UiO : Department of Informatics
University of Oslo

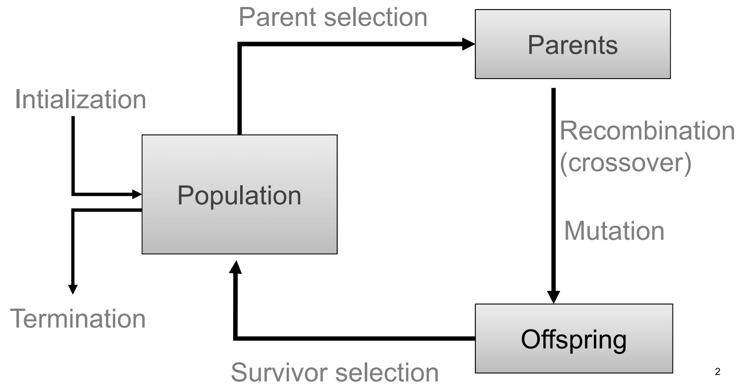
INF3490 - Biologically inspired computing

Lecture 3: Eiben and Smith, chapter 5-6

Evolutionary Algorithms - Population management and popular algorithms

Kai Olav Ellefsen

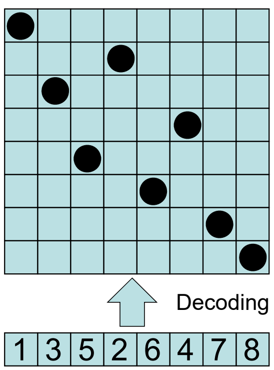
Repetition: General scheme of EAs



Repetition: Genotype & Phenotype

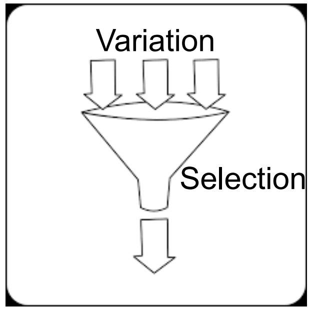
Phenotype:
A solution representation we can **evaluate**

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



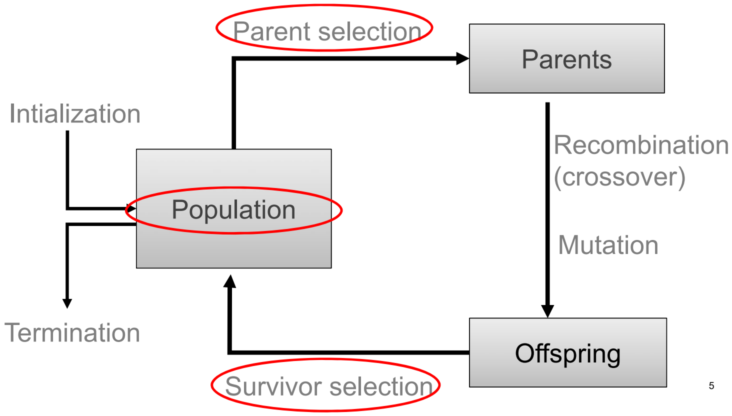
Chapter 5: Fitness, Selection and Population Management

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



UiO Department of Informatics
University of Oslo

Scheme of an EA: General scheme of EAs



UiO Department of Informatics
University of Oslo

Population Management Models: Introduction

- Two different population management models exist:
  - **Generational model**
    - each individual survives for exactly one generation
    - the entire set of parents is replaced by the offspring
  - **Steady-state model**
    - one offspring is generated per generation
    - one member of population replaced
- Generation Gap
  - The proportion of the population replaced
  - Parameter = 1.0 for G-GA, = 1/pop_size for SS-GA

