UiO : Department of Informatics
University of Oslo

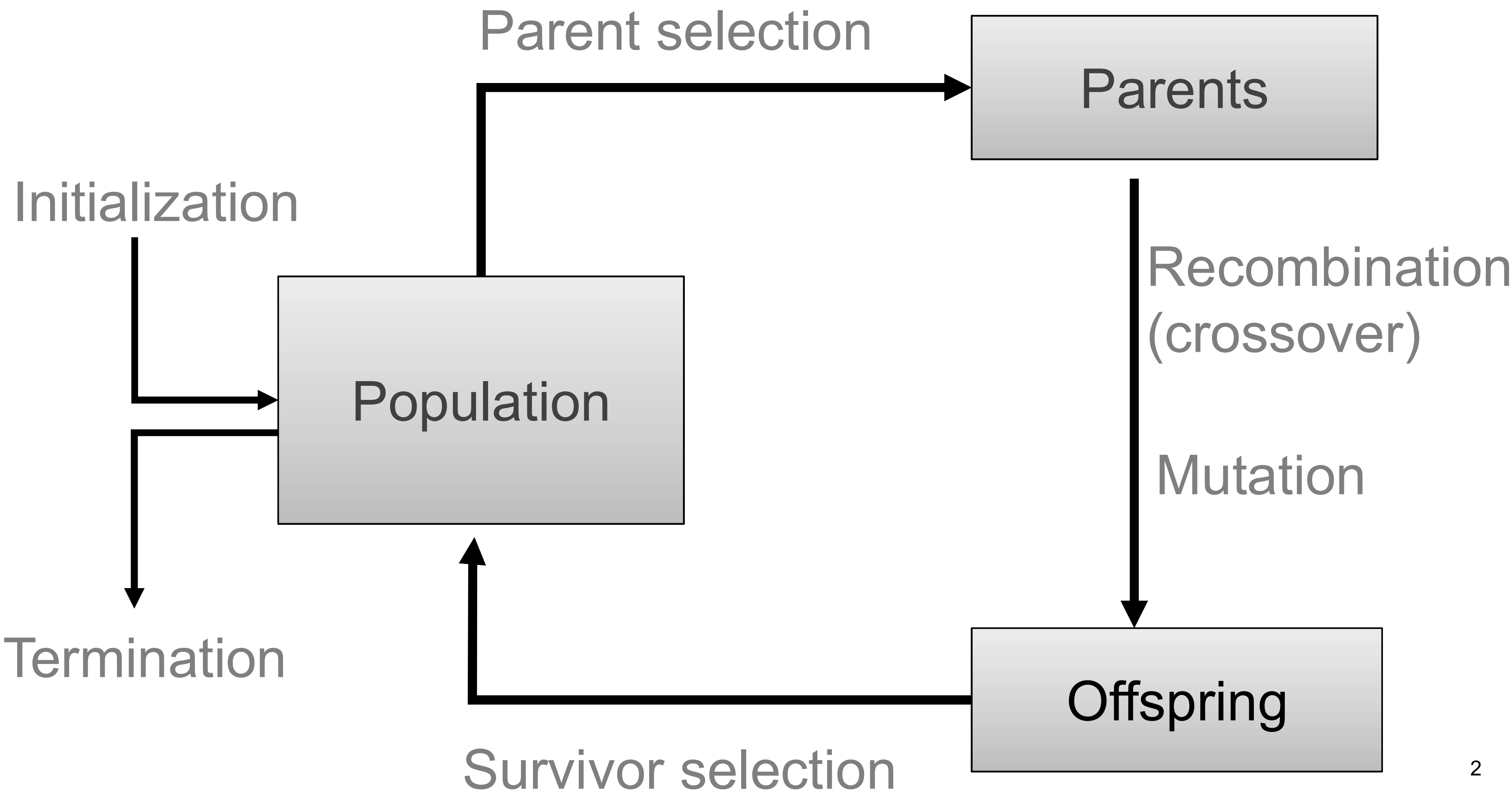
Biologically inspired computing

Lecture 3: Eiben and Smith, chapter 5-6

Evolutionary Algorithms - Population management and popular algorithms

Kai Olav Ellefsen

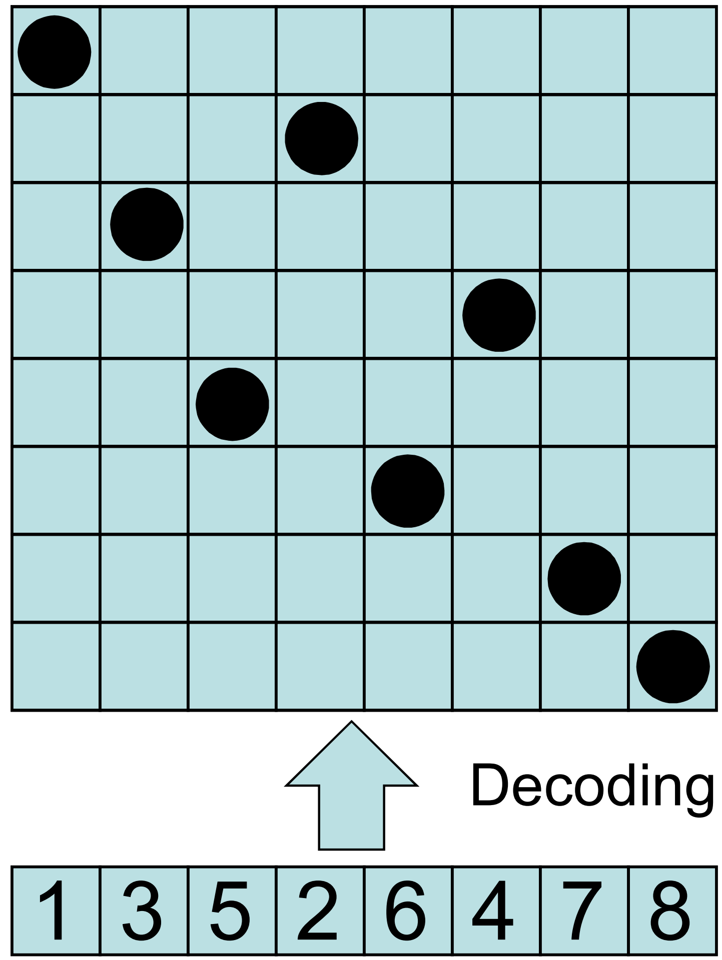
Repetition:
General scheme of EAs



Repetition: Genotype & Phenotype

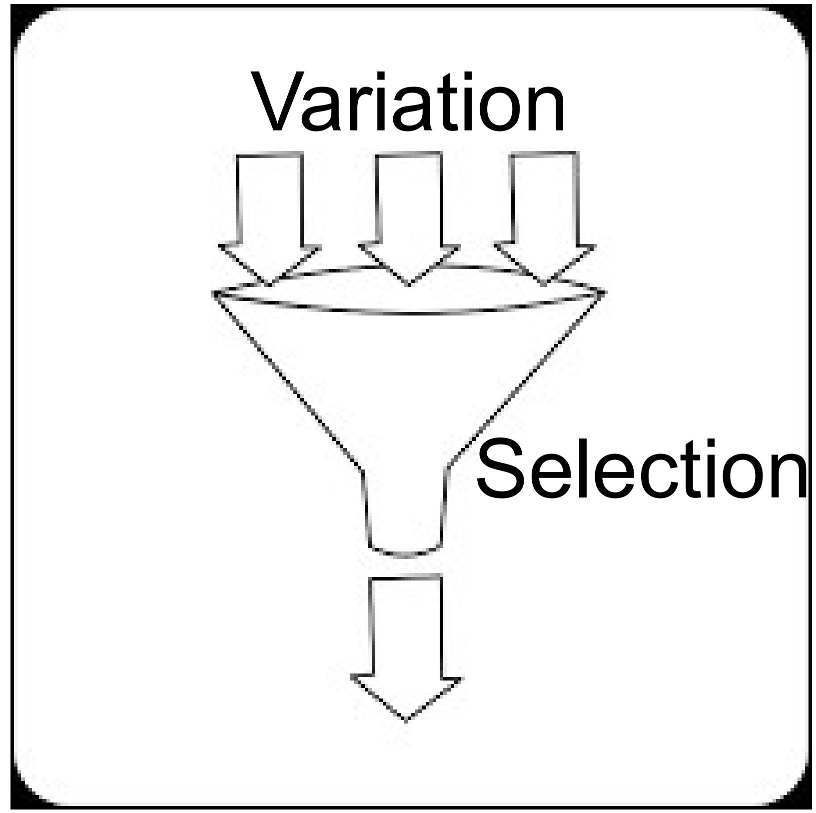
Phenotype:
A solution representation we can **evaluate**

Genotype:
A solution representation applicable to **variation**

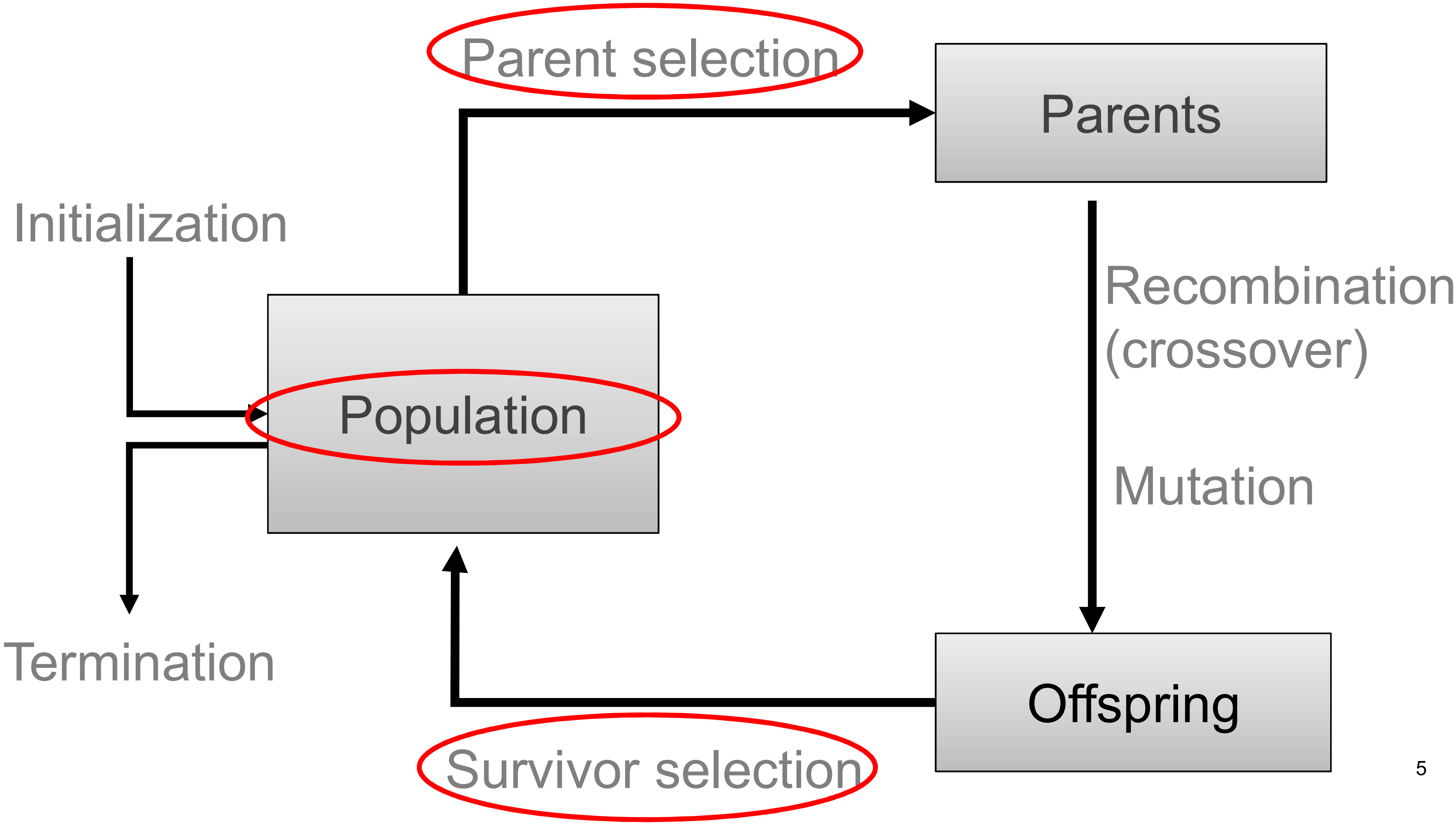


Chapter 5: Fitness, Selection and Population Management

- **Selection** is second fundamental force for evolutionary systems
- Components exist of:
  - Population management models
  - Selection operators
  - Preserving diversity



