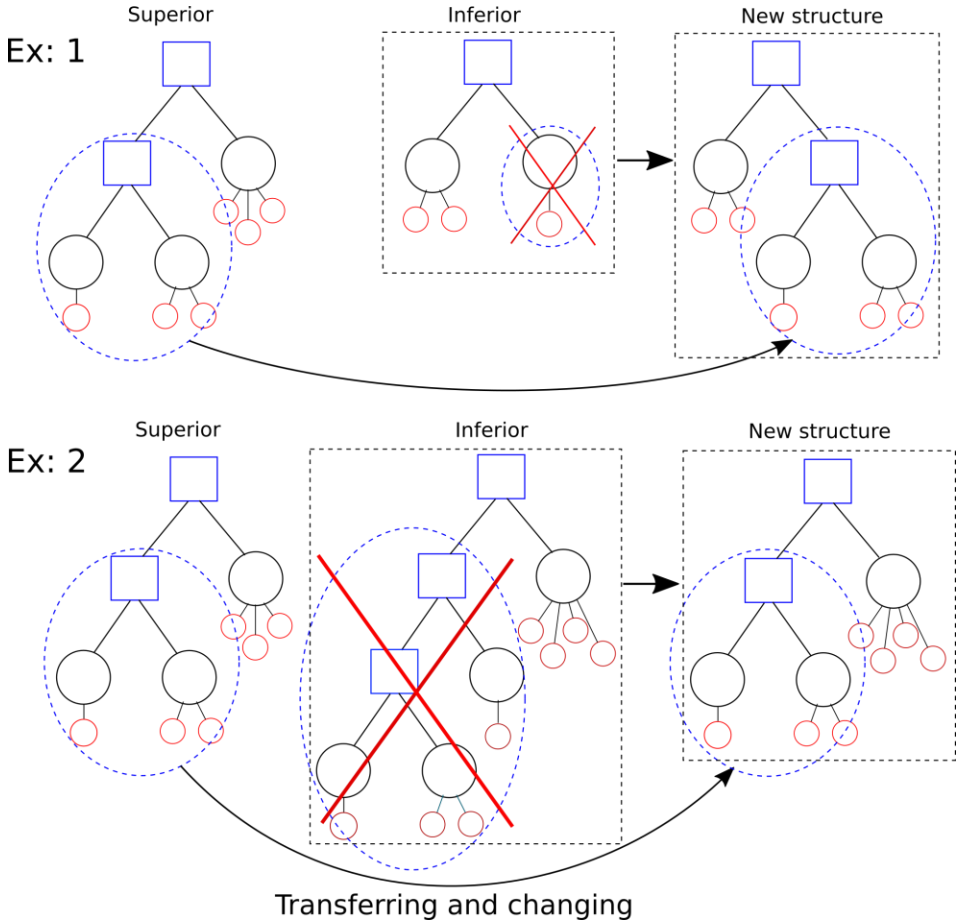


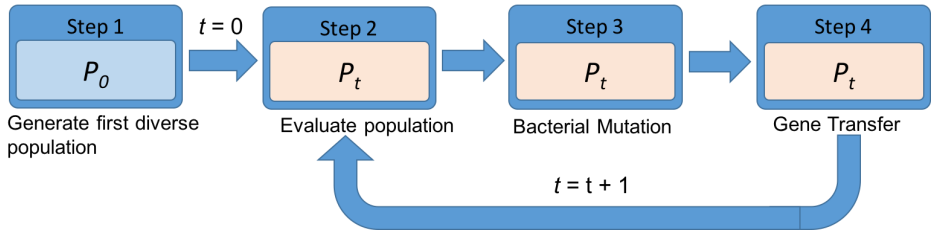


Algorithm 2 Bacterial Mutation Operation

```
for i ← 1 to N_ind do
    Cloning the ith individual
    repeat
        Defining a node randomly
        if node is a Joint node then
            //Joint node mutation in the clones
            for j ← 1 to N_clones do
                Processing Joint node mutation
                Evaluating the mutated clone
        else
            //Neuron node mutation in the clones
            for j ← 1 to N_clones do
                Processing Neuron node mutation
                Evaluating the mutated clone
        Selecting the best individual
        The best individual transfers the mutated subtree
        to the other individuals
    until all subtrees are mutated
    Keeping the best individual as the new ith individual
    Deleting the clones
```

