Evolutionary Computing

Chapter 6

Chapter 6: Popular Evolutionary Algorithm Variants

Historical EA variants:

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

Genetic Algorithms: Quick Overview (1/2)

- Developed: USA in the 1960's
- Early names: J. Holland, K. DeJong, D. Goldberg
- Typically applied to:
  - discrete function optimization
  - benchmark
  - straightforward problems binary representation
- Features:
  - not too fast
  - missing new variants (elitsm, sus)
  - often modelled by theorists

Genetic Algorithms: Quick Overview (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 technical summary tableau

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

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 xover, 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

Adapted from A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing 2014

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 (step 5 in SGA repr. cycle) can be improved with explicit survivor selection

Evolution Strategies: Quick overview

- Developed: Germany in the 1960's
- Early names: I. Rechenberg, H.-P. 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: ES technical summary tableau

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

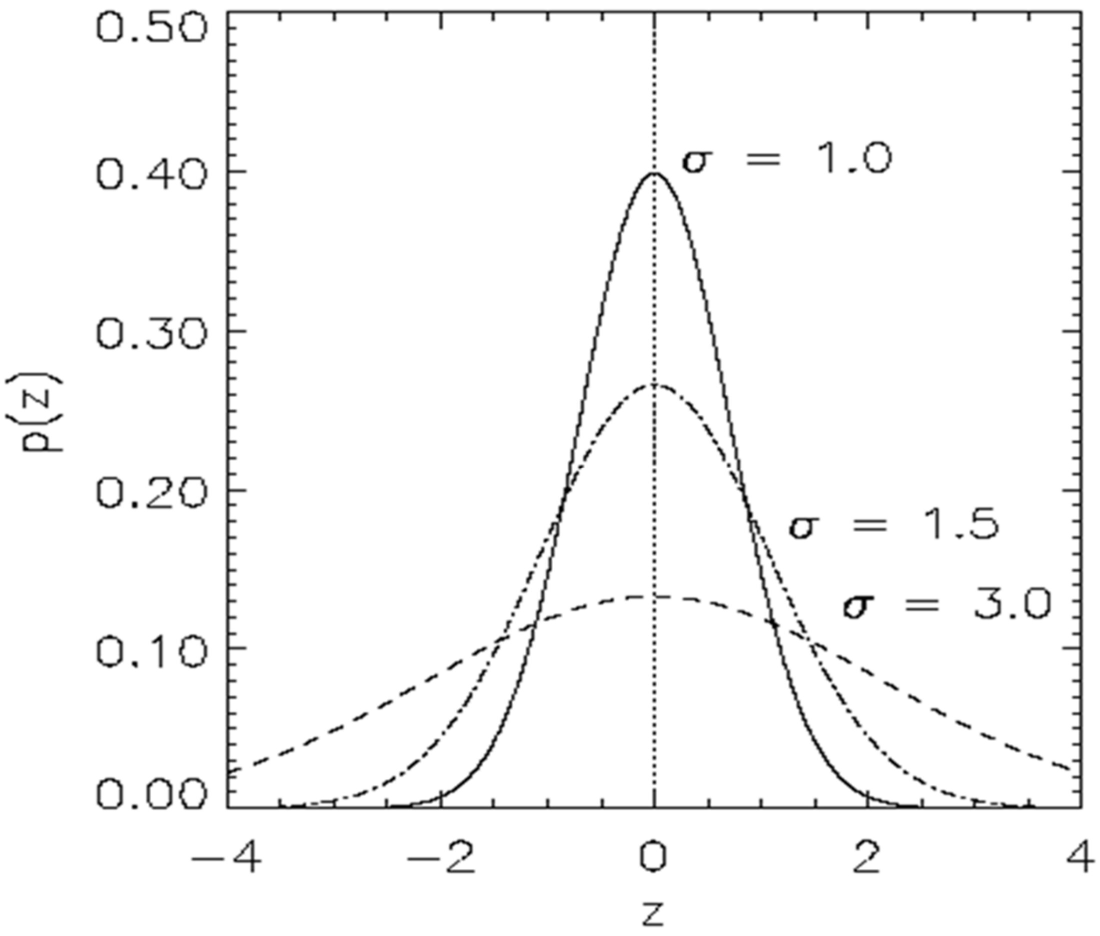
Evolution Strategies: Example (1+1) ES

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

Evolution Strategies: Introductory example: mutation mechanism

- 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":
- This rule 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, $0.8 \leq c \leq 1$