Scheme of an EA:
General scheme of EAs



Population Management Models: Introduction

- Two different population management models exist:
  - **Generational model**
    - each individual survives for exactly one generation
    - λ offspring are generated
    - the entire set of μ parents is replaced by μ offspring
  - **Steady-state model**
    - λ ($< \mu$) parents are replaced by λ offspring
- Generation Gap
  - The proportion of the population replaced
  - Parameter = 1.0 for G-GA, =λ/pop_size for SS-GA

Population Management Models: Fitness based competition

- Selection can occur in two places:
  - **Parent selection** (selects mating pairs)
  - **Survivor selection** (replaces population)
- Selection works on the population

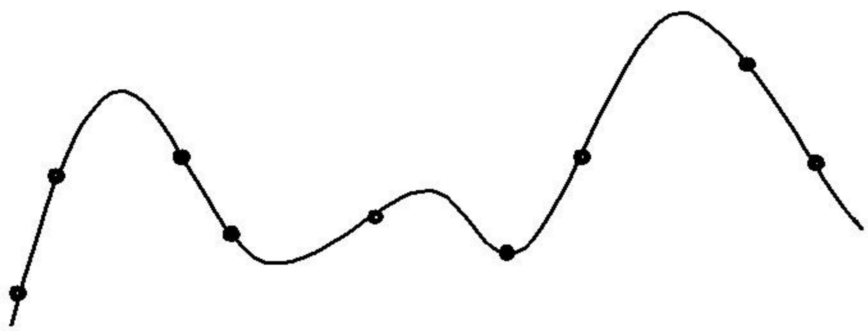
  -> selection operators are representation-independent !
- **Selection pressure**: As selection pressure increases, fitter solutions are more likely to survive, or be chosen as parents

Effect of Selection Pressure

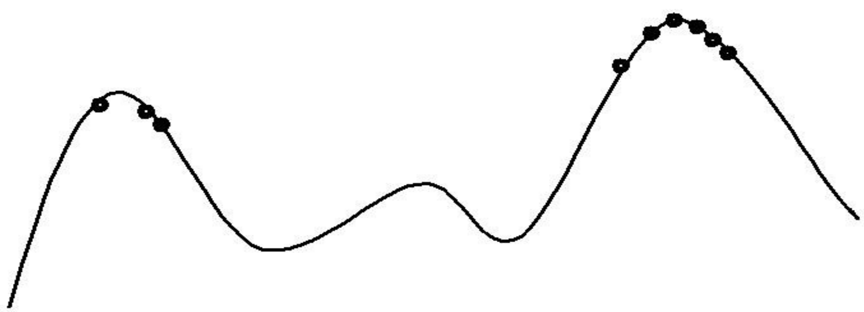
- Low Pressure
- High Pressure


Why Not Always High Selection Pressure?

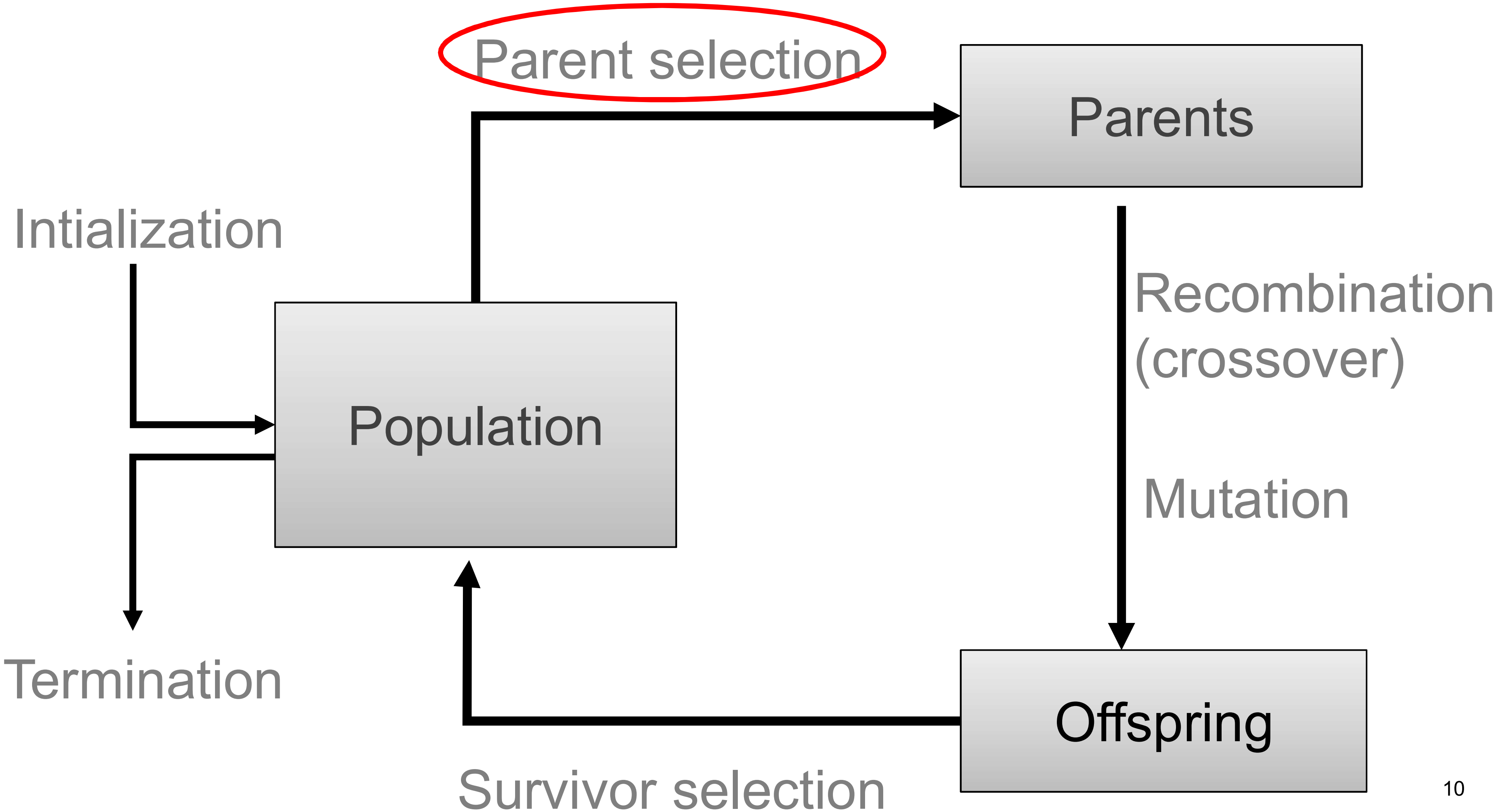
Exploration



Exploitation



Scheme of an EA: General scheme of EAs



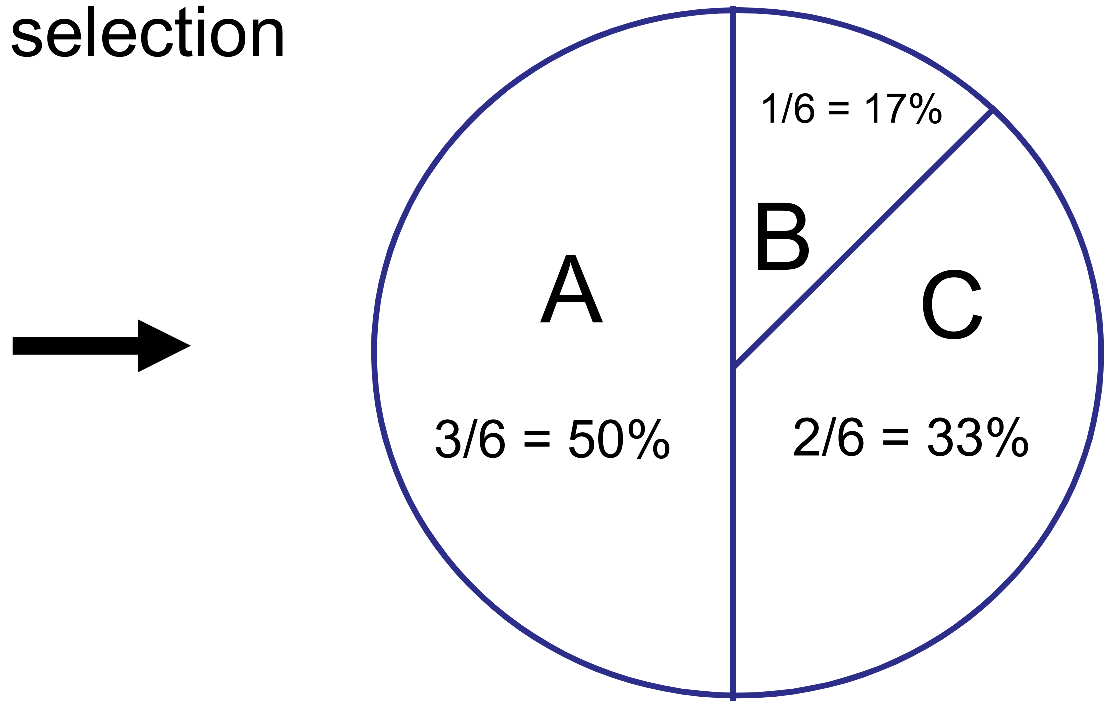
Parent Selection: Fitness-Proportionate Selection

