

Evolutionary Computation

The Algorithms

by George Chronis

Genetic Algorithm

GA($\Sigma, G, \rho, f, \mu, I, S, \text{Gap}, \Omega, R, \tau$)

Σ	the search space (phenotypes)
G	the representation space (genotypes)
$\rho: \Sigma \rightarrow G$	the representation (expression) function
$f: \Sigma \rightarrow \mathbb{R}^+$	the fitness function
μ	the size of the population
$I: \mu \rightarrow P(G)$	initialization function
S	selection mechanism (roulette, ranking, tournament)
Gap	percentage of the population genetically operated on
Ω	genetic operators (crossover, mutation, w/ parameters)
R	replacement policy (worst, random)
τ	termination criterion (number of steps, fitness)

GA Pseudocode



```
// start with an initial time
t := 0;

// initialize a usually random population of individuals
initpopulation P (t);

// evaluate fitness of all initial individuals of population
evaluate P (t);

// test for termination criterion (time, fitness, etc.)
while termination condition not met do
  // increase the time counter
  t := t + 1;

  // select a sub-population for offspring production
  P' := selectparents P (t);

  // recombine the "genes" of selected parents
  recombine P' (t);

  // perturb the mated population stochastically
  mutate P' (t);

  // evaluate its new fitness
  evaluate P' (t);

  // select the survivors from actual fitness
  P := survive P,P' (t);
end while
```



Evolutionary Programming

● EP(Σ , G, ρ , f, μ , I, S, Gap, Ω , R, τ)

Σ	the search space (phenotypes)
G	the representation space (genotypes)
$\rho: \Sigma \rightarrow G$	the representation (expression) function
$f: \Sigma \rightarrow \mathbb{R}^+$	the fitness function
μ	the size of the population
$I: \mu \rightarrow P(G)$	initialization function
S	selection mechanism (roulette, ranking, tournament)
Gap	percentage of the population genetically operated on
Ω	genetic operators (mutation, w/ parameters)
R	replacement policy (worst, random)
τ	termination criterion (number of steps, fitness)

EP Pseudocode

```
// start with an initial time
t := 0;

// initialize a usually random population of individuals
initpopulation P (t);

// evaluate fitness of all initial individuals of population
evaluate P (t);

// test for termination criterion (time, fitness, etc.)
while not done do

    // perturb the whole population stochastically
    P'(t) := mutate P (t);

    // evaluate its new fitness
    evaluate P' (t);





    // stochastically select the survivors from actual fitness
    P(t+1) := survive P(t),P'(t);

    // increase the time counter
    t := t + 1;

end while
```

EP Characteristics

Similar to GA

-  Emphasis on linkage between parents and offspring instead of emulating genetic operators
-  Ideal for rugged optimization surfaces with lots of optimal solutions
-  No crossover
-  No encoding

Evolution Strategies

- Very similar to EP
 - ES use deterministic selection instead of tournament that is used in EP
 - ES use recombination
 - ES and EP both use mutation based on a distribution

Evolution Strategies

● EP($\mu, \lambda, l, R, f, \chi, \Delta\sigma, \Delta\theta, \tau$)

μ	size of the population
λ	number of offspring created in every step
l	length of string in every individual
R	replacement policy (plus, comma)
$f: R^l \rightarrow R^+$	the fitness function
χ	recombination operator (none, local intermediate)
$\Delta\sigma$	step-size for modifying standard deviation σ
$\Delta\theta$	step-size for modifying correlated mutation control θ
τ	termination condition (optimum, number of steps)

GA, EP, ES, GP

- Genetic Algorithm

- Encoding, Mutation, Crossover

- Evolutionary Programming

- No crossover, No encoding

- Evolution Strategy

- Deterministic selection instead of tournament, No encoding, Crossover

- Genetic Programming

- Individuals are programs, no mutation, tree representation

Initial Steps in Building an EA

- In order to build an evolutionary algorithm there are a number of steps that we have to perform:
 - Design a representation
 - Decide how to initialize a population
 - Design a way of mapping a genotype to a phenotype
 - Design a way of evaluating an individual

More Steps in Building an EA

- Design suitable mutation operator(s)
- Design suitable recombination operator(s)
- Decide how to select individuals to be parents
- Decide how to select individuals to be replaced
- Decide when to stop the algorithm

Design a Representation

Phenotype and Genotype Definitions

 Binary alphabet

 Real-numbers

 Permutations

Population Initialization

- Uniformly on the search space (if possible)
 - Binary strings: 0 or 1 with probability of 0.5
 - Real-valued strings: Uniformly on a given interval (for bounded values only)
- Seed from previous results or heuristics
 - Possible loss of genetic diversity
 - Possible unrecoverable bias

Genotype to Phenotype

- Straightforward process
- Growth function operates on genotype and problem data to produce phenotype

Evaluating an Individual

- Most costly step
 - Do not re-evaluate unmodified individuals
- Subroutine, black-box sim or external process (e.g. robot environment)
- Approximate fitness may be used in the beginning

More on Evaluation

- Constraint handling
 - Penalize the fitness
 - Specific evolutionary methods
- Multi-objective evolutionary optimization gives a set of compromise solutions

Mutation Operators

- One or more
- Important points
 - At least one mutation operator should allow every part of the search space to be reached
 - Size of the mutation should be controllable
 - Mutation should produce valid chromosomes

Kinds of Mutation

- Flipping in Binary
- Distribution in Real
- Swapping in order based

Recombination Operators

- One or more
- Important points
 - Child should inherit something from **each** parent (o/w it's mutation)
 - Should not be catastrophic (design in conjunction with representation)
 - Produce valid chromosomes

Selection Strategy

- Better individuals should have better chance of being parents
- Give less good individuals some chance of being parents - they may include useful genetic material
- Tournament or Fitness

Stopping Criterion

- Optimum reached
- CPU resources limit reached
 - maximum number of fitness evaluations
- User's patience limit reached
 - after some generations with no improvement

Keep the Balance

Exploration vs Exploitation

 Exploration: Sample unknown regions

 Exploitation: Improve best-so-far individuals