Evolutionary Algorithms

Inteligencia Artificial en los Sistemas de Control Autónomo

Objectives

- Describe the most relevant EAs

Bibliography

- Eiben, A.E. and Smith, J.E. *Introduction to Evolutionary Computing*. Springer 2003.

Table of Contents

1. Genetic Algorithms
   - Introduction
   - Representation
   - Mutation
   - Recombination
   - Selection
2. Genetic Programming
   - Introduction
   - Representation
   - Mutation
   - Recombination
   - Initialization
   - Bloat in Genetic Programming
3. Evolution Strategies
   - Introduction
   - Representation
   - Mutation
   - Recombination
   - Parent and survivor selection

Genetic Algorithms

Introduction (I)

Introduced by Holland in the 70's

- John H. Holland ``*Adaptation in Natural and Artificial Systems*", MIT Press
- GA is the most popular EA
- Usually EAs confused with GA

Canonical GA (which is not canonical)

- Fixed length strings
- Binary codification
- Holland's Theorem

Representation	Bit strings
Recombination	1-point
Mutation	Bit flip
Parent select	Fitness prop
Survivor select	Generational

Genetic Algorithms

Introduction (II)

GAs are a family of algorithms, with common features

- Representation in strings, named chromosomes
- Mutation and recombination
- Usually fixed length

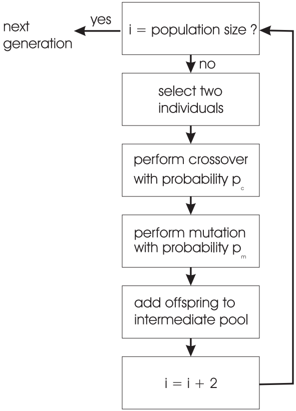
GAs are like a toolbox with customizable components

- Representations, genetic operators, selections mechanism, ...
- These components are interdependent

Rule of thumb: Small genotype changes $\Rightarrow$ Small phenotype changes

Genetic Algorithms

Introduction (III)



Genetic Algorithms

Representation: Binary

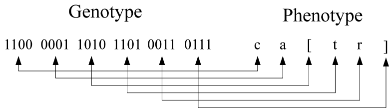
1	1	0	1	0	0	0	1

One of the oldest and widely used codifications

- Consequence of Holland's Theorem
- Strong historical influence

Often used to codify non-binary information (not recommended)

- Pure binary codification
- Gray coding
- Custom codification



Hint: Use binary codification to represent binary information

Genetic Algorithms

Representation: Integer

4	3	2	1	0	4	2	3	3

Chomosome as a sequence of integers

- More natural codification for many problems
- Optimization of integer values
- Integer representation ($\{1, 2, 3, 4\} = \{\mathit{North}, \mathit{East}, \mathit{South}, \mathit{West}\}$)

Genetic Algorithms

Representation: Floating-point

1.1	0.2	3.0	33.2	0.0	-3.2	130.1	88.3	-7.1

Chomosome as a sequence of floating-point values

- Common in optimization problems
- Solutions with continous nature

Genetic Algorithms

Representation: Permutation

4	3	2	5	6	1

Some problems involve order

- Sequence of integers
- No repeated numbers
- Range of valid numbers
- Special genetic operators

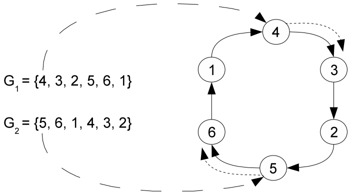
Information can be contained in

- The locus (position)

  $$[3, 1, 2, 4] \Rightarrow [C, A, B, D]$$

- The allele (value)

  $$[3, 1, 2, 4] \Rightarrow [B, C, A, D]$$



Integer codification to solve TSP

Genetic Algorithms

Mutation

Mutation: Genetic operator that uses one parent

- Introduces randomness into the genotype
- Depends on representation

Main objectives

- Avoid local minima (premature convergence)
- Enhances exploration

Often dependent on the mutation rate

- Significant influence in the algorithm behaviour
- Higher mutation rate, higher exploration

Genetic Algorithms

Mutation for binary representations

Flip bit with probability p_m

1	0	1	0	0	0	0	1	0	→	1	0	0	1	0	0	0	0	0

Optimal p_m depends on the problem and goals

- Need of high fitness population
- Need of high fitness individual
- Need of genetic diversity
- Modality of the problem
- Algorithm dynamics