UiO Department of Informatics
University of Oslo

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

UiO Department of Informatics
University of Oslo

Effect of Selection Pressure

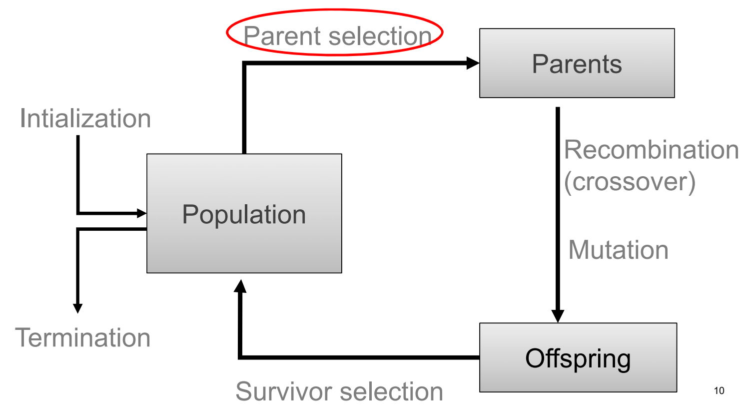
- Low Pressure
- High Pressure

UiO : Department of Informatics
University of Oslo

Why Not Always High Selection Pressure?

UiO : Department of Informatics
University of Oslo

Scheme of an EA: General scheme of EAs



UiO : Department of Informatics
University of Oslo

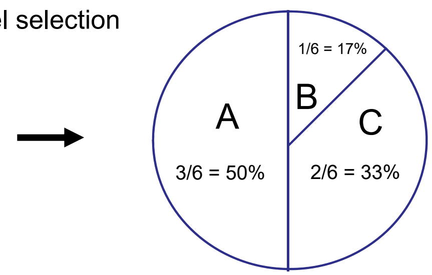
Parent Selection: Fitness-Proportionate Selection

Example: roulette wheel selection

fitness(A) = 3

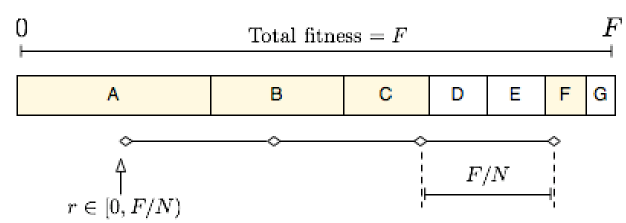
fitness(B) = 1

fitness(C) = 2



UiO : Department of Informatics
University of Oslo

Stochastic Universal Sampling



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

UiO : Department of Informatics
University of Oslo

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

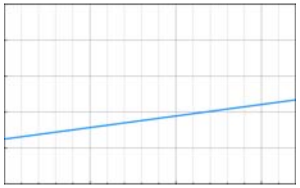
UiO : Department of Informatics
University of Oslo

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

UiO : Department of Informatics
University of Oslo

Rank-based Selection: Linear Ranking



$$P_{lin-rank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}$$

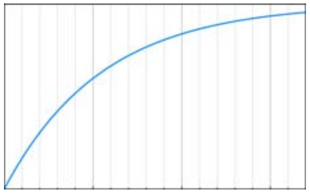
- Parameterised by factor *s:* $1 < s \leq 2$
  - Tunes selection pressure
- Simple 3 member example

Individual	Fitness	Rank	P_{selFP}	P_{selLR} $(s=2)$	P_{selLR} $(s=1.5)$
A	1	0	0.1	0	0.167
B	4	1	0.4	0.33	0.33
C	5	2	0.5	0.67	0.5
Sum	10		1.0	1.0	1.0

UiO : Department of Informatics
University of Oslo

Rank-based selection: Exponential Ranking

$$P_{\exp-rank}(i) = \frac{1-e^{-i}}{c}$$



- Linear Ranking is limited in selection pressure
- Exponential Ranking can allocate more than 2 copies to fittest individual
- Normalise constant factor *c* according to population size

