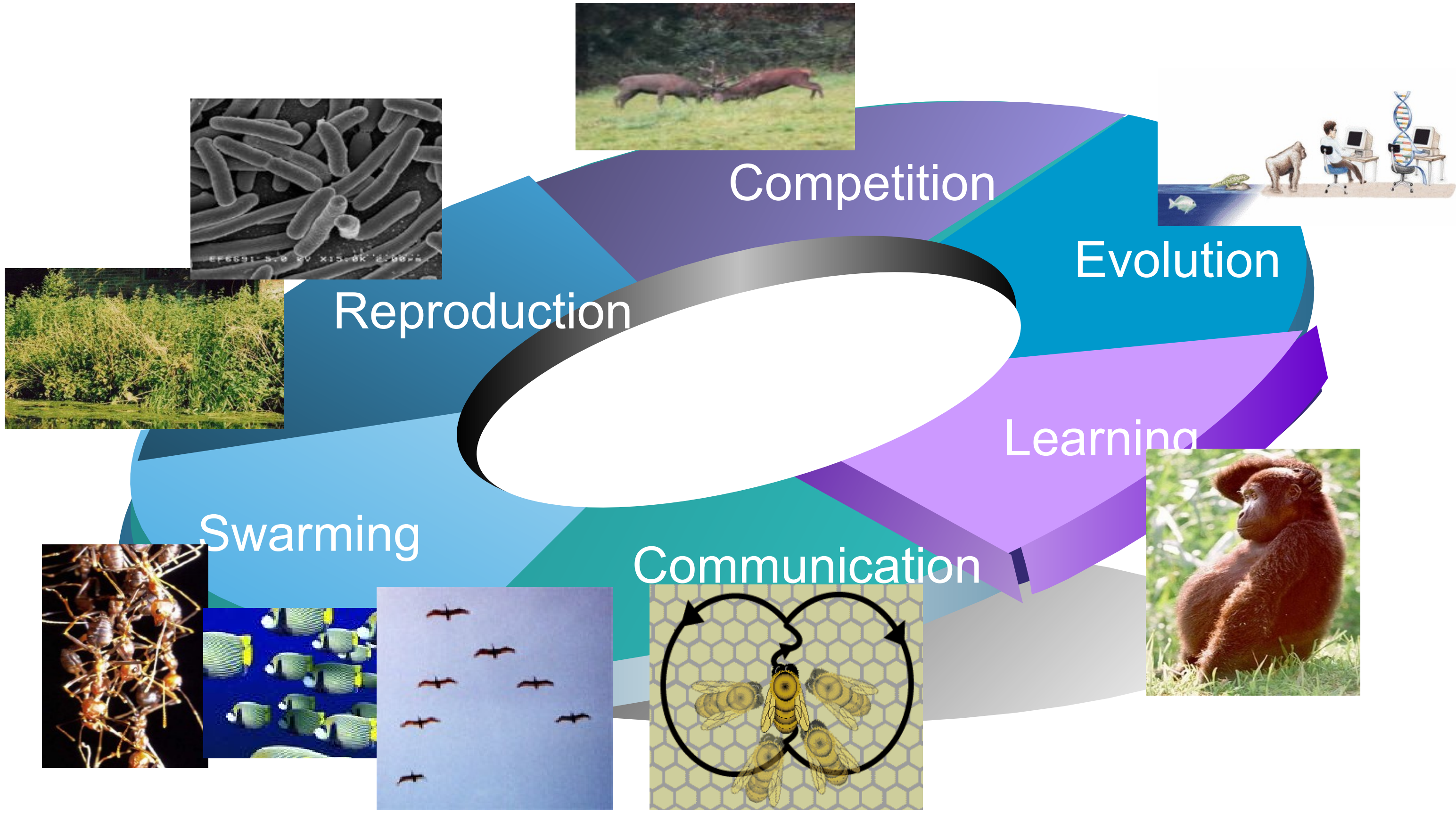
Evolutionary algorithms

Introduction, motivations

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

General optimization algorithms

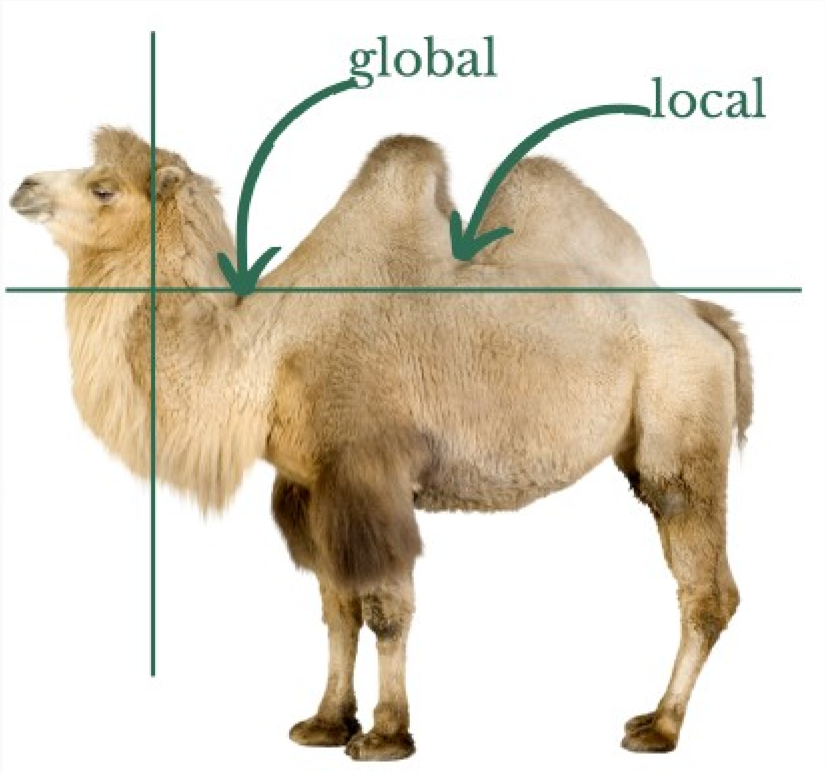
- Deterministic
  - Calculus based
  - Hill climbing
  - …
- Stochastic
  - Random search
  - Simulated annealing
  - …
- Evolutionary algorithms: Stochastic search methods, computationally simulate the natural evolutionary process using the idea of survival of the fittest

Continuous optimization problem

General Continuous Optimization Problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

- **Local Minimum**: x^* in D where $f(x^*) \leq f(x)$ in a small neighborhood.
- **Global Minimum**: x^* in D where $f(x^*) \leq f(x)$ for all x in the domain D.