Example: roulette wheel selection

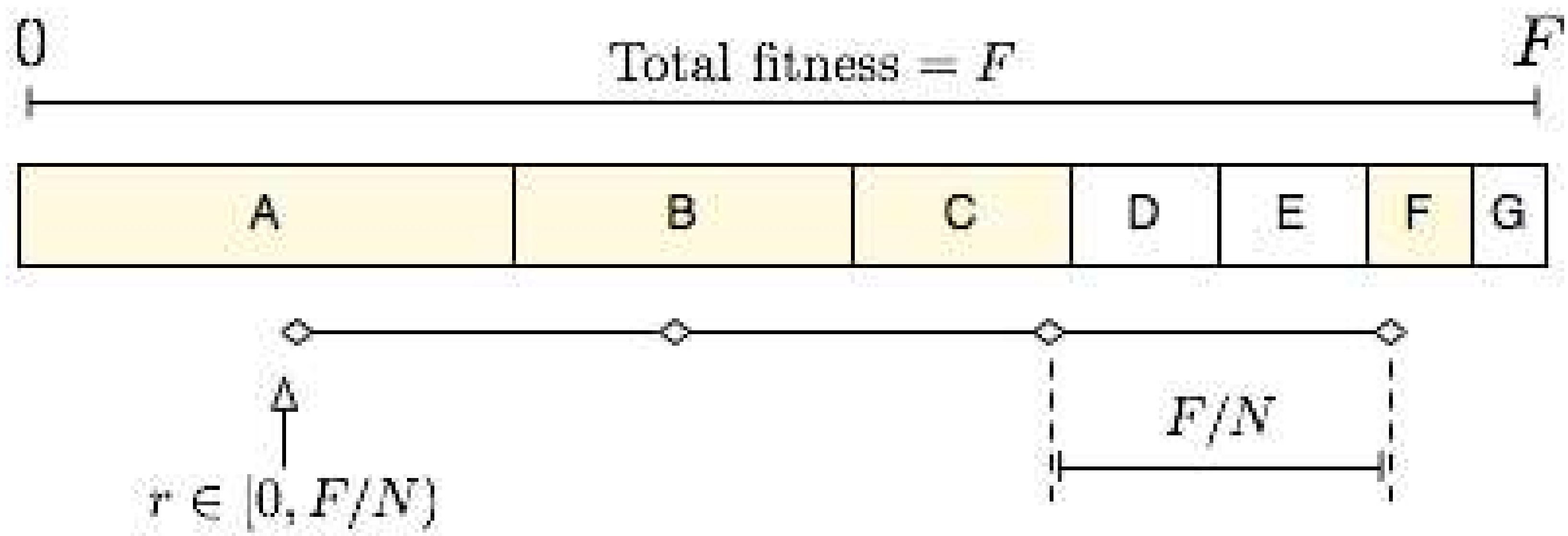
fitness(A) = 3

fitness(B) = 1

fitness(C) = 2



Stochastic Universal Sampling



Stochastic universal sampling (SUS)
Select multiple individuals by making **one** spin of the wheel with **a number of equally spaced arms**

Parent Selection: Fitness-Proportionate Selection (FPS)

- Probability for individual *i* to be selected for mating in a population size *μ* with FPS is

$$P_{FPS}(i) = f_i \Big/ \sum_{j=1}^{\mu} f_j$$

- Problems include
  - One highly fit member can rapidly take over if rest of population is much less fit: **Premature Convergence**
  - At end of runs when finesses are similar, loss of selection pressure
- **Scaling** can fix the last problem by:
  - **Windowing**: $f'(i) = f(i) - \beta^t$

  where β is worst fitness in this (last n) generations
  - **Sigma Scaling**: $f'(i) = \max(f(i) - (\bar{f} - c \bullet \sigma_f), 0)$

  where *c* is a constant, usually 2.0

Parent Selection: Rank-based Selection

- Attempt to remove problems of FPS by basing selection probabilities on ***relative* rather than *absolute* fitness**
- **Rank population** according to fitness and then base selection probabilities on rank (fittest has rank μ*-1* and worst rank 0)
- This imposes a sorting overhead on the algorithm

Parent Selection: Tournament Selection (1/3)

- All methods above rely on global population statistics
  - Could be a bottleneck esp. on parallel machines, very large population
  - Relies on presence of external fitness function which might not exist: e.g. evolving game players

Parent Selection: Tournament Selection (2/3)

Idea for a procedure using only local fitness information:

- Pick *k* members at random then select the best of these
- Repeat to select more individuals

Parent Selection: Tournament Selection (3/3)

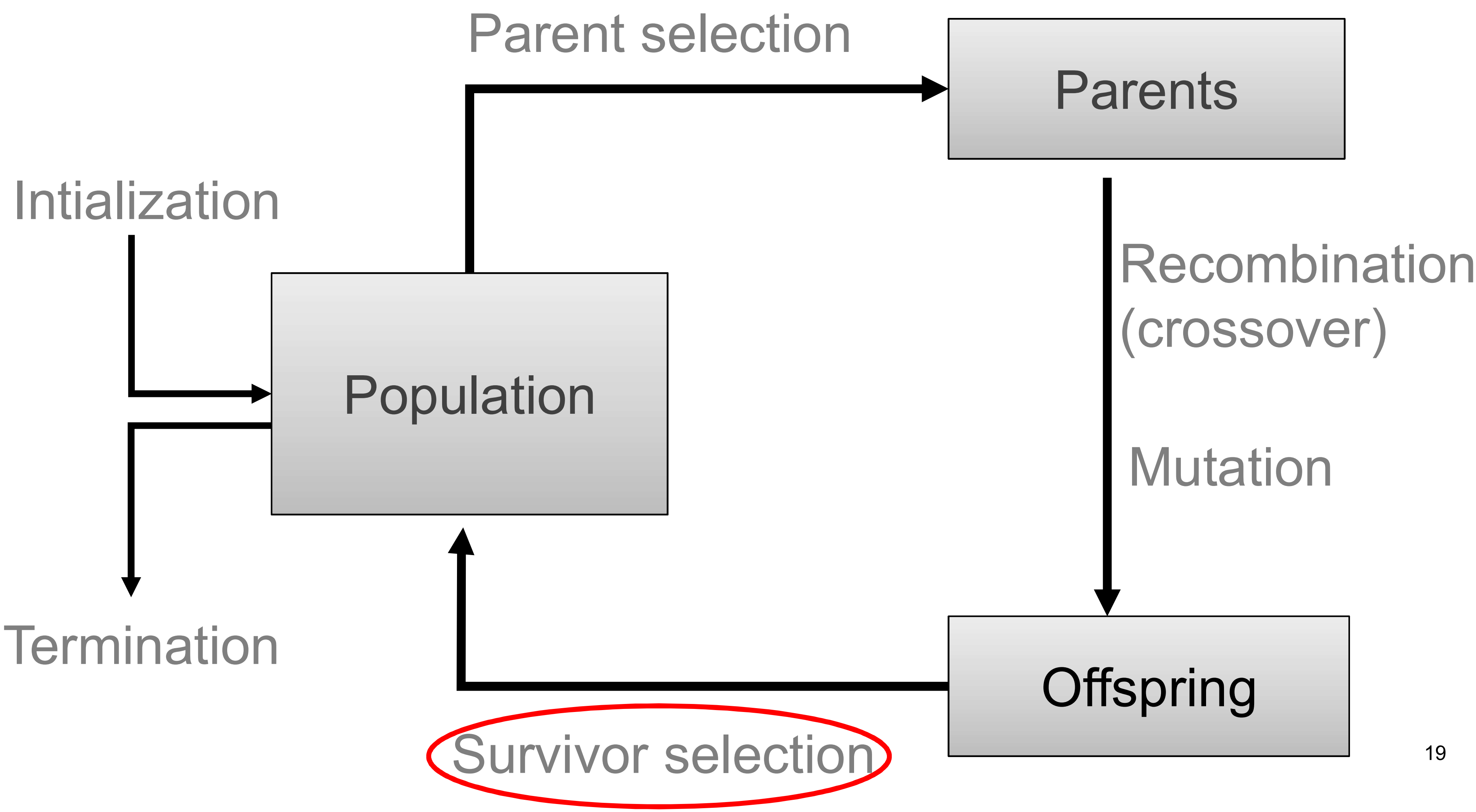
- Probability of selecting *i* will depend on:
  - Rank of *i*
  - Size of sample *k*
    - higher *k* increases selection pressure
  - Whether contestants are picked with replacement
    - Picking without replacement increases selection pressure
  - Whether fittest contestant always wins (deterministic) or this happens with probability *p*

Parent Selection: Uniform

$$P_{uniform}(i) = \frac{1}{\mu}$$

- Parents are selected by uniform random distribution whenever an operator needs one/some
- Uniform parent selection is unbiased - every individual has the **same probability** to be selected

Scheme of an EA: General scheme of EAs



Survivor Selection (Replacement)

- From a set of μ old solutions and λ offspring: Select a set of μ individuals **forming the next generation**
- Survivor selection can be divided into two approaches:
  - **Age-Based Replacement**
    - Fitness is not taken into account
  - **Fitness-Based Replacement**
    - Usually with deterministic elements

Fitness-based replacement (1/2)

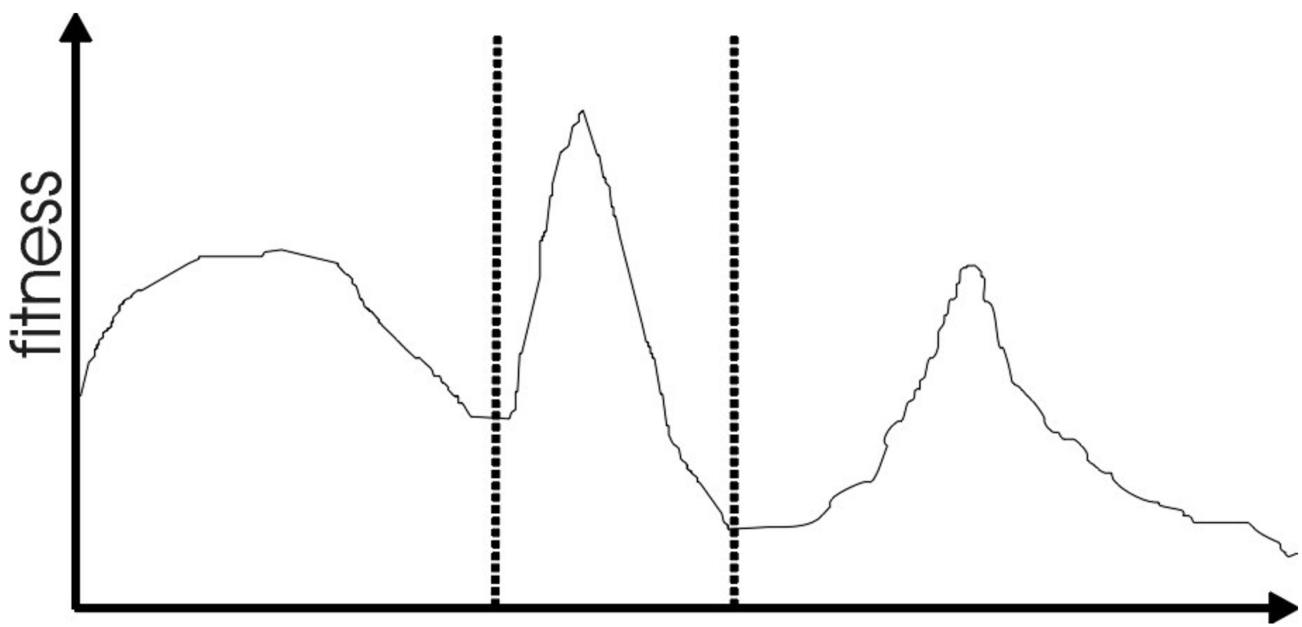
- Elitism
  - Always **keep** at least one copy of **the N fittest solution(s)** so far
  - Widely used in both population models (GGA, SSGA)
- Delete Worst
  - The worst λ individuals are replaced
- Round-robin tournament (from Evolutionary Programming)
  - Pairwise competitions in round-robin format:
    - Each individual x is **evaluated against q other** randomly chosen individuals in 1-on-1 tournaments
    - For each comparison, a "win" is assigned if x is better than its opponent
    - The μ solutions with the greatest number of wins are the winners of the tournament
  - Parameter q allows tuning selection pressure

Fitness-based replacement (2/2) (from Evolution Strategies)