Bacterial Programming - Gene Transfer

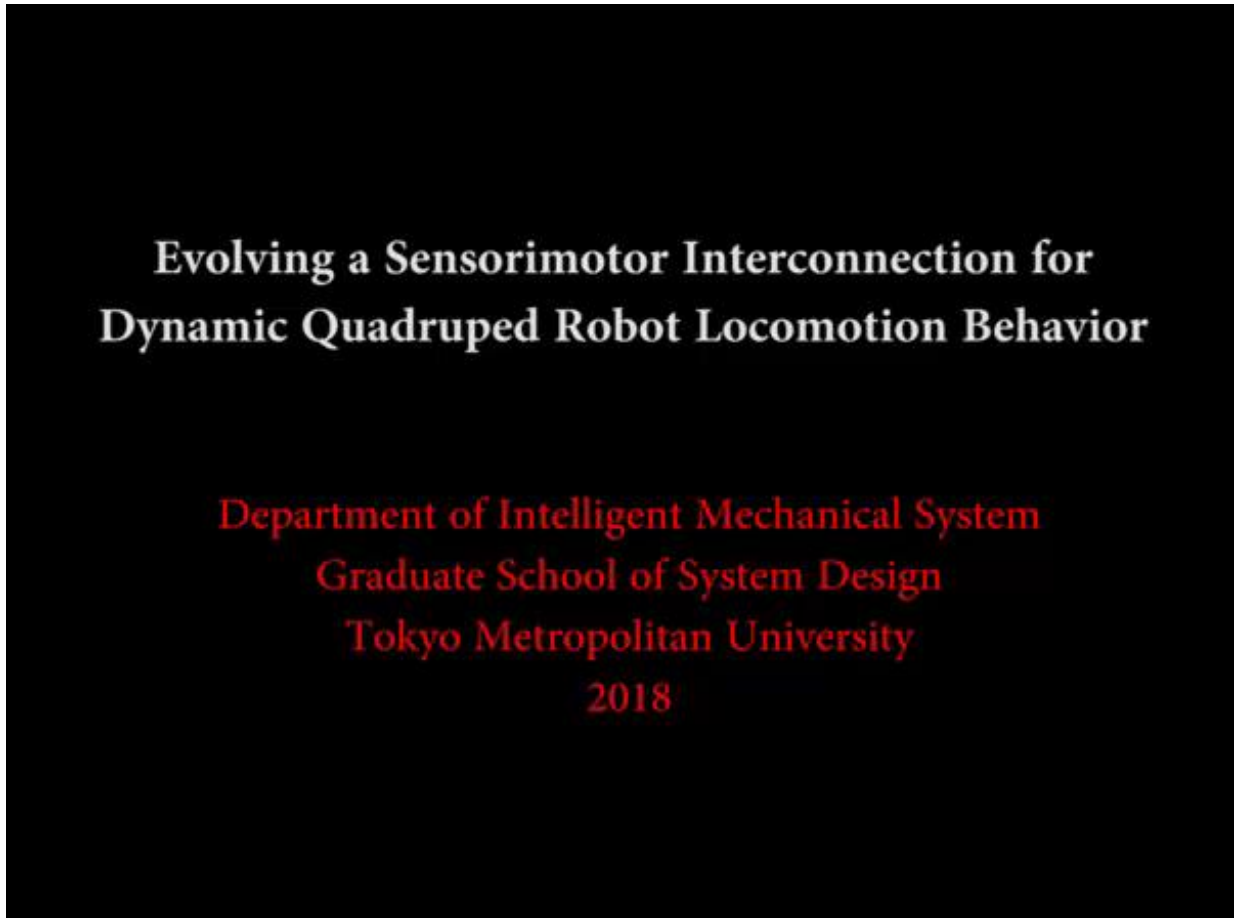




Algorithm 3 Gene Transfer Operation

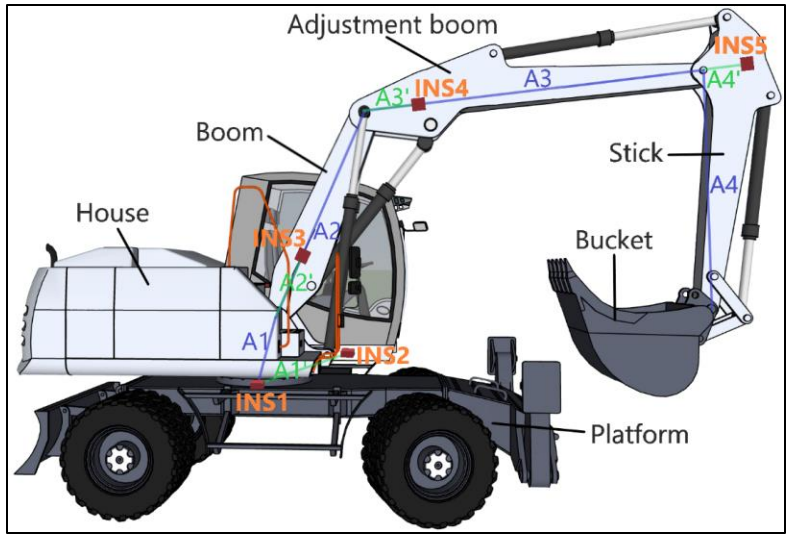
1: **for** $i \leftarrow 1$ to N_{inf} **do**
2: Ordering the population
3: //Selecting a random superior bacterium
4: $Superior = RandomValue(1, N_{ind}/2)$
5: //Selecting a random inferior bacterium
6: $Inferior = RandomValue(N_{ind}/2+1, N_{ind})$
7: Defining the subtree to be transferred
8: from the Superior bacterium
9: Defining the subtree to be overwritten
10: in the Inferior bacterium
11: Transferring and overwriting
12: Evaluating the new Inferior bacterium

Results

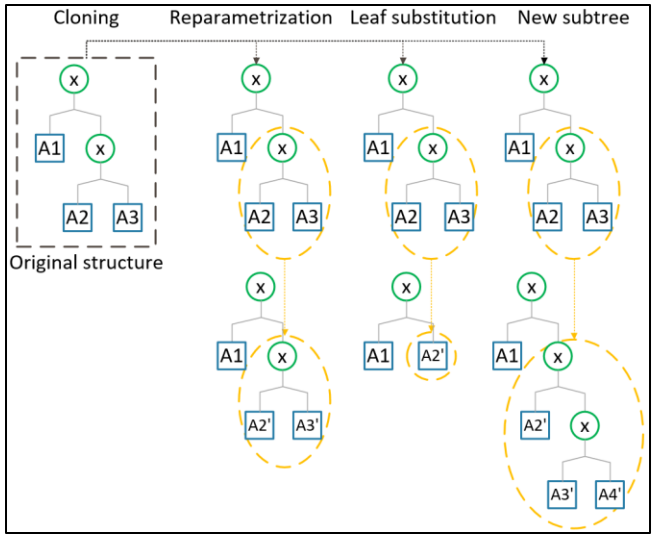

Evolving a Sensorimotor Interconnection for
Dynamic Quadruped Robot Locomotion Behavior
Department of Intelligent Mechanical System
Graduate School of System Design
Tokyo Metropolitan University
2018

Kinematic chain estimation

- Forward kinematic chain estimation of an excavator by bacterial programming based on randomly placed inertial navigation systems per segments with microelectromechanical sensors within.