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

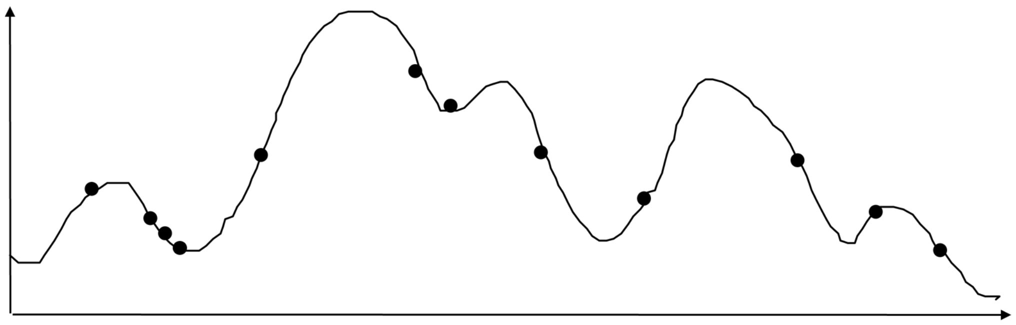
Evolutionary computation - history

- The idea of using simulated evolution to solve engineering and design problems have been around since the 1950's
- However, it wasn't until the early 1960's that we began to see three influential forms of EA emerge:
  - Evolutionary programming (Lawrence Fogel, 1962)
  - Genetic algorithms (Holland, 1975)
  - Evolution strategies (Rechenberg, 1965 & Schwefel, 1968)
- The designers of each of the EA techniques saw that their particular problems could be solved via simulated evolution
  - Fogel was concerned with solving prediction problems
  - Rechenberg & Schwefel were concerned with solving parameter optimization problems
  - Holland was concerned with developing robust adaptive systems

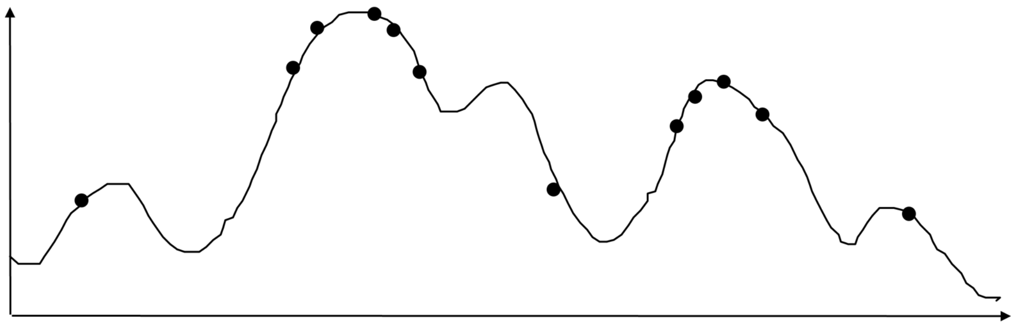
Terminology

- Gene: functional entity that encodes a specific feature of the individual (e.g. hair color)
- Allele: value of gene (e.g. blonde)
- Genotype: the specific combination of alleles carried by an individual
- Phenotype: the physical makeup of an organism
- Locus: position of the gene within the chromosome
- Individual: chromosome, represents a candidate solution for the problem
- Population: collection of individuals currently alive

Evolution of the population



Distribution of Individuals in Generation 0

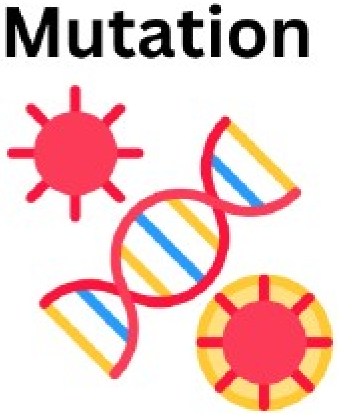


Distribution of Individuals in Generation N

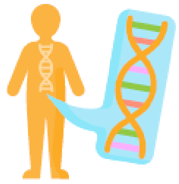
Genetic algorithms

Genetic algorithm

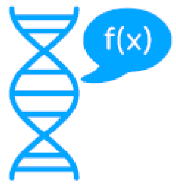
- Population-based optimization method
- Inspired by Darwinian theory of evolution
- Stochastic in nature
- Utilizes 3 bio-inspired operators