- **(μ,λ)-selection** (best candidates can be lost)
  - based on the set of **children only** ($\lambda > \mu$)
  - choose the **best** μ offspring for next generation
- **(μ+λ)-selection** (elitist strategy)
  - based on the set of **parents and children**
  - choose the **best** μ offspring for next generation
- Often (μ,λ)-selection is preferred because it is better in leaving local optima

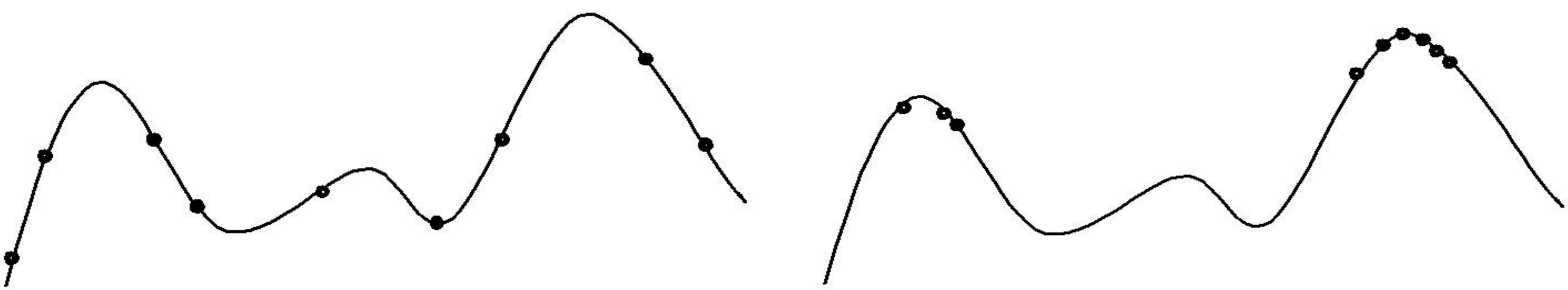
Multimodality

Most interesting problems have more than one locally optimal solution.



Multimodality

- Often might want to identify several possible peaks
- Different peaks may be different good ways to solve the problem.
- We therefore need methods to **preserve diversity** (instead of converging to one peak)



Approaches for Preserving Diversity: Introduction

- Explicit vs implicit
- Implicit approaches:
  - Impose an equivalent of geographical separation
  - Impose an equivalent of speciation
- Explicit approaches
  - Make similar individuals compete for resources (fitness)
  - Make similar individuals compete with each other for survival

Explicit Approaches for Preserving Diversity: Fitness Sharing (1/2)

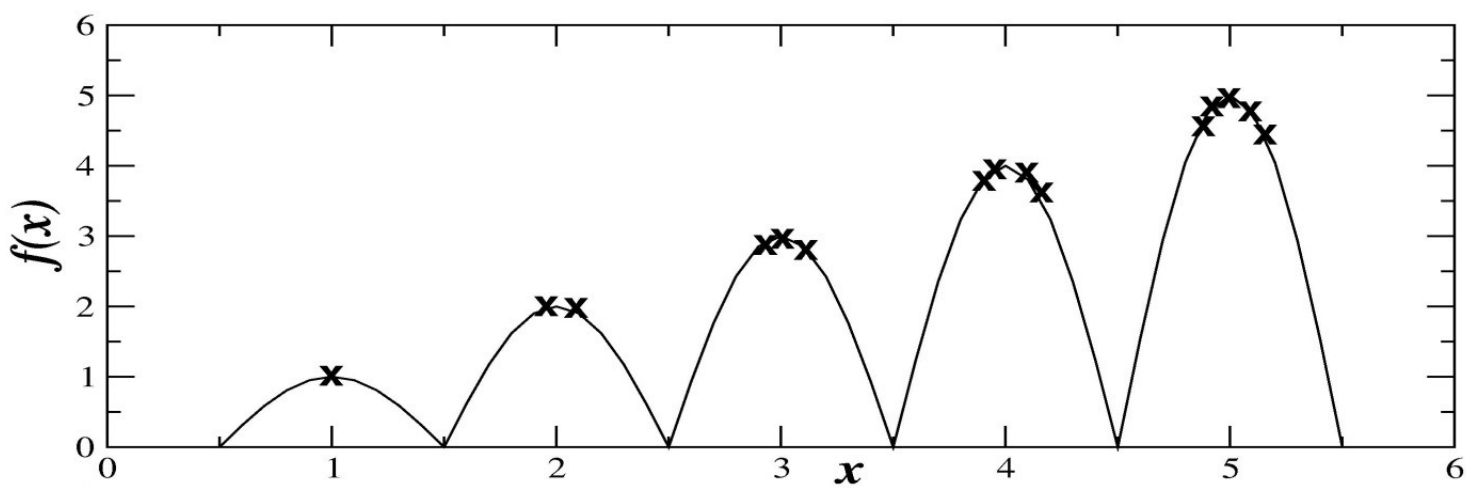
- Restricts the number of individuals within a given niche by "sharing" their fitness
- Need to set the size of the niche σ_{share} in either genotype or phenotype space
- run EA as normal but after each generation set

$$f'(i) = \frac{f(i)}{\sum_{j=1}^{\mu} sh(d(i,j))} \qquad sh(d) = \begin{cases} 1 - d/\sigma & d \leq \sigma \\ 0 & otherwise \end{cases}$$

Explicit Approaches for Preserving Diversity: Fitness Sharing (2/2)

$$f'(i) = \frac{f(i)}{\sum_{j=1}^{\mu} sh(d(i,j))}$$

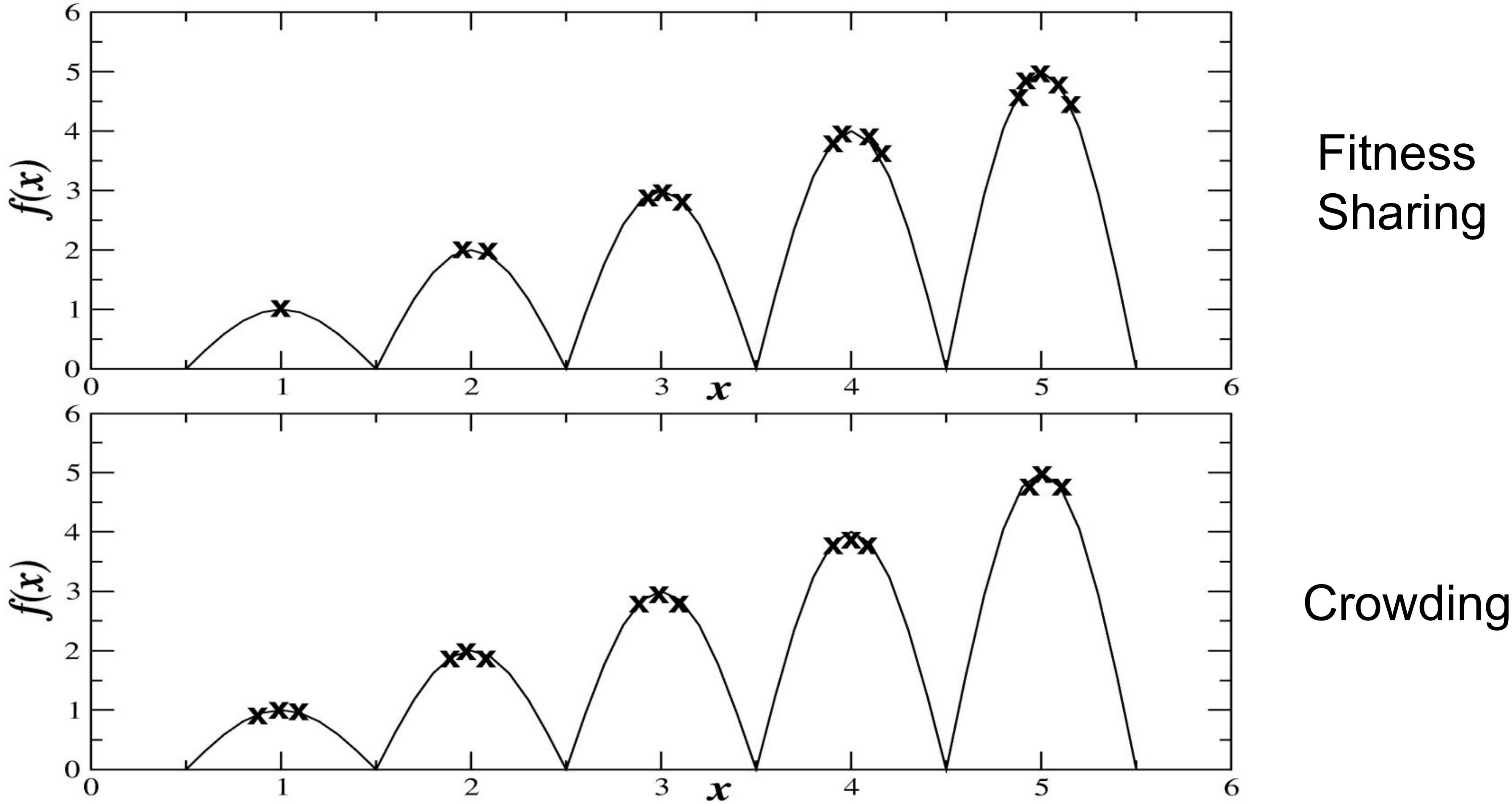
$$sh(d) = \begin{cases} 1 - d/\sigma & d \le \sigma \\ 0 & otherwise \end{cases}$$



Explicit Approaches for Preserving Diversity: Crowding

- Idea: New individuals replace *similar* individuals
- Randomly shuffle and pair parents, produce 2 offspring
- Each offspring competes with their **nearest** parent for survival (using a distance measure)
- Result: Even distribution among niches.

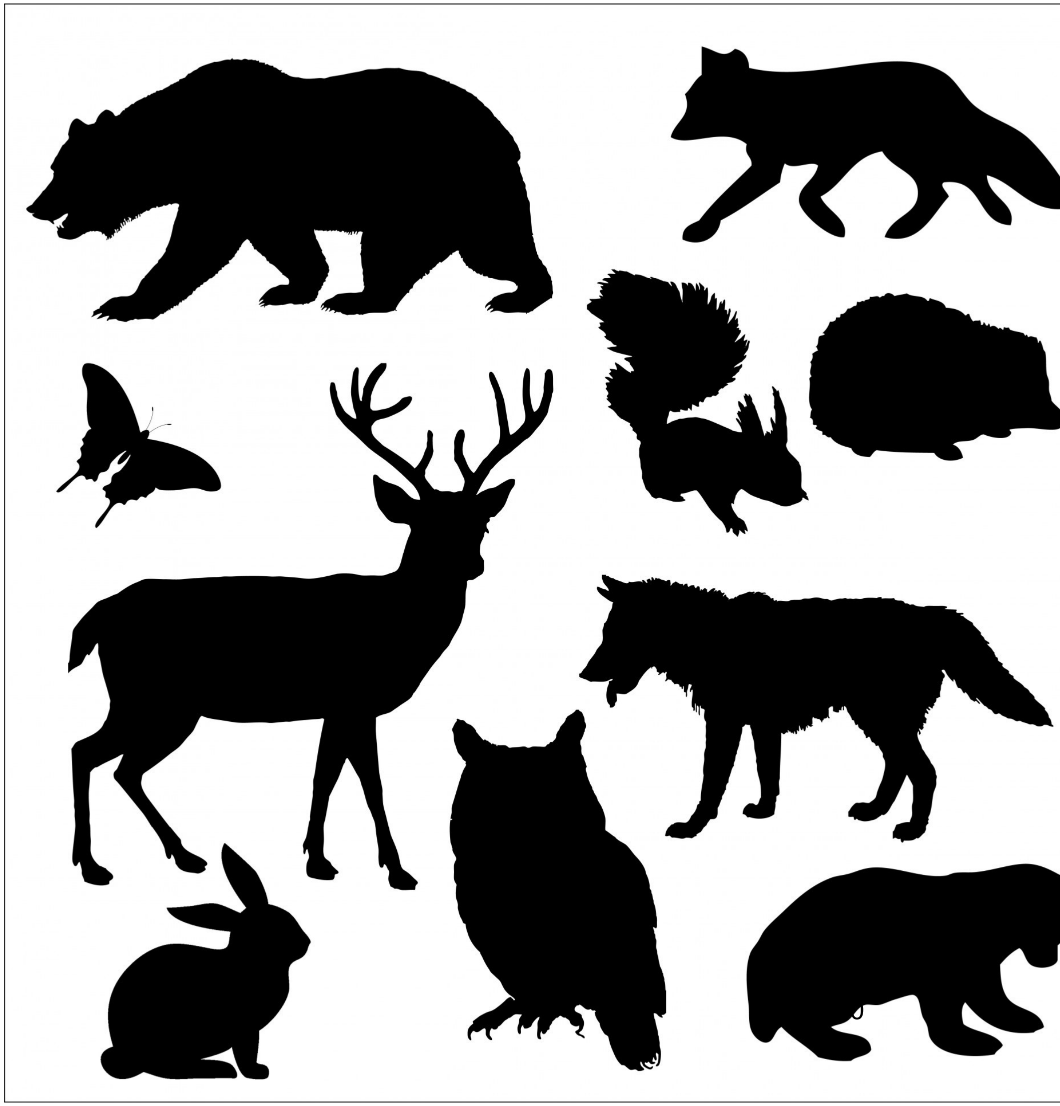
Explicit Approaches for Preserving Diversity: Crowding vs Fitness sharing