Evolution Strategies: Illustration of normal distribution



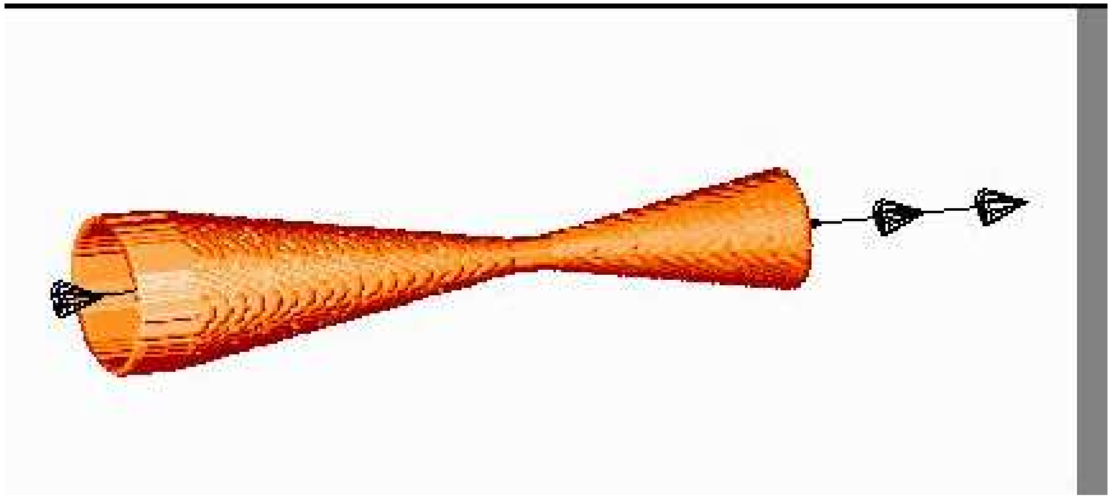
Another historical example: the jet nozzle experiment

Task: to optimize the shape of a jet nozzle
Approach: random mutations to shape + selection

Initial shape

Final shape

The famous jet nozzle experiment (movie)



Evolution Strategies: Representation

- Chromosomes consist of three parts:
  - Object variables: $x_1,\ldots,x_n$
  - Strategy parameters:
    - Mutation step sizes: $\sigma_1,\ldots,\sigma_{n_\sigma}$
    - Rotation angles: $\alpha_1,\ldots, \alpha_{n_\alpha}$
- Not every component is always present
- Full size: $\langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n ,\alpha_1,\ldots, \alpha_k \rangle$

  where k = n(n-1)/2 (no. of i,j pairs)

Evolution Strategies: Recombination

- Creates 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:
  - Using two selected parents to make a child
  - Selecting two parents for each position