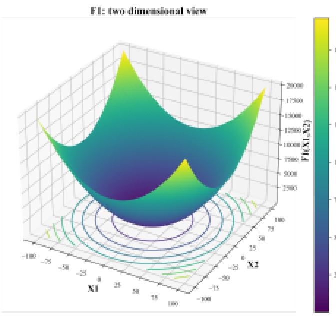


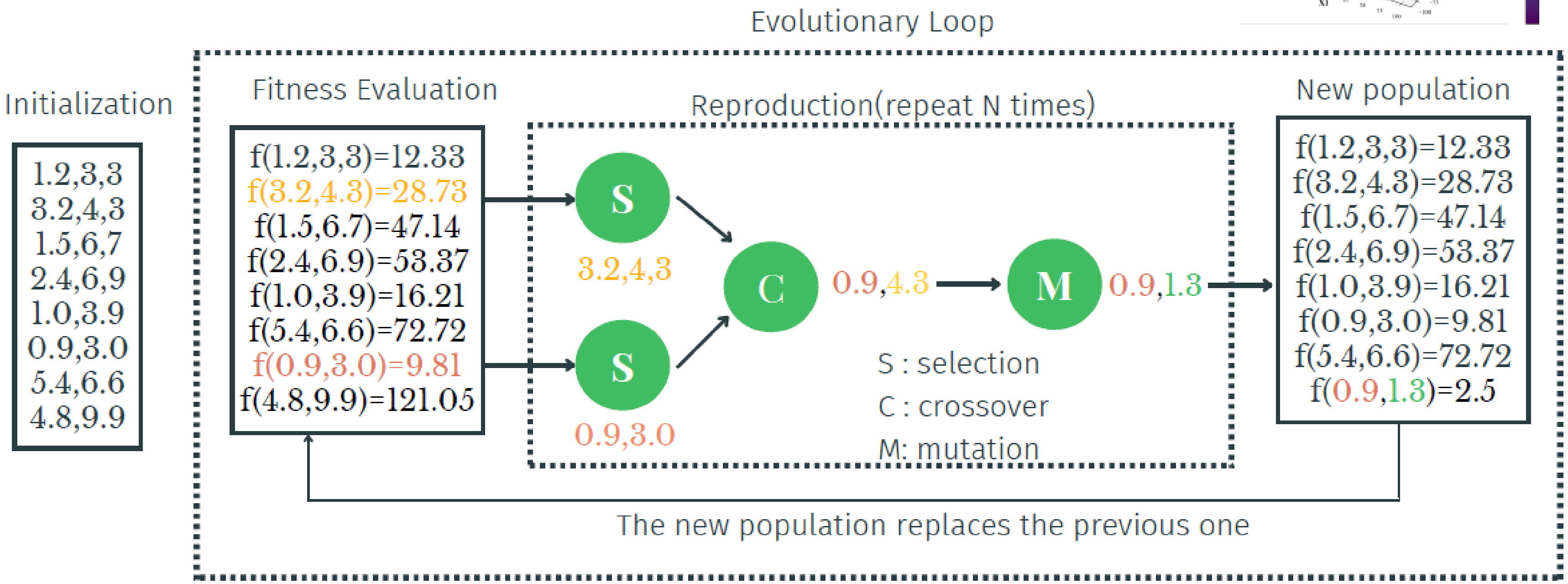
Genetic algorithms



Each individual has DNA. DNA(genotype) is a vector.
Each vector x is a candidate solution to a function.





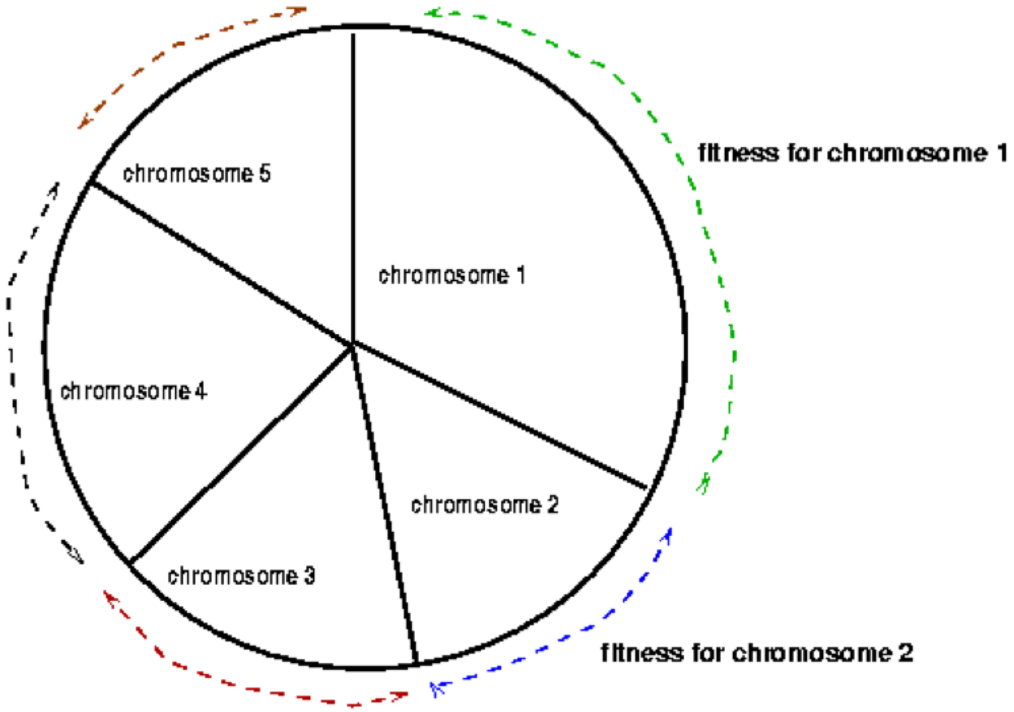


The individual

- The individual is a solution candidate to the problem
- A possible solution to the problem is encoded into the individual in some form
  - e.g. binary, or real
- Fitness value: the individuals are evaluated according to some criterion how good solution they can provide to the given problem
- Better individuals have a higher fitness value, thus they have a higher chance to survive