Observe the number of individuals per niche

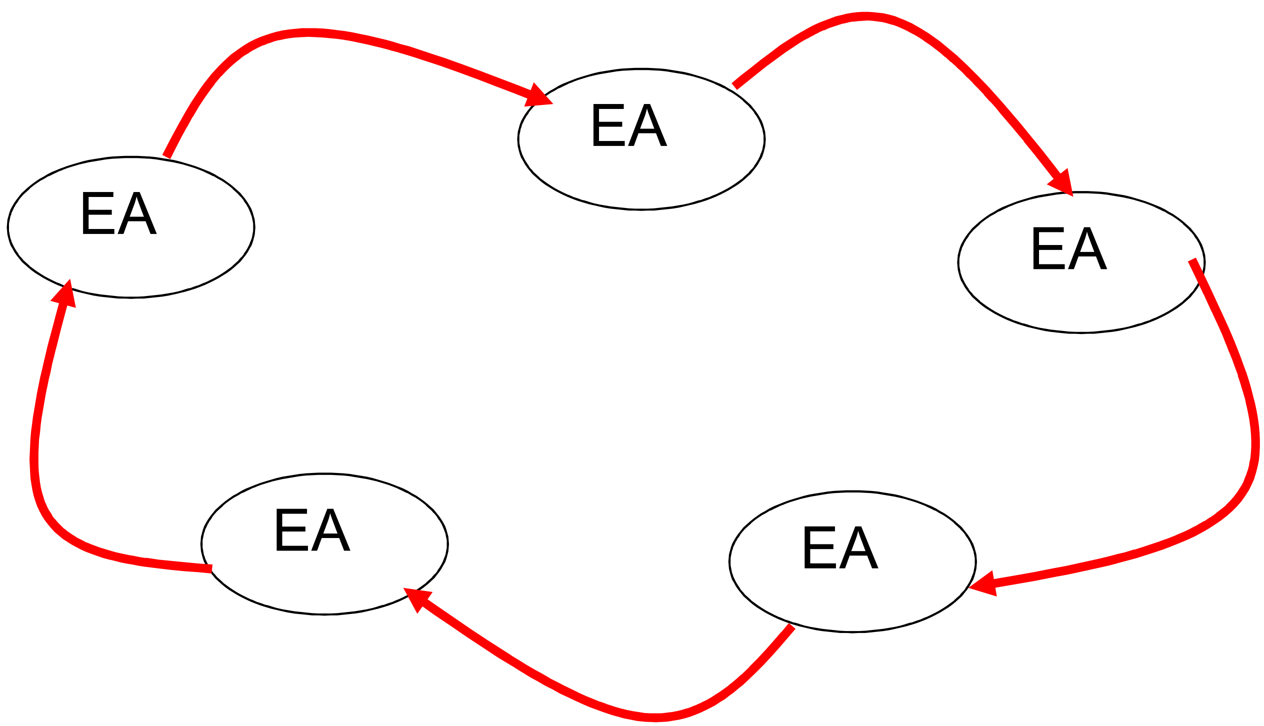
Implicit Approaches for Preserving Diversity: Automatic Speciation

- Either only mate with genotypically / phenotypically similar members or
- Add species-tags to genotype
  - initially randomly set
  - when selecting partner for recombination, only pick members with a good match



Implicit Approaches for Preserving Diversity: Geographical Separation

- "Island" Model Parallel EA
- Periodic migration of individual solutions between populations



Implicit Approaches for Preserving Diversity: “Island” Model Parallel EAs

- Run multiple populations in parallel
- After a (usually fixed) number of generations (an ***Epoch***), exchange individuals with neighbours
- Repeat until ending criteria met
- Partially inspired by parallel/clustered systems

Island Model: Parameters

- How often to exchange individuals ?
  - too quick and all sub-populations converge to same solution
  - too slow and waste time
  - can do it adaptively (stop each pop when no improvement for (say) 25 generations)
- Operators can differ between the sub-populations

Chapter 6: Popular Evolutionary Algorithm Variants

Historical EA variants:

- Genetic Algorithms
- Evolution Strategies
- Evolutionary Programming
- Genetic Programming

Algorithm	Chromosome Representation	Crossover	Mutation
Genetic Algorithm (GA)	Array	X	X
Genetic Programming (GP)	Tree	X	X
Evolution Strategies (ES)	Array	(X)	X
Evolutionary Programming (EP)	No constraints	-	X

Genetic Algorithms: Overview Simple GA (1/2)

- Developed: USA in the 1960’s
- Early names: Holland, DeJong, Goldberg
- Typically applied to:
  - discrete function optimization
  - benchmark for comparison with other algorithms
  - straightforward problems with binary representation
- Features:
  - not too fast
  - missing new variants (elitism, sus)
  - often modelled by theorists

Genetic Algorithms: Overview Simple GA (2/2)

- Holland's original GA is now known as the simple genetic algorithm (SGA)
- Other GAs use different:
  - Representations
  - Mutations
  - Crossovers
  - Selection mechanisms

Genetic Algorithms: SGA reproduction cycle

- **Select parents** for the mating pool (size of mating pool = population size)
- Shuffle the mating pool
- **Apply crossover** for each consecutive pair with probability p_c, otherwise copy parents
- **Apply mutation** for each offspring (bit-flip with probability p_m independently for each bit)
- **Replace the whole population** with the resulting offspring

Genetic Algorithms: An example after Goldberg ’89

- Simple problem: max x^2 over {0,1,…,31}
- GA approach:
  - Representation: binary code, e.g., 01101 $\leftrightarrow$ 13
  - Population size: 4
  - 1-point x-over, bitwise mutation
  - Roulette wheel selection
  - Random initialisation
- We show one generational cycle done by hand

X^2 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

X^2 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

X^2 example: Mutation

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	$\boxed{1}$ 1 1 0 0	26	676
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 $\boxed{1}$ 0 0	18	324
Sum				2354
Average				588.5
Max				729

Genetic Algorithms: The simple GA

- Has been subject of many (early) studies
  - still often used as benchmark for novel GAs
- Shows many shortcomings, e.g.,
  - Representation is too restrictive
  - Mutation & crossover operators only applicable for bit-string & integer representations
  - Selection mechanism sensitive for converging populations with close fitness values
  - Generational population model can be improved with explicit survivor selection

Genetic Algorithms: Simple GA (SGA) summary

Representation	Bit-strings
Recombination	1-Point crossover
Mutation	Bit flip
Parent selection	Fitness proportional – implemented by Roulette Wheel
Survivor selection	Generational

Evolution Strategies: Quick overview

- Developed: Germany in the 1960's by Rechenberg and Schwefel
- Typically applied to numerical optimisation
- Attributed features:
  - fast
  - good optimizer for real-valued optimisation
  - relatively much theory
- Special:
  - self-adaptation of (mutation) parameters standard

Evolution Strategies: Example (1+1) ES

- Task: minimise $f : R^n \rightarrow R$
- Algorithm: "two-membered ES" using
  - Vectors from R^n directly as chromosomes
  - Population size 1
  - Only mutation creating one child
  - Greedy selection

Evolution Strategies: Representation

- Chromosomes consist of two parts:
  - Object variables: $x_1,\ldots,x_n$
  - Strategy parameters (mutation rate, etc): $p_1,\ldots,p_m$

- Full size: $\langle\, x_1,\ldots,x_n, p_1,\ldots,p_n \,\rangle$

Evolution Strategies: Parent selection

- Parents are selected by **uniform random distribution** whenever an operator needs one/some
- Thus: ES parent selection is unbiased - every individual has the **same probability** to be selected

$$P_{uniform}(i) = \frac{1}{\mu}$$

Evolution Strategies: Recombination

- Two parents create one child
- Acts per variable / position by either
  - Intermediary crossover, or
  - Discrete crossover
- From two or more parents by either:
  - Local recombination: Two parents make a child
  - Global recombination: Selecting two parents randomly for each gene

Evolution Strategies: Names of recombinations

	Two fixed parents	Two parents selected for each i
$z_i = (x_i + y_i)/2$	Local intermediary	Global intermediary
z_i is x_i or y_i chosen randomly	Local discrete	Global discrete

Evolution Strategies: ES summary

Representation	Real-valued vectors
Recombination	Discrete or intermediary
Mutation	Gaussian perturbation
Parent selection	Uniform random
Survivor selection	(μ,λ) or $(\mu+\lambda)$