UiO : Department of Informatics
University of Oslo

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

17

UiO : Department of Informatics
University of Oslo

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

18

UiO : Department of Informatics
University of Oslo

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*

19

UiO : Department of Informatics
University of Oslo

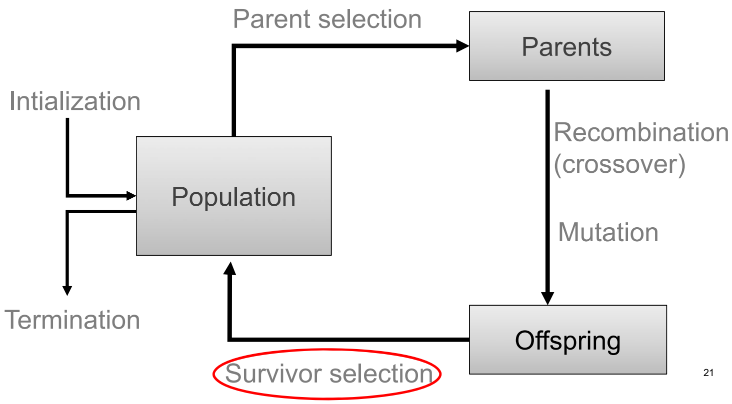
Parent Selection: Uniform

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

- 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

20

Scheme of an EA:
General scheme of EAs



Survivor Selection (Replacement)

- From a set of μ parents 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
    - In SS-GA can implement as “delete-random” (not recommended) or as first-in-first-out (a.k.a. delete-oldest)
  - **Fitness-Based Replacement**

Fitness-based replacement (1/2)

- Elitism
  - Always **keep** at least one copy of **the fittest solution** so far
  - Widely used in both population models (GGA, SSGA)
- Delete Worst
  - The worst λ individuals are replaced
- Round-robin tournament (from EP)
  - 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
  - Typically *q* = 10

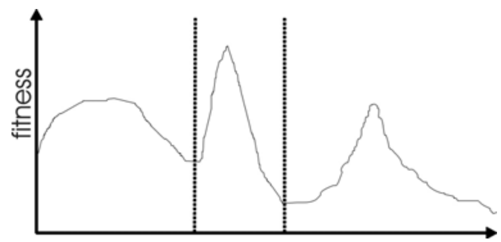
Fitness-based replacement (2/2)
(from ES)

- **(μ,λ)-selection** (best candidates can be lost)
  - based on the set of **children only** (λ > μ)
  - 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

UiO Department of Informatics
University of Oslo

Multimodality

Most interesting problems have more than one locally optimal solution.



UiO Department of Informatics
University of Oslo

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)

UiO Department of Informatics
University of Oslo

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

UiO Department of Informatics
University of Oslo

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

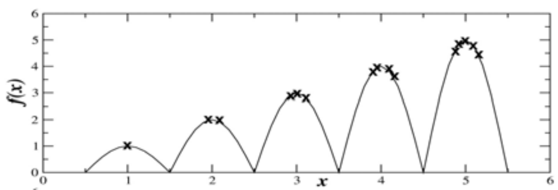
- Restricts the number of individuals within a given niche by "sharing" their fitness
- Need to set the size of the niche σ_{share} in either genotype or phenotype space
- run EA as normal but after each generation set

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

UiO Department of Informatics
University of Oslo

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

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



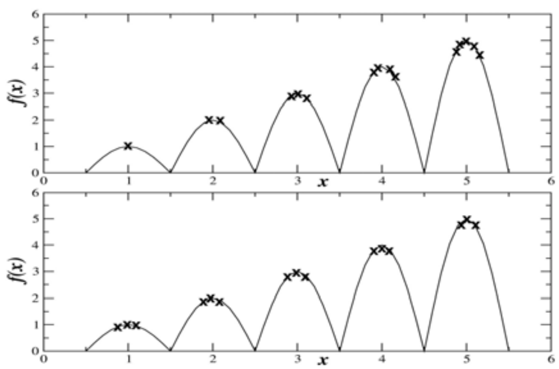
UiO Department of Informatics
University of Oslo

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.