Rule of thumb: $p_m = \frac{1}{length}$

Genetic Algorithms

Mutation for integer representations

Two main mutations applied to each gene

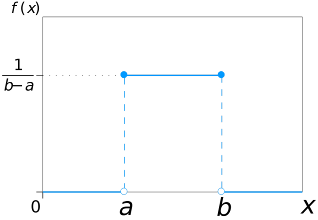
- **Random resetting**: Choose new random value with p_m
- **Creep mutation**: Add small (positive or genative) random value with p_m

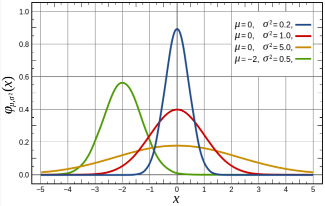
Genetic Algorithms

Mutation for floating-point representations

Set new value with value drawn from a distribution

- **Uniform mutation** Choose new random value from [L, U] with p_m
- **Non-uniform mutation** Usually adding a value drawn from a zero-mean gaussian distribution





Genetic Algorithms

Mutation for permutation representations

Genes are no longer independent

- No gene mutation, p_m affects the whole chromosome

Swap mutation

1 2 3 4 5 6 7 8 9 $\longrightarrow$ 1 5 3 4 2 6 7 8 9

Insert mutation

1 2 3 4 5 6 7 8 9 $\longrightarrow$ 1 2 5 3 4 6 7 8 9

Scramble mutation

1 2 3 4 5 6 7 8 9 $\longrightarrow$ 1 3 5 4 2 6 7 8 9

Inversion mutation

1 2 3 4 5 6 7 8 9 $\longrightarrow$ 1 5 4 3 2 6 7 8 9

Genetic Algorithms

Recombination

Recombination creates one individual from two or more parents

- Also known as crossover (specially for two parents)
- Basic feature in GA
- Parents selection mechanism needed

Usually applied to all new individuals

- Not used when elitism is applied
- Sometimes applied with $p_c \in [0{,}5, 1]$

Objectives of recombination

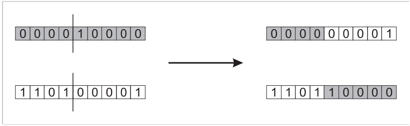
- Combine parents' behavior $\Rightarrow$ No new genetic material
- Constructive role
- Enhances explotation

Genetic Algorithms

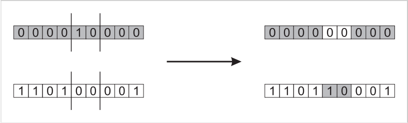
Recombination: Binary and integer representations

Three crossover mechanisms for binary and integer encodings

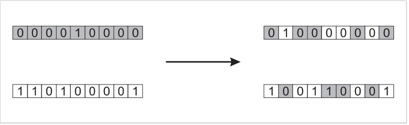
One-point crossover



Two-points crossover



Uniform crossover



Genetic Algorithms

Recombination: Floating point representations (I)

Discrete recombination

- Analogous to binary recombination
- No new genetic material

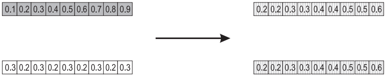
Arithmetic recombination

- Combines the parents' genes
- Weighted sums of genes: $z_i = \alpha x_i + (1 - \alpha) y_i$
- Usually, $\alpha = 0{,}5$ (average values)
- Different arithmetic recombinations

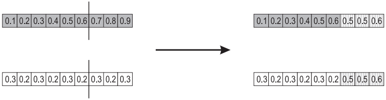
Genetic Algorithms

Recombination: Floating point representations (II)

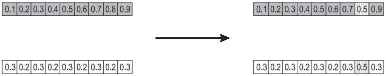
Whole arithmetic recombination
(All genes are included)



Simple arithmetic recombination
(Similar to one-point crossover)



Single arithmetic recombination
(Similar to uniform crossover)