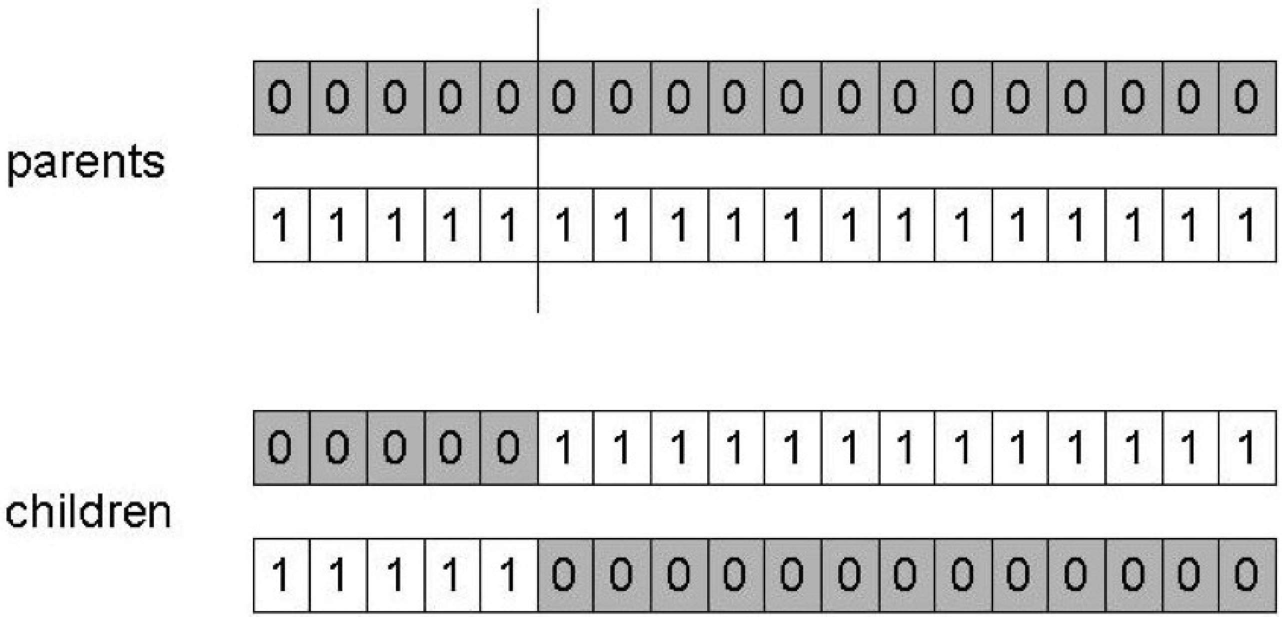
Selection methods

- There are many ways to select chromosomes to survive to the next generation
- *Roulette wheel selection*: the better the chromosome, the more chance for selection it possesses; imagine a roulette wheel where every chromosome is represented in proportion to its fitness function
- Then a 'roulette' ball is thrown and selects chromosomes – chromosomes with larger fitness will be selected with a higher probability



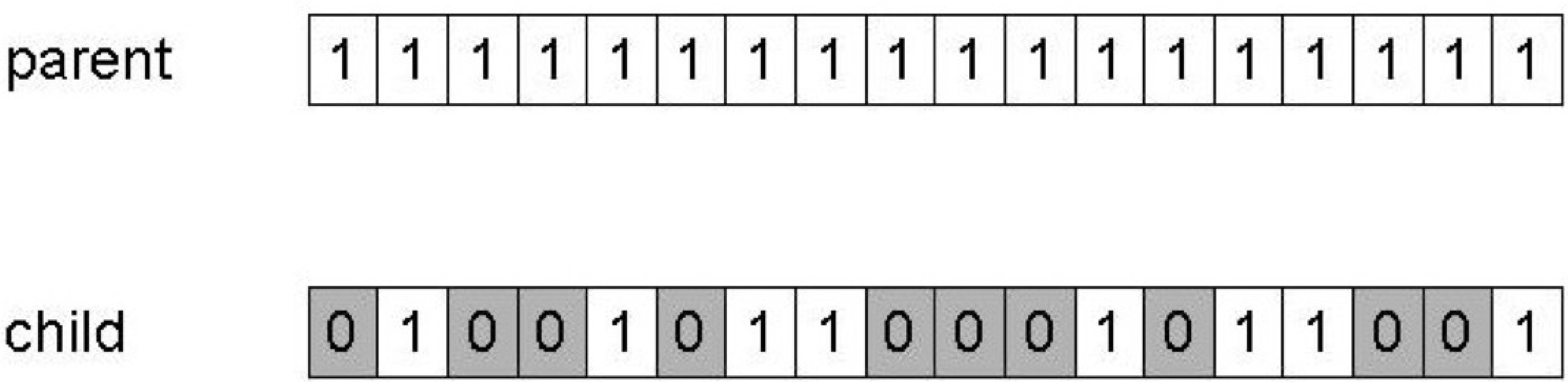
Crossover

- Choose a random point on the two parents
- Split parents at this crossover point
- Create children by exchanging tails



Mutation

- Alter each gene independently with a probability p_m (mutation rate)



An example (Goldberg)

- Simple problem: max x^2 over $\{0,1,\ldots,31\}$
- Genetic algorithm approach:
  - Representation: binary code, e.g. 01101 $\longleftrightarrow$ 13
  - Population size: 4
  - Crossover, mutation
  - Roulette wheel selection
  - Random initialization

An example: selection

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

An example: crossover

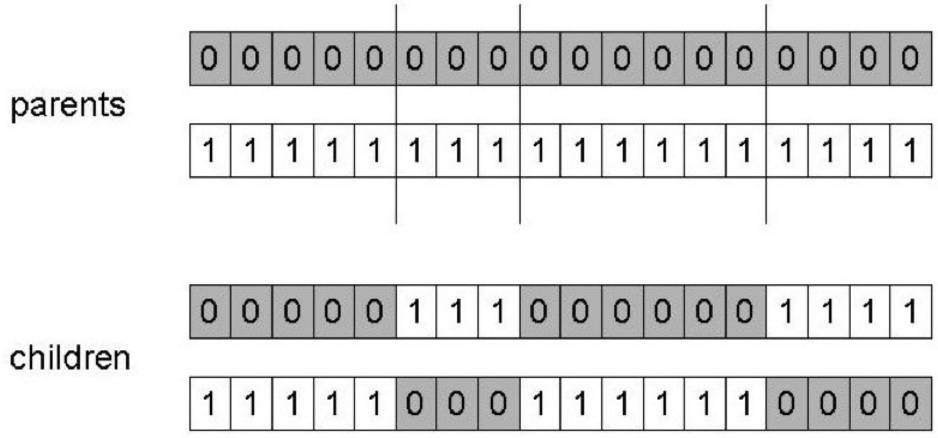
String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

An example: mutation

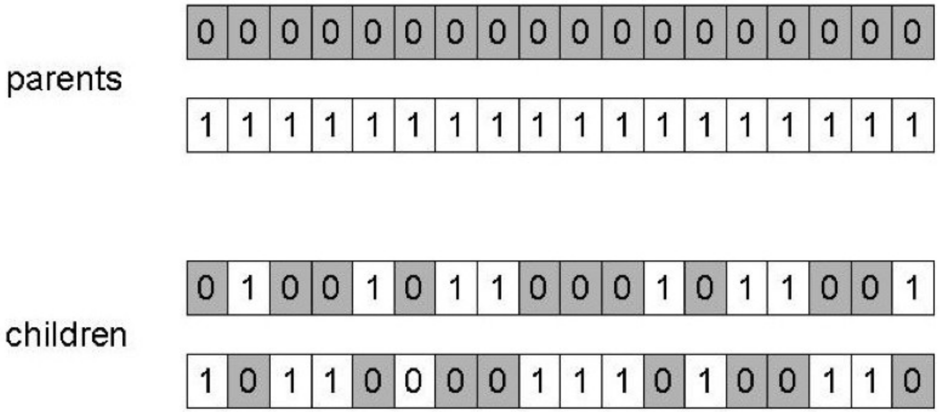
String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	28	784
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	20	400
Sum				2538
Average				634.5
Max				784