UiO Department of Informatics
University of Oslo

Explicit Approaches for Preserving Diversity: Crowding vs Fitness sharing



Fitness Sharing

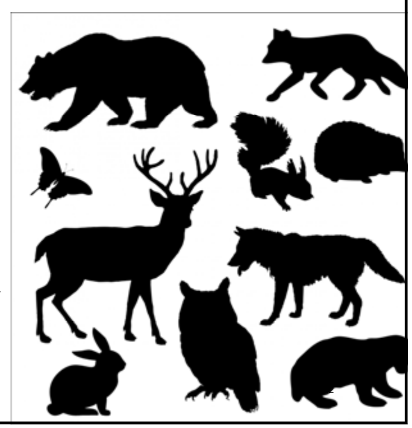
Crowding

Observe the number of individuals per niche

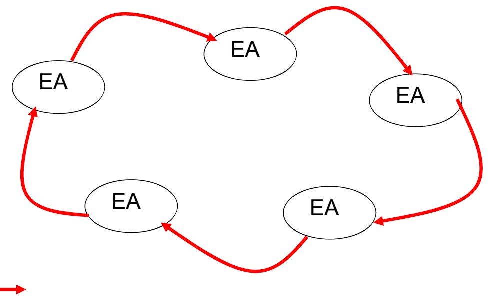
UiO Department of Informatics
University of Oslo

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: "Island" Model Parallel EAs



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

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

- 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

UiO : Department of Informatics
University of Oslo

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

UiO : Department of Informatics
University of Oslo

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

UiO : Department of Informatics
University of Oslo

Genetic Algorithms: An example after Goldberg '89

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

UiO : Department of Informatics
University of Oslo

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	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 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

UiO : Department of Informatics
University of Oslo

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

UiO : Department of Informatics
University of Oslo

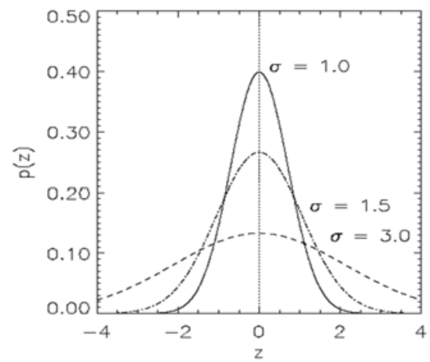
Evolution Strategies: Representation

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

UiO : Department of Informatics
University of Oslo

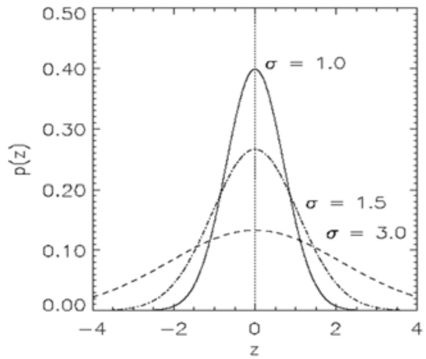
Evolution Strategies: Adaptive Mutation

- z values drawn from normal distribution $N(\xi,\sigma)$
  - mean ξ is set to 0
  - variation σ is called **mutation step size**
- σ is varied on the fly by the "1/5 success rule"



The "1/5 success rule"

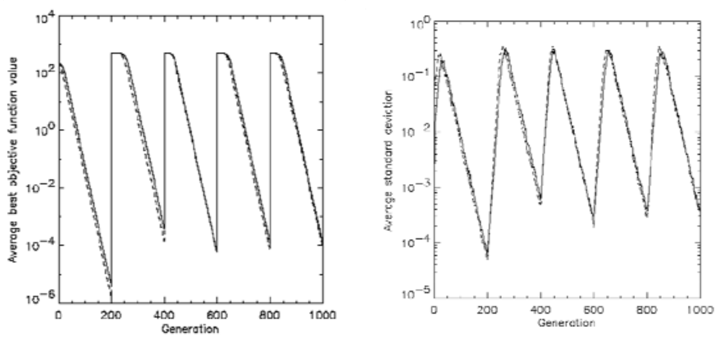
- Goal: Balance exploration and exploitation
- Resets σ after every k iterations by
  - $\sigma = \sigma / c$ if $p_s > 1/5$
  - $\sigma = \sigma \cdot c$ if $p_s < 1/5$
  - $\sigma = \sigma$ if $p_s = 1/5$
- where p_s is the % of successful mutations, $0.8 \le c \le 1$