- The unknown model structure and parameters are established and identified by BP without any a priori or given information about the device according to Denavit-Hartenberg transformation conventions



+



=

$$A^{G}_{INS_{AB}} =$$

$$Analytic \rightarrow \begin{Bmatrix} 0 \\ 0 \\ 0 \\ x_W \end{Bmatrix} \begin{Bmatrix} \psi_H \\ 0 \\ 0 \\ -1.250 \end{Bmatrix} \begin{Bmatrix} 3.141 \\ 0 \\ -1.571 \\ -0.400 \end{Bmatrix} \begin{Bmatrix} \theta_B \\ 0 \\ 0 \\ 0.580 \end{Bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1.285 \end{Bmatrix} \begin{Bmatrix} \theta_{AB} \\ 0 \\ 0 \\ 0.330 \end{Bmatrix}$$

$$BP \rightarrow \begin{Bmatrix} 0 \\ 0 \\ 0 \\ x_W \end{Bmatrix} \begin{Bmatrix} \psi_H \\ 0 \\ 0.001 \\ -1.262 \end{Bmatrix} \begin{Bmatrix} 3.141 \\ 0 \\ -1.571 \\ -0.400 \end{Bmatrix} \begin{Bmatrix} \theta_B \\ 0 \\ 0 \\ 0.575 \end{Bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1.286 \end{Bmatrix} \begin{Bmatrix} \theta_{AB} \\ 0 \\ 0 \\ 0.340 \end{Bmatrix}$$

ELTE | FACULTY OF INFORMATICS

Thank you for your attention!