Alternative crossover operators

n-point:

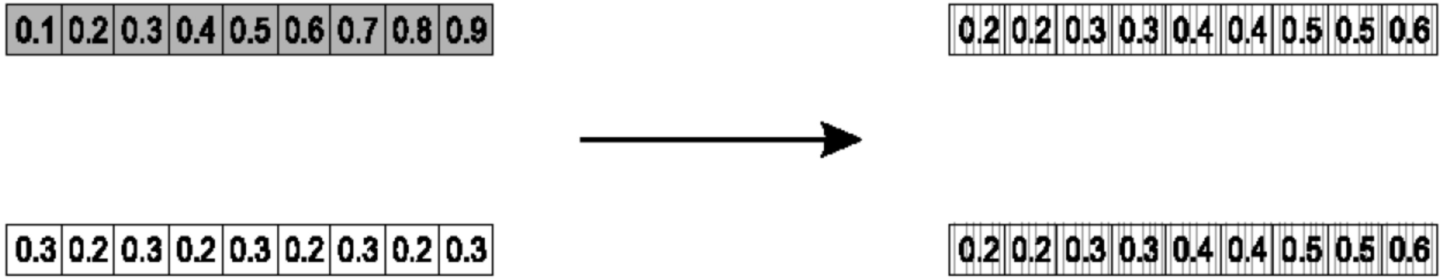


uniform:



Real-coded GA

- Crossover for example:
  - parents: $\langle x_1,\ldots,x_n \rangle$ and $\langle y_1,\ldots,y_n \rangle$
  - offspring$_1$: $\alpha\underline{x}+(1-\alpha)\underline{y}$
  - swapped for the other offspring
  - e.g.: ($\alpha = 0.5$)

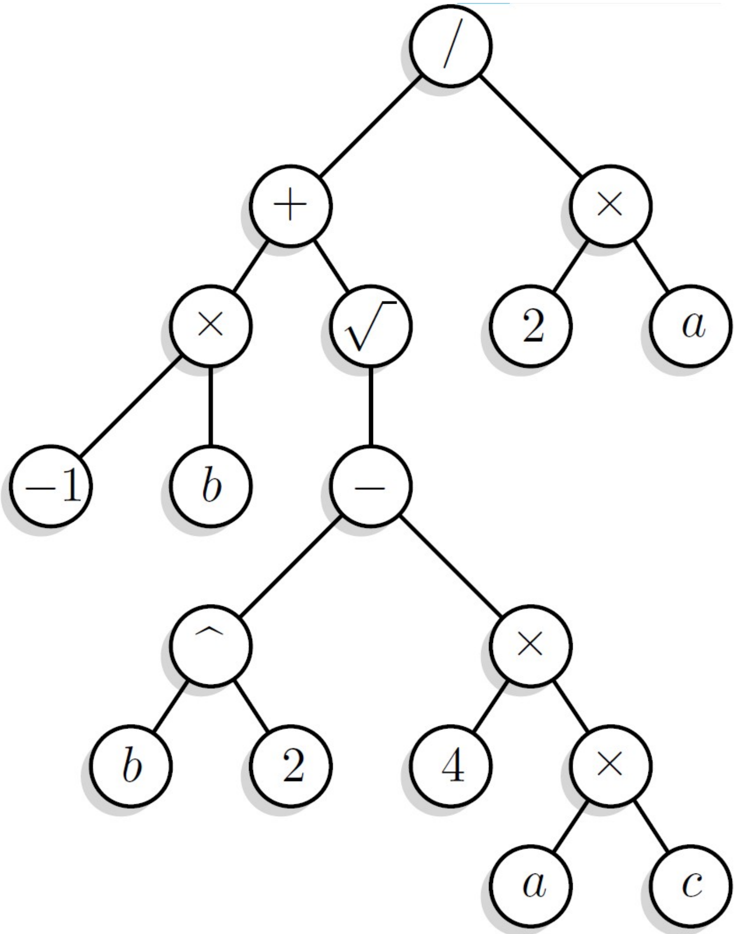


- Mutation: $\underline{x}= \langle x_1,\ldots,x_n \rangle$ $\underline{x}'= \langle x'_1,\ldots,x'_n \rangle$ $x_i,x'_i \in [LB_i,UB_i]$

Genetic programming

Genetic programming

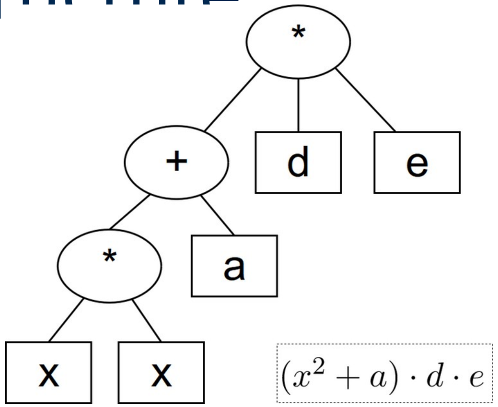
- John Koza (~1990)
- Genetic programming applies the approach of genetic algorithm to the space of possible computer programs
- A wide variety of seemingly different problems from many different fields can be reformulated as a search for a computer program to solve the problem
- Individuals are described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