Evolutionary Programming: Quick overview

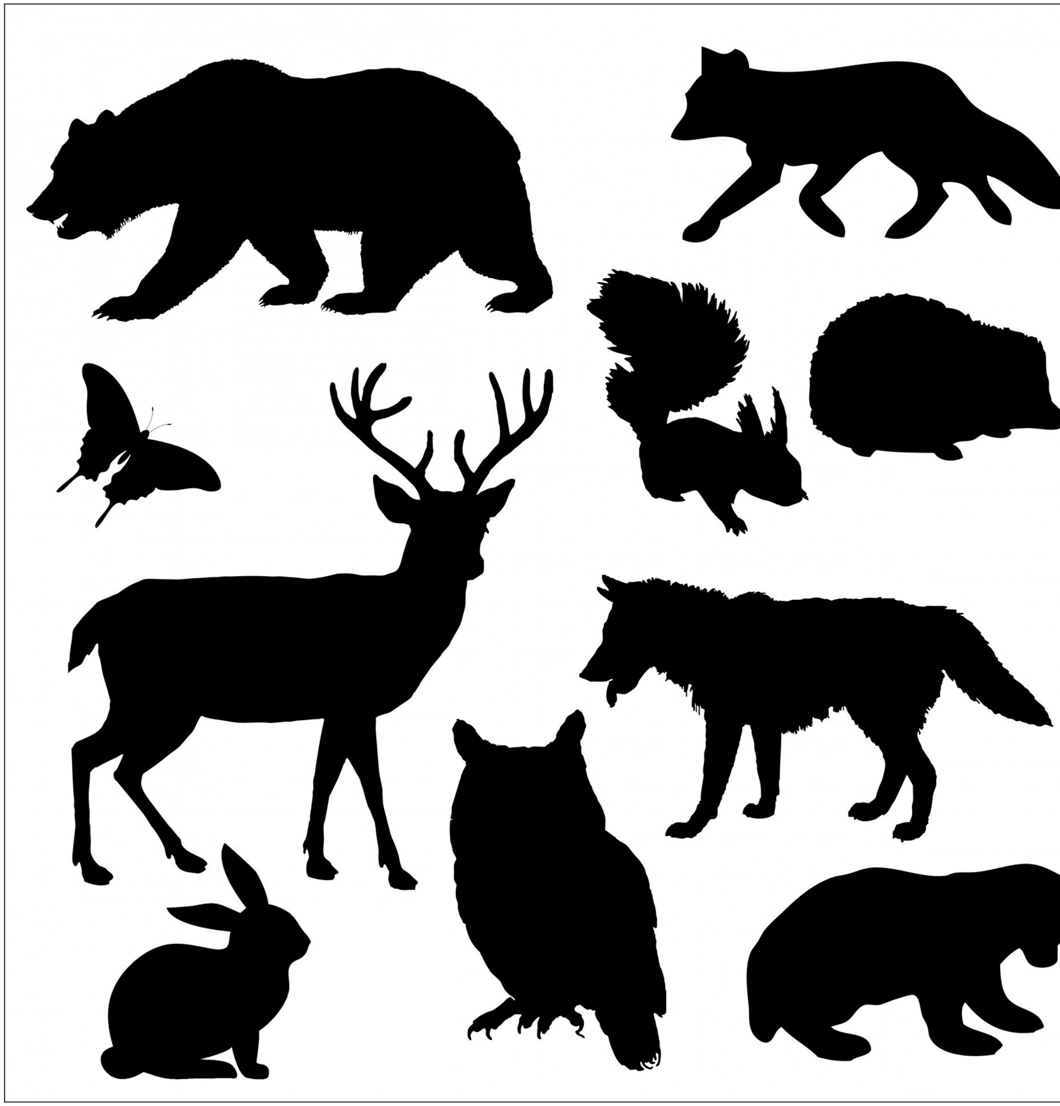
- Developed: USA in the 1960's by Fogel et al.
- Typically applied to:
  - traditional EP: prediction by finite state machines
  - contemporary EP: (numerical) optimization
- Attributed features:
  - very open framework: any representation and mutation op's OK
  - Contemporary EP has almost merged with ES
- Special:
  - **no recombination**
  - self-adaptation of parameters standard (contemporary EP)

Evolutionary Programming: Representation

- For continuous parameter optimisation
- Chromosomes consist of two parts:
  - Object variables: $x_1,\ldots,x_n$
  - Mutation step sizes: $\sigma_1,\ldots,\sigma_n$
- Full size: $\langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n \rangle$

Evolutionary Programming: Recombination

- None
- Rationale: one point in the search space stands for a species, not for an individual and there can be no crossover between species



Evolutionary Programming: Selection

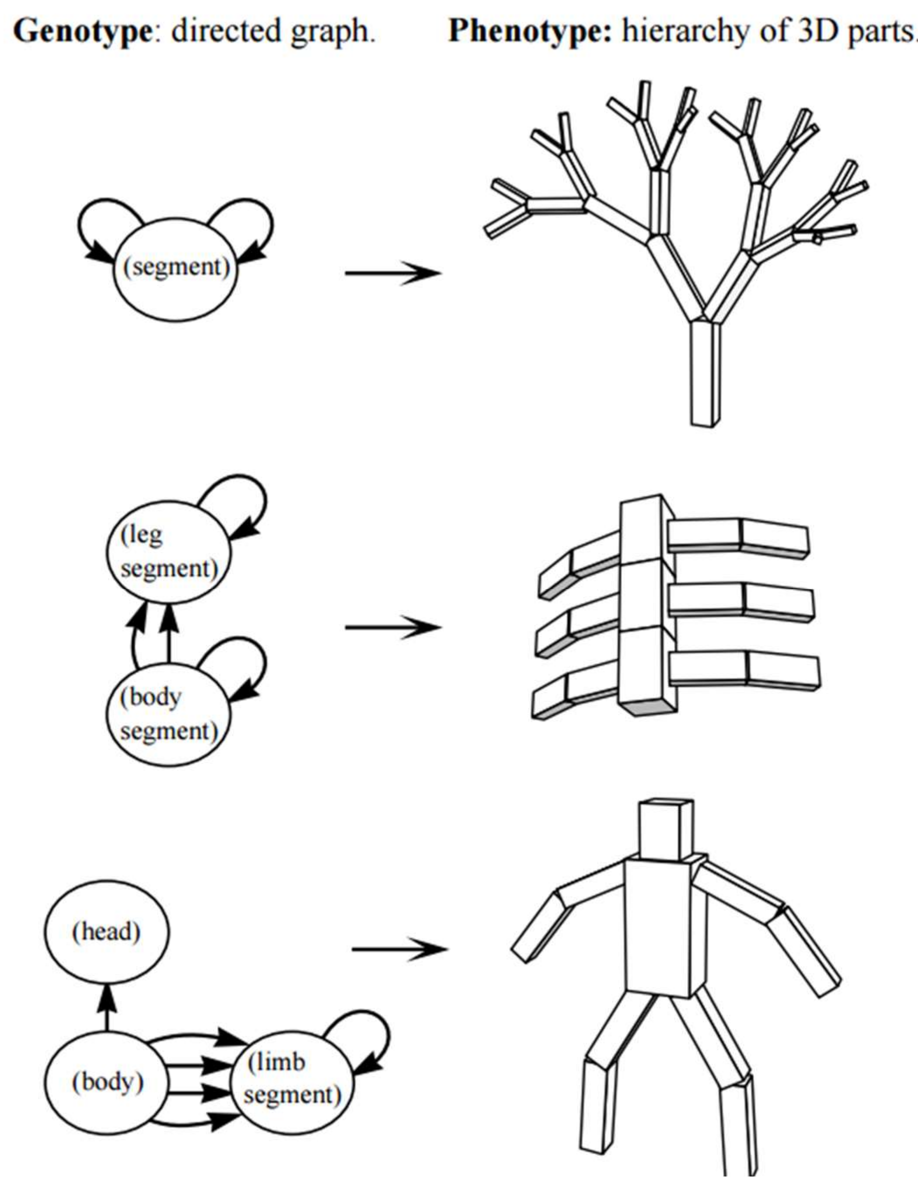
- Each individual creates one child by mutation
  - Deterministic
  - Not biased by fitness

- Parents and offspring compete for survival in round-robin tournaments.

Evolutionary Programming: Summary

Representation	Real-valued vectors
Recombination	None
Mutation	Gaussian perturbation
Parent selection	Deterministic (each parent one offspring)
Survivor selection	Probabilistic ($\mu+\lambda$)

Virtual Creatures (Karl Sims, 1994)



Genetic Programming: Quick overview

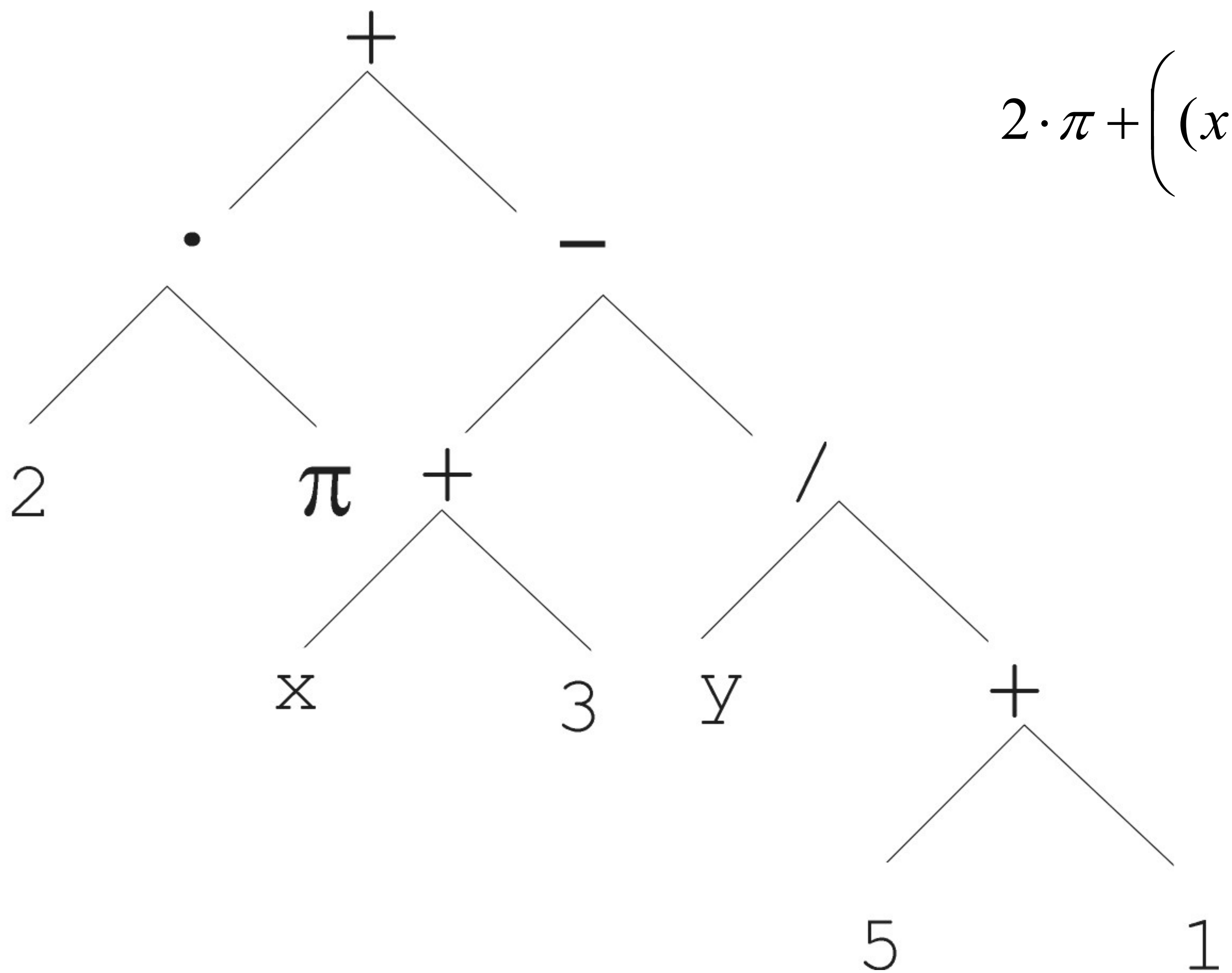
- Developed: USA in the 1990’s by Koza
- Typically applied to:
  - machine learning tasks (prediction, classification…)
- Attributed features:
  - “automatic evolution of computer programs”
  - needs huge populations (thousands)
  - slow
- Special:
  - non-linear chromosomes: trees
  - mutation possible but not necessary

Tree Representation

- Trees are a universal form, e.g. consider
- Arithmetic formula: $2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$
- Logical formula: $(x \wedge \text{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$
- Program:

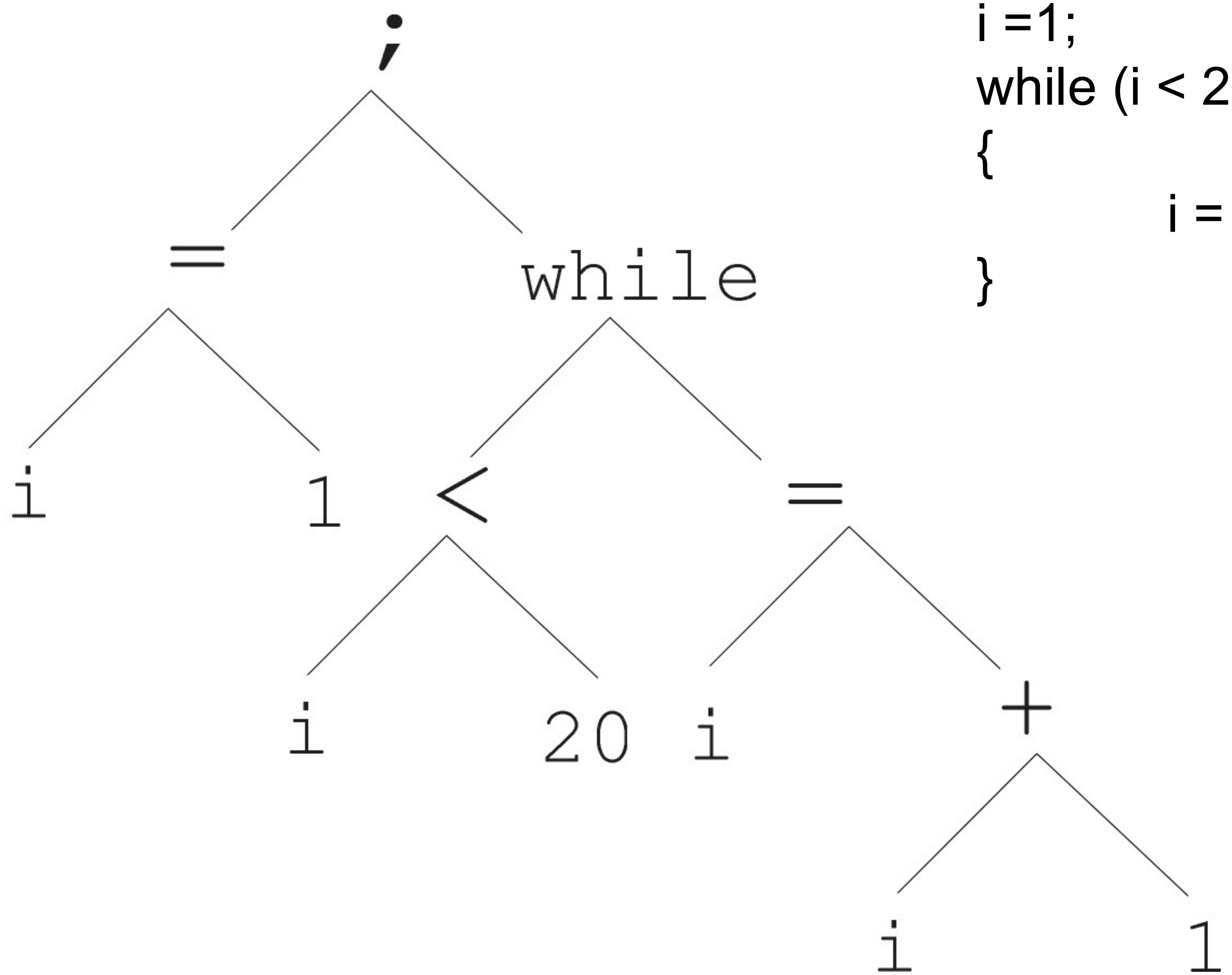
```
i =1;
while (i < 20)
{
        i = i +1
}
```

Tree Representation



$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

Tree Representation

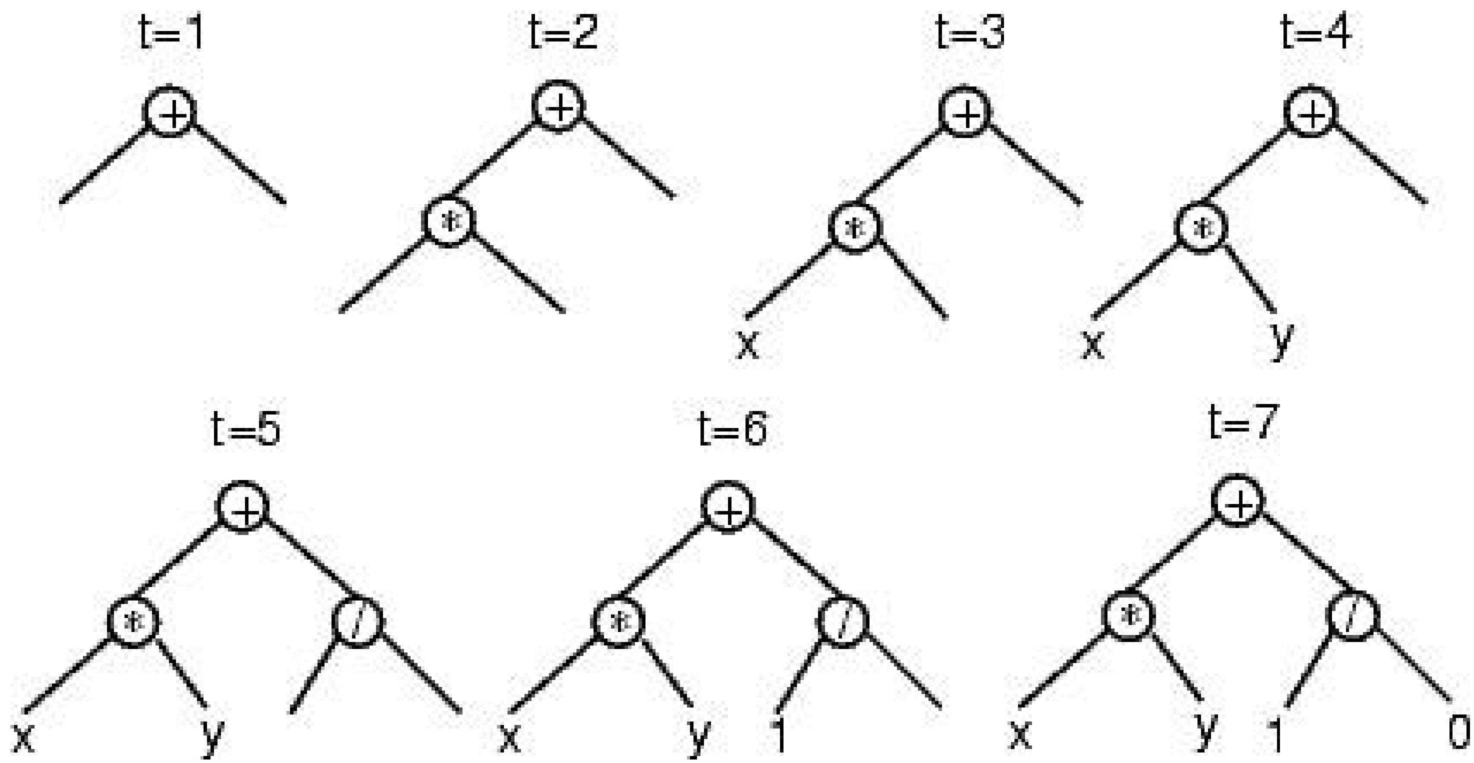



```
i =1;
while (i < 20)
{
        i = i +1
}
```

Tree Representation

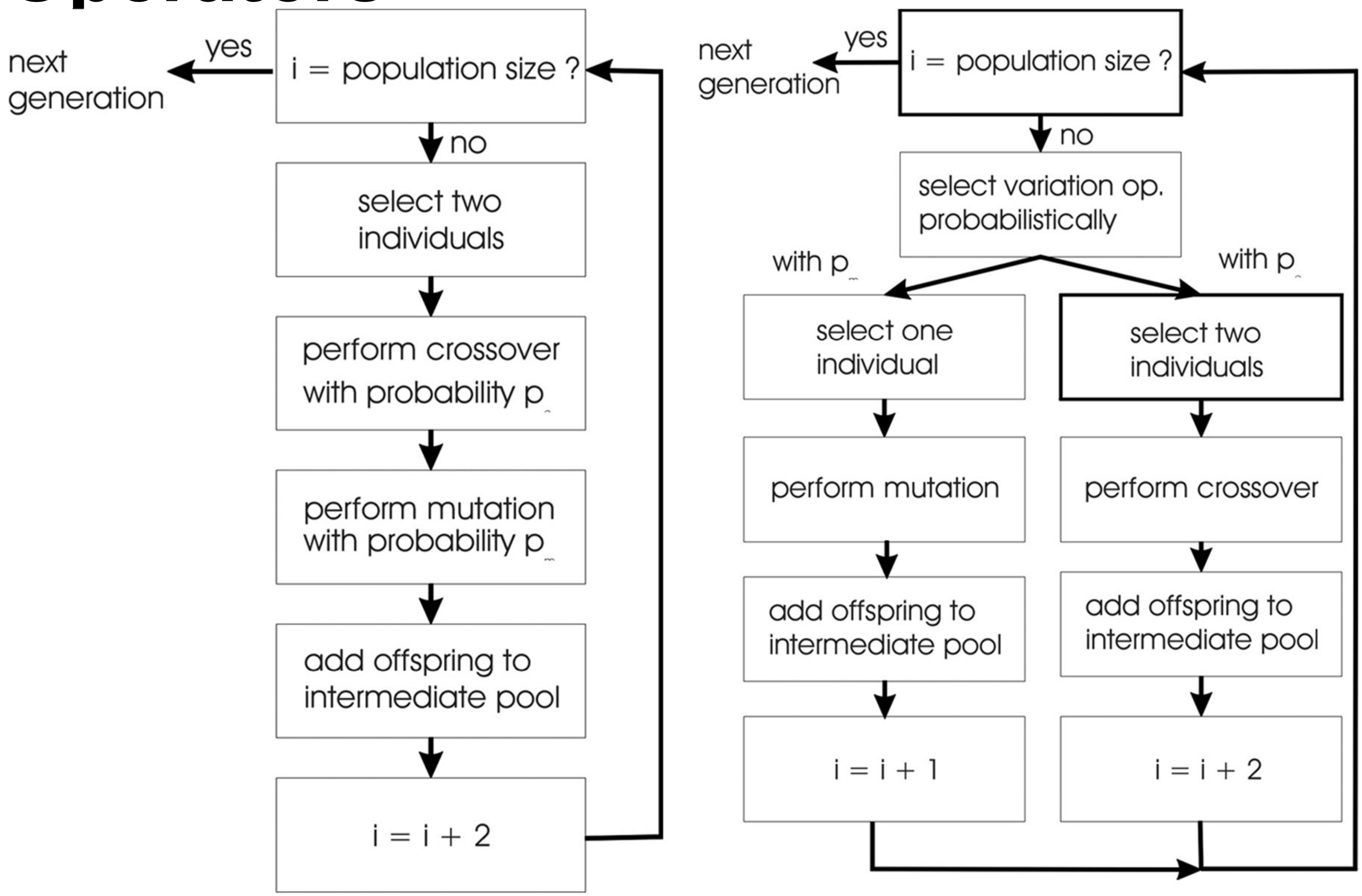
- In GA, ES, EP chromosomes are linear structures (bit strings, integer string, real-valued vectors, permutations)
- Tree shaped chromosomes are non-linear structures
- In GA, ES, EP the size of the chromosomes is fixed
- Trees in GP (Genetic Programming) may vary in depth and width