Evolution Strategies: Self-adaptation illustrated (1/2)

- Given a dynamically changing fitness landscape (optimum location shifted every 200 generations)
- Self-adaptive ES is able to
  - follow the optimum and
  - adjust the mutation step size after every shift !

Evolution Strategies: Self-adaptation illustrated cont'd (2/2)



Changes in the fitness values (left) and the mutation step sizes (right)

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

Evolution Strategies: Recombination

- Two parents create one child
- Acts per variable / position by either
  - Averaging parental values, or
  - Selecting one of the parental values
- 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

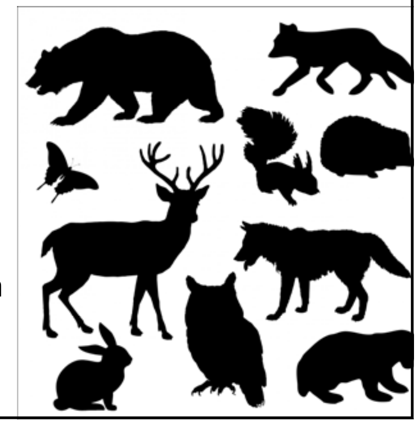
- Developed: USA in the 1960's by Fogel et al.
- Typically applied to:
  - traditional EP: prediction by finite state machines
  - contemporary EP: (numerical) optimization
- Attributed features:
  - very open framework: any representation and mutation op's OK
  - Contemporary EP has almost merged with 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

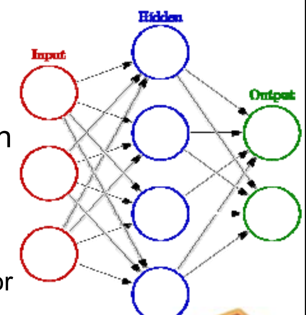


Evolutionary Programming: Selection

- Each individual creates one child by mutation
  - Deterministic
  - Not biased by fitness
- Parents and offspring compete for survival in round-robin tournaments.

Evolutionary Programming: Evolving checkers player (Fogel'02) (1/2)

- Neural nets for evaluating future values of moves are evolved
- NNs have fixed structure with 5046 weights, these are evolved
- Representation:
  - vector of 5046 real numbers for NN weights
- Population size 15



Evolutionary Programming: Evolving checkers player (Fogel'02) (2/2)

- Tournament size q = 5
- Programs (with NN inside) play against other programs, no human trainer
- After 840 generation (6 months!) best strategy was tested against humans
- Program earned "expert class" ranking outperforming 99.61% of all rated players

Evolutionary Programming: Summary

Representation	Real-valued vectors
Recombination	None
Mutation	Gaussian perturbation
Parent selection	Deterministic (each parent one offspring)
Survivor selection	Probabilistic (μ+λ)