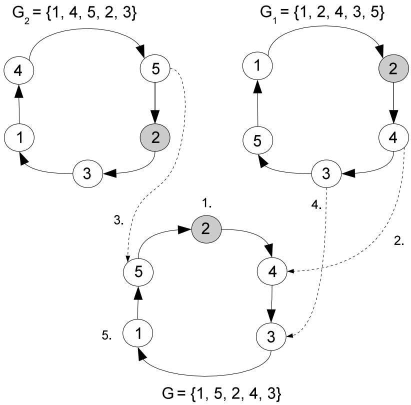


Genetic Algorithms

Recombination: Permutation representations

Specialized recombinations

- Partially Mapped Crossover
- Edge Crossover
- Order Crossover
- Cycle Crossover



Genetic Algorithms

Selection

Two purposes for selection

- Parent selection: Individuals to generate offspring
- Survivor selection: Individuals to remplace

Usually same methods applied to both

Genetic Algorithms

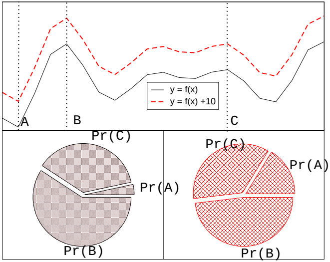
Selection: Fitness Proportional Selection

Selection probability proportional to fitness

- Premature convergence
- Lack of selective pressure for close fitness values
- Selective pressure not customizable
- Susceptibility to function transposition

Historically relevant

$$p_s = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$$



Genetic Algorithms

Selection: Ranking Selection

Selection probability proportional to rank

- Individuals are sorted by fitness
- Arbitrary rank to probability mapping
- Avoid problems with super individuals
- Selective pressure independent of fitness
- Selective pressure not customizable

Linear mapping

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

$1,0 < s < 2,0$

Exponential mapping

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

c = normalization factor

Genetic Algorithms

Selection: Tournament Selection

Algorithm of tournament size k

1. Select randomly k chromosomes
2. Compute their fitness
3. Select the fittest one
4. Go to 1

Customizable selective pressure

- Depends on k and μ

De facto standard

- Good for parallel computation
- Efficient implementation

Usually $k = 2$ in GA, in GP $k = 7$

Genetic Algorithms

Selection: Survival selection

Two strategies

- Generational (all the population is remplaced)
- Steady-stade (partial remplacement)

Survival selection algorithms

- Fitness-Based Replacement (inverse of the previous ones)
- Age-Based Replacement
- Elitism

Genetic Programming

Introduction (I)

GP is a family of algorithms

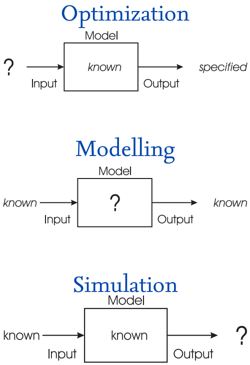
- Evolve programs
- Self-programming computers
- GP, Linear GP, Cartesian GP, EDA, ...

GP introduced by Koza in the 90's

Koza, J.R. ``*Genetic Programming: On the Programming of Computers by Means of Natural Selection*", MIT Press. 1992

GA and ES focused on optimization

- GP focused on Machine Learning



Genetic Programming

Introduction (II)

Example: Credit scoring problem within a bank. Develop a model describing good customers

Id	Children	Salary	Status	Credit
Id-1	2	45.000	Married	0
Id-2	0	30.000	Single	1
Id-3	1	40.000	Married	1
Id-4	2	60.000	Divorced	1
...				
Id-X	2	50.000	Married	1

Possible model:

```
IF (children=2) AND (Salary>80.000) THEN good ELSE bad
```

Genetic Programming

Introduction (III)

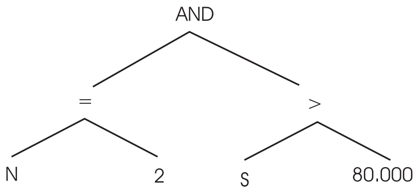
General form

```
IF (Formula)
 THEN good
  ELSE bad
```

In EC terms

Phenotype: Formula

Fitness: Classification accuracy



`(children=2) AND (Salary>80.000)`

Genetic Programming

Representation (I)

GP representation differs in two aspects

- Nonlinear structure
- Variable size

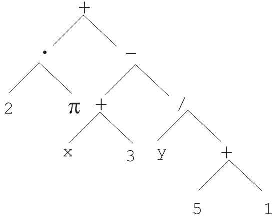
New representation and genetic operators