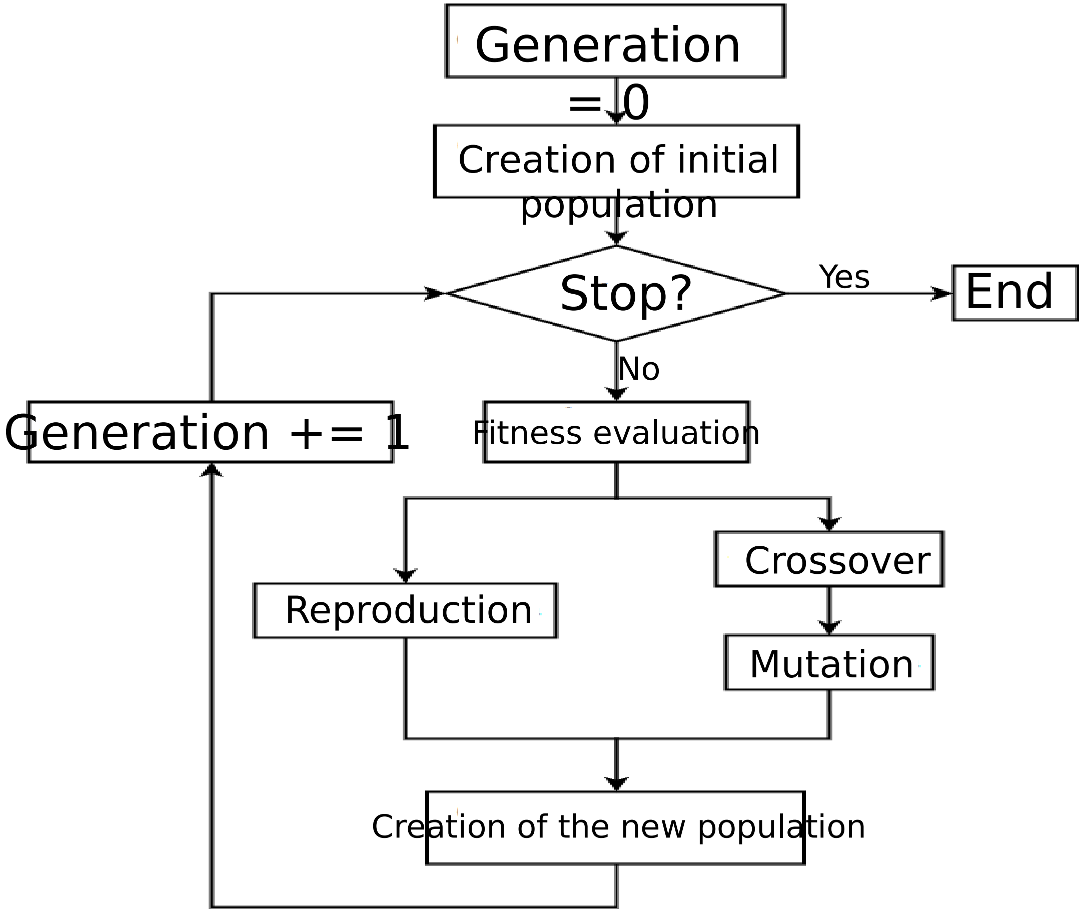
$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

Genetic programming

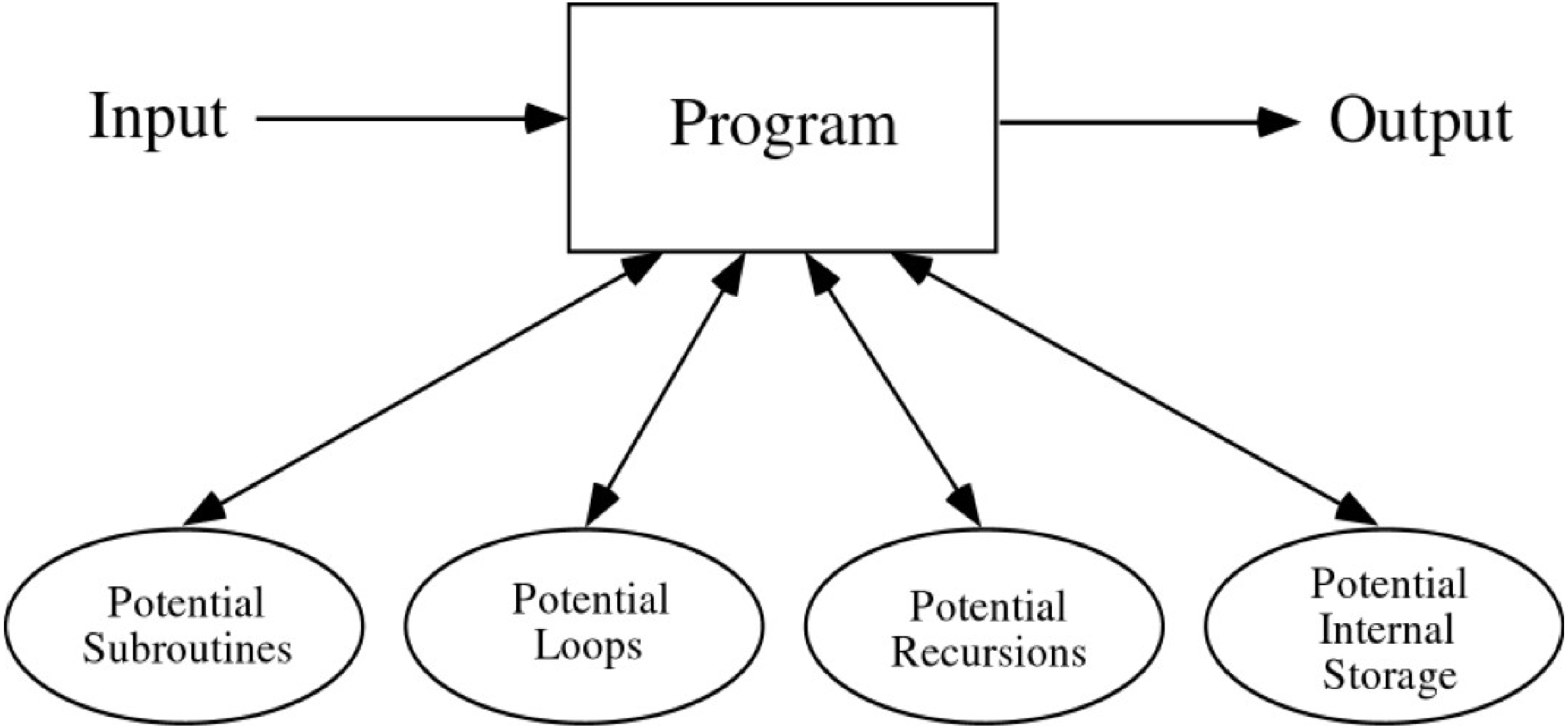
- The individual is a tree-based structure