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

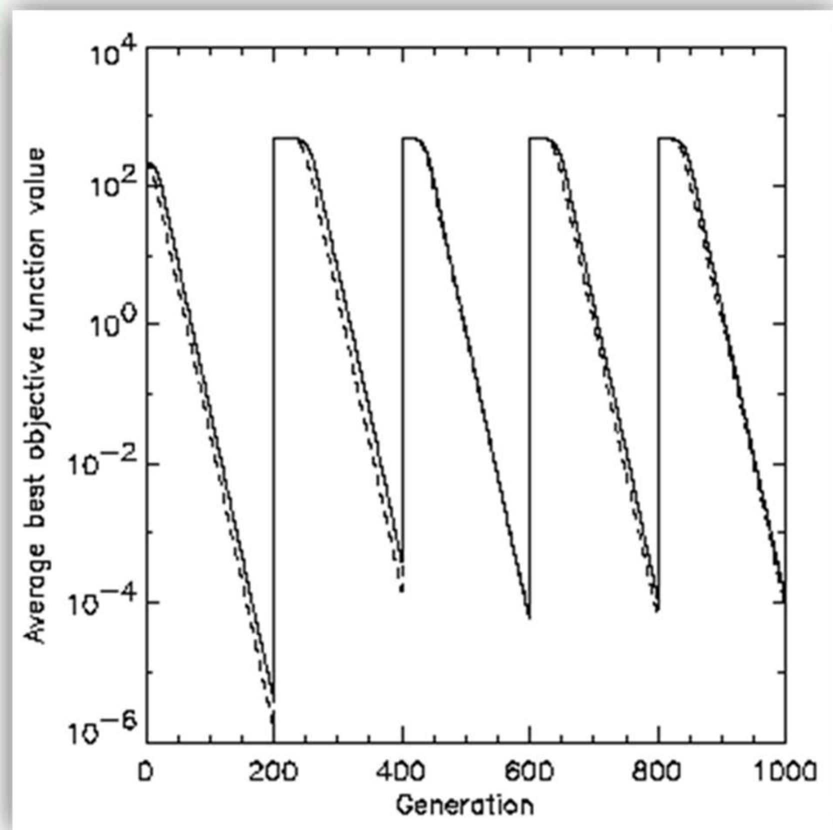
Evolution Strategies: Parent selection

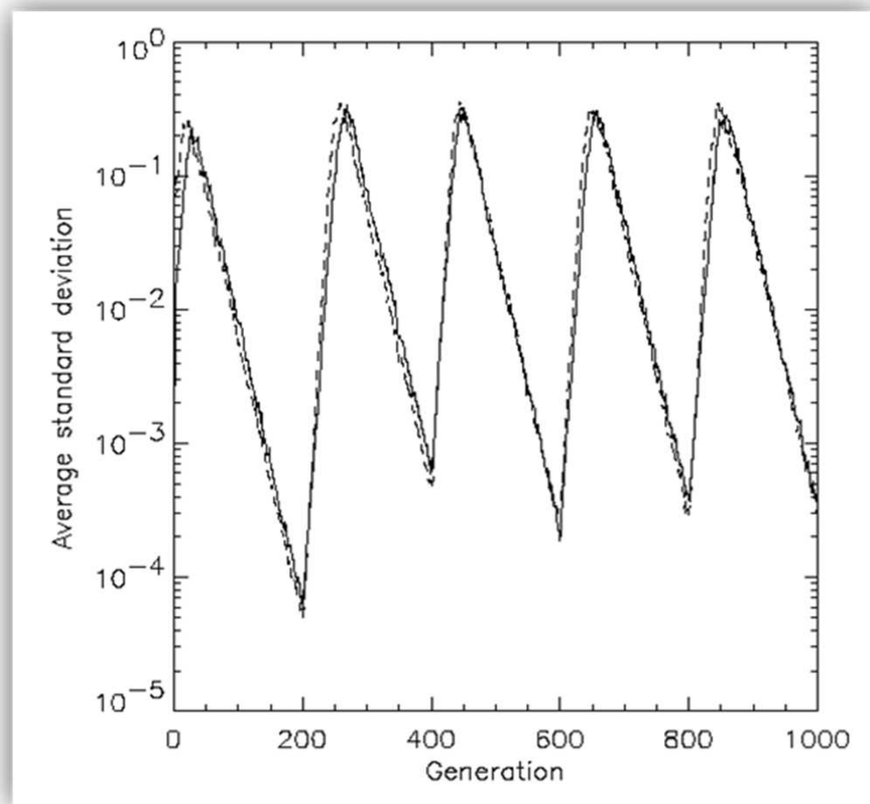
- Parents are selected by uniform random distribution whenever an operator needs one/some
- Thus: ES parent selection is unbiased - every individual has the same probability to be selected

Evolution Strategies: 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: Prerequisites for self-adaptation

- $\mu > 1$ to carry different strategies
- $\lambda > \mu$ to generate offspring surplus
- (μ,λ)-selection to get rid of misadapted σ's
- Mixing strategy parameters by (intermediary) recombination on them

Evolution Strategies: Selection Pressure

- Takeover time τ^* is a measure to quantify the selection pressure
- The number of generations it takes until the application of selection completely fills the population with copies of the best individual
- Goldberg and Deb showed:

$$\tau^* = \frac{\ln \lambda}{\ln(\lambda / \mu)}$$

- For proportional selection in a genetic algorithm the takeover time is $\lambda \ln(\lambda)$

Example application: The cherry brandy experiment (1/2)

- Task: to create a colour mix yielding a target colour (that of a well known cherry brandy)
- Ingredients: water + red, yellow, blue dye
- Representation: $\langle$ w, r, y ,b $\rangle$ no self-adaptation!
- Values scaled to give a predefined total volume (30 ml)
- Mutation: lo / med / hi σ values used with equal chance
- Selection: (1,8) strategy

Example application: The cherry brandy experiment (2/2)

- Fitness: students effectively making the mix and comparing it with target colour
- Termination criterion: student satisfied with mixed colour
- Solution is found mostly within 20 generations
- Accuracy is very good