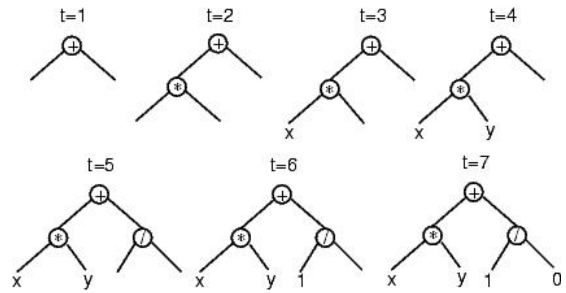
Genetic Programming: Quick overview

- 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

Genetic Programming: Initialisation

- Maximum initial depth of trees D_{max} is set
- Full method (each branch has depth = D_{max}):
  - nodes at depth d < D_{max} randomly chosen from **function set F (IF, AND, =, >, *, etc.)**
  - nodes at depth d = D_{max} randomly chosen from **terminal set T (x,y,5000,NOC, etc.)**
- Grow method (each branch has depth ≤ D_{max}):
  - nodes at depth d < D_{max} randomly chosen from $F \cup T$
  - nodes at depth d = D_{max} randomly chosen from T
- Common GP initialisation: ramped half-and-half, where grow & full method each deliver half of initial population

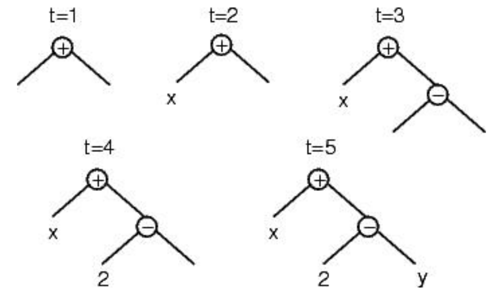
Genetic Programming: Full Initialisation to depth 2



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

UiO : Department of Informatics
University of Oslo

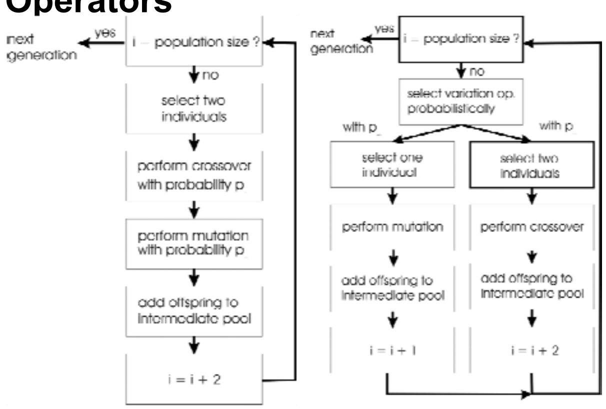
Genetic Programming: Grow Initialisation to depth 2



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

UiO : Department of Informatics
University of Oslo

Genetic Programming: Variation Operators

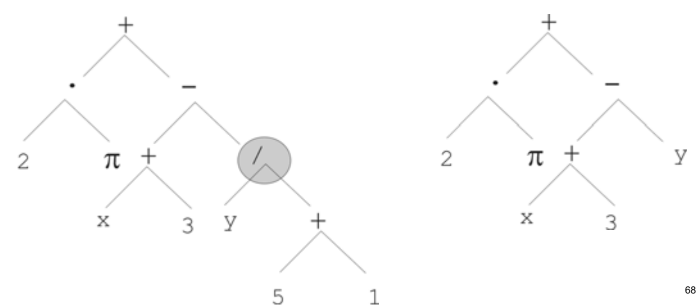


GA flowchart GP flowchart

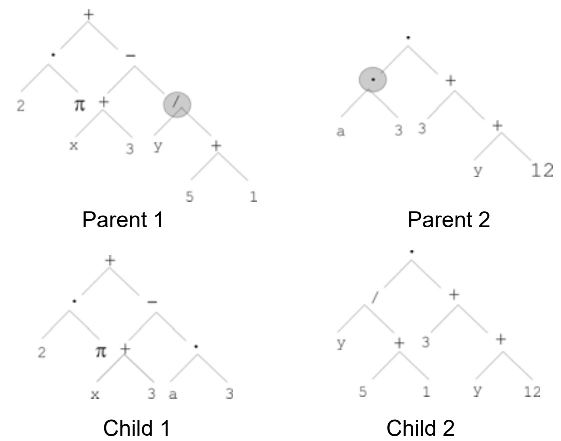
UiO : Department of Informatics
University of Oslo