- Evolutionary operators
  - Reproduction
  - Crossover
  - Mutation

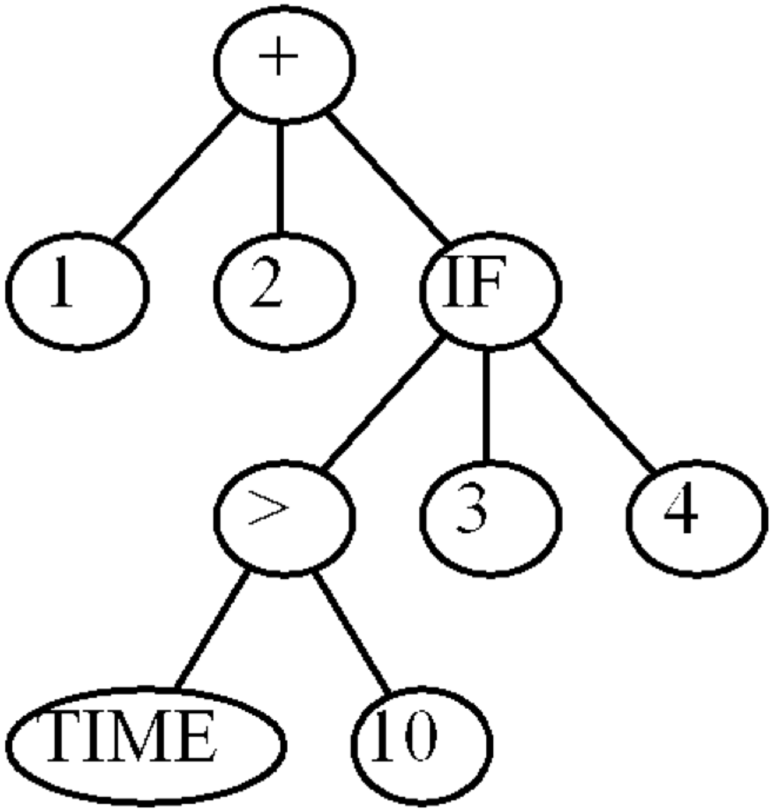


A computer program



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

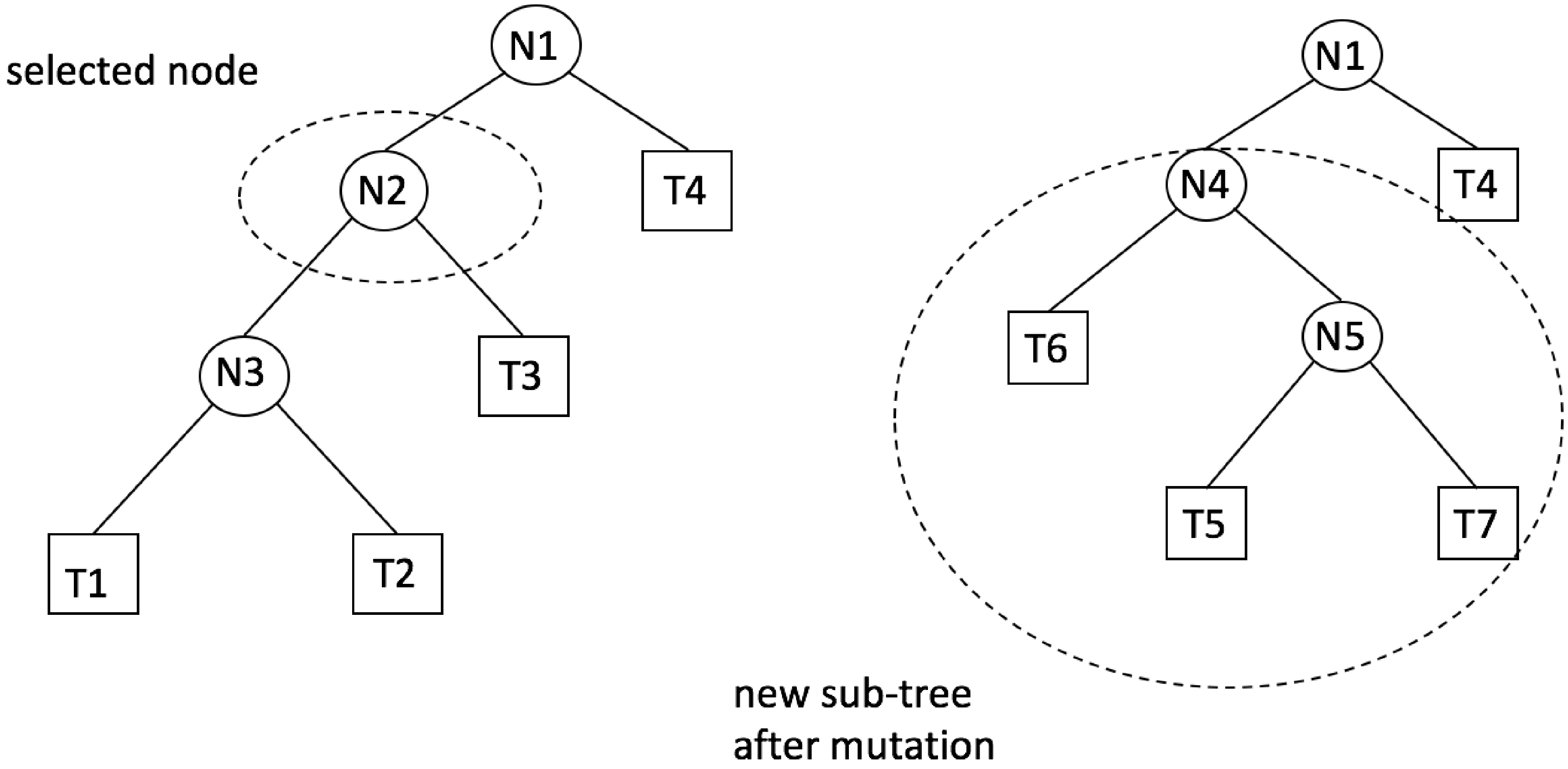
- Reproduction
- Mutation
- Crossover

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

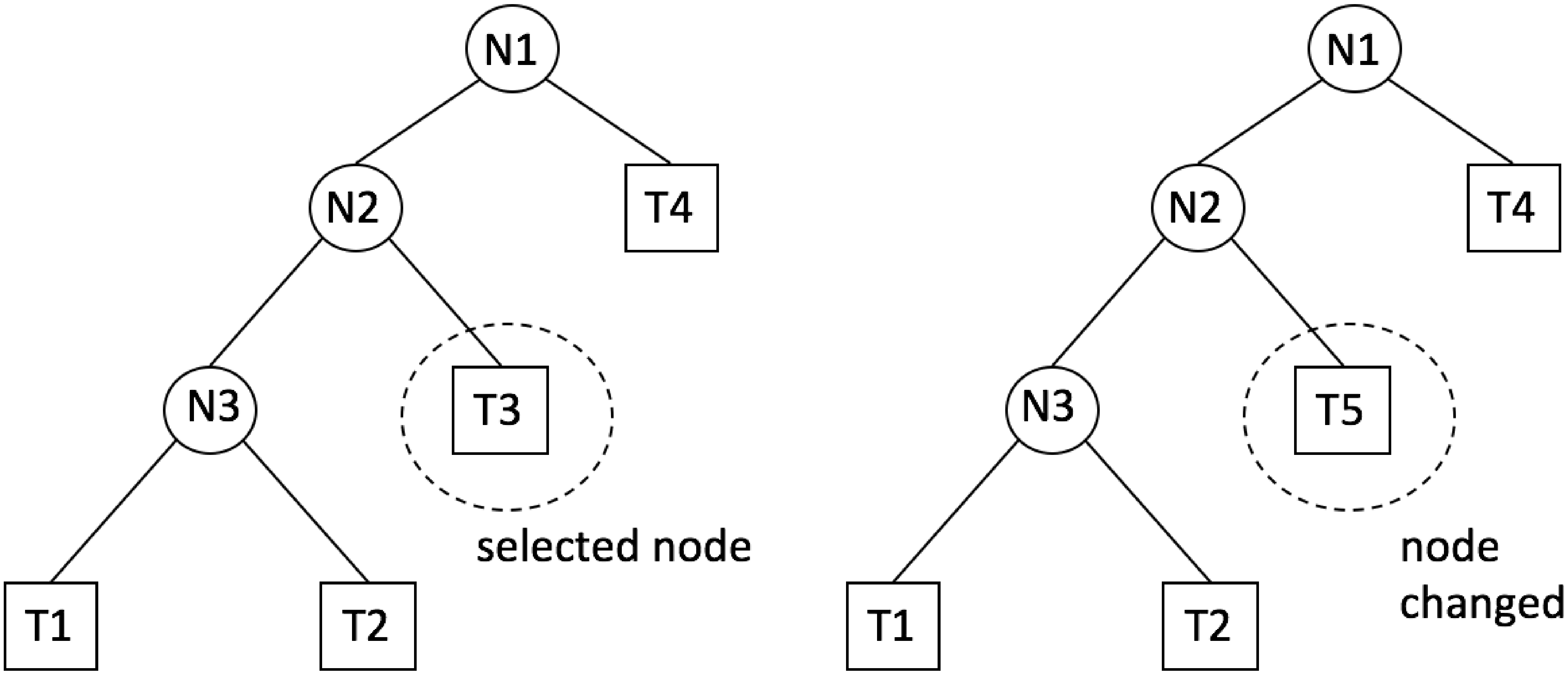