Example application: The Ackley function (Bäck et al '93)

- The Ackley function (here used with n =30):

$$f(x) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$$

- Evolution strategy:
  - Representation:
    - $-30 < x_i < 30$
    - 30 step sizes
  - (30,200) selection
  - Termination : after 200000 fitness evaluations
  - Results: average best solution is $7.48 \cdot 10^{-8}$ (very good)

Evolutionary Programming: Quick overview

- Developed: USA in the 1960's
- Early names: D. Fogel
- 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
  – crossbred with ES (contemporary EP)
  – consequently: hard to say what "standard" EP is
- Special:
  – no recombination
  – self-adaptation of parameters standard (contemporary EP)

Evolutionary Programming: Technical summary tableau

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

Evolutionary Programming: Historical EP perspective

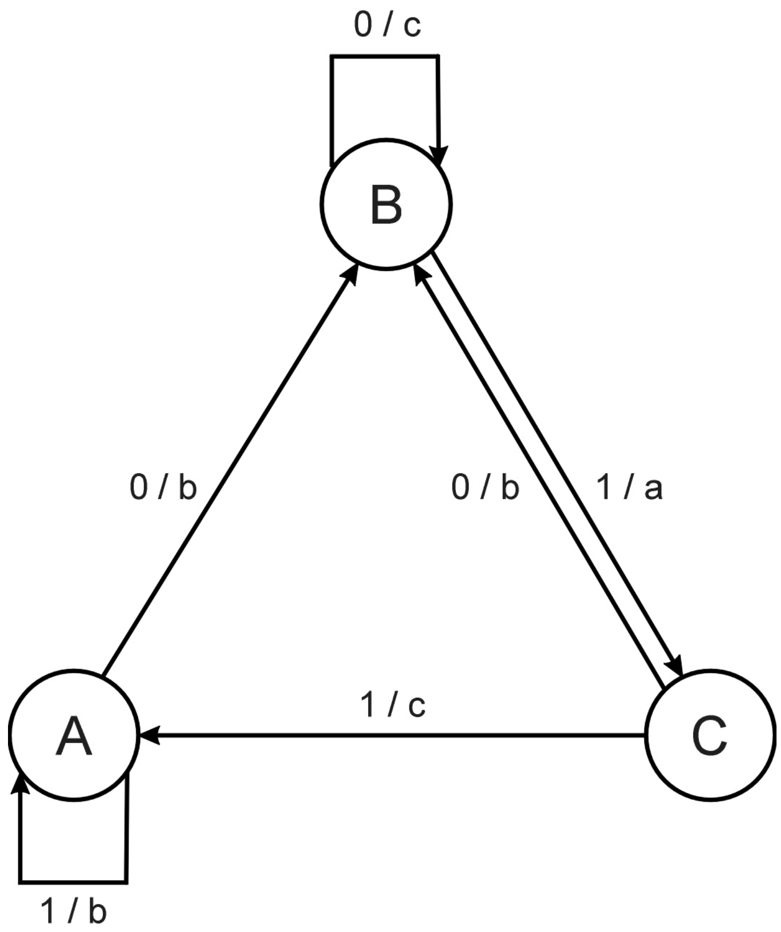
- EP aimed at achieving intelligence
- Intelligence was viewed as adaptive behaviour
- Prediction of the environment was considered a prerequisite to adaptive behaviour
- Thus: capability to predict is key to intelligence

Evolutionary Programming: Prediction by finite state machines

- Finite state machine (FSM):
  - States S
  - Inputs I
  - Outputs O
  - Transition function $\delta : S \times I \rightarrow S \times O$
  - Transforms input stream into output stream
- Can be used for predictions, e.g. to predict next input symbol in a sequence