- Same selection (done in phenotipic space)

Genetic Programming

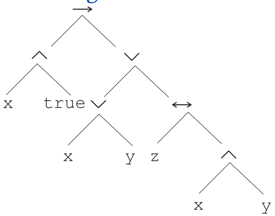
Representation (II)

Arithmetic formula



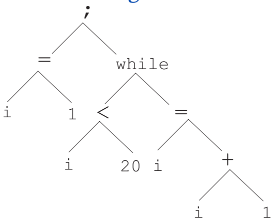
$$\left(2\pi+\left((x+3)-\frac{y}{5+1}\right)\right)$$

Logical formula



$$(x \wedge true) \rightarrow$$
$$((x \vee y) \vee (z \leftrightarrow (x \vee y)))$$

Program




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

Genetic Programming

Representation (III)

Two types of nodes

- **Function set** Internal nodes. It has an ssociated number of attributes
- **Terminal set** Leaves of the tree

Danger: Inviable trees

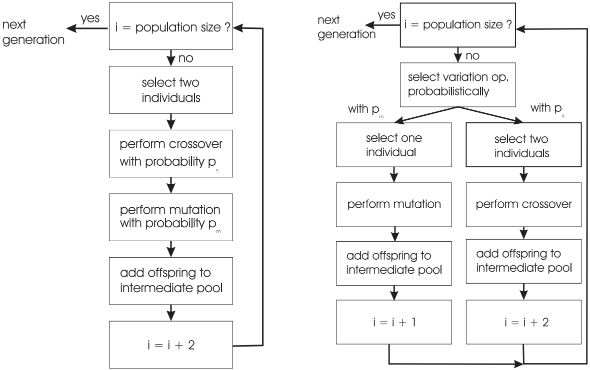
- Grammar-aware GP variants
- Strongly Typed Genetic Programming (STGP), Grammatical Evolution (GE), ...

(Complex representation example)

Genetic Programming

Mutation (I)

Application of genetic operators in GP contrast to GA



Genetic Programming

Mutation (II)

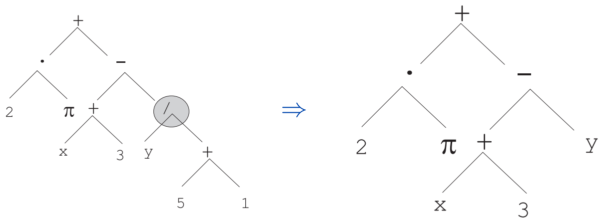
Subtree mutation

1. Select a random node
2. Delete subtree
3. Add new random subtree

Parameters

- Probability of choosing a terminal node

Highly correlated with code bloat



Genetic Programming

Mutation (III)

Alternative mutation operators

- Size-fair subtree mutation
- Node replacement mutation (point mutation)
- Hoist mutation
- Shrink mutation

Genetic Programming

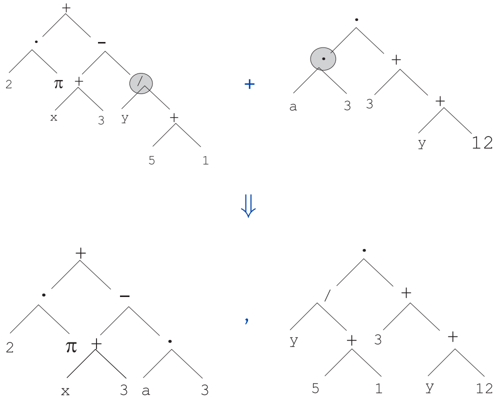
Recombination (I)

Subtree crossover

1. Take a random node from both parents
2. Swap subtrees

Parameters

- Probability of choosing a terminal node



Genetic Programming

Recombination (II)

Alternative recombination operators

- Homologous crossover
- Uniform crossover
- Size-fair crossover
- Node replacement mutation (point mutation)
- Hoist mutation
- Shrink mutation

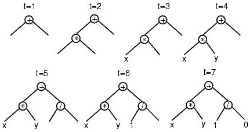
Genetic Programming

Initialization

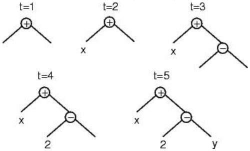
Three initialization methods

- **Full**. Introduces non-terminals nodes until max depth
- **Grow**. Introduces terminal or non-terminal with equal probability
- **Ramped half-n-half**. Applies full or grow with equal probability