Example of how to initialize trees: Full initialisation to depth 2



Source: www.gp-field-guide.org.uk

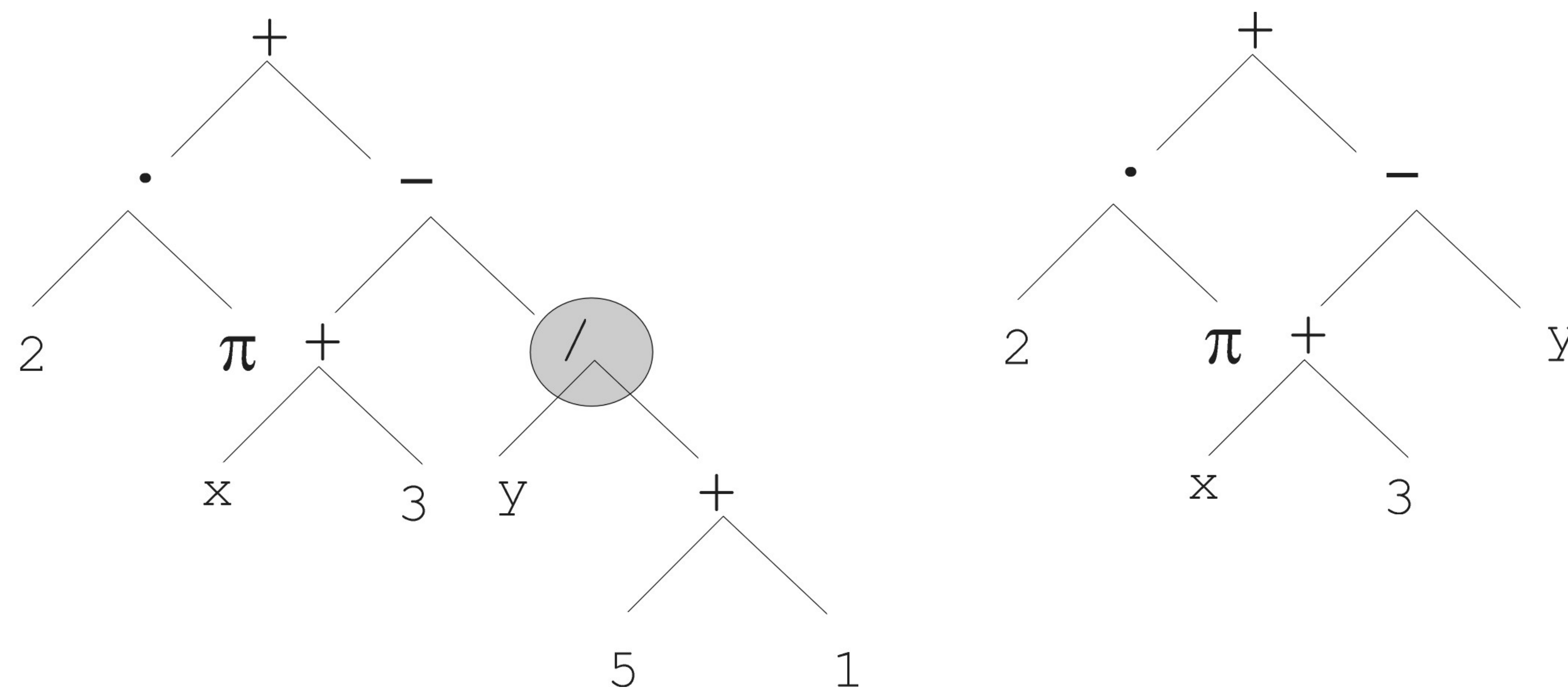
Genetic Programming: Variation Operators



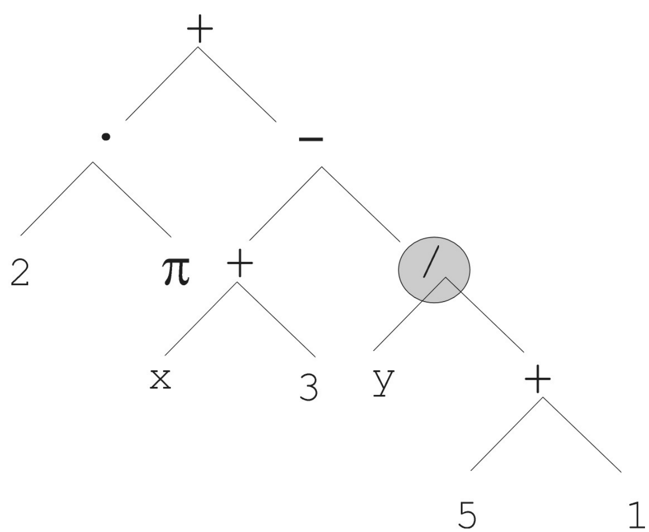
GA flowchart GP flowchart

Genetic Programming: Mutation

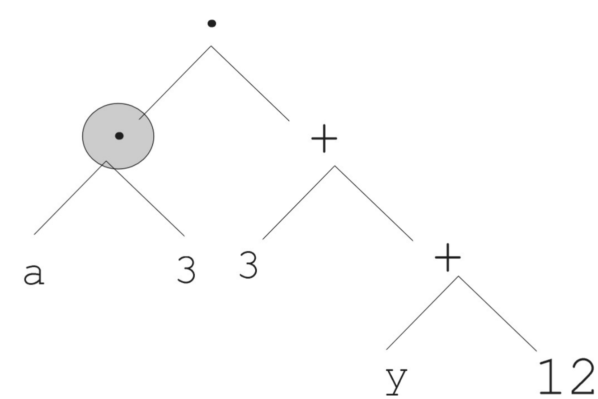
- Most common mutation: replace randomly chosen subtree by randomly generated tree



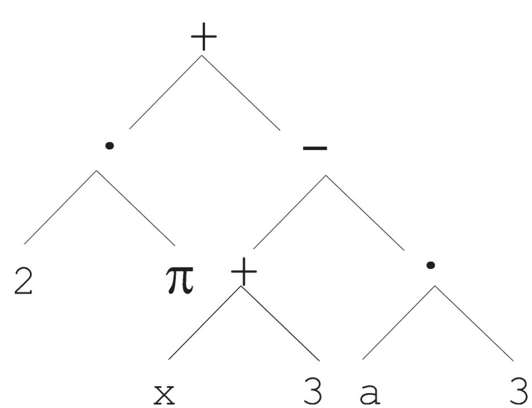
Genetic Programming: Recombination



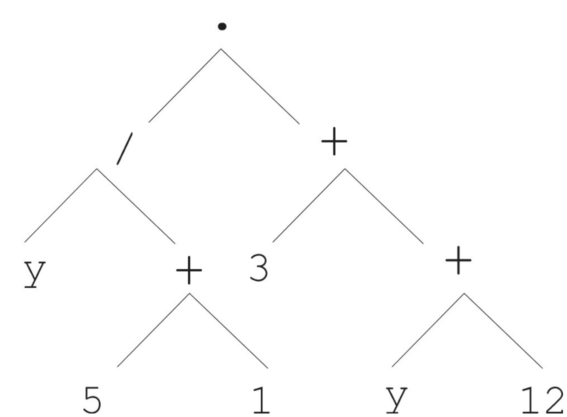
Parent 1



Parent 2

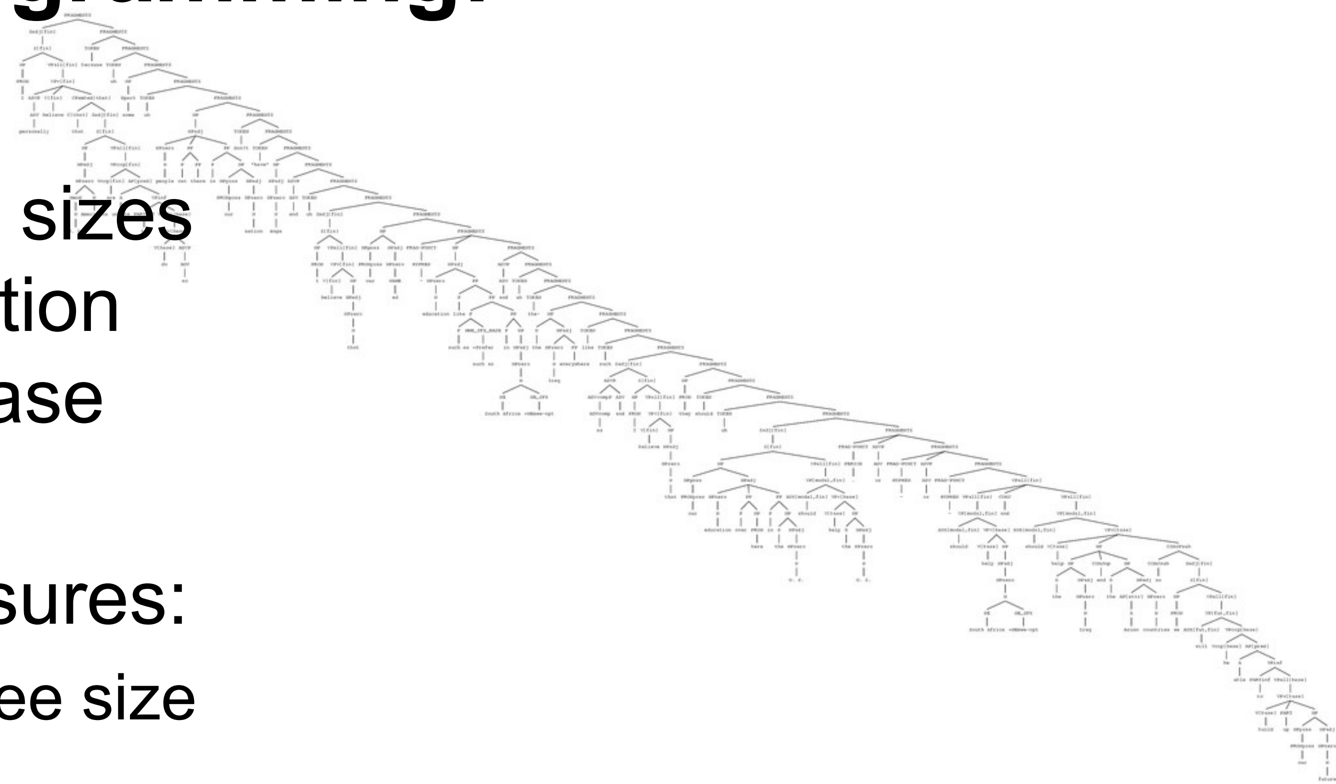


Child 1



Child 2

Genetic Programming: Bloat



- Average tree sizes in the population tend to increase over time
- Countermeasures:
  - Maximum tree size
  - Parsimony pressure: penalty for being oversized

Genetic Programming: Summary

Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

Summary: The standard EA variants

Name	Representation	Crossover	Mutation	Parent selection	Survivor selection	Specialty
Genetic Algorithm	Usually fixed-length vector	Any or none	Any	Any	Any	None
Evolution Strategies	Real-valued vector	Discrete or intermediate recombination	Gaussian	Random draw	Best N	Strategy parameters
Evolutionary Programming	Real-valued vector	None	Gaussian	One child each	Tournament	Strategy parameters
Genetic Programming	Tree	Swap sub-tree	Replace sub-tree	Usually fitness proportional	Generational replacement	None