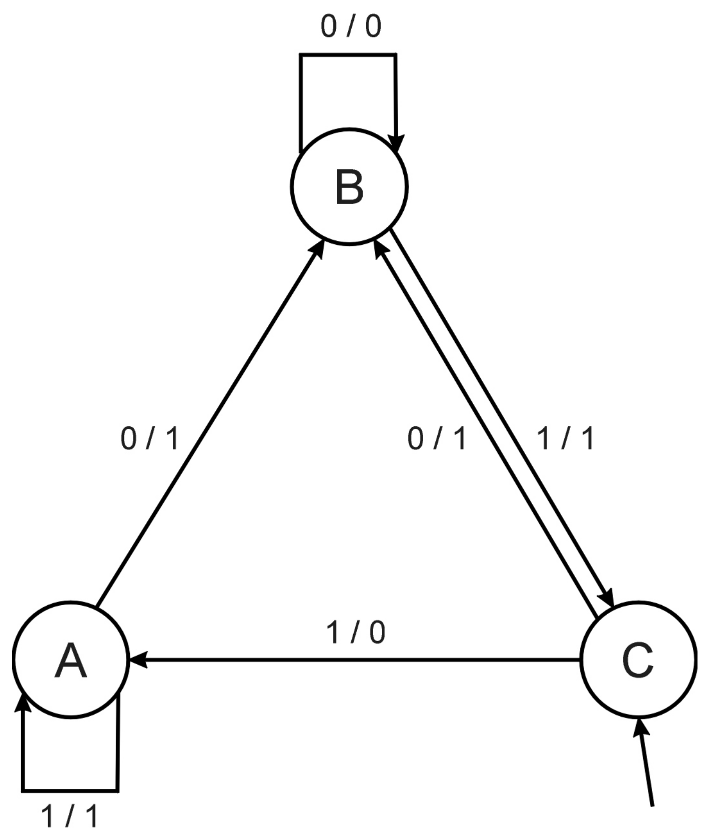
Evolutionary Programming: FSM example

- Consider the FSM with:
  - S = {A, B, C}
  - I = {0, 1}
  - O = {a, b, c}
  - δ given by a diagram



Evolutionary Programming: FSM as predictor

- Consider the following FSM
- Task: predict next input
- Quality: % of $in_{(i+1)} = out_i$
- Given initial state C
- Input sequence 011101
- Leads to output 110111
- Quality: 3 out of 5



Evolutionary Programming: Evolving FSMs to predict primes (1/2)

- $P(n) = 1$ if n is prime, 0 otherwise
- $I = N = \{1,2,3,\ldots, n, \ldots\}$
- $O = \{0,1\}$
- Correct prediction: $out_i = P(in(i+1))$
- Fitness function:
  - 1 point for correct prediction of next input
  - 0 point for incorrect prediction
  - Penalty for "too many" states

Evolutionary Programming: Evolving FSMs to predict primes (1/2)

- Parent selection: each FSM is mutated once
- Mutation operators (one selected randomly):
  - Change an output symbol
  - Change a state transition (i.e. redirect edge)
  - Add a state
  - Delete a state
  - Change the initial state
- Survivor selection: (μ+μ)
- Results: overfitting, after 202 inputs best FSM had one state and both outputs were 0, i.e., it always predicted “not prime”
- Main point: not perfect accuracy but proof that simulated evolutionary process can create good solutions for intelligent task

Evolutionary Programming: Modern EP

- No predefined representation in general
- Thus: no predefined mutation (must match representation)
- Often applies self-adaptation of mutation parameters

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: Mutation

- Chromosomes: $\langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n \rangle$
- $\sigma_i' = \sigma_i \bullet (1 + \alpha \bullet N(0,1))$
- $x_i' = x_i + \sigma_i' \bullet N_i(0,1)$
- $\alpha \approx 0.2$
- boundary rule: $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$
- Other variants proposed & tried:
  - Using variance instead of standard deviation
  - Mutate σ-last
  - Other distributions, e.g, Cauchy instead of Gaussian

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
- Much historical debate “mutation vs. crossover”

Evolutionary Programming: Parent selection

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

Evolutionary Programming: Evolving checkers players (Fogel’02) (1/2)

- Neural nets for evaluating future values of moves are evolved
- NNs have fixed structure with 5046 weights, these are evolved + one weight for “kings”
- Representation:
  - vector of 5046 real numbers for object variables (weights)
  - vector of 5046 real numbers for σ‘s
- Mutation:
  - Gaussian, lognormal scheme with σ-first
  - Plus special mechanism for the kings’ weight
- Population size 15

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

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

Genetic Programming: Quick overview

- Developed: USA in the 1990's
- Early names: J. Koza
- Typically applied to:
  - machine learning tasks (prediction, classification…)
- Attributed features:
  - competes with neural nets and alike
  - needs huge populations (thousands)
  - slow
- Special:
  - non-linear chromosomes: trees, graphs
  - mutation possible but not necessary

Genetic Programming: Technical summary tableau

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

Genetic Programming: Example credit scoring (1/3)