Genetic Programming: Mutation

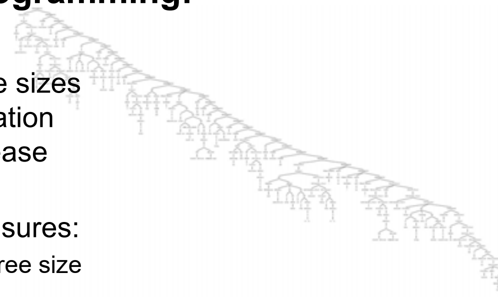
- Most common mutation: replace randomly chosen subtree by randomly generated tree



Genetic Programming: Recombination



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

Particle Swarm Optimisation: Quick overview

- Developed: in 1995 by Kennedy and Eberhart
- Inspired by social behavior of bird flocking/fish schooling

Particle Swarm Optimisation: Representation (1/2)

- Every population member can be considered as a pair $\langle \bar{x}, \bar{p} \rangle$ where the first vector is candidate solution and the second one a perturbation vector in IR^n
- The perturbation vector determines how the solution vector is changed to produce a new one: $\bar{x}' = \bar{x} + \bar{p}'$, where $\bar{p}'$ is calculated from $\bar{p}$ and some additional information

Particle Swarm Optimisation: Representation (2/2)

- A member is a point in space with a position and a velocity
- The perturbation is defined as the weighted sum of three components:
  - Current perturbation vector
  - Vector difference current position to best position of member so far
  - Vector difference from current position to best position of population so far

Image source: http://wirelesstechthoughts.blogspot.no/

A Reminder about Search Landscapes

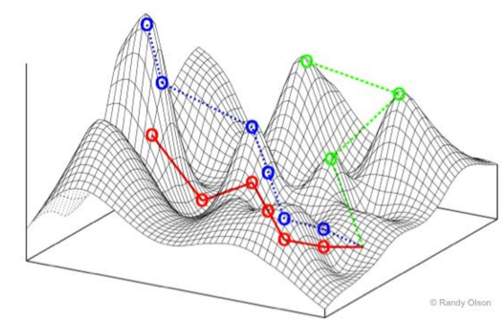


Image source: Randy Olson, wikipedia

PSO: Velocity and Position Update

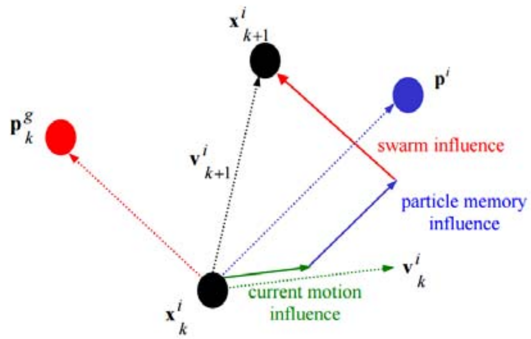
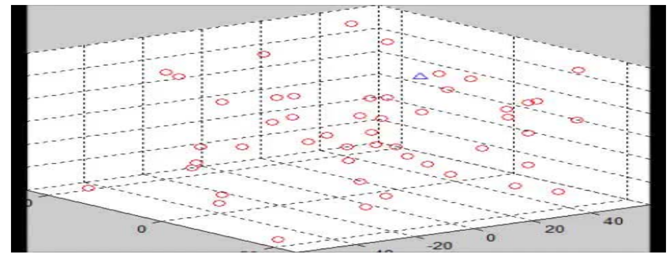


Image source: Hassan et al. 2004: A COPMARISON OF PARTICLE SWARM OPTIMIZATION AND THE GENETIC ALGORITHM

Particle Swarm Optimisation: Example moving target

- Optimum moves randomly
- Particles do not know position of optimum but do know which particle is closest and are attracted to that one



Particle Swarm Optimisation: Summary

Representation	Real-valued vectors
Recombination	None
Mutation	Adding velocity vector
Parent selection	Deterministic (each parent creates one offspring via mutation)
Survivor selection	Generational (offspring replaces parents)