Mutation

- 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



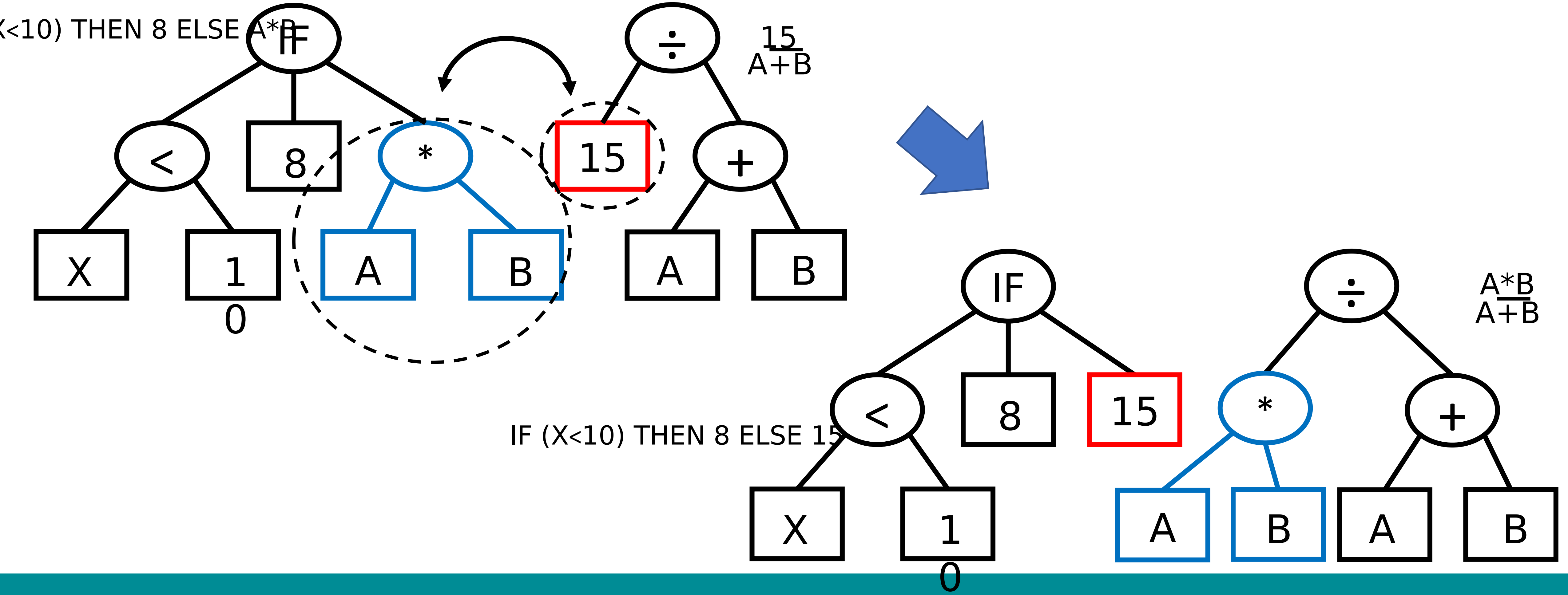
Node mutation



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) THEN 8 ELSE A*B
IF
<
8
*
X
10
A
B
÷
15
+
A
B
15 / A+B
IF
<
8
15
X
10
IF (X<10) THEN 8 ELSE 15
÷
*
+
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