- Bank wants to distinguish good from bad loan applicants
- Model needed that matches historical data

ID	No of children	Salary	Marital status	OK?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
…				

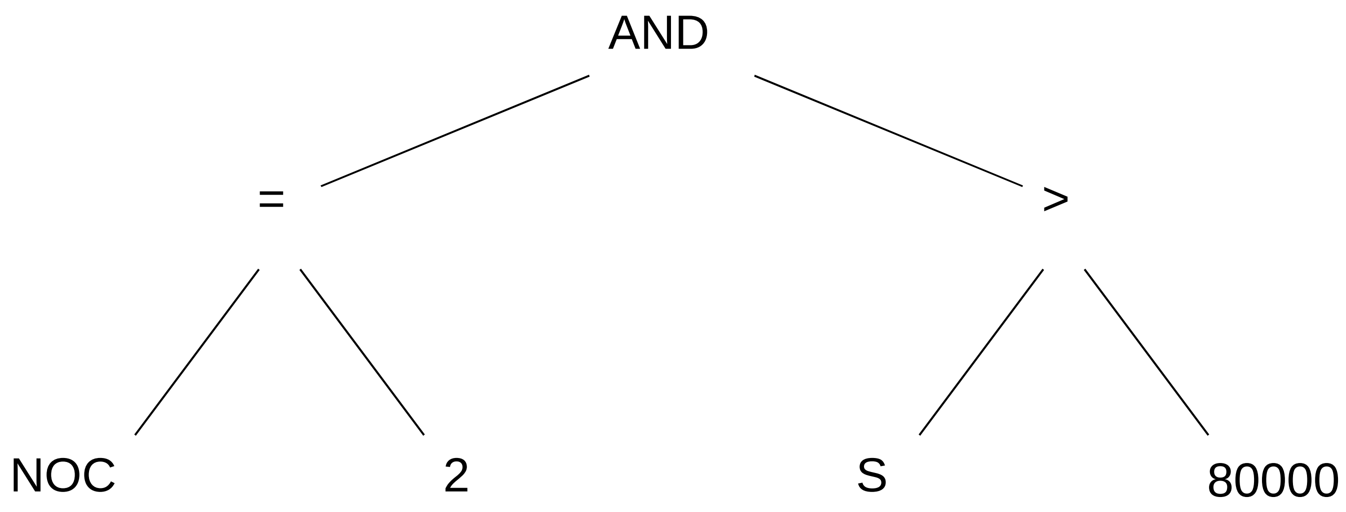
Genetic Programming: Example credit scoring (2/3)

- A possible model:
- IF (NOC = 2) AND (S > 80000) THEN good ELSE bad
- In general:
- IF formula THEN good ELSE bad
- Only unknown is the right formula, hence
- Our search space (phenotypes) is the set of formulas
- Natural fitness of a formula: percentage of well classified cases of the model it stands for

Genetic Programming: Example credit scoring (3/3)