Full (D=2)



Grow (D=2)



Genetic Programming

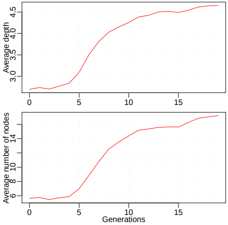
Bloat in Genetic Programming

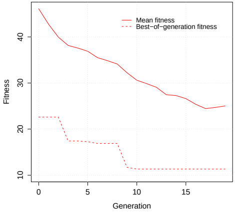
Code bloat: Uncontrolled grow of tree sizes

- Intrinsic to variable-length representations
- Undesirable effects
- Perhaps, the worse problem in GP

Countermeasures

- Depth limitation in genetic operators
- Parsimony pressure
- Tree plunning
- Multiobjective techniques





Cuadro 1: Main parameters used to obtain the approximations for secrets ID in the Genetic Tango attack against David-Prasad authentication protocol.

Parameter	ID
Population	500
Generations	10
Terminal Set	A, B, D, E, F, P_{ID1}, P_{ID2}
Function set	And, or, xor
Fitness	Hamming distance to secret
Fitness tags	5
Fitness sessions	100
Min. depth	1
Max. depth	3
Selection	Lexicographic tournament
Tournament size	4
Crossover	0.9
Reproduction	0.1
Elitism size	1
Terminals	0.1
Non terminals	0.9
Initialization	Rampled H-H

Evolution Strategies

Introduction (I)

Introduced by Rechenberg and Schwefel in the 60's

- Motivated by wing shape optimization
- Real-function optimization

ES properties

- Emphasis on mutation
- Mutation is gaussian noise
- Self-adaptation

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

Evolution Strategies

Introduction (II)

Example of basic ES

- Representation: Vector of real values
- Recombination: Not used
- Mutation: Gaussian noise with step-size σ

Adaptative σ (1/5 rule)

- Theorerical foundations
- Based on the ratio of success mutations (p_s)
- After k iterations a new σ is computed

$$\sigma = \begin{cases} \sigma / c & \text{if } p_s > 1/5, \\ \sigma \cdot c & \text{if } p_s < 1/5, \\ \sigma & \text{if } p_s = 1/5 \end{cases}$$

where $0{,}817 \leq c \leq 1$ is a parameter

Evolution Strategies

Representation

Nowdays ES is usually self-adapted

- Step size (σ) is included in the genotype
- Evolution includes variables and parameters

One or more σ values

- One σ: $\langle \underbrace{x_1, x_2, \ldots, x_n}_{\bar{x}}, \sigma \rangle$
- Several: σ : $\langle \underbrace{x_1, x_2, \ldots, x_n}_{\bar{x}}, \underbrace{\sigma_1, \sigma_2, \ldots, \sigma_{n_\sigma}}_{\bar{\sigma}} \rangle$

Evolution Strategies

Mutation

Genetic operators to modify σ

- Mutation with one step size:

$$x_i' = x_i + N_i(0, \sigma')$$
$$\sigma' = \sigma \cdot e^{\cdot N(0,\tau)}, \tau \propto 1/\sqrt{n}$$

τ is analogous to learning rate in ANN

- Mutation with n step sizes:

$$x_i' = x_i + N_i(0, \sigma_i)$$
$$\sigma' = \sigma \cdot e^{\cdot N(0,\tau') + N_i(0,\tau}$$

with $\tau' \propto 1/\sqrt{2n}$ and $\tau \propto 1/\sqrt{2\sqrt{n}}$

Evolution Strategies

Recombination

Secondary operator in ES

- **Discrete recombination**. Like uniform crossover in GA
- **Intermediate recombination**. Like arithmetic crossover in GA

ES tends to use **global recombination**

- More than two parents

Evolution Strategies

Parent and survivor selection

The whole population is seen as parent

- Select individual with uniform probability
- No selective pressure in parent selection

After creating the offspring, the λ fittests individuals are selected

- Deterministic procedure

Two selection mechamisms depending on who can be selected

- (μ, λ) **selection**. Only the offpring.
- $(\mu + \lambda)$ **selection**. Parents and offpring

(μ, λ) selection is more popular