IF (NOC = 2) AND (S > 80000) THEN good ELSE bad

can be represented by the following tree

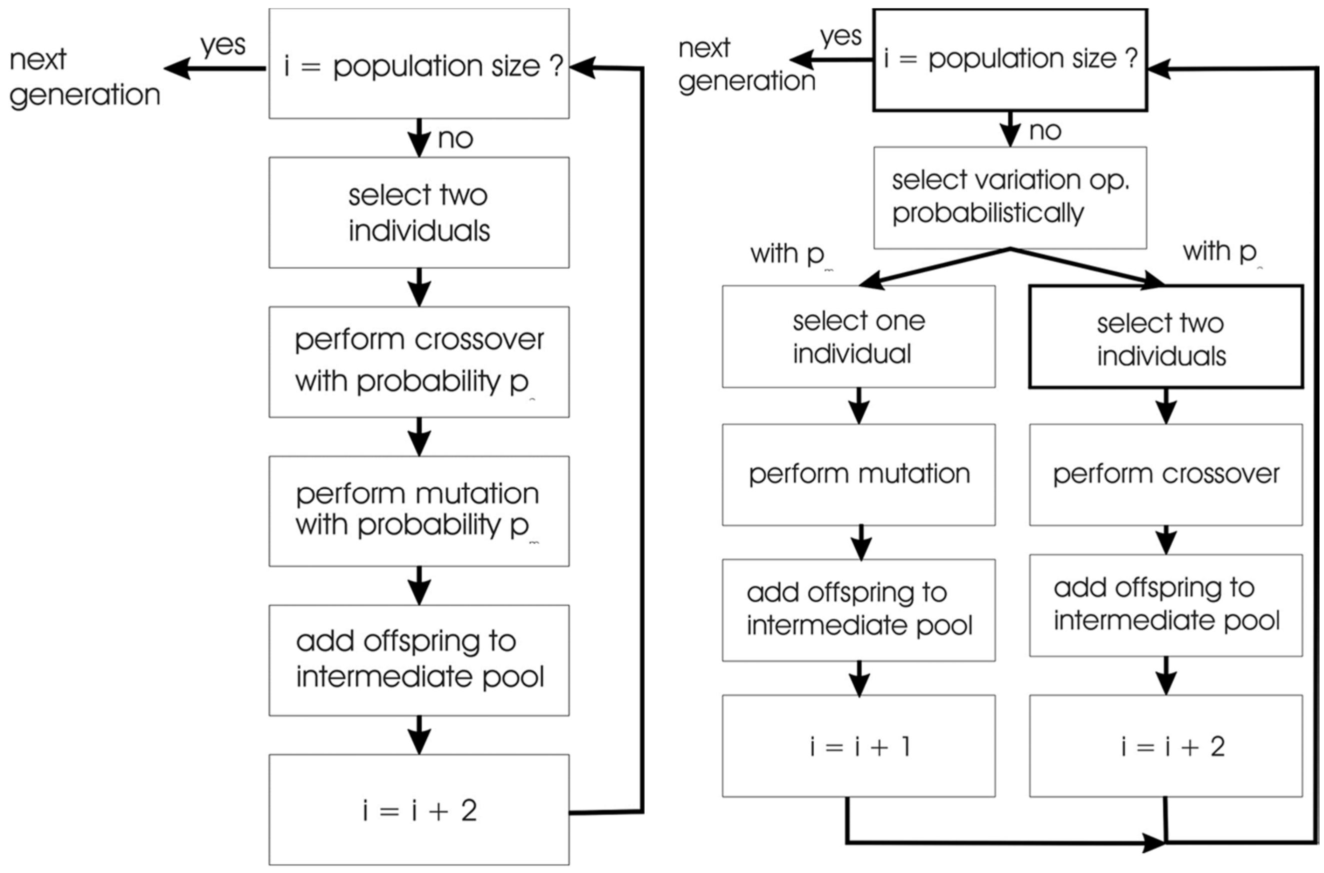


Genetic Programming: Offspring creation scheme

Compare

- GA scheme using crossover AND mutation sequentially (be it probabilistically)
- GP scheme using crossover OR mutation (chosen probabilistically)

Genetic Programming: GA vs GP



GA flowchart GP flowchart

Adapted from A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing 2014

Genetic Programming: Selection

- Parent selection typically fitness proportionate
- Over-selection in very large populations
  - rank population by fitness and divide it into two groups:
  - group 1: best x% of population, group 2 other (100-x)%
  - 80% of selection operations chooses from group 1, 20% from group 2
  - for pop. size = 1000, 2000, 4000, 8000 x = 32%, 16%, 8%, 4%
  - motivation: to increase efficiency, %'s come from rule of thumb
- Survivor selection:
  - Typical: generational scheme (thus none)
  - Recently steady-state is becoming popular for its elitism

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
  - nodes at depth $d = D_{max}$ randomly chosen from terminal set T
- Grow method (each branch has depth $\leq D_{max}$):
  - 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: Bloat

- Bloat = “survival of the fattest”, i.e., the tree sizes in the population are increasing over time
- Ongoing research and debate about the reasons
- Needs countermeasures, e.g.
  - Prohibiting variation operators that would deliver “too big” children
  - Parsimony pressure: penalty for being oversized

Genetic Programming: Example symbolic regression

- Given some points in $\mathbf{R}^2$, $(x_1, y_1), \ldots, (x_n, y_n)$
- Find function f(x) s.t. $\forall i = 1, \ldots, n : f(x_i) = y_i$
- Possible GP solution:
  - Representation by F = {+, -, /, sin, cos}, T = $\mathbf{R} \cup \{x\}$
  - Fitness is the error $err(f) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$
  - All operators standard
  - pop.size = 1000, ramped half-half initialisation
  - Termination: n "hits" or 50000 fitness evaluations reached (where "hit" is if $| f(x_i) - y_i | < 0.0001$)